IBM WebSphere Application Server 64-bit Performance Demystified
11/11/2005
Version 2.0

WebSphere. software

IBM® WebSphere® Application Server and 64-bit platforms

64-bit Performance

J. Stan Cox, Piyush Agarwal, Nikola Grcevski, Hong Hua
stancox@us.ibm.com
agarwalp@us.ibm.com
nikolag@ca.ibm.com
honghua@us.ibm.com

WebSphere Application Server Performance
Research Triangle Park, NC

WebSphere. software

Contents

Executive Summary

The release of IBM® WebSphere® Application Server (WAS) version 6.0.1 and 6.0.2 introduces support for 64-bit computing platforms. These include AIX® and Linux® on POWER™ as well as Linux® and Windows® on x86-64.

Applications capable of taking advantage of WAS 64-bit features can see significant performance gains

64-bit WAS versions leverage 64-bit computing features providing two key advantages:

- the capability to configure and run with Java™ heaps much larger then the ~2-3Gb limitations of the 32-bit platforms
- automatic and transparent Just-In-Time (JIT) code generation enhancements to leverage 64-bit CPU performance extensions

The downside for 64-bit WAS applications is a significantly greater memory footprint that can lead to performance loss.

Applications running on WAS 64-bit can see both performance gains and performance loss. Significant performance gains can be observed in applications capable of taking advantage of WAS 64-bit features. For example, reducing database requests by leveraging a large heap space to cache database data can provide significant gains. However, there is also a downside for WAS 64-bit applications. All address references are 64-bit wide, twice the size of address references in 32-bit deployments. This results in an increased memory footprint and can reduce hardware cache efficiency. Therefore, applications may actually see a performance loss. General performance expectations for applications that do not specifically leverage 64-bit features are +/- 10% of 32-bit WAS applications.

WAS is available in both 32 and 64-bit editions for supported 64-bit platforms. Choosing the preferred edition for optimal performance requires an assessment of the application's requirements and characteristics. Also, 32 and 64-bit WAS installs can co-exist allowing applications to run in a mixed mode. This provides a simplified migration path for 32-bit applications. They can initially be deployed on WAS 32-bit and gradually be migrated to a 64-bit version.

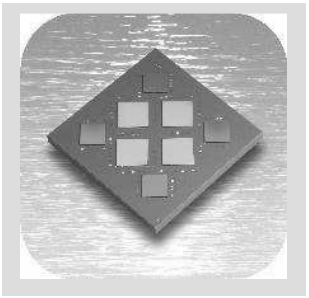
This report details WAS performance on 64-bit platforms including:

- 64-bit processor technology overviews
- IBM Java/WAS enhancements for 64-bit
- Detailed WAS 64-bit performance results
- WAS cross platform performance comparison

WAS and 64-bit computing

WAS 64-bit platforms

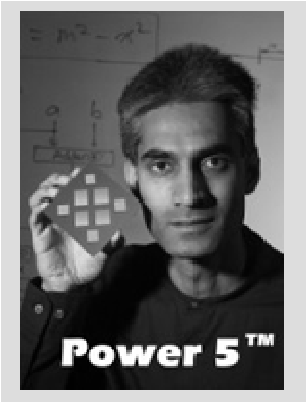
IBM POWER (Performance Optimization With Enhanced RISC) Architecture



The POWER performance RISC (**R**educed **I**nstruction **S**et **C**omputing) chip was first developed by IBM in the 1970s. The chips in the POWER line are numbered 1 through 5, with POWER5+ being the latest available.

The RISC architecture attempts to increase CPU speeds by reducing size and complexity of the instruction set. It supports a limited and simple set of fixed length and fixed format instructions running at fast clock speeds and supporting high degrees of parallelism. They make extensive use of a large bank of registers to reduce memory accesses and an extensive use of pipelining. They are designed to substitute raw speed for complexity and have limited addressing modes to simplify decoding.

The POWER Architecture incorporates characteristics common to most other RISC architectures. Instructions are a fixed length (4 bytes) with consistent formats, permitting a simple instruction decoding mechanism. Load and store instructions provide all of the accesses to memory. The architecture provides a set of general purpose registers (GPRs) for fixed-point computation, including the computation of memory addresses. It provides a separate set of floating-point registers (FPRs) for floating-point computation. All computations retrieve source operands from one register set and place results in the same register set.



The IBM POWER architecture supports both 32-bit and 64-bit computing. It is unique among the existing RISC architectures in that it is functionally partitioned, separating the functions of program flow control, fixed-point computation, and floating-point computation. The architecture's partitioning facilitates the implementation of superscalar designs, in which multiple functional units concurrently execute independent instructions.

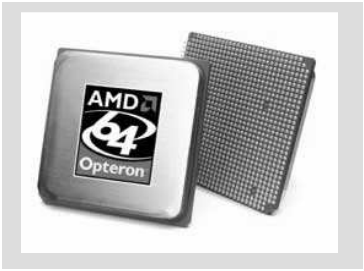
IBM now offers a full line of POWER5 products that range from desktops to supercomputers. It features the following key improvements

- 120 GPRs (increased by 40 compared to POWER4+) and 120 FPRs (increased by 48 compared to POWER4+)
- Simultaneous multithreading (SMT) - two threads executing simultaneously per processor. Takes advantage of unused execution unit cycles for better performance.

- Processors use Dual core module or Quad core module sharing the L2 cache.
- Cache improvements - larger, faster, highly associative L2 and L3 caches
- Improved chip-memory bandwidth - ~16 GB/sec. This is 4x faster than POWER4.
- Micro-partitioning: running up to 10 copies of the OS on a processor.
- Dynamic Resource balancing for efficient prioritization of threads on the SMT stream
- Advanced power management, dedicated single-tasking mode, Hypervisor (virtualization technology - allows multiple operating systems to run, unmodified, at the same time), and eFuse (hardware re-routing around faults).
- Supports increased CPUs - up to 64 per node

Since POWER3, the POWER platform offers 64-bit implementations of the architecture only, but they provide native support for both 64-bit and legacy 32-bit applications to run on the same system. The POWER architecture has native support for byte (8-bit), halfword (16-bits), word (32-bit), and doubleword (64-bit) data types. The Linux and AIX software "stack" that runs on the POWER platform includes both 32-bit and 64-bit applications, and IBM supports 64-bit distributions on POWER that run 32-bit and 64-bit applications (from two major Linux distributors, Red Hat and Novell SUSE) and IBM AIX.

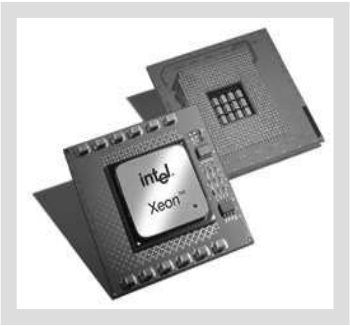
AMD/Intel x86-64



In 2003 AMD introduced x86-64 extension technology. Many prior 64-bit processor designs, such as the Intel® Itanium™, abandoned former 32-bit designs for a full 64-bit re-design. The x86-64 architecture however, builds directly on top of 32-bit. Extensions were added on top of the ubiquitous x86-32 instruction set to provide 64-bit addressing.

Intel followed the AMD x86-64 with a compatible design of their own. The Intel EM64T line of processors supports the same 64-bit addressing extensions found in the AMD design. Thus, Intel EM64T processors are, for the most part, binary compatible with AMD64-bit. Key features of the x86-64 designs which are common to both AMD and Intel processors include:

- Extended memory addressability (64-bit pointers & registers).



- Eight new 64-bit general purpose registers in 64 bit mode
- Eight new 128-bit SSE registers
- Double precision (64-bit) integer support.

Many 64-bit processor architectures rely on some degree of software emulation to support 32-bit applications. This usually results in significant performance penalties. Both the AMD and Intel x86-64 processors however, run entrenched x86-32 bit applications natively. This enables a straight forward migration path to 64-bit technology by providing both compatibility and performance for 32-bit applications. 32-bit applications can be run simultaneously with 64-bit applications while migrating individual applications to 64-bit technology as needed.

WAS 64-bit support

WAS introduces support for 64-bit platforms starting with release 6.0.1 in the first half of 2005. WAS 6.0.1 is available in 32-bit and 64-bit installs on Linux® (Suse Linux Enterprise Server (SLES), Red Hat) for the x86-64 platform and the POWER platform. WAS 6.0.2 extends support for Microsoft® Windows® server on x86-64 and 64-bit AIX on POWER platforms.

The key advantages for WAS 64-bit applications include dramatically increased Java heap size capability and Java generated code optimizations to fully leverage 64-bit performance extensions.

The key advantages for 64-bit WAS applications include dramatically increased Java heap size capability and Java generated code optimizations to fully leverage 64-bit performance extensions. WAS 32-bit provides a maximum heap size of around 2 gigabytes (GB). 64 bit platforms can address as much as 256 Terabytes (TB), far beyond current practical limits on platform physical memory. WebSphere/Java set no artificial limitation to the size of the heap below the 64-bit addressing maximum. In practice, however, the largest of applications that can exploit very large heaps will generally not need more than several GB of memory.

Both POWER and x86-64 processors provide performance extensions in 64-bit mode over 32-bit. For example, x86-64 provides 8 extra general purpose registers that are only available in 64-bit mode. WAS applications do not require re-compilation or other porting to take advantage of these 64-bit platform features. Applications launched by a WAS 64-bit base install will automatically support 64-bit addressing. Also, the IBM Java Just-In-Time compiler (JIT) recognizes 64-bit platforms and automatically generates optimized code to leverage 64 bit extensions available on each platform.

WAS 64-bit performance

Leveraging large heaps available in 64-bit

Caching vast amounts of data in large heaps can avoid latencies in accessing data from slower resources like database or disk resulting in large performance gains.

Applications that can take advantage of the large memory addressing capability of 64-bit can gain significant performance advantages. For example, caching vast amounts of data in main memory and thus avoiding latencies in accessing data from slower resources like database or disk can result in dramatic performance gains.

Figure 1 shows a modified WebSphere Trade6 scenario in which the Trade application has been modified to store and retrieve large binary objects (Jpeg images) from a database. The application uses WebSphere caching technology to cache the retrieved images in the WAS application server. For this scenario the in-memory cache size required for the entire dataset is greater than the addressable limits of a 32-bit WAS platform.

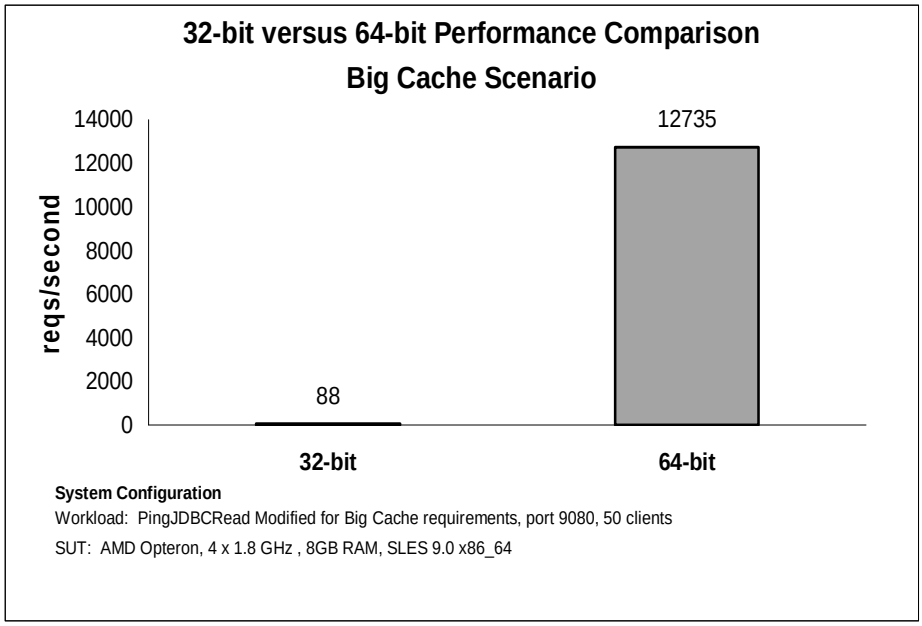


Figure 1: Leveraging a large heap for performance

The results show that on the 32-bit application server on the x86-64 platform the performance is very low as it is not able to cache the entire dataset in Java heap memory. This causes the application to access the

database repeatedly and fetch large objects over the network introducing significant overhead and latency. The 64-bit application server on the same x86-64 platform is able to allocate a cache large enough to cache the entire dataset. The application exploits this large heap by caching the entire working dataset, avoiding the repeated costly access to the remote database and dramatically improving application performance.

This example shows a "best-case" scenario for large heap usage by a WebSphere application. For example, applications that require cache entry invalidations will typically not see performance gains that are this dramatic. However, this example does show the significant gains possible with large heaps for applications that can leverage them.

Leveraging 64-bit platform computing features

Besides supporting larger Java heaps, 64-bit processors also provide hardware support for double precision (64-bit) mathematical computations and wider 64-bit integer and floating point registers. These features in 64-bit processors can lead to significant performance improvements on computationally intensive applications.

The IBM Java JIT compiler generates code automatically to improve performance on 64-bit processors. For example, computationally intensive applications usually contain tight loops with high levels of register pressure, meaning that, the more registers we have at our disposal the better the performance of the loop will be. An example of this kind of application, in the WebSphere application world, is the WebSphere Web services security code. Figure 2 shows a performance comparison of WebSphere security performance running in 32-bit and 64-bit mode. In general, having more registers helps reduce the need of the stack for temporary storage, also known as register spills. By reducing the register spills the IBM JIT reduces the memory loads and stores, improving the bus utilization and overall memory performance, which is critical in WebSphere based workloads.

Figure 2 and Figure 3 detail performance improvements for a set of Web services workloads. These tests use Web services security for authentication and message encryption/decryption. These security algorithms are high precision and computationally intensive. The benchmark was run with varying input and response SOAP payloads. For the smaller sized payloads, more time is spent in the security algorithms. For larger sized payloads XML parsing of the payload content is the main contributing factor. Hence you see a larger benefit for

Applications heavy in statistics, security and encryption, simulations, etc can benefit from the high precision computation support on 64-bit WAS

smaller size payloads compared to the larger size payloads. The 64-bit enabled WAS application outperforms the 32-bit version on both platforms. The average improvement is 47% across these workloads on the x86-64 platform. The POWER platform also provides significant improvement on average in 64-bit mode. However, the improvements are not as significant as the x86-64 due to the fact that the x86-64 platform has eight extra registers available only in 64-bit mode. This penalizes 32-bit performance on x86-64. All registers are available on the Power platform for both 32 and 64-bit applications.

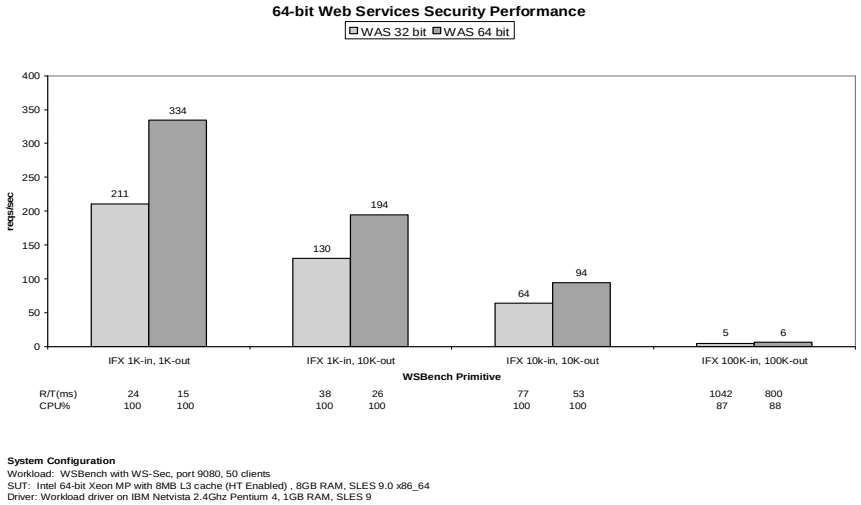


Figure 2: Web services security performance – x86-64

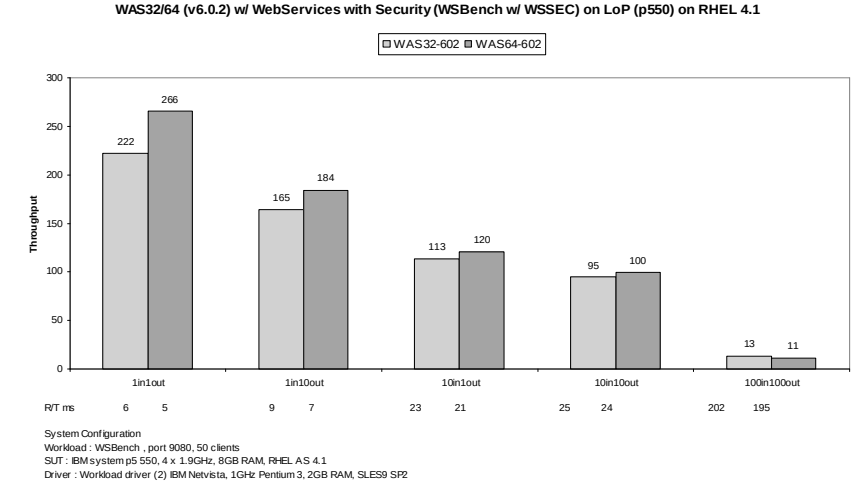


Figure 3: Web services security performance – POWER

IBM Java JIT technology can improve performance for WAS applications on 64-bit in other areas as well. For example, this is particularly important in WebSphere when dealing with java/util/Date objects. The java/util/Date timestamp is 64-bit value in Java, meaning, all java/util/Date based computations are 64-bit mathematical operations, which can see a performance gain on the 64-bit platforms.

WAS 64-bit increased memory footprint

As stated earlier in this paper, the downside for WAS 64-bit applications is increased memory footprint. In 64-bit deployments, each Java memory address reference is 8 bytes, versus 4 bytes in a 32-bit deployment. **Figure 4** shows the average memory growth for the WebSphere Trade6 performance application on 64-bit. The Trade6 memory footprint grows by 60-70% on 64-bit due to the increase in pointer size. Not only does this increase platform memory requirements, it will also generally cause a decrease in processor cache efficiency. This is due to higher number of caches misses caused by the increase in working set size.

In 64-bit deployments, each Java memory address reference is 8-bytes, versus 4 bytes on 32-bit. This results in a 60-70% memory growth for the WebSphere Trade application.

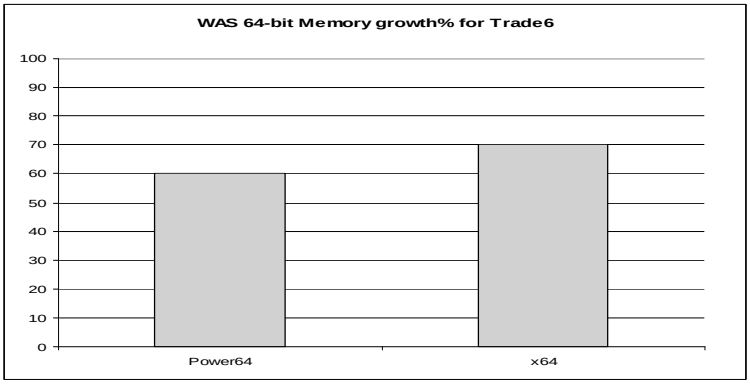


Figure 4: WAS 64-bit memory growth for Trade6

Figure 5, Figure 6 and **Figure 7** show a performance comparison running Trade 6 on 32-bit and 64-bit version of WAS on various platforms. With this configuration and database population, the Trade6 application does not benefit from the large heap capability 64-bit. Also, computationally intensive operations such as encryption are not enabled and the application does not have mathematically intensive algorithms.

The results show that Trade on WAS 32-bit configuration slightly outperforms the 64-bit configuration for both Linux and AIX on the POWER platform (3-7%). On the Intel 64-bit Xeon™ MP with 8MB L3 cache hardware the 64-bit version of WebSphere is slightly faster then the 32-bit version (~5%). However, the same tests were run on an x86-64 platform without L3 cache support. In those results, the 64-bit version of WebSphere lagged the 32-bit version by 15%.

The difference in performance is primarily due to growth of the memory reference size when going from 32 bit to 64 bit, and the corresponding decrease in the efficiency of the hardware processor cache. The large L3 cache in the prior results helped to mitigate this and thus reduced the impact on performance. It is generally recommended to use systems with large L2/L3 caches for optimal performance of applications on the 64-bit WebSphere.

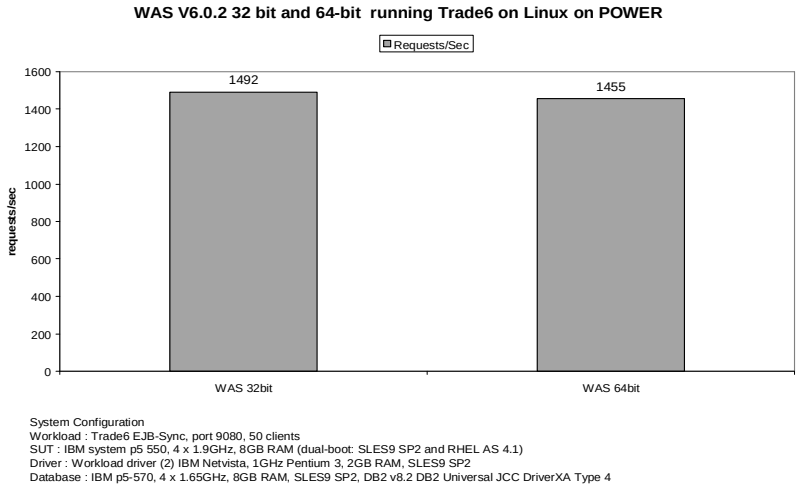


Figure 5: Trade6 64-bit performance on Linux on POWER

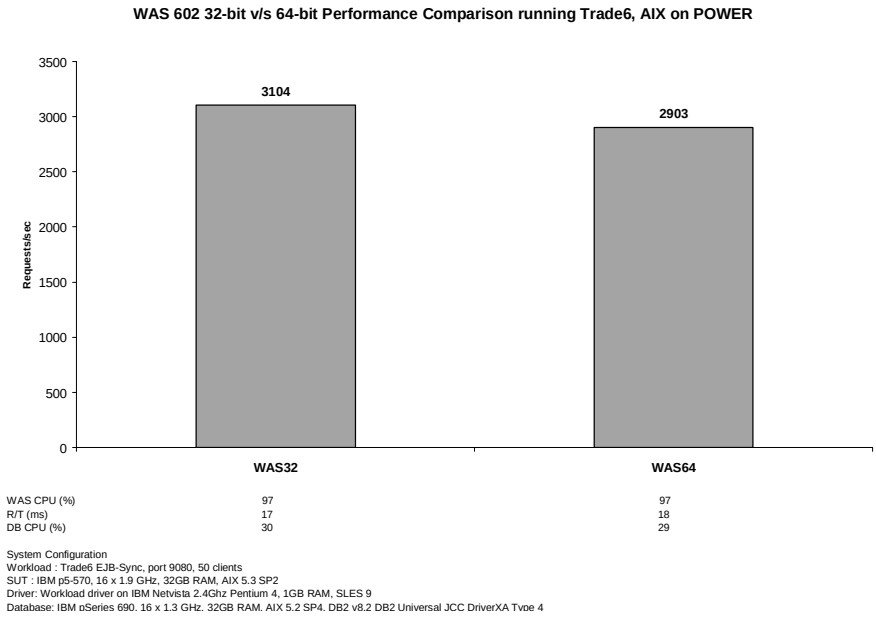


Figure 6: Trade6 64-bit performance on AIX on POWER

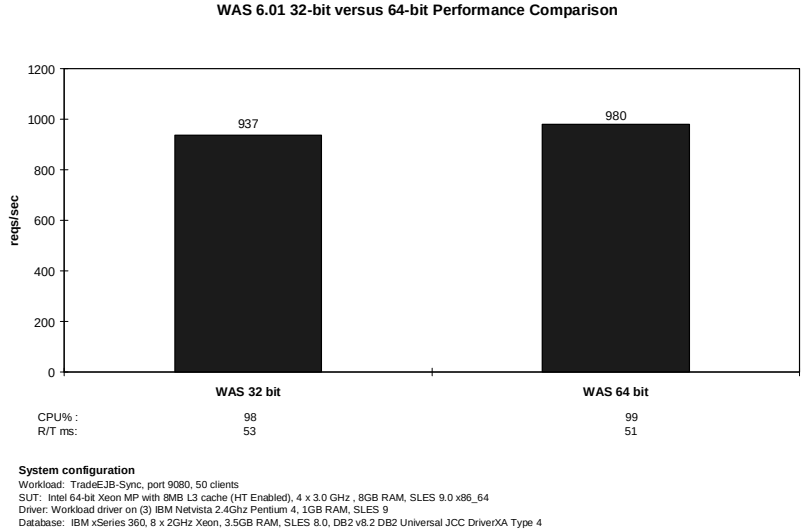


Figure 7: Trade6 64-bit performance on x86-64

Applications that do not benefit from 64-bit features can run with full native performance on the 32-bit version of WebSphere for the POWER and x86-64 platforms.

Figures 5, 6 and 7 also show the 32-bit version of WAS runs applications at full native hardware performance on the POWER and x86-64 platforms. Unlike some 64-bit processor architectures, the POWER and x86-64 hardware does not emulate 32-bit mode. Therefore applications that do not benefit from 64-bit features can run with full performance on the 32-bit version of WebSphere running on the abovementioned 64-bit platforms.

In fact, both 32 and 64-bit WAS installs can co-exist and applications run in a mixed mode. Migration is also simplified. Applications can initially be deployed on WAS 32-bit for x86-64 and gradually be migrated to the 64-bit version.

WAS 64-bit scalability performance

Figure 8, 9 and **10** detail scalability for the POWER and x86-64 versions of WAS. These charts show that WAS scales effectively (near linear) on both hardware platforms. At 4-way, the AIX on POWER and Linux on POWER platform shows a slightly better scaling factor (3.76/4.0 and 3.73/4.0 respectively) as compared the x86-64 platform (3.63/4.0). Further results were gathered for AIX on POWER platform showing an impressive scaling factor of 6.85 at 8-way.

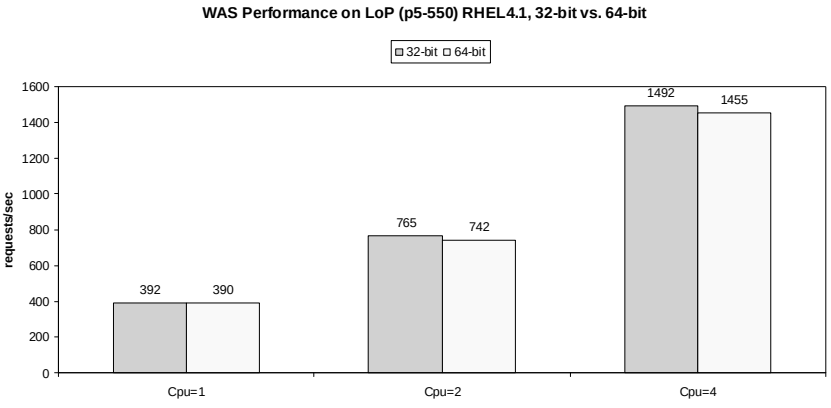
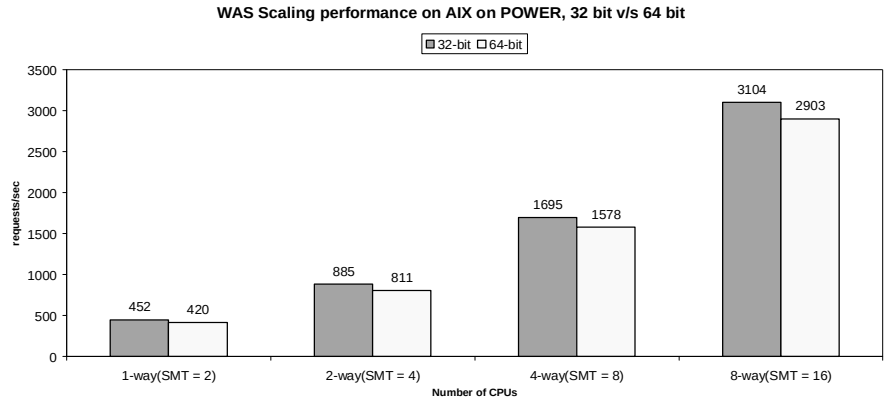


Figure 8: WAS Vertical Scalability on Linux on POWER

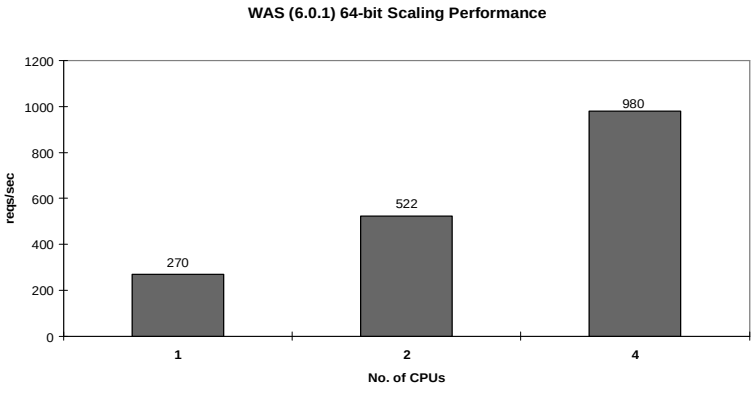
Both 32 and 64-bit performance results are shown for POWER in figures 8 and 9. From this, you can see that 64-bit performance and scalability is relatively comparable to 32-bit on the POWER platforms.



Scaling	1	1	1.93	1.93	3.70	3.76	6.87	6.85
WAS CPU (%)	100	100	99	99	99	99	97	97
R/T (ms)	116	126	59	64	31	33	17	18
DB2 CPU (%)	4	4	8	8	16	16	30	29

System Configuration
Workload : Trade6 EJB-Sync, port 9080, 50 clients
SUT : IBM p5-570, 16 x 1.9 GHz, 32GB RAM, AIX 5.3 SP2
Driver: Workload driver on IBM xSeries 335, 2 x 3.0 GHz Intel Xeon (HT Enabled), 4GB RAM, SLES 9
Database: IBM pSeries 690, 16 x 1.3 GHz, 32GB RAM, AIX 5.2 SP4, DB2 v8.2 DB2 Universal JCC DriverXA Type 4

Figure 9: WAS Vertical Scalability on AIX on POWER



Scaling factor:	1	1.93	3.63
CPU%:	100	99	99
R/T ms:	185	95	51

System configuration
Workload: TradeEJB-Sync, port 9080, 50 clients
SUT: Intel 64-bit Xeon MP with 8MB L3 cache (HT Enabled), 4 x 3.0 GHz , 8GB RAM, SLES 9.0 x86_64
Driver: Workload driver on (3) IBM Netvista 2.4Ghz Pentium 4, 1GB RAM, SLES 9
Database: IBM xSeries 360, 8 x 2GHz Xeon, 3.5GB RAM, SLES 8.0, DB2 v8.2 DB2 Universal JCC DriverXA Type 4

Figure 10: WAS Vertical scalability on x86-64

WAS Cross platform performance comparison

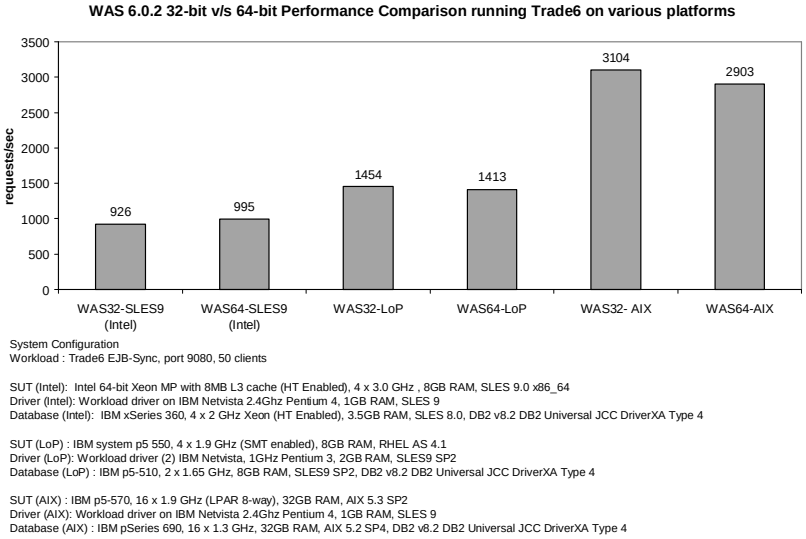


Figure 11: POWER vs. x86-64 performance

Figure 11 provides a performance comparison between WAS on x86-64 and POWER hardware. Accurate cross platform performance comparisons are not possible due to significant differences in process architectures, clock speeds and other factors. For the systems tested however, the results imply a significant performance advantage for WAS on POWER platforms.

Conclusions/Summary

WAS introduces support for 64-bit platforms in incremental updates to version 6.0. Version 6.0.2 supports AIX/Linux (64-bit) on POWER and Linux/Windows (64-bit) on x86-64. WAS 64-bit versions leverage 64-bit platform features providing the capability to configure and run with Java heaps much larger then the ~2-3GB limitations of 32-bit platforms. The WebSphere JIT code generation also automatically generates native code leveraging 64-bit CPU performance extensions.

Applications that leverage these features can see significant performance gains when deployed on a WAS 64-bit install. However, applications can expect to see an increase memory footprint requirement due to 8-byte

pointers used in 64-bit versus 4-byte in 32-bit. This can also result in reduced hardware cache efficiency and performance.

WAS is available in both 32 and 64-bit editions for 64-bit platforms. This provides significant application deployment flexibility. For applications that cannot leverage 64-bit features, the 32-bit version provides full native performance. Otherwise, for migration, applications can initially be deployed on WAS 32-bit and gradually be migrated to the 64-bit version. Also, both WAS versions can be installed simultaneously and applications run in a mixed mode.

References

A good 64-bit technology overview and reference
http://en.wikipedia.org/wiki/64-bit

New to Power Architecture technology
http://www-128.ibm.com/developerworks/power/newto/#17

InfoWorld – Inside POWER5
http://www.power.org/news/infoworldpower5.pdf

Linux on POWER
http://www-03.ibm.com/servers/eserver/linux/power/powerarticle.pdf

Power.org http://www.power.org

The AMD64 Computing Platform: Your Link to the Future of Computing
http://www.amd.com/us-en/assets/content_type/white_papers_and_tech_docs/30172C.pdf

AMD Opteron
http://www.amd.com/us-en/assets/content_type/white_papers_and_tech_docs/23932.pdf

The 64-bit Tipping Point
http://www.intel.com/business/bss/products/server/64-bit_tipping_point.pdf