IBM AIX Dynamic System Optimizer – A New Approach to Autonomous System Performance

January 14, 2013

Thanh Do, Bret Olszewski, Bruce Mealey

1.0 Introduction

Advances in hardware system design, virtualization technologies, and increased workload have created challenges and opportunities for system optimization. These opportunities are frequently not fully exploited by organizations more focused on managing basic service delivery than on achieving highly tuned system environments. To some users, the risks of static tuning on highly variable workloads outweigh the potential benefits. Against this backdrop, there are obvious benefits to organizations in improving service delivery through optimization. IBM® AIX® Dynamic System Optimizer (DSO) is a technology that addresses many of the existing AIX and POWER® performance optimization opportunities through dynamic monitoring and system adjustment.

DSO monitors and optimizes workloads running in an AIX system or partition. It is tightly coupled with the AIX operating system, giving it a degree of control over the operating system's optimization algorithms. The process of optimization is continuous in operation and autonomically adjusts system tuning based on instantaneous analysis of system dynamics. Autonomic tuning with ASO/DSO employs a set of algorithms which provide the basis for the feedback directed self-tuning.

2.0 Concepts

DSO observes characteristics of the workload running under AIX and dynamically applies a range of optimizations. The application of each particular optimization is tested to ensure that benefit was realized. Optimizations with negative impacts are quickly disengaged with the results stored for future consideration. As workloads change over time DSO reacts to those changes allowing continuous optimization for an environment. All of the major optimizations are applied with the goal of improving the workload's execution on the system hardware.

A DSO daemon implements monitoring, evaluation of potential optimization strategies, triggering of optimization requests to the AIX operating system, and the evaluation of optimization. DSO optimizes the performance of single partitions on a partitioned system.

DSO is built on the Active System Optimizer (ASO) framework introduced in AIX 7.1 TL1 SP1. The ASO framework includes a user-space daemon, namely "aso", advanced instrumentation methods based on kernel data and the hardware Performance Monitoring Unit (PMU). ASO provides the technology for cache and memory affinity optimization. DSO extends ASO by adding optimizations to use large pages and data stream pre-fetch. Each will be explored in detail in the next section. The ASO/DSO infrastructure is optimized for self tuning, however it will not override manual tuning. The ASO/DSO product architecture allows for expansion of optimization over time.

3.0 DSO Optimizations

Modern hardware systems have a wide range of capabilities and features. DSO applies optimizations which are supported by AIX, but are frequently unexploited in Information Technology (IT) organizations.

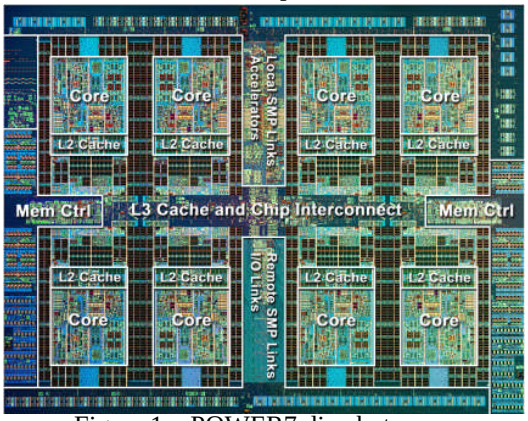


Figure 1 – POWER7 die photo

3.1 Optimization of Cache Affinity

Multi-processor systems based on POWER7® or POWER7+™ use a modular architecture which associates memory with processor chips. Each processor chip contains memory controllers which typically manage a set of

dual inline memory modules (DIMM's). Multi-processor systems include inter-processor interconnects which facilitate data movement between processors. Figure 1 is a die photo of the POWER7 processor chip. The chip contains two memory controllers and remote SMP links for inter-processor connectivity. The POWER7+ chip is similar, though with higher clock rates, more on-chip cache and additional accelerators for encryption and compression.

The cache affinity optimization identifies the threads of a multi-threaded process and considers them for fit on a single resource domain which may map to a chip. As an example, consider a Java process running with eight concurrent threads of execution. Figure 2 describes how at any instant in time, the threads may be spread over two chips in a situation that the total processor load required by the process could be handled by a single chip. Figure 3 shows that the related threads have their execution compacted onto a single chip. The compaction onto a single chip improves processor efficiency by reducing the amount of communications between chips. When the threads are co-resident on a chip, the data shared between the threads flows over the on chip L3 interconnect. This reduces processor stalls associated with accessing caches lines across the inter-chip interconnect.

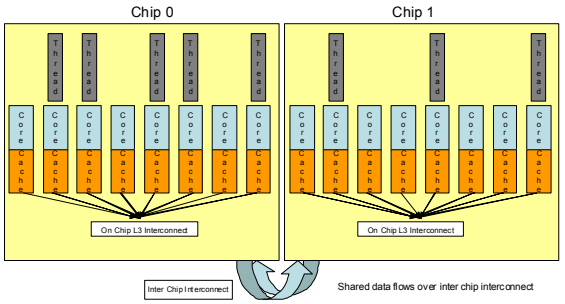


Figure 2 – Before Cache Affinity

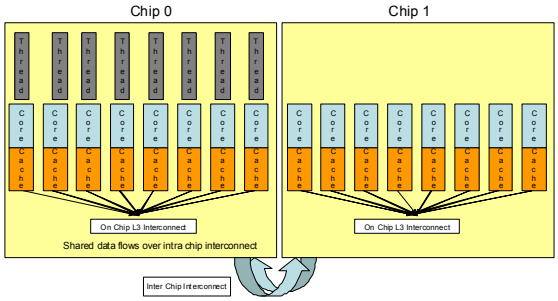


Figure 3 – After Cache Affinity

The cache affinity optimization supports only multi-threaded processes. This support only applies to multi-core workloads. Cache affinity can also be applied to partitions running in single thread (ST) mode. Thresholds are used such that the optimization occurs only when the workload is currently consuming certain minimum resources. Workloads must be relatively long-lived (thirty minutes or more) in duration to observe cache affinity. Cache affinity could be obtained with manual tuning by binding workloads to resource sets. However, such tuning is less flexible than DSO when workload resource usage is variable. Additionally, it can be difficult to fully exploit resource sets in an environment that exploits adding and removing processors via dynamic reconfiguration.

3.2 Optimization of memory affinity

A partition may have memory which is distributed over two or more physical processor sockets. Such resource distributions may be a result of partitions requiring more than one microprocessor chip of capacity, more memory than can be sourced from a chip, or as a result of previously activated partitions allocation of physical resource. While the distribution of memory is not an issue if there is not heavy sharing of objects in memory, there are many cases in which sharing will impact performance. Consider the case of a threaded application which uses a large amount of shared memory to cache frequently used data. In this case, the shared memory for the application will likely be spread across physical processors. Since data may be accessed with near-random locality, those accesses will be delayed by inter-chip latencies. DSO can analyze optimizations done in cache locality to identify frequently accessed memory that can be moved to be local to the executing threads.

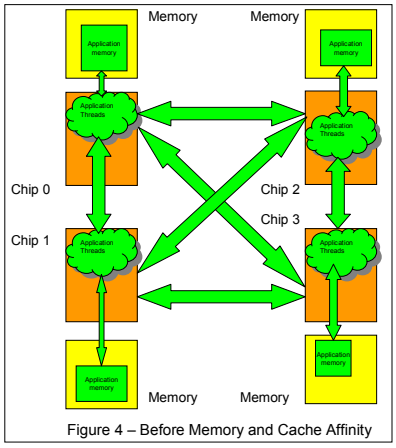


Figure 4 – Before Memory and Cache Affinity

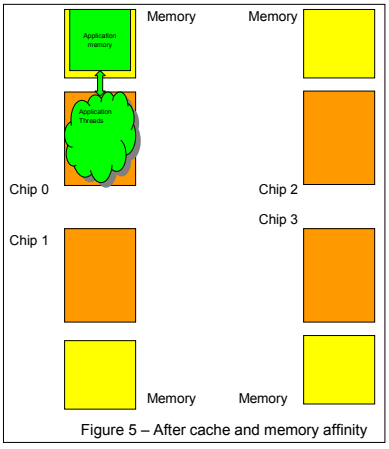


Figure 5 – After cache and memory affinity

3.3 Optimization using large pages

Virtual memory requires a means of translating virtual addresses to physical addresses. In order to make translation a low latency operation, as it is critical to processor efficiency, computer systems typically provide high speed caches of translations. Figure 6 details the translation mechanism used in AIX. Effective addresses are translated to virtual addresses through a segment-level translation, which uses the segment look-a-side (SLB) buffer cache. A second level of translation converts the virtual addresses to physical addresses. The two caches of translations that are used for this level of translation are referred to as the Effective-to-Real Address Translation table (ERAT) a small table and the larger Translation Look-a-side Buffer (TLB). Translations that cannot be mapped through the ERAT go to the TLB. As programs execute, each instruction fetch or data access passes through the translation mechanisms. Because the TLB may need to cache potentially terabytes of memory and because its physical size is limited by physical factors, the TLB contains translations for only a portion of memory for most partitions. On POWER7/POWER7+ cores, each TLB entry may represent a small size page (4096 bytes), a medium size page (65536 bytes), or a large page (16,777,216 bytes). When an address space range that an application uses is fairly contiguous in nature, the AIX operating system automatically creates medium sized pages for the program. Normally, the use of large pages requires explicit operational intervention managing a large page pool. Table 1 shows the potential coverage of the 512 entries of the POWER7 core TLB for supported page sizes.

Page Size	TLB Coverage
4096 Bytes	2 M Bytes
65536 Bytes	32 M Bytes
16,777,216 Bytes	8192 M Bytes

Table 1 – POWER7 TLB Coverage

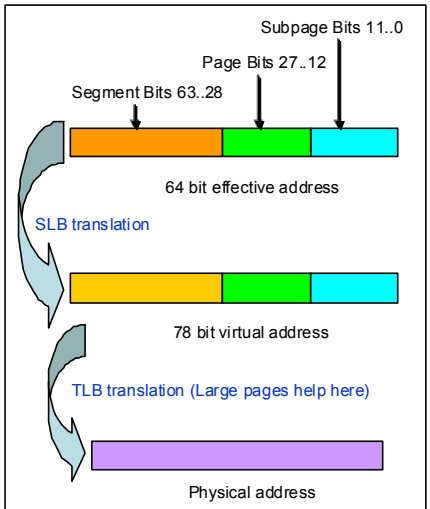


Figure 6 – POWER translation

As an address goes through translation, if it is not currently in the TLB additional steps are required to complete the translation. These steps are referred to as a TLB miss. The miss operation involves the hardware accessing memory to resolve the miss and replace an existing entry in the TLB with the translation information for the current address. The rate of TLB misses for an application will be affected by the locality of access of pages.

As an example, a relational database will have a mix of per thread storage and global storage. Thread storage will tend to have good locality, particularly for things like stacks which are used over and over. Global storage will likely have lower locality, as it contains large scale items such as database pages and indexes. The cost of a TLB miss is directly accumulated in processor time for the application. The cost of each miss will vary, depending on the resolution point for the translation information, which could be in a cache, local memory, or remote memory. Misses in the TLB are all but invisible to running applications, but a high rate of misses will slow down program execution.

AIX provides support for large page function. However, the usage requires advanced planning and management. It usually requires coordination between the system management and application management teams to properly implement. Historical usage of large pages with AIX has established some best practices around static application environments. More complex and dynamic environments involve additional management and reduce overall resource flexibility.

DSO avoids the historical static and inflexible limitations by automatically identifying ranges of address space which can be converted into large pages and performing the conversion. For applications with large address spaces, the large page optimization improves CPU efficiency by reducing translation misses. The large page optimization is supported for both multi-threaded processors or groups of single threaded processes attached to the same shared memory region. Only System V shared memory regions are supported. The large page optimization is ideal for long running large database environments. Workload must have at least two cores of CPU utilization and 16GB of shared memory to make use of the large page optimization.

3.4 Optimization of hardware pre-fetch

Hardware pre-fetch is a capability in POWER7 and POWER7+ processors that detects sequential data accesses and speculatively initiates memory accesses in anticipation of potential future use. The mechanism detects both forward and backward access patterns with various stride ranges. While hardware pre-fetch is generally beneficial, there are two specific situations where it can actually degrade application performance. The first situation is when applications pre-fetch cache lines that are never used. This may be seen in object-oriented programming applications which have many small randomly accessed but widely dispersed patterns of data access. The second situation is when memory access is not local. This may be the result of a partition being spread across multiple chips or having generally low memory locality.

DSO identifies cases where tuning hardware pre-fetch can be beneficial and adjusts the behavior dynamically. Since memory pre-fetch optimization is very fast and requires low overhead, DSO rechecks the optimization every 15 minutes, based on a fixed tuning cycle. If more than 50% of the system workload has been tuned for hardware pre-fetch, DSO will apply the optimization system-wide.

As a general rule, hardware pre-fetch tuning has more benefit on POWER7 systems than POWER7+ systems. The POWER7+ processor has significantly larger on chip cache - normally resulting in a smaller number of cache misses. Workload must have at least eight cores of CPU utilization and 16GB of shared memory to utilize the hardware pre-fetch optimization.

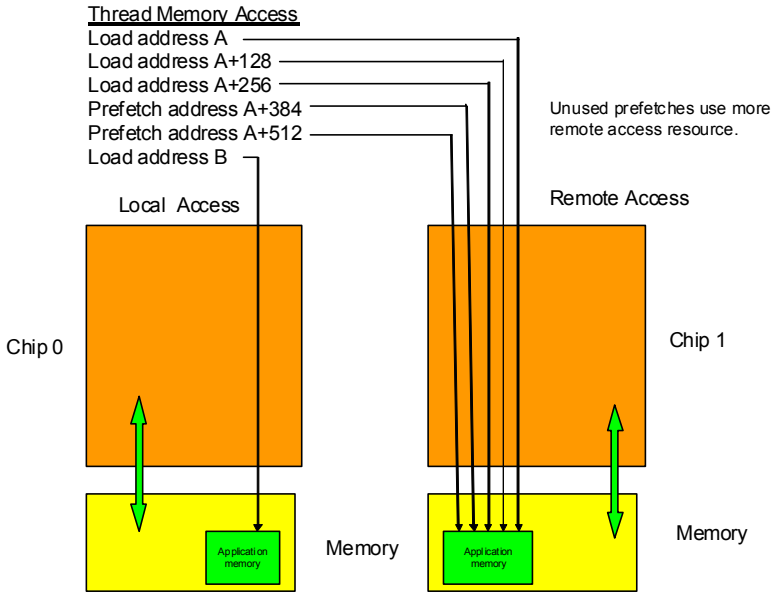


Figure 7 – Hardware Prefetch

4.0 An illustrative example using DB2®

The specific optimizations of DSO can be observed using a relational database performance workload leveraging DB2 v9.7 Fix Pack (FP) 5. This environment was done on a shared processor partition with 16 virtual processors and 16 entitled processors. Before DSO is enabled, the system operates consuming approximately 12.7 cores of processor resource as described in Figure 8. The first step is to enable cache affinity. To explain the observed behavior, we need to review AIX's POWER7 SMT4 usage. As the default, AIX optimizes processor usage based on providing the highest per software thread performance. To meet this goal, individual software threads tend to be allocated first to the primary threads of a core, essentially fanning out the software threads onto the primary threads of all of the cores in the system. When all available primary threads are busy and there are additional software threads to run, AIX begins to use the secondary threads of the cores. This is done irrespective of the Scheduler Resource Allocation Domain (SRAD) topology of the partition – as long as processor and memory resources are available on the SRADs.

The SRAD configuration for our test environment was obtained from the *lssrad* command with the *–av* options and is detailed in Table 2. This table shows a fraction of the virtual processors in the partition, but the pattern is consistent. This partition has two SRADs, with the virtual processor to SRAD mapping alternating cores to SRADs. The virtual processor to resources mapping is done in the hypervisor.

The AIX *mpstat* tool provides a good means of detailing the partition behavior before and after the cache affinity optimization. Table 3 shows before cache affinity begins, the primary threads, threads CPU 0 and 24 (as well as the other primary threads) are relatively busy. The other threads on the cores are relatively unused. This is best represented as the PC or processor consumed in the *mpstat* output. For example, on the first core, primary thread CPU 0 shows a processor consumed of 0.57, while the secondary thread CPU 1 shows a processor consumed of 0.11. The secondary, tertiary and quaternary threads show minimal activity, consistent with being essentially idle. The point is further reinforced by the observation that the tertiary and quaternary threads are performing no context switches whatsoever in the example.

After cache affinity has completed, the threads of the target workloads have been compacted onto SRAD0. The table shows CPUs 0 and 1 on core 0 now have both the primary and secondary thread active. On the core represented by CPUs 24, 25, 26, and 27 which are on SRAD1, the threads are now idle. The active software threads in the workload have migrated to cores in SRAD 0, reducing the load on SRAD 1 considerably. The compaction of workload onto SRADs has a number of performance benefits. First, the data that is shared between threads in the partition will have better locality. That is, objects like variables and locks shared by the threads will move with less latency as cache-to-cache operations. Second, in a shared processor partition, it is possible to disuse virtual processors and let them return to the shared processor pool. This benefits overall system throughput by reducing cycles consumed as idle time in partitions. DSO will weigh these benefits against the potential response time effects of compaction during its optimization methodology. DSO will only do compression when there is no predicted loss in response time.

REF1	SRAD	MEM	CPU
0	0	120458.94	0-15 20-23 28-31 36-39 44-47 52-55 60-63
	1	78186.00	16-19 24-27 32-35 40-43 48-51 56-59

Table 2 – Partition SRAD configuration

	Before DSO		After DSO	
CPU	cs	PC	cs	PC
0	2487	0.57	740	0.41
1	0	0.11	441	0.33
2	0	0.11	0	0.07
3	0	0.11	0	0.08
24	1461	0.39	123	0.01
25	4	0.08	0	0.00
26	0	0.08	0	0.00
27	0	0.08	0	0.00

Table 3 – Partial mpstat comparison

Figure 8 shows the effect of cache affinity on the physical processor consumption of the partition under load. The processor consumption drops from 12.7 cores to 9.7 cores during the optimization with no loss in throughput.

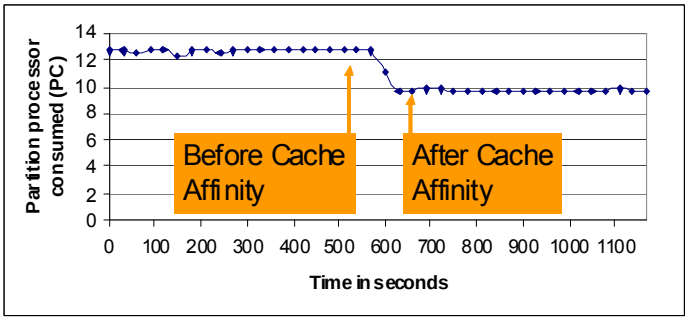


Figure 8 – Partition Processor Consumed

A more complete view of the system-level changes observed with cache affinity can be found in Figure 9. This figure details the *mpstat*-level per core statistics before and after cache affinity. An *mpstat* processor is a hardware thread. The figure shows before cache affinity the load is relatively evenly divided amongst processors on SRAD0 and SRAD1. The highest used processors are the primary threads for each core. The middle-range used processors tend to be secondary threads. The lowest used processors are tertiary and quaternary threads. After cache affinity, most (but not all) of the processors in SRAD1 become idle whereas the processors in SRAD0 all become busy. As cores in SRAD1 become unused AIX disuses them. This allows added capacity in the shared pool.

After cache affinity is complete, memory affinity can take place. If cache affinity cannot identify optimization, memory affinity will be skipped. The infrastructure attempts to identify memory shared between threads and move it to the SRAD local to the software threads. If there is not enough physical memory on an SRAD to move the entire workload, portions of the workload's memory are moved. Only workloads that can be contained in a single SRAD are optimized. The memory affinity optimization can take a considerable period of time, as the movement of memory is paced to avoid overhead impacts to other workloads running in the partition. Observation of memory usage and placement is done with the "*svmon –P <pid> –O affinity=detail,segment=on*" command. Since the db2sysc process does most of the processing in our chosen workload, we use the *svmon* command on this process. Below is a subset of the *svmon* output before memory affinity. It shows a distribution of memory for the process with 60.7% in domain affinity 0 (SRAD0) and 39.3% in domain affinity 1 (SRAD1).

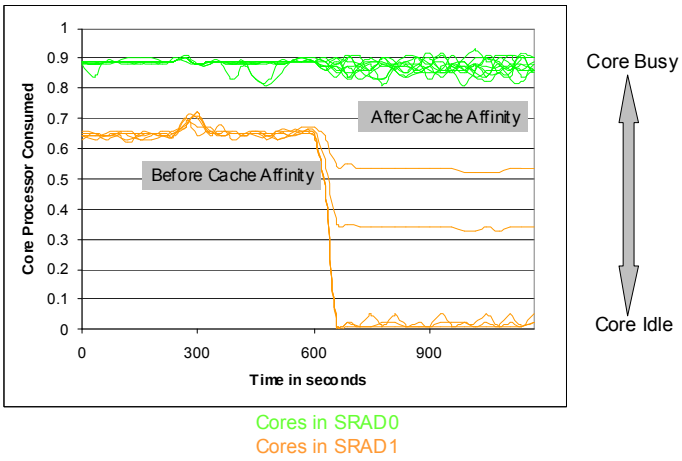


Figure 9 – Core Processor Consumed

```
5177706 db2sysc      22922126   10072       0 22922071

  Domain affinity      Npages  Percent    Private

                0    13913830     60.7      11968

                1     9018212     39.3       8693
```

The total memory of the DB2 process including kernel is 13,913,830+9,018,212, or about 22.9 million pages (Npages). However, the private memory (not in shared memory) is 11968+8693=20,661 (Private) pages. In this case, DSO chose the large page optimization and did not apply memory affinity to the shared memory. DSO will only affinitize up to 70% of the private memory.

One hour later, after memory affinity has occurred, the distribution of memory is changed only slightly, with approximately 3200 pages moving between domains. The small amount of memory moved is characteristic of a workload that has most of its memory allocated as shared memory. Once shared memory has been upgraded to 16M byte pages, it will no longer be considered for affinity. Only the private pages for the individual threads benefited from DSO memory affinity. The increase in total memory is due to additional memory required as the workload progressed.

```
5177706 db2sysc      22939475    10040        0 22939420

          Domain affinity      Npages  Percent     Private

                        0    13927408     60.7       15613

                        1     9021916     39.3        5437
```

The effects of the large page optimization are also observed with the *svmon* command. Note that *svmon* without options will not correctly identify the large pages promoted through DSO. This is due to DSO using a new mechanism for large pages which allows them to be created and re-split into small pages as required. The *svmon –P <pid> -Ompss=on* option, which details the memory segments mapped to processes, produces a sufficient level of detail to identify the changes. Below is a subset of the *svmon* output for a db2sysc process before the large page optimization begins. A number of the DB2 shared memory segments are in the range of a0003000 and higher. In this example, the first two segments displayed, Esid's a00030a2 and a000314c are fully populated, each with 4096 pages of 64k (PSize is m for medium).

```
      Pid Command          Inuse        Pin     Pgsp     Virtual
  5046328 db2sysc       22951784      10638        0   22950692
     Vsid      Esid Type Description              PSize  Inuse   Pin  Pgsp Virtual
   6212e2   a00030a2 work N/A                       m   4096     0     0    4096
  10c138c   a000314c work N/A                       m   4096     0     0    4096
```

Approximately 1 ½ hours later, we run the *svmon* command again and see a change in the segment-level output. Now segment a000314c is a mixed segment, with zero 64KB pages and 16 16MB pages. The change in pinned pages is not related to memory optimization.

```
      Pid Command          Inuse        Pin     Pgsp     Virtual
  5046328 db2sysc       22951901      10142        0   22951846
     Vsid      Esid Type Description              PSize  Inuse   Pin  Pgsp Virtual
   6212e2   a00030a2 work N/A                       m   4096     0     0    4096
  10c138c   a000314c work N/A                       m      0     0     0    4096
                                                    L     16     0     0       0
```

Migration of pages to large pages takes time. Figure 10 shows summarization of *svmon* data for the DB2 processes over 30 minute periods of workload execution. The DB2 processes consume a relatively small amount of small pages, at least when the total system memory is considered.

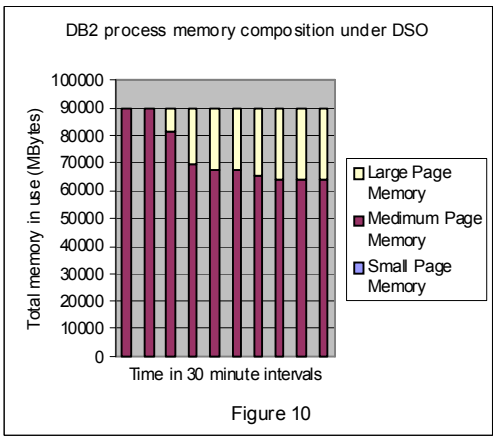


Figure 10

The memory pre-fetch optimization is much harder to observe with standard tooling. The end user would merely notice a reduction in CPU consumption. Additional diagnosis is difficult, as ASO/DSO requires full use of the hardware performance monitor, commands like *hpmstat* and *hpmcount* perturb the behavior of ASO/DSO. Additionally, the use of *tprof* with the *–E* option will also disturb ASO/DSO. Usage of these tools should be minimized as they can reduce the

effectiveness of DSO optimization. Measurements on this workload indicated that DSO pre-fetch optimization reduced the fraction of memory bandwidth used by pre-fetching from approximately 43% to 3%, with only a small corresponding increase in demand misses.

5.0 ASO/DSO Laboratory Benchmark Results

In this section we introduce a variety of workloads and describe performance comparisons with and without ASO/DSO. The workloads chosen for evaluation were based on a variety of attributes. One characteristic that was specifically considered was the applicability to ASO/DSO optimization. For workloads that couldn't benefit, our goal was to ensure that ASO/DSO did not actually have any negative implications to performance. Other workloads were selected based on representative commercial applications and environments that are common in industry today.

5.1 SDET Workload

Experiments were run using the long deprecated SPEC system development multi-tasking (SDM) 05.sdet workload. This workload mimics interactive users running scripts of common commands. The workload was measured on two configurations on a Power 740 system using POWER7 cores. One measurement used eight cores and 30GB memory, and the other used 15-cores and 90GB memory. The workload has relatively high kernel usage due to heavy stat, fork, exit, and exec system call usage. Both ran at 100% CPU utilization. There was no performance difference comparing ASO/DSO enabled vs. disabled. None of the DSO/ASO optimizations are useful for this workload, but we observed no measurable workload degradation when DSO was active.

5.2 JAVA Workload

A modified SPECjbb2005™ benchmark was used to study the behavior of Java performance under ASO/DSO. This benchmark evaluates the performance of server side Java, exercising the implementation of the JVM (Java Virtual Machine) JIT (Just-In-Time) compiler, garbage collection, threads and some main aspects of the operating system such as the performance of CPUs, caches, memory hierarchy and the scalability of shared memory processors (SMPs).

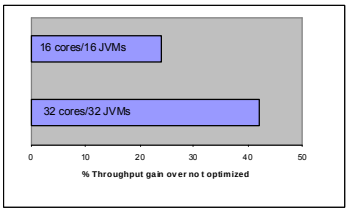


Figure 11 – Modified SPECjbb2005 Results

The benchmark was modified to lengthen execution time to meet the long-running criteria required by ASO/DSO. The benchmark was run on a Power 750 in varying configurations. The smaller configuration used 16 cores of 128GB of memory. The larger configuration used 32 cores and 256GB of memory. Both configurations are measured with Java 6 Service Refresh (SR) 7 at or very near 100% CPU utilization.

There is a throughput improvement of 24% and 42% in the two hardware configurations used. The improvement is mainly seen with cache and memory affinity optimization under both AIX V6.1 TL8 SP1 and AIX V7.1 TL2 SP1. For this particular JAVA workload, memory affinity has played a major role in improving JVM memory traffic - resulting good throughput improvement. The workload did not benefit from large page and memory pre-fetch optimization as the required CPU and memory threshold criteria were not met. In these configurations, each JVM utilized roughly one core of CPU utilization and 8GB of memory.

The performance gain is identical whether using dedicated or shared processor partitions on an otherwise idle server.

5.3 The WebSphere® DayTrader Workload

The DayTrader workload simulates an online stock trading system. The benchmark was configured in three-tier mode with two POWER6® database server machines running DB2 V9.7 FP5, and the System Under Test (SUT) is a Power 730 both with 8-cores and

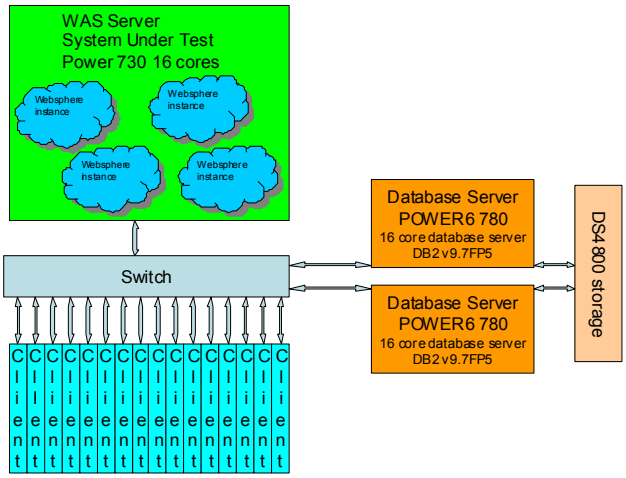


Figure 12 DayTrader workload configuration

16-cores with 4 WAS instances. Figure 12 shows the system diagram for the 16 core test. WebSphere Application Server (WAS) Version 7.0 is configured on the SUT. Each WAS instance used a 13GB heap size. The client workload was generated from running 16 client systems operating concurrently. The environment is representative of a horizontal scale-out WebSphere environment, with multiple Java virtual machines participating in execution of a workload.

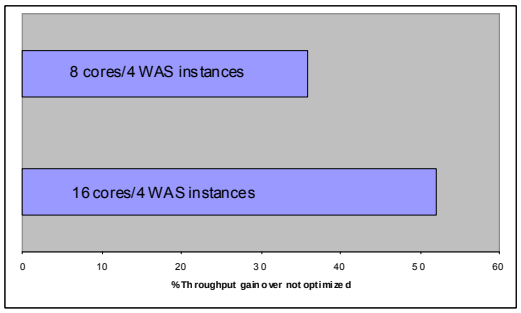


Figure 13 – Websphere Day Trader Results

Figure 13 shows throughput improvements of 36% and 52% were measured, primarily from cache and memory affinity in the two hardware configurations used. The memory was 64GB and the CPU utilization is roughly about 80% in both tests on the system under test. The SUT has been measured with both AIX V6.1 TL8 SP1 and AIX V7.1 TL2 SP1 and similar performance gain observed. The performance gain is identical whether using dedicated or shared processor partitions on an otherwise idle server. As seen in the Java workload case, there is no performance gain with large page or memory pre-fetch for this Day Trader workload as the criteria to trigger these two optimizations did not occur.

One other observation was made in another Day Trader environment also on a POWER7 based system with 16-cores but running at low CPU utilization. The system under test was about 17% CPU busy. Hand-tuning was applied to each WAS instance via *bindprocessor* and *execrset* commands to distribute the load evenly to a particular set of CPUs. There was roughly 5% throughput degradation due to hand-tuning. We then unset hand-tuned and enabled ASO/DSO optimization, we did not see any performance benefit or loss with low utilization. DSO was able to detect that cache placement could not produce a performance benefit and avoided the optimization.

5.4 COPR Workload on DB2

COmmercial Performance Rating (COPR) workload represents a modern OnLine Transaction Processing (OLTP) application system. It consists of 35 tables using 37 associated explicit indexes. COPR is intentionally designed to be user-representative, avoiding extreme optimization. For these tests DB2 V9.7 FP5 was used to evaluate ASO/DSO performance improvement for threaded database applications. DS/4800 storage was used for the database disk configuration. The benchmark is long-running, typically more than one hour in steady-state. In the DSO runs, the total run time has been increased to greater than eight hours in order to observe the full effect of large page promotion.

Figure 14 shows the performance gains on three configurations.

On a Power 795, a 26.57% improvement was measured for cache, memory affinity, large page and memory pre-fetch optimization over an unoptimized workload at approximately 85% utilization. The database buffer pool size is about 170GB. The throughput improvement was a direct consequence of lower transaction response time, as the workload has a fixed input load. The measurement was done under AIX 6.1 TL8 SP1. There was about 3.26% improvement due to cache and memory affinity, and about 23.31% improvement due to large page and memory pre-fetch.

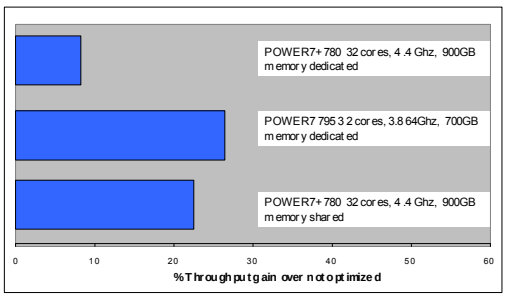


Figure 14 – COPR DB2 Performance Results

Figure 15 shows the comparative response time of the COPR transactions for the Power 795 test. Although we show just Power 795 example here, the improvements in response times for the tests were well correlated with the increase in throughput in all the tests. The COPR workload runs with 10 distinct transactions with a fixed mix. All of the transactions yielded improvements in response time. The improvements in response time ranged from 12% to 39% on the various transaction types on the Power 795.

The performance gains due to ASO/DSO vary somewhat depending on partition topology and hardware considerations. It was observed that the gain in cache and memory affinity is much more significant while running the SUT at a lower CPU load and/or having database working set spread on multi-books/cores vs. single book/core. The throughput gain over the POWER7+ based system was larger in part to bigger gains with the memory pre-fetch optimization on the POWER7 system.

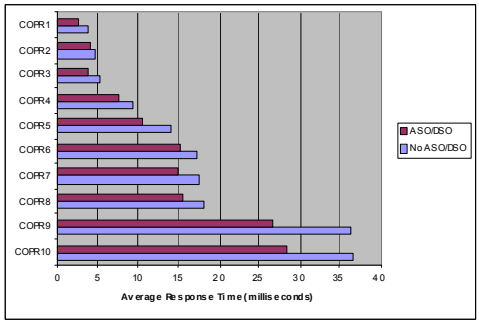


Figure 15 – Comparable COPR transaction response times on Power 795

On a POWER7+ Power 780, a throughput gain of 8.3% was measured with gains from large page and memory pre-fetch optimizations. That measurement was under AIX V7.1 TL2 SP1 or AIX V6.1 TL8 SP1 dedicated on a POWER7+ based system. The database buffer pool size was roughly 170GB in size. In this configuration, the unoptimized workload was running at 90% utilization, hence no benefit seen for cache and memory optimization.

On a POWER7+ Power 780 32-core system, a 22% improvement was measured. This workload was on AIX V7.1 SP2 TL1 or AIX V6.1 TL8 SP1 shared partition. The database buffer pool is same at roughly 170GB in size. The unoptimized workload was running at 90% CPU busy. There is about 13% throughput improvement due to cache and memory affinity. The partition spanned books (or hardware domains) in the shared partition case which results in more benefit attributed to cache and memory affinity optimization. The benefit due to large page and memory pre-fetch optimization is roughly 9%.

5.5 COPR Workload on a process-based database

This test also uses the COPR workload, in this case with a different commercially available database. This database uses a process-based model, distinguishing it from previous section. The workload run time extended to beyond eight hours in consideration of the time necessary for large page promotion.

The hardware configuration consisted of a 16 core POWER7+-based 780 running 3.7Ghz with 600GB of memory. Both AIX V6.1 TL8 SP1 and V7.1 TL2 SP1 were tested separately on the target configuration in both dedicated and simple shared partition mode. The database memory was roughly 90GB in size. CPU utilization was kept around 60% and 90% respectively for dedicated and shared partition runs. In both dedicated and shared partition mode, there was about 4.3% improvement seen, primarily due to large page and memory pre-fetch optimization. The hardware configuration and database size are smaller in this set up comparing to those of COPR DB2 resulting a smaller overall throughput improvement for large page and memory pre-fetch optimization. There is no performance gain for cache/memory affinity optimization as ASO only considers multi-threaded processes for such optimizations. The performance gain is identical whether using dedicated or shared processor partitions on an otherwise idle server.

5.6 Virtualization – With two COPR database workloads

Another evaluation was done on a Power 780 running at 3.86Ghz, POWER7 chips with 32 cores. A shared pool of 32-cores and two shared processor partitions was created, each with 120GB of memory, eight cores of processor entitlement, 16 virtual CPUs and uncapped. Both the AIX V7.1 TL2 SP1 and AIX V6.1 TL8 SP1 were tested. Each partition ran a separate copy of previously described process based database and separate copies of the COPR workload. On one partition, the partition's memory was split across two SRADs, forcing it to expand across chips. On the other partition, the memory fit onto a single SRAD. Figure 16 describes how such an environment can occur. When partitions are started, they are assigned memory and processors, ideally on the same socket. Generally, if sufficient processors and memory are available on a socket for a partition, the partition processors and memory will be collocated on a single socket. Assume for a moment that Partitions A, C, D and E are all created before Partition B. In that case, it is quite possible that partition B will have cores and memory spread across chips.

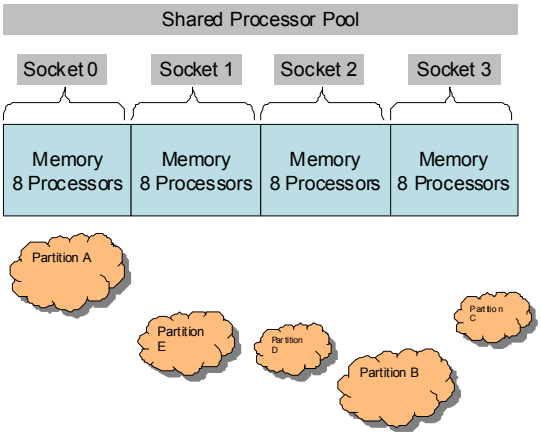


Figure 16 – COPR DB2 Performance Results

Each partition utilized about 80% of 15-cores physical processor consumed. The DB memory size was about 40GB for each partition. The total frame throughput improved by 6%, seen with both AIX levels, for large page and memory pre-fetch optimization. The database memory size was small for both partitions, thus the gain from large page optimization was modest and the bulk of the performance gain could be attributed to the memory pre-fetch optimization. The workloads were process-based so there was no improvement for cache and memory affinity.

A second evaluation was performed on a Power 780 running 4.42Ghz POWER7+ chips and 32 cores. A shared pool of 32-cores and two shared processor partitions were created, each with 120GB of memory, eight core processor entitlement, 16 virtual CPUs and uncapped. One partition ran a separate copy of the process-based database COPR workload using AIX V6.1 TL8 SP1, with a utilization of 80% of the 15-cores physical processor consumed. The other partition ran a separate copy of the DB2 V9.7 FP5 COPR workload, using AIX V7.1 TL2 SP1, with a utilization of 80% of 14-cores physical processor consumed. On the DB2 partition, the DB buffer pool size was about 52GB, and the workload's memory was split across two SRADs, forcing it to expand across chips. On the process-based database partition, the database memory size was about 40GB and fit on a single SRAD. The total frame throughput improved 8.1%, primarily due to large page and memory pre-fetch optimization. There was no performance benefit for cache and memory affinity optimization due to the high CPU load usage for the partitions.

5.7 The Tradelite Workload with multiple WebSphere and DB2 tiles

The Tradelite workload is quite similar to the Daytrader workload, but with simpler execution logic. HTML-based sequences of operations are presented from client systems to a WebSphere instance. Each WebSphere instance has an associated DB2 instance which maintains the persistent data associated with the workload. Because the logic in the WebSphere instance is simpler than DayTrader, a relatively heavier workload is presented to the DB2 instance. Figure 17 shows how two tiles of Tradelite run upon the same 32 core Power 750 server concurrently. The workload is driven to maximum capacity in both tiles by the clients.

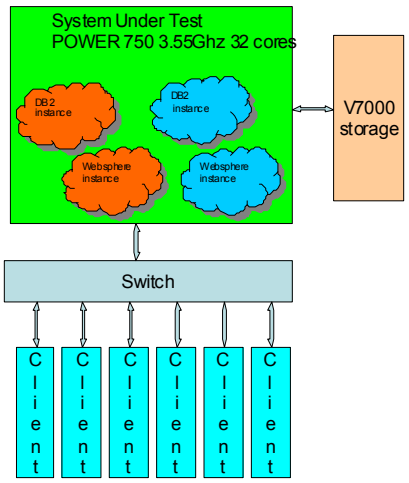


Figure 17 Tradelite tile-based workload configuration

An environment like this with multiple multi-threaded processes is ideally suited to ASO's cache and memory optimization. Each multi-threaded process can be optimized onto a distinct SRAD, resulting in good locality. The small size of the Websphere heap (0.5GB) and the relatively small database size (600MB) make the elements of the tile too small to get benefits from the additional optimizations in DSO. The high levels of communication between the clients and the server, as well as the loopback-based communications between WebSphere and DB2 result in approximately one-third of the CPU utilization in the operating system. This workload can be hand-tuned without ASO by binding the processes to RSETs. The hand-binding approach is effective in a benchmark environment where workload is steady and balanced produces very good performance. However, care must be taken in identifying how much capacity each workload can absorb. Such hand placement can be an inefficient use of staffing, particularly considering how well ASO can do automatic optimization. Additionally, in practice the world is rarely as orderly and predictable as benchmarks. The inflexibility imposed by RSET binding is a liability not experienced with ASO.

Three cases were run varying the number of concurrent tiles from one to three. The SUT was a Power 750 running at 3.55GHz with 256GB memory. The measurements were done under AIX V7.1 TL2 SP1. Figure 18 shows the aggregate throughput improvements realized with ASO over un-tuned for the cases. For the case of a single tile and the system is not highly busy, ASO did not provide any benefit. There are 10% and 63.6% throughput improvements for the 2 and 3 tile cases respectively.

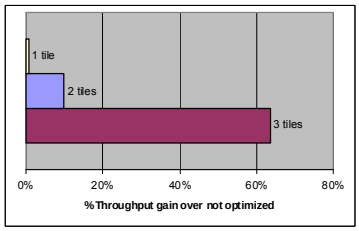


Figure 18 Tradelite tile-based workload ASO benefit

Figure 19 shows the CPU utilization measurements for the points of comparison. One important observation is that ASO changed the CPU utilization modestly in the environment, at least when compared to the increased throughput.

Figure 20 shows the average response time measurements from the clients. The response times are linearly related to the increased throughput for the workload.

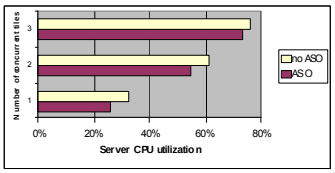


Figure 19 Tradelite tile-based workload CPU utilization

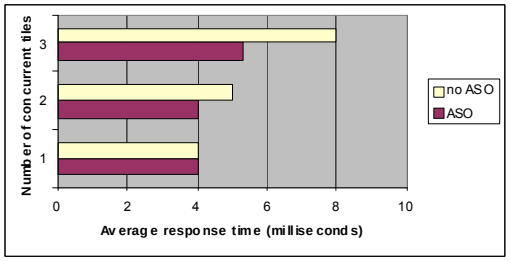


Figure 20 – Tradelite response time comparison

6.0 Conclusion

DSO has demonstrated significant value on a variety of key commercial workloads. Many threaded applications such as DB2 and WebSphere are well suited to the optimizations available. High value has been proven in compressing such workloads onto SRAD granularity resource domains when that is applicable. DSO also has high value in exploiting hardware features available on POWER7 and POWER7+ systems such as large pages and memory pre-fetch optimization. DSO optimizations work well for both dedicated processor partitions and shared processor partitions. Shared processor partition optimization is frequently high value, as additional capacity can be made available to the shared pool.

The optimization framework is capable of managing multiple optimizations across different workloads and is designed to be extensible to future enhancements and hardware platforms. The framework is scalable, workloads such as Tradelite show DSO is capable of optimizing multiple concurrent workloads simultaneously.

The true value of DSO is that it is flexible, easy to use, and does not require on-going management. The optimizations supported are applied automatically as proven to be beneficial. The framework automatically adapts to changing workload, relieving responsibility of workload specific optimizations from IT. This reduces partition administrative costs by eliminating significant hand tuning processes that are in use today. This enables IT organizations to extract maximum value from their system investments with minimal costs.

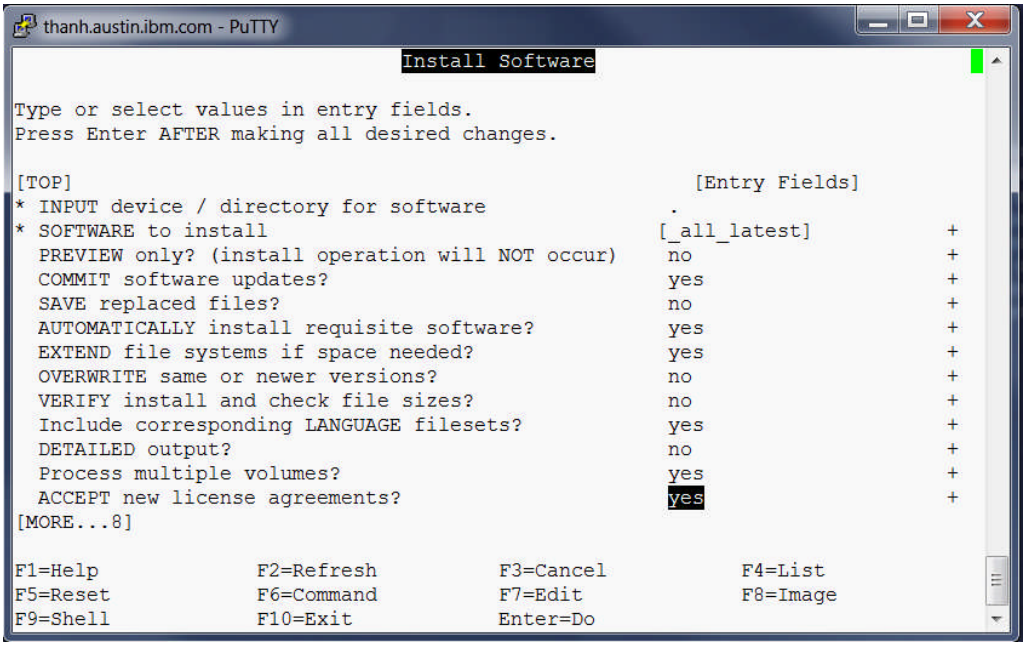
Appendix

Installation and Usage

ASO and DSO are only supported on POWER7 or POWER7+ systems. ASO and DSO are supported on AIX V6.1 TL8 SP1 and AIX V7.1 TL2 SP1. ASO is supported on AIX V7.1 TL1 SP1.

DSO is available as a separately price offering. It can be ordered individually or as part of the AIX Enterprise Edition 7.1 and AIX Enterprise Edition 6.1. Refer to IBM Announcement dated 11/6/2012 for ordering information. http://www-01.ibm.com/common/ssi/rep_ca/8/897/ENUS212-398/ENUS212-398.PDF

The DSO installation package contains DVDs that include product installation documentation and files. The procedure involves installation of the dso.aso fileset via either AIX smitty or "installp" command. When using smitty, adjust the "PREVIEW only?" and "COMMIT software updates" setting to "yes" or "no" as appropriate and make sure to set "ACCEPT new license agreements?" as "yes".




```
$ mount -rv cdrfs /dev/cd0 /mnt
$ cd /mnt
$ installp -ad /mnt/dso.aso all
```

Starting DSO

There are two aso executables that come as part of ASO/DSO offering:

- aso – the user-level daemon

```
$  ps -ef|grep aso
    root 5898444 4653296   0 11:12:43      -  0:00 /usr/sbin/aso
    root 8978496 5832798   0 11:18:26  pts/0  0:00 grep aso
```

- asoo – the AIX command tool to configure unrestricted and restricted tunables

```
$ asoo -a
aso_active = 0
debug_level = -1
```

Once DSO is installed, it is activated with the asoo command.

```
asoo –po aso_active=1
```

The system default is ASO/DSO not active (aso_active=0).

For additional information on usage, ASO/DSO commands, variable environments, and log files, consult the POWER7 and POWER7+ Optimization and Tuning Guide (**http://www.redbooks.ibm.com/redpieces/abstracts/sg248079.html?Open&pdfbookmark**).

Acknowledgements

Rick Peterson, Yong Cai, Hong Hua, Qunying Gao, Basu Vaidyanathan, Paul Mazzurana, Kapileswari Chandragiri, Aravinda Herle, Rajesh Kumar, Alan Jiang, Nikhil Hegde

The Power Architecture and Power.org wordmarks and the Power and Power.org logos and related marks are trademarks and service marks licensed by Power.org.

Java and all Java-based trademarks and logos are trademarks of Sun Microsystems, Inc. In the United States and/or other countries.

All performance information was determined in a controlled environment. Actual results may vary. Performance information is provided “AS IS” and no warranties or guarantees are expressed or implied by IBM. Buyers should consult other sources of information, including system benchmarks, to evaluate the performance of a system they are considering buying.

SPECjbb2005, is a trademark of the Standard Performance Evaluation Corporation (SPEC).

Photographs show engineering and design models. Changes may be incorporated in production models.

Copying or downloading the images contained in this document is expressly prohibited without the written consent of IBM

© IBM Corporation 2013
IBM Corporation
Systems and Technology Group
Route 100
Somers, New York 10589

Produced in the United States of America
January 2013
All Rights Reserved

This document was developed for products and/or services offered in the United States. IBM may not offer the products, features, or services discussed in this document in other countries.

The information may be subject to change without notice. Consult your local IBM business contact for information on the products, features and services available in your area.

All statements regarding IBM future directions and intent are subject to change or withdrawal without notice and represent goals and objectives only.

IBM, ibm.com,AIX, Power Systems, POWER6, POWER7, POWER7+, DB2, and Websphere, are trademarks or registered trademarks of International Business Machines Corporation in the United States, other countries, or both. If these and other IBM trademarked terms are marked on their first occurrence in this information with a trademark symbol (® or ™), these symbols indicate U.S. registered or common law trademarks owned by IBM at the time this information was published. Such trademarks may also be registered or common law trademarks in other countries. A current list of IBM trademarks is available on the Web at "Copyright and trademark information" at www.ibm.com/legal/copytrade.shtml

Other company, product, and service names may be trademarks or service marks of others.

IBM hardware products are manufactured from new parts, or new and used parts. In some cases, the hardware product may not be new and may have been previously installed. Regardless, our warranty terms apply.

This equipment is subject to FCC rules. It will comply with the appropriate FCC rules before final delivery to the buyer.

Information concerning non-IBM products was obtained from the suppliers of these products or other public sources. Questions on the capabilities of the non-IBM products should be addressed with those suppliers.

When referring to storage capacity, 1 TB equals total GB divided by 1000; accessible capacity may be less.

The IBM home page on the Internet can be found at: http://www.ibm.com.

The IBM Power Systems home page on the Internet can be found at: http://www.ibm.com/systems/power/

POW03093-USEN-01