Further Implementation Aspects of the Server Efficiency Rating Tool (SERT)

Klaus-Dieter Lange
Hewlett-Packard Company
klaus.lange@hp.com

Jeremy A. Arnold
IBM Corporation
arnoldje@us.ibm.com

Hansfried Block
Fujitsu Technology Solutions GmbH
hansfried.block@ts.fujitsu.com

Nathan Totura
Intel Corporation
nathan.totura@intel.com

John Beckett
Dell Inc.
john_beckett@dell.com

Mike G. Tricker
Microsoft Corporation
mike.tricker@microsoft.com

ABSTRACT

The Server Efficiency Rating Tool (SERT) has been developed by the Standard Performance Evaluation Corporation (SPEC) at the request of the US Environmental Protection Agency (EPA). Almost 3% of all electricity consumed within the US in 2010 went to running datacenters. With this in mind, the EPA released Version 2.0 of the ENERGY STAR for Computer Servers program in early 2013 to include the mandatory use of the SERT. Other governments world-wide that are also concerned with growing power consumption of servers and datacenters are considering the adoption of the SERT.

Categories and Subject Descriptors

H.3.4 [**Systems and Software**]: Performance evaluation (efficiency and effectiveness)

General Terms

Design, Experimentation, Measurement, Performance, Reliability, Standardization.

Keywords

SPEC, Benchmark, Energy Efficiency, Server, System Performance, Performance Engineering, Memory, System Discovery, Affinitization, Framework, Reporting, Energy Star, Environment Protection Agency (EPA).

1. INTRODUCTION

This paper builds on those published in 2011 [2] and 2012 [4], which described the initial design and development phases of the SERT [5].

As the SERT is now released, this paper first provides an overview of the SERT explaining its different components, a description of the Chauffeur framework, the graphical user interface (GUI), and the automated discovery of hardware and software information. Next, this paper states the challenges of automation of processor affinitization. The SERT is an efficiency evaluation tool that does not offer a single scoring model or metric (unlike a benchmark [8]); therefore, in a section of this paper, the memory subsystem is taken as an example in order to provide a detailed description of how the scores are derived.

A future aspect of the SERT is addressed with initial test results and comparisons between DC and AC powered servers.

ICPE'13, April 21–24, 2013, Prague, Czech Republic.

2. SERT OVERVIEW

The use of multiple power analyzers and temperature sensors is supported by the SERT in order to measure a large scope of system configurations. The most basic SERT measurement configuration requires one **power analyzer**, one **temperature sensor**, a system under test **(SUT),** and a **Controller** system.

The SERT's test harness, named **Chauffeur**, controls the software installed on the SUT and **Controller**. Chauffeur also handles the logistical side of measuring and recording the power consumption and inlet temperature of the SUT.

The SUT gets instructed by the **Director** (Chauffeur instance) to execute the suite, which is comprised of a set of **workloads**. The workload consists of a set of **Worklets**, which exercise the SUT while Chauffeur collects the power and temperature data. The Worklets are the actual code designed to stress a specific system resource or resources, such as the CPU, memory, or storage IO.

Each power analyzer and temperature sensor interacts with its dedicated instance of the **SPEC PTDaemon**, which gathers their readings while the Worklets are executed.

The **Reporter**, executed after all measurements phases are completed, compiles all of the environmental, power, and performance data for a complete test run into an easy-to-read HTML report as well as an extensible markup language (XML) report; the HTML report includes a graphical visualization of the results.

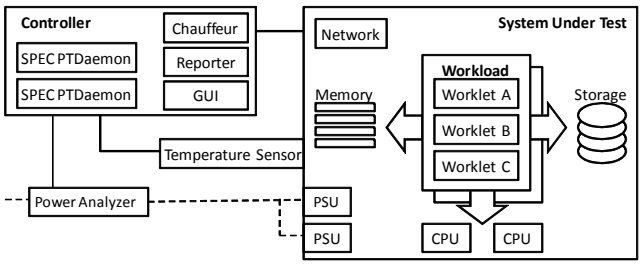


Figure 1 - Discovery Workflow

3. CHAUFFEUR FRAMEWORK

SPEC implemented the Chauffeur framework to support most of the following common functionality, and built the SERT on top of this framework. Future tools and benchmarks could be developed more quickly by taking advantage of Chauffeur.

Energy Measurements

One of the key features of Chauffeur compared to other benchmark frameworks is support for measuring power consumption under a variety of loads. The SPEC PTDaemon interfaces to power analyzers and temperature sensors, and Chauffeur makes the necessary calls to PTDaemon in order to collect the data for the appropriate time intervals. The resulting data is stored together with the runtime performance information so that the data does not have to be correlated after the test run is completed.

The ability to run at multiple utilizations is also an inherent property of Chauffeur. This support is based on the same principles used in SPECpower_ssj2008 to vary the load by first determining the maximum transaction throughput during a calibration process, and then scheduling transactions with appropriate delays to drive the system at lower levels of utilization. This support is implemented generically in Chauffeur and can be applied to any workload that is composed of a series of short-running transactions.

Scalable

The Chauffeur framework is inherently multi-process and multi-threaded, providing scalability across a wide range of servers. Multiple-node runs are also supported, enabling Chauffeur-based workloads to run across multiple blade servers. This support also could be extended to do runs that span multiple virtual servers on one or more physical hosts, although the SERT does not currently allow this.

Chauffeur's Director (the component that instructs the Host JVM to start executing the workload) normally runs on a system other than the SUT, and is usually collocated with the SERT GUI and SPEC PTDaemon. It communicates with a host process that runs on each system that is being measured. The Chauffeur Host will automatically launch one or more Client JVMs as needed, using platform-specific affinity commands as described in Section 4.

Chauffeur can adjust heap settings for each Client JVM based on the amount of memory in the system and the number of Clients being used. The algorithms for calculating heap sizes can be adjusted for individual Worklets; for example, the SERT uses nearly all available memory when running the memory Worklets, but normally uses only 256 MB per processor for CPU Worklets.

Ease of Use

Benchmark configurations can be complex, particularly when power measurements are involved. Chauffeur includes a number of features intended to simplify testing as much as possible.

A particular challenge for industry standard benchmarks and tools like SERT is reporting a complete and accurate description of the system configuration. To assist users in this process, Chauffeur supports automatic collection of system configuration data. This data is included in the results file, and also made available to the SERT GUI to allow users to review and edit the information. This process is described in more detail in Section 4.

Chauffeur also supports automatic validation of results, both at runtime and for completed results. These validation checks can confirm that the configuration is valid, that transactions did not fail, that power analyzer data met the requirements specified by the run rules, and various other requirements. The current validation checks in Chauffeur are defined for the SERT, but the framework is generic to allow checks to be added, changed, or removed for other workloads. This validation gives users confidence that the workload is configured and running properly and that the results are accurate.

Portable

Chauffeur is implemented primarily in Java to simplify portability across different systems. The SERT currently supports 64-bit Windows and Linux on x86 processors, and AIX on the Power architecture. Limited testing has also been performed on other platforms with minimal difficulty.

Platform-specific code is included in two areas of Chauffeur: System Configuration Discovery (see Section 4) and Affinity (see Section 5). In both cases Chauffeur will continue to run on platforms without explicit support. Future versions of Chauffeur can be extended easily by adding support for these features on other platforms.

Although the framework is written in Java, Worklets can make use of other programming languages, as the Storage Worklets in the SERT do. Communication between the Chauffeur Host and Clients is intentionally language-neutral, enabling a possible future native implementation of the Chauffeur Client.

Flexible

Chauffeur implements the core functionality required for the SERT. It also offers flexibility for changing the runtime behavior, either for research purposes or for future Chauffeur-based benchmarks. Many aspects of Chauffeur behavior can be changed via configuration files, without modifications to Chauffeur itself. For example, a virtualization benchmark may require the ability to run different virtual servers at different utilization levels in order to mimic dynamic heterogeneous usage patterns. Chauffeur does not currently support this usage model, but a custom implementation of a “Sequence” [6] could be implemented and plugged in through the configuration file.

It is often desirable to collect a variety of different types of data during a benchmark run, particularly for research and development purposes. Chauffeur provides a Listener interface that allows custom data collection to be performed without modifications to Chauffeur. These listeners are notified at various stages of the run, such as at the beginning and at the end of each measurement interval. Listeners can launch platform-specific tools or collect data that is not directly supported by Chauffeur. Data obtained by listeners can be included in the main Chauffeur results file so it does not need to be correlated after the run completes.

The Chauffeur Reporter is also designed for flexibility. It reads the XML-based results file from a Chauffeur run and produces HTML or plain-text reports. The contents of these reports are defined using an XSL transform, which allows changes to be made to the reports without code changes to Chauffeur. Advanced users can create their own reports, or generate output in comma-separated values (CSV) format for easy import into spreadsheets or statistical packages.

4. SYSTEM CONFIGURATION DISCOVERY

One of the largest challenges in benchmark and rating tool submissions is correctly identifying and capturing all of the characteristics of the SUT. Unintentional errors readily occur when collecting identification details of the various hardware components and recording the details into the benchmark report. Formatting errors overlooked while entering data may only become obvious once the final report is ready for submission. Correcting such errors and oversights in the final report can be cumbersome. The SERT addresses these issues with an automated hardware discovery process and an easy-to-use GUI workflow that assists the user in generating high quality accurate reports. The GUI therefore reduces the burden of test configuration, execution and report editing so that the user can focus on obtaining results.

4.1 Workflow

The relationship between the main components (Hardware Discovery, Test Environment Editor, and Preview Report) of the SERT discovery workflow is shown in Figure 2.

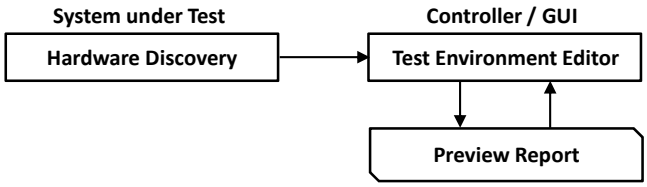


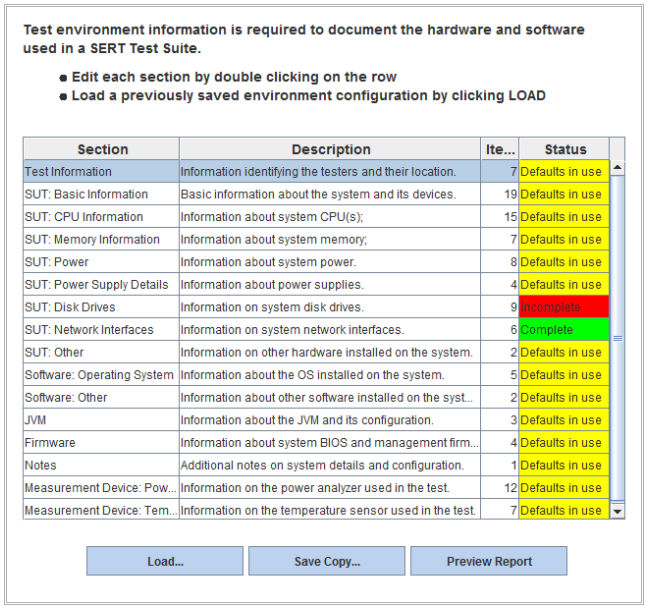
Figure 2 - Discovery Workflow

The initial phase of discovery occurs immediately after the SERT host is started on the SUT. On Windows systems, the hardware discovery scripts execute on the SUT using Windows Management Instrumentation (WMI). The solution on Linux distributions is a combination of scripts that parse the data found in the `/proc` and `/sys` file systems. The burden added to startup is small, and hardware discovery on the Windows platform completes within approximately 15 seconds. The process is efficiently executed once for the lifetime of the SUT host. Once the Controller connects to the SUT host, the discovered data is transferred and is made available for multiple purposes. The data is first used to attach directly to the final report. This ensures that the report reviewer always has an unmodified reference of the discovered data for any discrepancies encountered while reviewing. The discovered data is then sent to the GUI for assisting the user in updating the test environment configuration.

The GUI makes a request to the Controller to retrieve the discovery data as part of one of the first workflow panels the user encounters. The discovered name value pairs are presented in a list and the end-user has an opportunity to review the individual discovered items.

Once the user reviews the data, the GUI workflow is advanced and the discovered data is inserted into the final test configuration file. A mapping is contained within the raw discovered data that assists the GUI in aligning discovered values with the corresponding test environment configuration values. One of the GUI's main responsibilities is to provide an interface for the end-user to access the test environment configuration values. The Test Environment Panel of the GUI, as seen in Figure 3, serves this purpose and is used for verifying, editing, adding, removing, and viewing test environment configuration values.

The panel in Figure 3 illustrates some of the main features of the GUI that assist the user with system reporting. Color coding and status of the individual sections draw the user's attention to the components that still need manual intervention. "Incomplete" items are colored red and indicate that a value within the collection is completely missing.

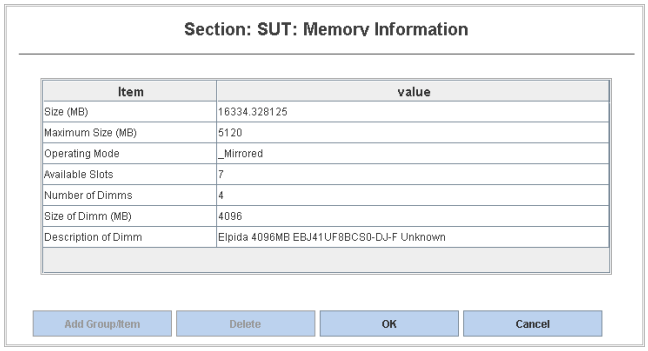


Section	Description	Ite...	Status
Test Information	Information identifying the testers and their location.	7	Defaults in use
SUT: Basic Information	Basic information about the system and its devices.	19	Defaults in use
SUT: CPU Information	Information about system CPU(s);	15	Defaults in use
SUT: Memory Information	Information about system memory;	7	Defaults in use
SUT: Power	Information about system power.	8	Defaults in use
SUT: Power Supply Details	Information about power supplies.	4	Defaults in use
SUT: Disk Drives	Information on system disk drives.	9	Incomplete
SUT: Network Interfaces	Information on system network interfaces.	6	Complete
SUT: Other	Information on other hardware installed on the system.	2	Defaults in use
Software: Operating System	Information about the OS installed on the system.	5	Defaults in use
Software: Other	Information about other software installed on the syst...	2	Defaults in use
JVM	Information about the JVM and its configuration.	3	Defaults in use
Firmware	Information about system BIOS and management firm...	4	Defaults in use
Notes	Additional notes on system details and configuration.	1	Defaults in use
Measurement Device: Pow...	Information on the power analyzer used in the test.	12	Defaults in use
Measurement Device: Tem...	Information on the temperature sensor used in the test.	7	Defaults in use

Figure 3 - Screenshot: SERT Test Environment Panel

Sections that contain default values are color-coded yellow and have a status of "Defaults in use". The default values are used to provide a hint to the user as to what a report value should contain. These default values are demarked with a leading underscore.

The section editor dialog is visible in Figure 4 and is opened when editing a specific section.



Item	value
Size (MB)	16334.328125
Maximum Size (MB)	5120
Operating Mode	_Mirrored
Available Slots	7
Number of Dimms	4
Size of Dimm (MB)	4096
Description of Dimm	Elpida 4096MB EBJ41UF8BCS0-DJ-F Unknown

Figure 4 - Screenshot: SERT Section Editor Dialog

The example Memory Information section contains several values and demonstrates a couple of key GUI features. Visible is the value for "Operating Mode" set to "_Mirrored"; the leading underscore indicates a parameter is currently at a default value and needs updating. The default value is there to give an example of what type of information is expected in this field.

The remaining Memory Information values were automatically populated by the hardware discovery process and a user simply needs to verify that the values represent the system correctly. In

the example, the Size (MB) field is not formatted in the best possible human readable format. The user now has the opportunity to edit each line and fix any other issues. The benefit of hardware discovery is evident in the "Description of DIMM" item value. Determining the dual in-line memory module (DIMM) description without the assistance of hardware discovery would have been tedious and error prone if a lengthy description such as this was entered by hand.

Once all items have been reviewed and updated, the GUI contains a very useful "Preview Report" button that launches a browser with a SERT Report populated with the current test environment values. Figure 5 is an example of a SERT Report preview. A similar status color coding is used to draw attention to values that still need to be updated.

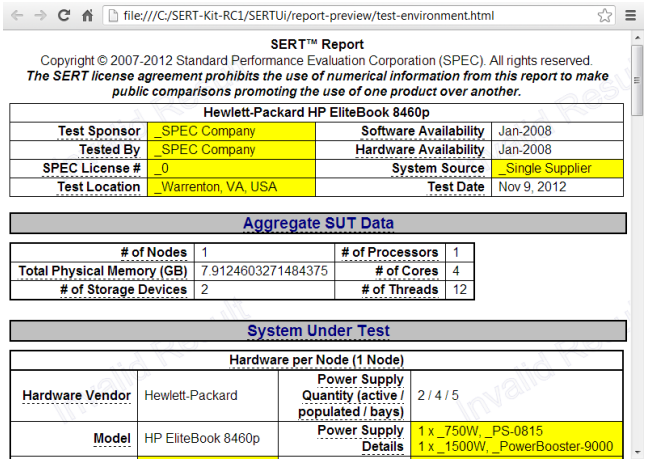


Figure 5 - Screenshot: SERT Preview Report

Another key feature of a SERT Report is the hyperlink found on each field name. Following the link brings the user to a help page with a full, detailed description and definition of the field.

The GUI additionally provides "Save" and "Load" buttons to organize test configurations, a useful feature if one Controller system is used for multiple SUTs. Each new configuration can be saved under a descriptive unique name and the applicable configuration file can be reloaded easily when needed. Configurations can also be moved to different machines to serve as a starting point or reference for the next system to test.

4.2 Implementation Challenges

Several major challenges presented themselves when scoping the effort of automated Hardware Discovery. Perhaps one of the greatest challenges among these was the realization that, depending on the system where Discovery was run, different vendor implementations meant that hardware information was reported in varying ways. To better understand how to filter and format the discovery results, a survey was undertaken by the SPECpower Committee. This survey characterized an early version of the discovery script in order to analyze which fields were consistent across vendor platforms. It also identified which fields could be salvaged with additional post-discovery script logic, for inclusion in the final formatted SERT result output.

This survey sampled 21 separate platform types across five vendors. It enabled the researchers to characterize the suitability of each discovery field in determining which fields were consistent across vendors, and which fields needed additional logic to transform varied output strings into data that could be useful for the final report. The survey found that many details required interpretation and transformation across platforms. Vendor-specific fields such as Vendor and Model number were generally consistent, but other elements such as CPU details, Memory DIMM population, and Disk information were less easily interpreted. In addition other elements, such as Network Device information, were hampered by large numbers of pseudo network interface controller (NIC) devices, depending on the software installed on the target SUT.

In the analysis of CPU details, many separate elements were characterized such as the name, characteristics, frequency, core and thread counts, and the total number of CPUs populated on the system. The survey showed that while many of these elements were correct, others needed considerable rework in order to return correct and relevant information. In particular, the CPU cache fields presented problems. Reporting L1 and L2 cache in a per physical processor combined manner made it difficult to parse and almost impossible to transform into per-core information that would match with standard processor vendor specifications. The number of cores and threads per core were also initially problematic, but allowed for the insertion of additional logic to transform these values into data that could be readily transformed to a per-core/per-processor/per-system format.

Memory population details were another challenging area of the discovery process implementation. All the servers tested were correctly reporting the total installed memory and individual DIMM capacity in a consistent manner. However, elements such as individual slot population and memory manufacturer information could differ significantly depending on the platform. Another focal point of the discovery process was reporting the number and type of storage devices on the SUT, and the Host Bus Adapter (HBA) to which these devices were connected. It was also observed that almost all RAID HBAs analyzed would hide detailed information about the storage device vendor and any other element beyond simple capacity information. However, if the HBA was a simple SATA controller, this information was typically not obscured and additional detail could be derived from the underlying storage devices.

One of the persistent challenges in the development of the discovery process was filtering out extraneous NIC information that is common in some operating systems (OS) and driver models. In some cases, many pseudo-NIC devices were being discovered along with the actual network adapters that were the target. In such circumstances, it was determined that additional logic needed to be added to the discovery script to avoid reporting additional OS pseudo-NIC devices and other interface entries created by VPN and similar drivers. Significant effort was made to correctly determine which interface was connected and to report the connection speed accurately. As it is common on many servers for multiple NIC devices of the same type to be present but only one connected for the SERT run, the discovery process needed to be able to account for this in a fashion that could be correctly interpreted for the purposes of creating a final report.

Another major challenge for the implementation of the discovery mechanism was the fact that some fields were found to be duplicates of one another in cases where multiple instances of the same type of device were discovered. This applies not only to multiple processors, but also to DIMMs, NICs, and storage devices. To allow latter portions of the discovery process to

operate correctly on this information after the discovery script itself was executed, an additional key/value pair was added for discovery elements where this behavior was present. The "_Uniquefields" tag was added to the discovery script output for subsets of elements that could be identified as multiples of the same device type, which enabled the Reporter to automatically generate device counts and correctly represent device elements.

The SERT Automated Discovery process supports two major operating systems: Microsoft Windows Server and multiple Linux distributions. The challenges presented in implementing the Windows discovery process were somewhat more straightforward than Linux due to the common Windows Management Instrumentation (WMI) query structure provided by Microsoft. For Linux, there was no unified query mechanism that could be counted on being installed by default on any major distribution. This meant that all the major discovery elements that were successfully discovered in Windows needed to be compiled into a list, and a determination made as to what Linux resources could be queried in Linux to report the same information. System files in the `/proc` and `/sys` filesystems that could be queried either directly or through common commands made the writing of the Linux Perl discovery script less challenging than originally envisioned. After refinement, both the Windows and Linux discovery scripts reliably report the same information in similar formats.

4.3 Additional Advantages

The SERT Automated Discovery process significantly eases the user burden in documenting and tracking different SUT configuration details. In addition, the discovery data can be used to cross-check the validity of user edits made to the included data after the fact during the submission checking process. Discovery is executed independently of the GUI for any SERT run, and that information is encoded into the results.xml file. This gives the result reviewer the ability to detect whether the final report system description details vary significantly from what was actually found during initial discovery. Almost all the system description fields that show up in the final text and HTML reports can be user-modified, either through the GUI or after the run is complete and the HTML report is regenerated. However, this could lead to erroneous or intentional obfuscation of important system hardware configuration details, misrepresenting the state of the platform that was characterized by a particular SERT run. Since the raw discovery data is present in a form that prevents tampering the result reviewer has greater confidence that the result was actually executed on the same type of hardware as described by the submitter in the HTML report.

5. PROCESSOR AFFINITIZATION

A major challenge in the design of SERT was the need to remove the burden of manually setting Java client (JVM) core affinity. This is required to take the best advantage of processor cache sharing and non-uniform memory access (NUMA) nodes present on multi-socket systems. One of the major design goals was to automate this sometimes complex and burdensome process with an automated affinity generator. This was required to operate across three separate operating systems, one- to eight-processor sockets, up to two NUMA nodes per processor socket, and with different platform ACPI presentations to the OS. The SERT affinity generator also needed to generate the correct affinity masks regardless of how many JVMs needed to be affinitized. This was due to the JVM count being expected to vary depending on the processor type, populated socket count, and workload choice.

Proper NUMA node affinity for multi-socket systems is important for optimal performance. This may be impacted by the additional latency costs incurred when a thread requests memory non-local to the core to which that worker thread is bound. It is therefore critical to be able to identify the NUMA layout of a particular system. This information will be used to group worker threads of a JVM instance to a single physical CPU for maximal resource sharing through explicit binding, and to ensure that memory allocation occurs only on the local NUMA node. Failure to adhere to these best practices will cause performance degradation and high run-to-run variability.

All three OS families supported by the SERT, including Microsoft Windows Server, 64-bit Linux Server distributions, and IBM AIX, were enabled with automatic affinity generator support. Some assumptions were made regarding the topology of the SUT to reduce complexity. Among these assumptions are that the total number of logical processors on the system is evenly divisible by the number of JVMs. It is also assumed that the total JVM count will be evenly spread across all NUMA nodes of a given system. This assumption should hold true if each NUMA node has the same number of logical processors. There are potential situations where these assumptions may not hold true, in which case the SERT will still operate correctly but the affinity may not be completely optimal. The SERT developers expect that configurations with which automatic affinity generators perform sub-optimally will be very rare.

5.1 Linux Affinity

In Linux operating systems, the cores enumerated during kernel boot time are based on the presentation by the ACPI table. This ACPI presentation differs considerably between separate vendors and even between platforms and BIOS releases in some cases. This means that two platforms from different vendors populated with the same processor type and socket count can order cores very differently due to the ACPI table presentation. It was therefore recognized during the SERT development process that the Linux affinity generator needed to be able to handle any type of core enumeration strategy.

Figure 6 and Figure 7 display two different core enumeration strategies utilizing the same 8-core Intel Xeon processor type and socket count with Hyper-Threading enabled. The NUMA Node notations show which set of memory is local to which physical processor and associated cores. Note that the numbers separated by the "/" character indicate the real core and Hyper-Threaded sibling from an OS logical processor enumeration standpoint. As an example, if one were to compare the actual location of OS CPU 1 between the two illustrations, one would see that the first example is located as the first core on the second socket with memory local to NUMA Node 1. The second illustration shows OS CPU 1 as the second core on the first socket, with memory local to NUMA Node 0. These illustrations show two of several enumeration strategies known to exist. It is clear, given the disparate enumeration strategies encountered, that the SERT developers were unable to count on consistency from vendor to vendor, so all affinity strategies needed to be calculated on a per-system basis.

Modern server-oriented 64-bit Linux distributions come with tools such as `numactl` to determine the NUMA node count as well as displaying which cores are local to any given NUMA

node. However, the challenge of Hyper-Threaded platforms meant that it was also necessary to be able to pair individual cores with their Hyper-Threaded siblings. This meant that all logical processor data from the Linux `/proc/cpuinfo` file needed to be parsed to locate this information in order to ensure optimal affinity mask generation.

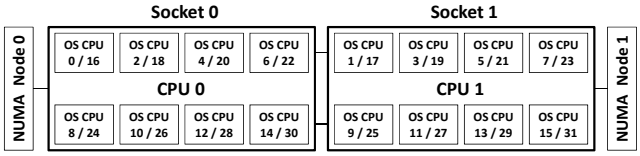


Figure 6 – Core Enumerations – Example 1

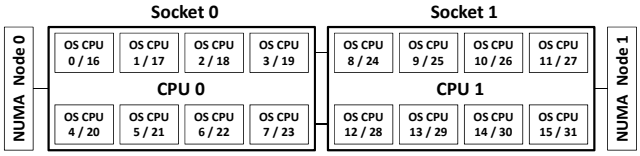


Figure 7 – Core Enumerations – Example 2

Each JVM needs a different affinitization command, which is provided as a set of OS CPUs in a comma-separated list, and added to a `numactl` command which will pin the JVM instance to the logical processor list and closest NUMA node.

5.2 Windows Affinity

Unlike Linux operating systems, the Microsoft Windows Server OS variants enforce their own core enumeration strategy regardless of ACPI presentation. This means that for two platforms with the same processor type and socket count, the location of each core or Hyper-Threaded sibling will be the same regardless of the vendor. This advantage was balanced with the difficulty of easily detecting the number of NUMA nodes on a given system, where no default system tools exist to discover this information.

The lack of the ability to easily determine the NUMA node count meant that the SERT developers needed to design a method for determining this information as part of the SERT package. This was accomplished with .dll helper files called through Java JNI that call specific Windows APIs to provide the NUMA node count and total number of logical processors.

Given the discovered JVM and NUMA node counts the WindowsAffinityProvider Java class generates individual hexadecimal affinity masks for each JVM as it is being spawned. These individual hexadecimal masks are appended to a Windows `start /affinity` command. This ensures that each JVM being spawned is bound to a particular set of logical processors that are on the same physical CPU and NUMA node, and are capable of sharing cache resources for optimal performance. In addition, the `/NODE` switch is used in cases where there is more than one NUMA node on a particular platform. This ensures correct NUMA node locality and makes hexadecimal masks considerably shorter for systems with a high logical processor count. In addition, the use of the `/NODE` switch allows correct affinity masks to be generated for systems with greater than 64 logical processors for Windows Server 2008 R2 and newer operating systems.

5.3 AIX Affinity

The IBM AIX automatic affinity generator was developed after the Linux implementation was complete, and shares some of the same general structure. However, AIX system commands for topology discovery and affinitization are completely different, and the IBM Power Processor Architecture shares little similarity with x86 implementations. Much like the Linux affinity generator, the AIX affinity generator relies on system commands to discover NUMA topology and core/thread enumeration and to bind JVM instances to specific processor resources. The `lssrad` command is used to determine the logical CPU set associated with each NUMA node. The `smtctl` command is utilized to identify which logical processors are actually hardware threads on the same physical processor core. This is used because, although OS logical processors are typically grouped in a predictable fashion, Symmetrical Multi-Threading (SMT) levels can change post-boot in AIX environments. Using this command allows the SERT to handle processors in ST (single-thread), SMT-2 (two threads per core) and SMT-4 (four threads per core) modes. The `execrset` command is used to launch JVM instances and bind them to a list of appropriate hardware threads. To ensure correct memory allocation on local NUMA nodes, `MEMORY_AFFINITY=MCM` environmental variable is set.

6. MEMORY WORKLETS

The Memory Worklets included in the SERT [1] RC1 release have been used for an extensive series of experiments on various system configurations, including different numbers and models of CPUs and DIMMs. The tests were executed on a Fujitsu PRIMERGY computer server model with 24 DIMM slots under both Microsoft Windows Server 2008 R2 and Red Hat Enterprise Linux (RHEL) 6.2. SPEC PTDaemon was employed for the power measurements of total system power.

The reporting capabilities of the SERT were used to document the performance and power results of these experiments. The scaling properties of the two memory Worklets, comparing performance and power consumption of the tested configurations at different load levels, are shown.

The server side Java (SSJ) workload of SPECpower_ssj2008 [7] was the first industry-standard benchmark for measuring computer server efficiency. It primarily stresses the CPU and a limited amount of main memory of computer servers. Higher memory capacities do not increase SSJ performance but do increase power consumption, thus worsening the efficiency result. This deficit restricts its usability for governmental regulatory programs; e.g. the EPA ENERGY STAR [1].

The Memory Worklets developed for SERT generate synthetic loads on server storage devices mimicking basic access patterns from real world usage models. The design goals specified in the SERT Design Document [6] state that the Memory Worklets should give credit to higher bandwidth, lower latency, and increased total size of computer server main memory. The tests described in this paper were performed to check the suitability of the implementation for the designed purpose, especially testing whether these design goals are met.

6.1 Test Configurations

The experiments described in this paper are based on the RC1 version of SERT. The results are representative for this specific SERT release and may differ for the final release.

In order to show the scaling capabilities of the Memory Worklets, a series of tests was executed on a two-socket Fujitsu PRIMERGY RX300 S7 rack server with 24 DIMM slots. Throughput and power consumption have been measured running the SERT Memory Worklets under different CPU and Memory configurations.

6.1.1 Power Measurement Setup

For the experiments described in this paper the SERT standard power measurement setup was used, only measuring the total system power at the Power Supply Unit (PSU) input. Because no adapters for measuring memory power consumption on the system main board have been available for our tests, only overall system power can be provided.

A temperature sensor was used in all test scenarios to measure the ambient temperature and ensure that it always stays above the required minimum of 20°C.

6.1.2 The SERT Memory Workload

The SERT memory workload includes two memory Worklets named Flood and Capacity, briefly introduced below. A detailed description of these Worklets is given in the SERT Design Document [6].

Memory Flood Worklet

The Flood Worklet is based upon STREAM, a popular benchmark that measures memory bandwidth across four common and important array operations. For the *long* (64-bit) integer arrays used in Flood, the following amounts of memory are involved per assignment:

1. **COPY:** *a(i) = b(i)* -- 8 bytes read + 8 bytes write
2. **SCALE:** *a(i) = k * b(i)* -- 8 bytes read + 8 bytes write
3. **ADD:** *a(i) = b(i) + c(i)* -- 16 bytes read + 8 bytes write
4. **TRIAD:** *a(i) = b(i) + k * c(i)* -- 16 bytes read + 8 bytes write

The Flood score is based upon the aggregate system memory bandwidth calculated from the average of these four tests multiplied by the amount of physical memory installed in the SUT. While Flood is based upon STREAM, it uses no STREAM code and is implemented entirely in Java.

Memory Capacity Worklet

The Capacity Worklet uses modified code from the SERT XMLvalidate Worklet, which exercises Java's XML validation package.

Memory scaling in Capacity is done through a scheme known as input data caching (IDC). In IDC, the universe of possible input data (here, randomized XML file data) is pre-computed and then cached within memory before the start of the workload. During workload execution, the input data for a particular transaction instance is then chosen randomly and retrieved from this cache rather than computed on the fly.

The data store size is increased incrementally with each interval. If the data store size is less than the amount of physical memory available to the Worklet, data is retrieved from the cache. Once the data store size is larger than the maximum size of the data cache, a 'cache miss' penalty is incurred when the transaction randomly chooses a data store element that is not currently in the cache. When this occurs, multiple iterations of re-generating a cache element are performed to apply a cache miss penalty and the transaction rate decreases. The more memory the system has, the larger a data store access can be executed before the transaction rate begins to lower as a result of cache misses.

In addition to the transaction characteristics, the maximum cache size is applied to the scoring algorithm. Cache size is computed as: Physical Memory * data-cache-to-heap-ratio (currently 0.6)

While this Worklet does contain transactions that are memory-oriented, there is still a component that is influenced by CPU performance.

Memory Workload Execution

The Flood Worklet uses the SERT fixed iteration execution model (i.e., it always deploys a fixed number of iterations). The amount of memory under test will automatically adjust to fully utilize installed DRAM, so runtime will vary depending upon system configuration.

The Capacity Worklet measurement intervals run for a fixed amount of time, 2 minutes plus pre- and post-measurement phases of 15 seconds Each interval consists of a No Delay Series (i.e., the code runs unrestricted at highest possible speed), where the parameter data-store-size changes with each interval. Table 1 describes the load levels of the two Memory Worklets.

6.1.3 The Tested Configurations

The basic configuration of the SUT used for the test series is described below. The CPU and memory configuration as well as the OS, have been varied to measure how these configuration changes influence Worklet performance and power consumption.

- CPU: 2 x Intel Xeon E5-2690, E5-2620, E5-2603
- RAM: 8, 16, 24 x 16GB PC3L-12800R DIMMs
  8, 16, 24 x 8GB PC3L-12800R DIMMs
- OS: Microsoft Windows Server 2008 R2
  Red Hat Enterprise Linux 6.2
- JVM: Oracle HotSpot 1.7.0_09-b05
- RAID Controller: 1 x LSI 2108 SAS
- Storage Device: 1 x 146GB SAS 2.5" 10k rpm (boot dev.)
- PSU: 1 x 800W/230V AC CSCI Platinum Standard
  1 x 800W/48V DC CSCI Gold Standard

Table 1. Memory Workload Load Levels

Worklet	Load Levels	Description
Flood	Full, Half	2 load levels using full and half of the available memory capacity
Capacity	4, 8, 16, 32, 64, 128, 256, 512, 1024	9 load levels with increasing IDC data store sizes specified in GB

Due to time constraints, each of these configurations was tested with a single SERT run only. The SERT test configuration file (config-all.xml) was modified to execute the Flood, Capacity, and Idle Worklets only, resulting in a reduced execution time of about 40 - 50 minutes per test run depending on the configuration.

6.2 Memory Worklet Test Results

This section presents the results of the experiments executed on the different configurations. Specifically, it shows the scaling capabilities of the memory Worklets and compares their power

consumption. The different configurations tested are described in Table 2. Pairs of three Intel Xeon CPU models with differing number of cores/threads and clock frequencies were used. All tested CPUs include an on-chip memory controller providing four memory channels (i.e. a total of eight memory channels). DIMMs of size 16GB and 8GB have been tested in configurations of 8, 16 and 24 [i.e., 1, 2, and 3 DIMMs Per Channel (DPC)]. Table 2 shows the effective memory frequencies which are defined by the capabilities of DIMMs and CPUs (e.g., the E5-2603 CPU restricts memory frequency to 1066MHz although the tested DIMMs support up to 1600MHz). For all CPU models the memory frequency is reduced to 1066MHz for configurations with 24 DIMMs or 3 DPC.

Table 2. Memory Scaling Configurations

#	Configuration Description
1	E5-2690 8C/16T 2.90G.Hz, 8x16=128GB@1600MHz
2	E5-2690 8C/16T 2.90GHz, 16x16=256GB@1600MHz
3	E5-2690 8C/16T 2.90GHz, 24x16=388GB@1066MHz
4	E5-2690 8C/16T 2.90GHz, 8x8=64GB@1600MHz
5	E5-2690 8C/16T 2.90GHz, 16x8=128GB@1600MHz
6	E5-2690 8C/16T 2.90GHz, 24x8=192GB@1066MHz
7	E5-2620 6C/12T 2.00GHz, 8x8=64GB@1600MHz
8	E5-2620 6C/12T 2.00GHz, 16x8=128GB@1600MHz
9	E5-2620 6C/12T 2.00GHz, 24x8=192GB@1066MHz
10	E5-2603 4C/4T 1.80GHz, 8x8=64GB@1066MHz
11	E5-2603 4C/4T 1.80GHz, 16x8=128GB@1066MHz
12	E5-2603 4C/4T 1.80GHz, 24x8=192GB@1066MHz

In the following description we will reference the tested configurations using the numbers defined in Table 2.

6.2.1 Memory Flood Results

Figure 8 shows the performance results of the Flood Worklet for both load levels, **Full** (solid lines) and **Half** (dotted lines), on all tested hardware and software configurations. The corresponding values are printed in Table 3.

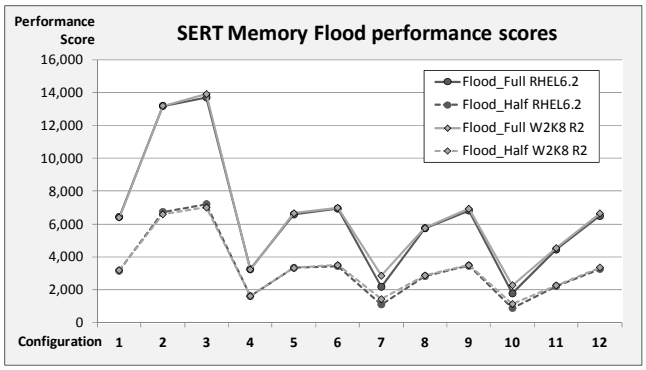


Figure 8 – Flood Worklet Performance Scores

Remarks:

- The performance scores presented here are calculated by the Flood Worklet code from measured aggregate system memory bandwidth multiplied by the amount of physical memory.
- The scores increase with capacity for configurations with the same bandwidth [e.g., (1, 2), (4, 5), (7, 8) and (10, 11, 12)].
- Configurations 3, 6, and 9 show a smaller performance increase with capacity because of lower bandwidth due to reduced memory frequency.
- The scores from the load level **Half** are lower because tested physical memory is cut by half for both measurement and score calculation (i.e., although the bandwidth is about the same the score is halved). This load level was included in the Flood Worklet to facilitate higher efficiency scores with memory power management potentially powering off unused DIMMs.
- The scores are almost identical under both OSs.

Generally, the measured results reflect the desired behavior.

Table 3. Flood Worklet Performance Scores

	RHEL6.2		W2K8 R2	
#	Full	Half	Full	Half
1	6,434	3,163	6,396	3,200
2	13,185	6,743	13,189	6,591
3	13,698	7,216	13,901	7,005
4	3,257	1,622	3,238	1,620
5	6,586	3,345	6,659	3,331
6	6,940	3,451	6,991	3,502
7	2,192	1,117	2,860	1,433
8	5,747	2,836	5,752	2,875
9	6,826	3,460	6,926	3,501
10	1,790	897	2,283	1,143
11	4,454	2,231	4,538	2,274
12	6,493	3,274	6,651	3,365

In order to get performance values in the same order of magnitude from all Worklets the individual performance scores of each Worklet are divided by a fixed reference score. The reference score for each Worklet was determined by averaging the performance scores across several SERT test runs on a well-defined reference configuration under different operating systems.

Figure 9 shows the normalized performance results of the Flood Worklet for the peak load level (i.e., **Full**, on all tested hardware and software configurations together with the corresponding power readings and Idle power).

The bars represent the normalized performance score (left y-axis) and the lines show the corresponding power consumption in watt (right y-axis).

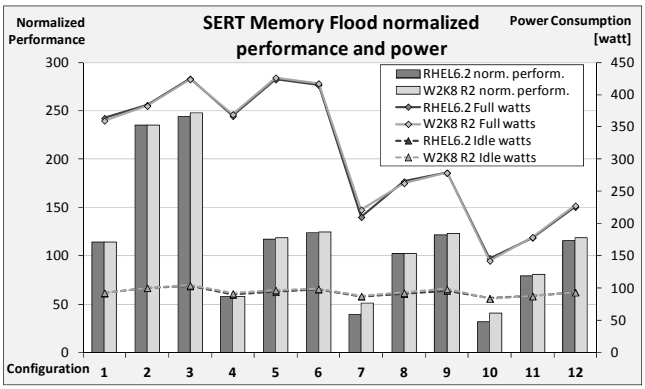


Figure 9 – Flood Worklet Normalized Performance and Power

Normalized performance follows the basic performance score described previously. Idle power for all configurations is almost the same and is only marginally influenced by the number and the capacity of the DIMMs. Peak load power is dominated by CPU power as can be seen from the three power levels in Figure 9 corresponding to the 3 CPU models. Within each group power

increases with the number of DIMMs. Configurations 6 and 9 show a smaller power increase or even decrease due to lower bandwidth caused by reduced memory frequency. This effect is countered by higher DIMM power consumption in configuration 3. Power consumption is the same under both operating systems.

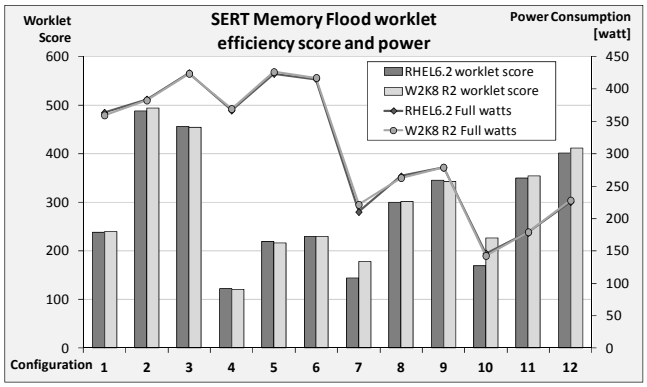


Figure 10 – Flood Worklet Efficiency Score and Power

Figure 10 finally shows the Flood efficiency scores which are calculated as the sum of the normalized performance for each interval divided by the sum of the average-watts for each interval. Efficiency for the Idle Worklet is undefined as the performance part is missing by definition. Please note that Idle power is **not** included in the per Worklet efficiency score calculation.

Efficiency is best for the high end configuration and it increases for most configurations with memory capacity, which is due mainly to the high weight of capacity in the performance calculation. Only configuration 3 has a lower efficiency than the smaller configuration 2, caused by the big increase of power consumption.

6.2.2 Memory Capacity Results

Figure 11 shows the performance results of the Capacity Worklet for selected load levels, 4GB, 64GB, 128GB, and 1024GB data store size, on all configurations with 8GB DIMMs under RHEL6.2 OS. The corresponding values are printed in Table 4.

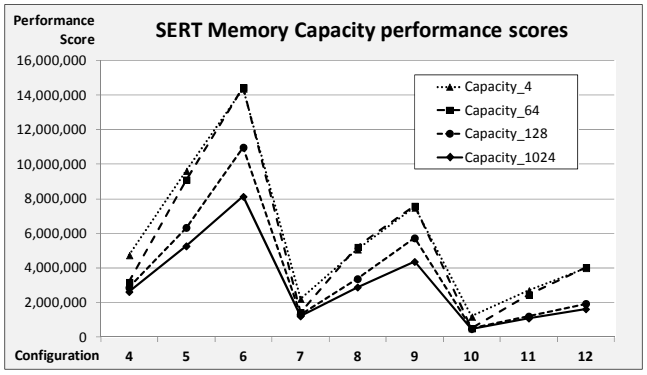


Figure 11 – Capacity Worklet Performance Scores

Remarks:

- The performance scores presented here are calculated by the Capacity Worklet code as explained in chapter 6.1.2.
- Performance is dominated by CPU capabilities as can be seen from the three performance levels in Figure 11 corresponding to the three CPU models.
- In the peak load level the complete data store of 4GB can always be cached completely in available physical memory (i.e., there are no cache misses and the XML transformation rate only depends on CPU capabilities). This load level defines the upper performance limit.
- In the lowest load level, the 1024GB data store never fits into available physical memory resulting in high cache miss rates causing high CPU load for regenerating cache elements. It defines the lower performance boundary.
- Because of additional memory overhead needed for XML translation work, the 64GB physical memory in configurations 4, 7, and 10 is not sufficient for fully caching the 64GB data store. For the configurations with 128GB and 192GB the performance is close to the 4GB results because now the data store is totally cached.
- Cache hit rates increase significantly with additional physical memory for 128GB data store, but it cannot be cached fully in the 192GB configurations due to the overhead explained above.
- The remaining load levels are not shown in this chart for better readability. They reach scores between the upper and lower limit defined by the 4GB and 1024GB data store sizes depending on how much data can be cached.

Generally, the measured results reflect the desired behavior.

Table 4. Capacity Worklet Performance Scores

	Capacity Performance Scores RHEL6.2			
Load	**4**	**64**	**128**	**1024**
4	4,752,281	3,192,736	2,864,201	2,621,257
5	9,629,606	9,116,005	6,334,995	5,264,811
6	14,395,461	14,464,419	10,978,881	8,130,920
7	2,219,895	1,437,212	1,302,491	1,213,172
8	5,085,719	5,221,489	3,369,048	2,886,206
9	7,508,036	7,602,138	5,730,170	4,362,105
10	1,192,931	516,047	490,574	467,977
11	2,712,958	2,469,347	1,203,025	1,084,737
12	4,011,082	4,053,182	1,936,115	1,621,643

Figure 12 shows the normalized performance results of the Capacity Worklet for the peak load level on all tested hardware and software configurations together with the corresponding power readings and Idle power.

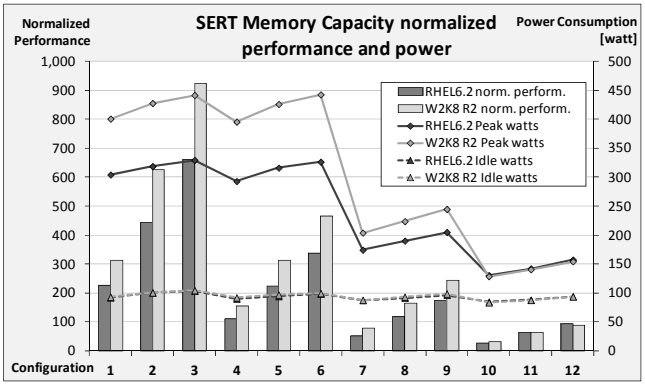


Figure 12 – Capacity Worklet Normalized Performance and Power

Normalized performance follows the basic performance score. Peak load power is dominated by CPU power as can be seen from the three power levels in Figure 12 corresponding to the three

CPU models. Within each group both power and performance increase with the number of DIMMs.

Performance is significantly higher under the Windows Server 2008 R2 OS compared to Red Hat Enterprise Linux 6.2. Power increases proportionally with the performance. Currently, there is no explanation for these differences. Additional experiments are required to analyze this anomaly.

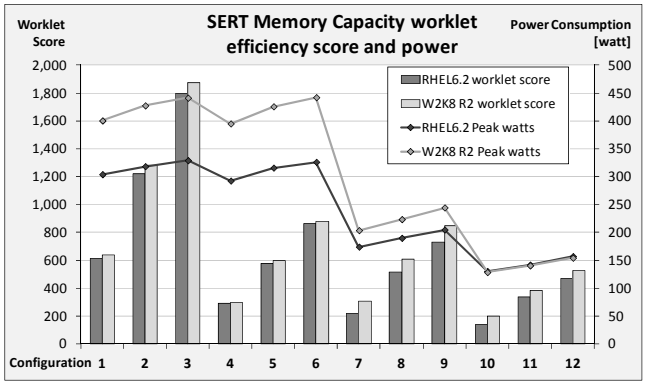


Figure 13 – Capacity Worklet Efficiency Score and Power

The Capacity efficiency scores are shown in Figure 13. Again, efficiency is best for the high end configurations and it increases for all configurations alongside memory capacity, which is mainly due to the high weighting of capacity in the performance calculation. Although performance was higher under Windows, the efficiency scores are close to each other or even equal for some configurations. This is due to the higher power consumption under Windows which compensates the performance advantage.

7. AC-DC COMPARISON

The EPA has received requests from stakeholders to support DC-powered servers with Version 2.0 of the ENERGY STAR Enterprise Servers Specification [1]. Currently, this cannot be achieved because the SERT does not support DC loads, as stated in the SERT Design Document [6]. The general SERT design allows for such measurements, once a DC-capable version of the SPEC PTDaemon becomes available. In order to evaluate the DC capabilities of the SERT and to compare the characteristics of AC and DC power consumption a modified version of SPEC PTDaemon has been implemented, which supports DC measurements for a test series. Typically the power analyzer uncertainties of DC measurements are significantly higher than those of AC measurements, specifically with lower voltages. In order to stay below the 1% uncertainty threshold required for SERT measurements, a high precision power analyzer had to be used. This special beta version of the SPEC PTDaemon is for internal use only. Currently there are no plans for releasing this version with the final SERT kit.

Three consecutive full SERT runs have been executed on the test system described in chapter 6.1.3 with 8 x 8GB DIMMs using an 800W/230V AC PSU. A second series of full SERT tests was performed on the same configuration but exchanging the PSU against an 800W/48V DC model.

Figure 14 shows the normalized peak performance for all SERT Worklets in both configurations and the corresponding power consumption values.

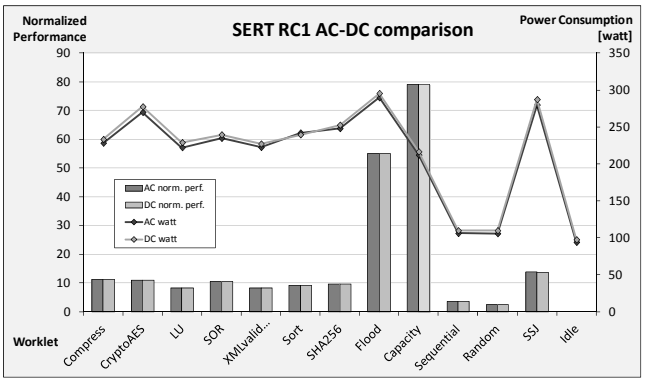


Figure 14 – SERT RC1 AC – DC Comparison

The detailed result values are provided in Table 5, which also includes a column showing the power consumption delta.

Table 5. SERT AC – DC comparison test results

PSU	800W AC CSCI Platinum		800W DC CSCI Gold		Delta
Worklet	**Norm. Perf.@ 100%**	**Avg. Watts @ 100%**	**Norm. Perf. @ 100%**	**Avg. Watts @ 100%**	**DC - AC watts**
Compress	11.1	228	11.1	233	2.2%
CryptoAES	10.9	270	10.9	277	2.8%
LU	8.3	222	8.3	229	3.1%
SOR	10.5	235	10.5	240	2.0%
XMLvalidate	8.3	222	8.2	227	2.1%
Sort	9.1	242	9.1	240	-1.0%
SHA256	9.6	248	9.6	253	1.9%
Flood	54.9	290	54.9	296	2.0%
Capacity	79.0	212	79.0	216	2.1%
Sequential	3.5	106	3.5	110	3.4%
Random	2.4	106	2.4	110	3.9%
SSJ	13.7	280	13.7	287	2.6%
Idle		94		97	3.6%

7.1 AC-DC Comparison Results

The performance scores are almost identical for all Worklets. Power consumption is however typically 2 – 4% higher for the DC PSU configuration. Only the Sort Worklet consumes less power in the DC configuration. This is probably due to high run-to-run variations seen during our test, which may be caused by the aggressive JVM tuning flags currently used by default. These flags will be revised for the final SERT release with the goal of minimizing run-to-run variations.

The higher power draw of the DC configuration is partly caused by the lower efficiency standard of the DC PSU, which is CSCI Gold compared to CSCI Platinum for the AC PSU. Based on the efficiency curves for both PSUs, it is estimated that about half of the power delta is due to higher power losses in the DC model.

Generally it is assumed that DC PSUs would show the higher efficiency because AC-DC conversion losses will not occur. These tests have shown the opposite behavior. This is most probably due to the lower voltage of the DC PSU (48V) compared to the AC PSU (230V), which results in much higher currents having to be handled by the PSU for voltage conversion. Higher

currents cause increased power loss and this effect dominates the missing AC-DC conversion loss.

Repeating this comparison with a 400V DC PSU would most probably end in favor of the DC configuration, because of the absence of AC-DC conversion losses and lower currents resulting in reduced voltage conversion losses.

8. CONCLUSION

The SERT was released in February 2013, for use in Version 2.0 of the EPA ENERGY STAR for Computer Servers program. While previous papers described the initial design and implementation of the SERT, this one has focused on the design decisions, implementation trade-offs and validation performed to actually deliver the tool.

This paper discusses in detail the design decisions taken to simplify configuring and running the SERT, with customer feedback during the beta phases as a key input. Experience gained with customers of SPECpower_ssj2008, and the results reviewed by the SPECpower Committee also inspired simplification of use. Five years of ssj2008 submission highlighted some of the complexities of configuring test hardware and power analyzers to perform measurements. This directly drove the development of the affinity mask generator, and most significantly, the GUI and hardware discovery components.

It has been stressed throughout the development of the SERT that it is not a benchmark. However there continue to be requests for scores and metrics that can be used to help differentiate between similar servers from different vendors.

The SERT implementation also shows that the underlining Chauffeur framework provides the features needed for future performance and energy efficiency benchmark implementations.

Finally, this paper describes some of the future evolution of the SERT that is under active development or consideration. Adding support for DC power is in the experimental phase. The SERT is currently providing results that realistically represent what might be observed with production workloads on servers running on DC power. The intent is for a future release to add this support, broadening the appeal and usability of the SERT to non-traditional server industries, such as telecommunications.

Considerable interest has been shown in the SERT by several countries. It is therefore reasonable to hope for widespread adoption in the next few years. This should enable consistency across markets, which will benefit computer manufacturers and users, as well as the environment as a whole.

The SERT offers ease of use, cross platform support, a strong range of synthetic Worklets, and a highly modular and extensible architecture. These features are intended to ensure its ability to evolve along with the computer industry, leading to a common baseline for server power measurements across geographies.

Work is already underway to further improve and extend the SERT. It is hoped that this leads to even broader adoption across other industries, and for other types of workloads.

9. ACKNOWLEDGMENTS

The authors would like to acknowledge Karin Wulf for providing a modified version of SPEC PTDaemon supporting DC power measurements, as well as Thomas Brand and Charlie Cha for executing numerous measurements for this paper.

The authors also wish to acknowledge current and past members of the SPECpower Committee who have contributed to the design, development, testing, and overall success of the SERT: Sanjay Sharma, Greg Darnell, Karl Huppler, Van Smith, Paul Muehr, David Ott, Cathy Sandifer, Jason Glick, and Dianne Rice, as well as the late Alan Adamson and Larry Gray.

10. REFERENCES

[1] ENERGY STAR Enterprise Servers Specification Version 2.0: http://www.energystar.gov/products/specs/node/142

[2] Lange, K. D., and Tricker, M. G. 2011. The Design and Development of the Server Efficiency Rating Tool (SERT). In *Proceedings of the second joint WOSP/SIPEW international conference on Performance engineering* (Karlsruhe, Germany, March 14 - 16, 2011). DOI= http://dx.doi.org/10.1145/1958746.1958769

[3] Lange. K. D., Identifying Shades of Green: The SPECpower Benchmarks, IEEE Computer, V42 #3 2009, 95-97, DOI= http://dx.doi.org/10.1109/MC.2009.84

[4] Lange, K. D., Tricker, M. G., Arnold, A. A., Block, H., and Koopmann, C. 2012. The Implementation of the Server Efficiency Rating Tool (SERT). In *ICPE '12 Proceedings of the third joint WOSP/SIPEW international conference on Performance Engineering* (Boston, USA, April 22 - 25, 2012). DOI= http://dx.doi.org/10.1145/2188286.2188307

[5] Server Efficiency Rating Tool - Home Page: http://www.spec.org/sert/

[6] Server Efficiency Rating Tool - Design Document: http://www.spec.org/sert/docs/SERT-Design_Document.pdf

[7] SPECpower_ssj2008 - Home Page: http://www.spec.org/power_ssj2008/

[8] SPEC Power and Performance Benchmark Methodology: http://www.spec.org/power/docs/SPEC-Power_and_Performance_Methodology.pdf