Characterization of SPECpower_ssj2008** benchmark

Larry D Gray, Anil Kumar and Harry H Li, *Intel Corporation*

***Abstract*— SPEC** recently released SPECpower_ssj2008, the first industry benchmark to measure performance and power of volume server class computers using graduated load levels. In this paper, we present a brief overview and an initial characterization of SPECpower_ssj2008 by measuring the utilization of system resources with the aid of processor monitoring events, at graduated load levels and by comparing the sensitivity of final metric and other related data between various configurations consisting of hardware changes as well as software changes on Intel® Xeon® processor based servers. Even though this is early data is from a specific platform and operating system, it does validate many expected behaviors and patterns opening exciting new opportunities for researchers to investigate specific areas as well as in-depth characterization as a next step.**

***Index Terms*— SPEC, SPECpower, SPECpower_ssj2008, performance to power ratio, Graduated load levels, overall ssj_ops/watt, energy efficiency, volume class Servers.**

I. Introduction

December of 2007 brought a significant milestone for SPEC, the Standard Performance Evaluation Corporation, with the release of the industry's first benchmark to measure the power and performance of volume server platforms with an innovative graduated workload.

Formally named SPECpower_ssj2008, this new benchmark measures eleven levels of server loads from zero to 100% of a given platform's full capacity to process business transactions with a server side Java application. Full disclosure reports using this benchmark provide an unprecedented amount of new information on the power consumption and performance of the tested platform.

In this paper we strive to provide some insights into workload behavior and server resource utilization characteristics of this benchmark above and beyond the wealth of information included in the now available SPEC provided documentation cataloged on the SPEC public website[1][2].

The authors have been active members of the SPECpower benchmark development team from the outset and therefore are capable of providing valuable insights on the workload, the rationale for design decisions, and the strengths and inevitable weaknesses inherent in any such product. We share this information to enhance the understanding of the benchmark and its intended usage. Our intent is that others will benefit and therefore be more interested in using the benchmark as an evaluation tool across the wide array of studies to which it can apply.

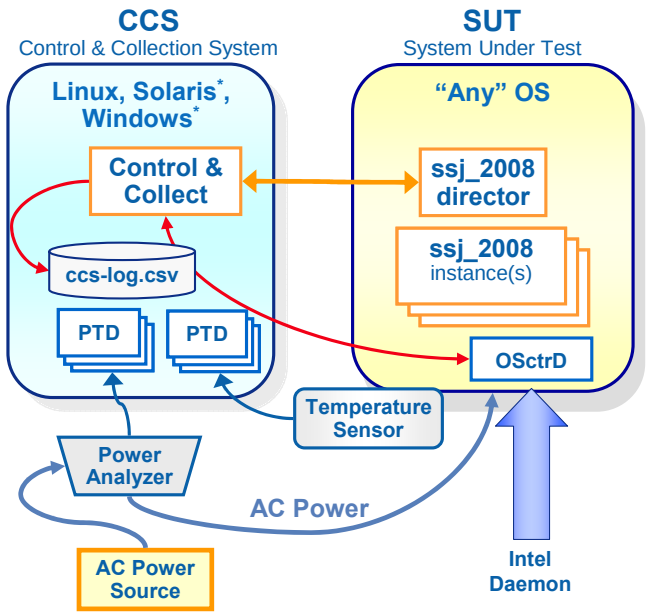

Fig 1 – Elements of the SPECpower Framework

A. A little History

The SPECpower committee was chartered in January of 2006 to create a benchmark that would address the emerging need to measure power consumption and performance of server class computer systems under application-like loads. The intersection of performance and power has become an important attribute of computer systems, sometimes labeled efficiency. A standard method of measuring and reporting both would require a disciplined approach beyond what was then available on the open market.

The fact that one workload or benchmark could not represent the spectrum of server usage was generally accepted and therefore the SPECpower committee was determined to create more than one "benchmark" in more than one application segment.

The committee is and was staffed by engineers and managers from these companies: AMD*, Dell*, Fujitsu Siemens Computers*, HP*, IBM*, Intel*, and Sun Microsystems*. The manufacturers were assisted by representatives from academia including the University of California Berkeley, Virginia Polytechnic Institute and State University, and Lawrence Berkeley National Laboratory.

After two years of constant collaboration, design, coding and extensive testing efforts, the SPECpower committee released [13] its first benchmark named SPECpower_ssj2008 on December 11, 2007 to very positive reviews from industry and trade press.

II. SPECpower_ssj2008 Overview

A. Measuring Power with Performance

This section provides a brief overview of the SPECpower measurement framework described in more detail in the set of documents freely available at the SPEC public web site on the SPECpower_ssj2008page[2][3][4][5][6][8][9].

For deeper understanding of the design of the benchmark and its essential elements, refer first the to the "SPEC Power and Performance Design Overview" [8][9][10][11][12].

Several challenges are presented by the requirement to measure power consumption with performance, in particular at multiple load levels.

A "measurement methodology" was established and then realized by implementing a measurement framework that requires a separate platform to which power and temperature measurement devices are attached, with the necessary logging and reporting functions.

The two systems required are the system under test (SUT) and the Control and Collection System (CCS)[9]. Communications between the two systems is enabled by a standard ethernet local area network (LAN).

The addition of a "measurement server" enables a host of benefits that include but are not limited to:

- independence from the workload to enable quick integration of new workloads,
- multiples measurements; the ability to manage a number of SUTs and multiple measurement devices.
  multiple JVM instances are also supported.
- low impact to the loads on the SUT for data consolidation and logging

Altogether, the design permits extending the framework from the current capability to measure a stand-alone server with a single OS, to environments or topologies with multiple OS images, for instance blade servers and virtualized servers, with workloads appropriate to those environments.

B. The Measurement Framework

To better understand the terminology and characterization data later in this paper, a brief overview of the SPECpower framework software elements is provided. Figure 1 is a graphic representation of the framework with the interconnections.

On the left side of Figure 1, is the Control and Collect system (or measurement server). On the right is the SUT where the workload runs. The "up arrow" on the right, under the SUT, points to a non-standard element (not provided by SPEC), the OS counters daemon (OSctrD). Created by Intel, this software implements the capability to collect resource utilization data from a Windows OS platform, passing a configurable set of counter data to the CCS for logging, second by second, along side the power, performance and other essential data items. It is this additional element of the framework that enables producing the data shown later in this report.

The ssj_2008 workload runs on a SUT plugged into a power analyzer plugged into the building's power infrastructure, measuring power of the entire SUT platform. This is sometimes described as "watts at the wall".

The power analyzer is connected to the CCS machine via a data cable where purpose built software, the PTD (Power and Temperature Daemon) records electrical activity from a power analyzer, and ambient temperature from a temperature sensor device placed at the air

in-flow to the SUT.

The SPECpower_ssj2008 workload uses TCP/IP protocol to pass time, performance and status data to the CCS system which then consolidates that with power and temperature, and in this case the OS counters, logging all together into one record in a comma separated file.

C. A Graduated Workload

The notion of a graduated workload was inspired by the advent of processor power management technologies on volume server platforms, which are most effective at low loads and usually required to operate without a negative performance impact.

All this is driven by the global need to conserve energy, reduce carbon footprints, and the general movement to be more green. Platform power consumption has become a competitive differentiator for the system manufacturers.

Add then that it has become widely recognized that most (commercial) data-center servers generally run at low loads with resources underutilized except during periods of peak business activity – which will vary widely for various types of businesses and geographies.

Since there is no "typical" load level, the graduated load was conceived to assess power management across what has come to be known as the "load line".

The benchmark reports the power consumption and the performance at each load level, allowing the reader to reasonably match their usage and determine power usage for that platform.

1. Platform Capacity Adjustment

Systems of widely different capacity must be fairly measured, so a method was conceived to determine the full transaction throughput capacity of a given system, and then increment the workload gradations accordingly.

A benchmark run begins with 3 or more "calibration levels" where an ungated stream of transactions is presented to the application. The calibration workloads are unrealistic but they serve to determine the full performance potential of the system – with the SPECpower_ssj2008 application and transaction mix.

The calibration throughput is used to set a "throughput target" for the 100% load level. The other load levels are then graduated percentages of the calibrated target load. In the normal case, the levels are increments of 10%. Fewer or more levels are configurable.

It is important, when interpreting data in this paper and from the SPECpower_ssj2008 disclosure reports that the load levels labeled as percentages are a percent of "target calibrated throughput".

It is a common misconception that gradations are governed by processor utilization.

Processor (or CPU) utilization is an outcome of the benchmark and considered to be unique to a given platform. Since there are a number of vagaries and sometimes gross differences to the meaning of CPU utilization from one architecture to another, this point is emphasized. (We encourage someone to use this as a topic a future paper).

2. Measurement Intervals

The graduated method loads the system with a given throughput for a fixed amount of time during which power is measured every second along with the effective transaction rate at that second.

Figure 2 provides a graphic example of second by second transaction throughput across five levels of transaction load. The benchmark form of SPECpower_ssj2008 implements 10 load levels plus the state known as "active idle".

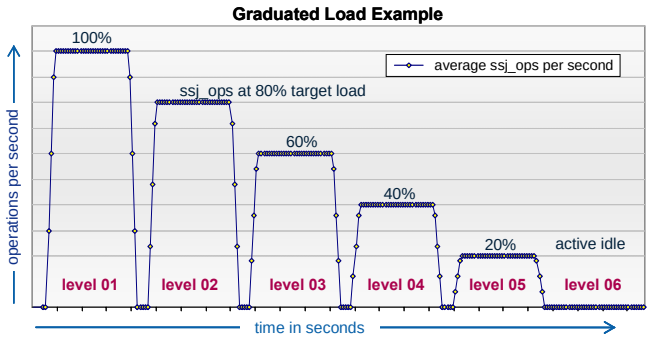

Fig 2 – Graduated Load Example

3. Active Idle

Idle is generally the state when the system is running no applications nor performing any operating system management tasks. CPU utilization is zero. We could label this state "OS idle".

The duration of idle states can vary from fractions of seconds to minutes. Modern operating systems run many asynchronous

background tasks and therefore most servers are never totally idle for long periods.

"Active idle" is a SPEC defined state where an application is running and no transactions are incoming or in process; the system is ready to quickly respond to any incoming transactions.

Given that servers are usually operating 24 x 7 they are also ready to accept transactions therefore “active idle” is the most common operating state.

In this benchmark, active idle is handled and measured virtually the same as the other 10 load levels, except no transactions are scheduled.

1. *Workload States and State Changes*

Accurate, consistent and repeatable measurement of performance and power together requires that there be mechanisms to assure that a period known as the measurement interval is carefully defined, delineated and controlled.

This control is implemented through the definition of “states” which identify the various phases of the workload in the detailed CCS log file. In the case of a graduated workload, the load type and level number is included.

These states and the change rules are built in to the ssj2008 code and passed to the director along with the per second average performance, time stamps and other meta data.

There are four distinct phases of any given load level:

1. “inter” is a period between load levels. This method creates a break between load levels that eases post run visual analysis.

2. “ramp up” (pre-measurement) is a period of time that allows the application to reach a level of processing that will continue for the duration.

3. “recording” is where data is collected and summarized in post-processing steps. This is the “measurement interval”.

4. “ramp down” (post-measurement) is a period of time where the application will continue to process transactions till the very end of the load level.

Following ramp down, the cycle begins again with another “inter” level or if all configured levels have been completed, the workload can terminate normally.

State changes for one workload level are illustrated below in the chart in figure 3.

Note that power is measured continuously to enable detailed analysis. For reporting purpose in benchmark disclosures, only the average power in the measurement interval is used. Also all these intervals are long enough to provide sufficient settle time for consistent power and performance measurements.

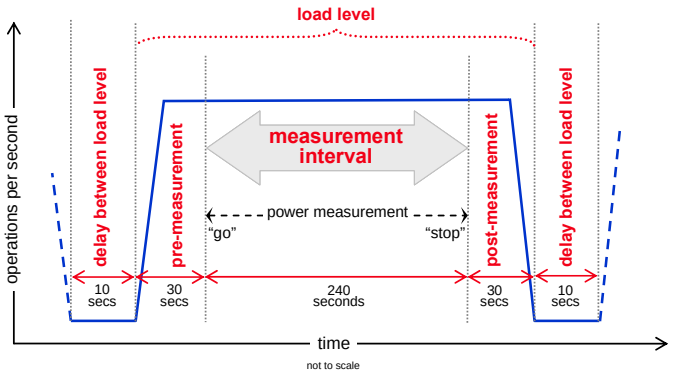

Fig 3 – State changes in a load level

III. SERVER RESOURCE UTILIZATION

A. *Overview*

The SPECpower_ssj2008 benchmark emulates a server side Java transaction processing application. It exercises processors, processor caches, the memory hierarchy, implementations of the JVM (Java Virtual Machine), JIT (Just-In-Time) compiler, garbage collection, threads and some aspects of the operating system.

A Java application was chosen for the very important advantage of cross operating system portability. The opportunity to leverage existing code from the SPECjbb2005 benchmark was irresistible.

Base code and transaction types [15] are from SPECjbb2005, but many substantive changes make the two not comparable. Some notable differences are a modified transaction mix, transaction scheduling and arrival method, calibration to seek the platform peak transaction capacity, altered throughput accounting, data collection via a network with TCP/IP, additional logging that increases disk I/O, plus other less significant changes. Overall, even though ssj2008 is derived from SPECjbb2005, it is very different.

While running, the application makes some use of the network and does minimal disk I/O. Actual data rates are shown in a later section.

With the arrival of multi-core processors in symmetric multi-processor systems, a high degree of scalability was a top benchmark design goal.

It is expected that the benchmark will be run on a very wide range of low end and mid-range servers which span the space from a single socket single processor core (uni-processor servers) up to servers that support multiple processors (SMP or symmetric multiprocessor) where each processor can incorporate 1, 2, 4 and likely more processing cores – then some implementations will support SMT (Simultaneous Multi-threading).

Conscious design decisions were made such that additional disks or network interfaces would not be necessary with increases in available processing capacity.

The scalability of the benchmark is an incredibly positive attribute when setting out to measure power and the performance of basic system infrastructure (processors, chipset, memory, fans, power supply, etc.) across platforms with a very broad range of transaction processing capacity.

B. Resource Usage and Platform Power Consumption

To the above that we also understand that platform power consumption under varied loads is largely driven by the power requirements of the processors (a generalization that applies to most platforms available today) which changes with the applied load. This may seem counter-intuitive since memory and disks are both subject to dynamic and random access.

Memory power consumption does change with load, however, as a percentage of total platform power, the range from idle to full load might be only 1-2% of platform power. Use this information only as a guide since memory designs, types, and densities can be quite different in their behavior from one to another.

Modern high density disk drives show similar behavior. Once spun up, power changes are small with usage, again relative to total platform power.

Network interface cards (NICs) follow the same pattern that when enabled, with a LAN cable plugged in, a NIC is consuming power very near its maximum and very small power increase is seen with higher traffic, again on the order of 1% or less of total platform power.

As a caveat, it is important to note that the observations above apply to the types of memory, disks and NICs found in high volume platforms common to x86 servers. Exhaustive studies of peripheral and component power consumption are yet to be completed.

IV. SPECpower_ssj2008 METRIC DEFINITION

A. The primary metric

The primary metric for SPECpower_ssj2008 is "overall ssj_ops/watt" where:

"overall ssj_ops/watt"
= ∑ 11 avg-trans-rate pts / ∑ 11 power pts

B. Unprecedented data in Full Disclosure Report

The SPECpower_ssj2008 Full Disclosure Report (FDR) presents and abundance of data on performance, power as well as detailed configuration data. Table 1 below has been copied from FDR of SPECpower_ssj2008 publication [14] and highlights important data fields [7] and values:

Table 1 – Performance and power data

Performance			Power	
Target Load	Actual Load	ssj ops	Average Power (W)	Performance to Power Ratio
100%	99.10%	220,306	276	799
90%	90.40%	200,860	269	746
80%	79.50%	176,684	261	677
70%	70.30%	156,344	254	616
60%	59.60%	132,525	245	541
50%	49.60%	110,222	237	465
40%	40.20%	89,388	229	390
30%	30.10%	66,875	221	302
20%	19.90%	44,157	213	207
10%	10.20%	22,649	206	110
Active Idle		0	198	0
		∑ssj_ops / ∑power =		**468**

In table 1 above, ssj_ops column, first row, is ssj_ops@100%. The fourth and fifth columns contain average power (in watts) and a performance to power ratio at each level. The "primary" metric is highlighted in the last row. Following page one of the FDR, are several more pages with important configuration, environment and electrical data from the benchmark run.

V. PLATFORM HARDWARE AND SOFTWARE DETAILS

A. Platform configuration details

To understand and characterize this benchmark, we used an Intel® Xeon® based, 2 socket Intel "white box" server with the following configuration described in Table 2:

SUT:	SUT: Intel® "White Box"
HW	Dual and Quad Core Intel® Xeon® 2.0 & 3.0 GHz
	Supermicro* X7DB8/ Main Board, Super Micro 5000P
	4x 2GB FBDIMMs
	1x 700W PSU
	5U Tower Platform
OS	Microsoft* Windows Server 2003 64 bit
Power Options	Server Balanced Processor Power and Performance
JVM	JVM: BEA* JRockit* P27.4.0 64 bit
JVM Options	JVM Command Line similar to published results
Sampling Rates	Power: 1 second (average from meter)
SPECpower_ssj2008 setup	
	SSJ Director on SUT
	Load levels 120 seconds

Table 2 – Platform hardware - software details

Load levels of 120 seconds were used to reduce total run time as we have observed that measurements from shorter load levels are reasonably consistent with that of 240 sec load levels.

Also note that we do not use SPECPower_ssj2008 metrics, since the measurements in this report are largely "non-compliant"; that is, they can not be published along side full disclosure reports. Data herein is intended for "academic" use only.

The measurements and observations in the following sections are in large part exclusive to the Microsoft Windows Server operating environment. Disk write frequency and rates are largely governed by policies of the OS used. Platforms other than those used in this study may also affect the resource utilization characteristics.

VI. SPECPOWER_SSJ2008 CHARACTERIZATION DATA

A. SSJ_2008– per JVM instance

Code footprint size

Each SSJ (JVM) instance has a code size of ~1.5 MByte; when totaling the size of all methods that have been JITed and optimized.

Data footprint size

Each warehouse thread has ~50 MBytes of long lived "database" objects and produces ~8Kbytes of short lived transient objects per SSJ transaction. The overall data footprint depends on the number of threads (warehouses) and maximum throughput produced.

Java Heap Size and Sizing

The Java heap size is user configurable where the best size is dependent upon available memory and the number of JVMs chosen for a particular run. An optimal heap size is necessary for optimal performance.

A heap size too big could cause memory swapping (total heap size > RAM). Too small a heap will incur a performance penalty due to frequent Garbage Collections (GC).

Overall, due to the nature of the Java heap, an application can exercise any amount of memory and a user could measure the energy consumption impact, but the performance component only benefits to a certain extent.

The optimal physical memory size is throughput capacity - processing capability – dependent and does vary by platform and its hardware expandability. As an example, for Quad-Core Intel Xeon based Dual Processor systems, ~8GB RAM is optimal when running SPECpower_ssj_2008.

B. Processor Utilization

Figure 4 below show CPU % utilization tracking closely with the transaction loads on the Intel Core 2 architecture. On other micro-architectures it will vary (SMT etc.).

Load level targets are set to be percentages of ssj_ops@calibrated, the average of the last two calibration levels.

We repeat that readers must be aware that CPU utilization is no part of the benchmark.

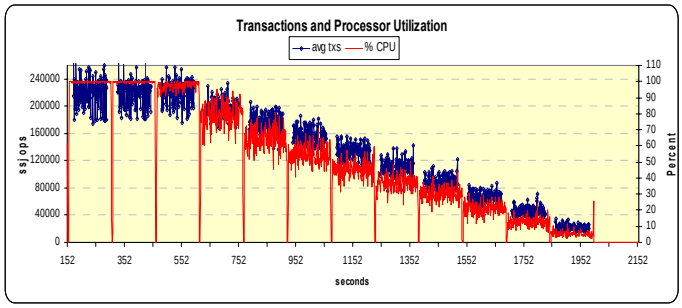

Fig 4 – CPU % utilization

Average second by second ssj_ops are exhibiting the expected variability within a load level because the inter-arrival time of transactions is modeled with a negative exponential distribution to better simulate random arrival of work.

C. Power and processor utilization

Figure 5 below shows that Power consumption varies with load. Also the variability of transaction throughput is being reflected in power consumption changes (watts).

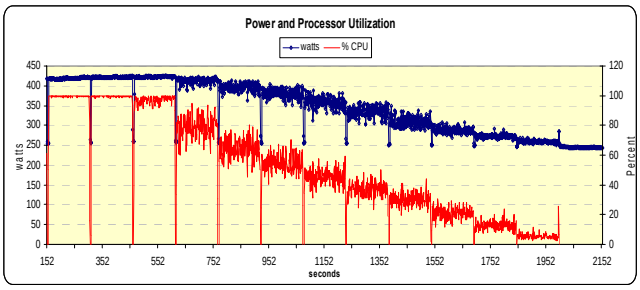

Fig 5 – Power and CPU % utilization

D. Power, ssj_ops, and processor utilization

Plotted points in Figure 6 shows that ssj_ops, Power and CPU % utilization are changing together – showing a distinc relationship one to the other

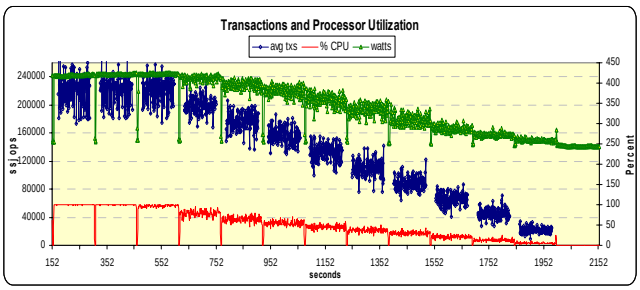

Fig 6 – Power, ssj_ops, and CPU % utilization

E. % time in C1 state

Figure 7 below shows that % time in C1 state is the inverse of CPU % utilization at all load levels. Time in C1 state contributes to power saving which varies with architecture, OS and policies. For example Intel EIST "enabled" in BIOS will result in more power saving at lower utilizations.

"C" states are lower processor power states. Their specific definition is architecture and implementation dependent

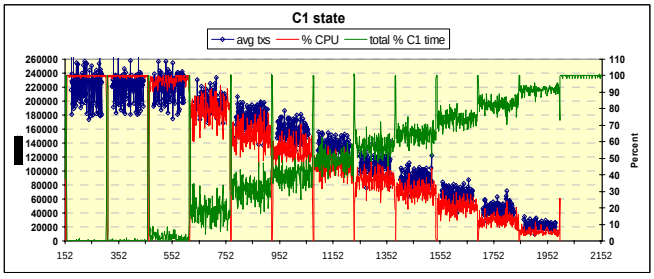

Fig 7 – % of Time in C1 state

F. Memory utilization

Data in figure 8 below has been collected using typical tuning (Xmx==Xms) where Java heap allocated remains same throughout the run. As a result committed memory in use remains constant at all load levels including active idle.

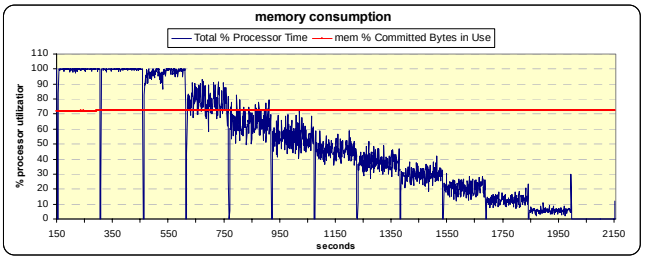

Fig 8 –Memory utilization

G. Network I/O

Data in Figure 9 indicates ~1500 Bytes/sec of network I/O at all load levels including active idle. As expected network traffic is similar at all load levels and does not track load. Most of the Network I/O is from per sec request/response between Control & Collect (CCS) and SSJ_2008 Director.

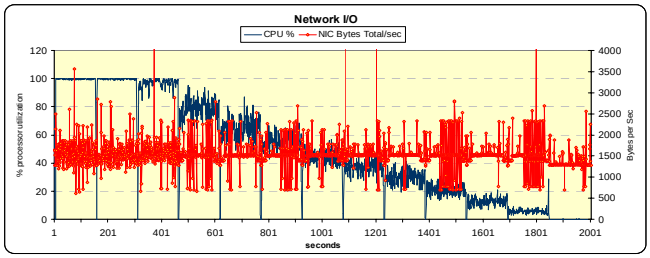

Fig 9 –Network I/O

H. Disk I/O

Disk I/O in figure 10 shows regular bursts of ~140Kbyte writes. On an average there is ~3.3Kbytes/sec of Disk I/O at all load levels. Most disk writes are related to SSJ_2008 logging. Disk reads average is zero.

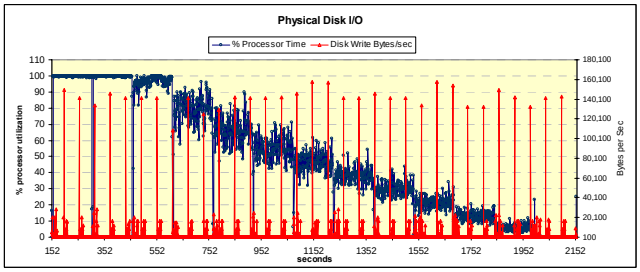

Fig 10 –Disk I/O

I. *Basic system events*

Figure 11 below shows interrupts rates of ~700 per second at all load levels including active idle. Context switches are ~800 /sec at higher utilization levels and decline at lower utilization while dropping to ~400 at active idle. These events are OS and platform dependent. Since these events are showing strange patterns, more investigation is needed.

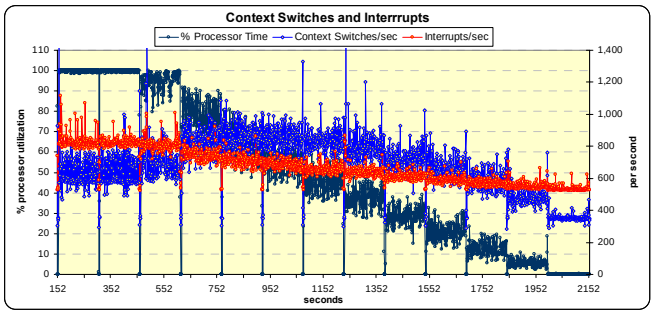

Fig 11 –Basic system events

J. *Impact of JVM optimizations*

Selection of JVM options can have significant impact on performance.

In this experiment, we compared "no options" to the set of "best known JVM options". Figure 12 shows the difference in performance and power

When using no JVM options (default options), performance dropped by ~50% while power dropped by 0 to 3%. Please note that any findings from these experiments are dependent on the JVM and its options.

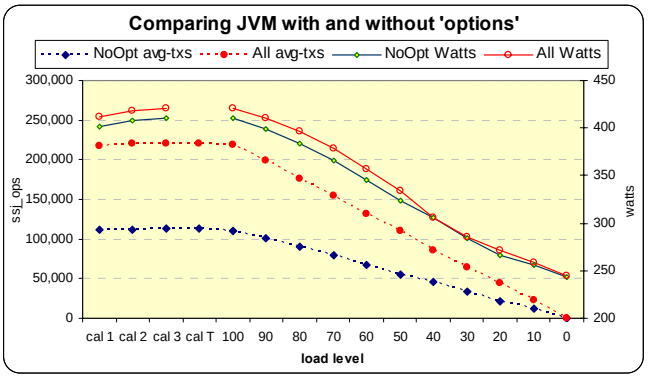

Fig 12 –Impact of JVM Options

JAVAOPTIONS_SSJ=""
(None, default heap and optimizations)

JAVAOPTIONS_SSJ="-Xms3000m -Xmx3000m -Xns2400m -XXaggressive –XxlazyUnlocking -Xgc:genpar -XXcallprofiling -XXlargePages -XXtlasize:min=12k,preferred=1024k"

K. *Processor scaling*

Figure 13 shows that when ssj_ops are plotted on the x-axis, the additional capacity of Quad Core Intel Xeon 2.0GHz/2x4MB L2 compared to Dual Core Intel Xeon 2.0GHz/4MB L2 is clearly evident.

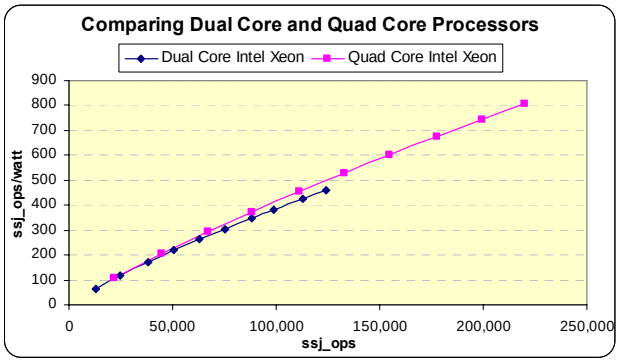

Fig 13 –Processor scaling

Table 3 below shows that when comparing these two types of processors, performance improves drastically - ssj_ops@100% increased by ~77% while power consumption@100% increases by only ~1%.

Dual Core to Quad Core scaling (Intel Xeon processors)	**% increase**
ssj_ops@100%	77%
Power@100%	1%

Table 3 – Processor scaling

L. *Frequency scaling*

To view the impact of frequency scaling, in Figure 14 below, we compared Quad Core Intel® Xeon® (2x6MB L2) running at 2.0GHz and 3.0GHz respectively.

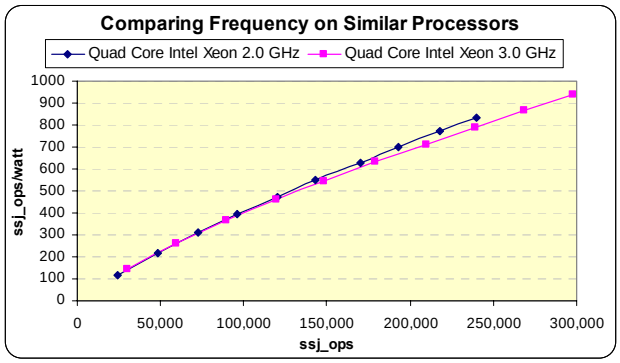

Fig 14 –Frequency scaling

Table 3 shows that for 2.0GHz to 3.0GHz Quad Core Intel Xeon / 2x6MB L2, ssj_ops@100% improves by ~24% while power consumption@100% increases by ~10%. Overall ssj_ops/Watt improves by ~76%.

Frequency Scaling (Intel Xeon Quad Core processors)	**% increase**
2.0 GHz to 3.0 GHz	50%
ssj_ops@100%	24%
Power@100%	10%

Table 3 – Frequency scaling

M. Platform generation scaling

New generation platforms almost always deliver more performance, consume less power and exhibit overall better energy efficiency. The chart below shows three generations of platforms from 2005, 2006 and 2008. Figure 15 below compares the results from three SPECpower_ssj2008 benchmarks reports [14] found on the SPEC web site, described in more detail in Table 4 All dual processor platforms.

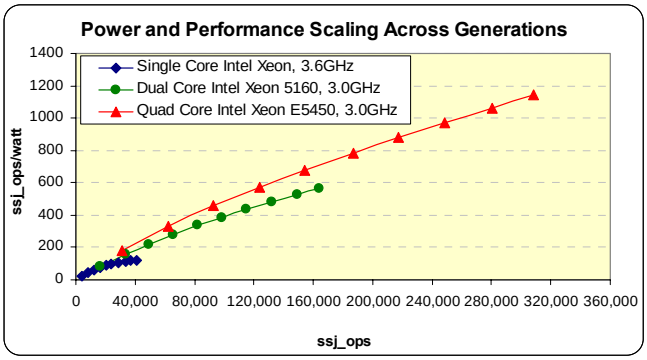

Fig 15 –Platform generation scaling

The SPECpower_ssj2008 benchmark results provide compelling evidence that the latest generation of Intel processors in commercial servers are improving efficiency. With a 650% increase in performance, the latest generation platforms (with Intel Xeon E5450 processors) consumes 20% less power, an overall improvement of 700% in overall ssj_ops/watt (from last row of Table 4 below).

Processor	Performance ssj_ops@100%	Power(watts) @100%	Overall ssj_ops/watt
Single Core Intel Xeon, 3.6GHz	40,852	336	87
Dual Core Intel Xeon 5160, 3.0GHz	163,768	291	338
Quad Core Intel Xeon E5450, 3.0GHz	308,022	269	698
Improvement	**654%**	**-20%**	**702%**

Table 4 – Platform generation scaling

VII. System Configuration Considerations

The SPECpower_ssj2008 benchmark metrics have two primary components: performance (ssj_ops) and power consumption (average Watts). In this section, software and hardware choices are listed that may impact performance, power or both.

A. Performance Factors

The following factors can have a significant impact on performance with unknown impact to power consumption.

Java Virtual Machine (JVM)

Different JVMs will deliver different performance.

JVM Parameters

A JVM can run by default (no options) but very likely will not deliver optimal performance. To find parameters for optimal performance, one can search published results at SPEC website or otherwise will need to find their own best tuning parameters for a JVM.

Multiple SSJ instances

If a system has large number of logical cores, often increasing the number of SSJ instances with each JVM instance at no more than ~8 warehouse threads results in better performance.

Affinity of SSJ instances

When running multiple SSJ instances, affinitizing them to shared caches or each socket or each NUMA node results in better performance.

HW and OS settings

Some HW settings like enabling or disabling features in BIOS or OS settings and use of large pages can result in better performance.

B. Power Factors

The following factors can impact power consumption significantly while minimally impacting performance. They are provided here for awareness purposes. Systems vary widely with options; consult the manufacturer's documentation.

1. BIOS Power management options

Many systems provide BIOS options for power management. The best choice of options depends on your priorities. It is wise to check

the BIOS options since the best setting may or many not be enabled by default.

Twp such power management options on Intel® processor based systems is called “Intel EIST”. Another is “C states” or C1 or C1E. Both should be enabled if your objective is to reduce power consumption.

2. Fan speed control in BIOS

Fans consume significant amount of power. Selecting optimal settings for fan speed control, when available, can reduce power consumption without performance impact. In some cases if fan speed is drastically reduced, it could lead to lower performance due to system level thermal throttling.

3. OS Power Management

Most operating systems have some power management settings. With Microsoft Windows Server 2003 for example, choosing “power options” from the control panel and then the option “balanced power and server performance” will conserver power without severe impact on performance.

4. Power Supplies (PSUs)

Many systems are ordered with optional redundant power supplies. Reducing the number of power supplies (without going below the minimum needed) will result in lower power consumption.

C. Memory Size and Performance

The most important factor in this category is total system RAM size and configuration. The platform configuration is the primary determinant of power consumption. Memory configuration can have following impact:

As RAM size is increased, both performance and power consumption will increase. Performance will increase (with associated heap size adjustments) to some limit up to the optimal amount of RAM. .

If RAM size is beyond the optimal size, there may be no measurable increase in performance but power consumption will increase with the number (and type of) DIMMs.

RAM configuration or slot placement can have an impact on performance if the platform supports more than one memory channel and memory interleaving which can improve performance. Consult system documentation.

VIII. Conclusions

The SPECpower_ssj2008 benchmark and the associated Full Disclosure Reports present an unprecedented amount of data on the power consumption and performance of server systems across the graduated load levels.

The benchmark framework, with the power data capture from the Power and Temperature Daemon combined with the OS counters collection daemon, with information captured by the logging capability in CCS and SSJ makes this benchmark a powerful and capable toolset for new areas of behavioral data collection exposing new fields of systems analysis.

Based on the information presented in this paper, we observe that most system resource utilizations are following the expected patterns. Processor Utilization follows the load line for Intel Core 2 based platforms (note that this is architecture dependent and CPU utilization is no part of the benchmark).

Power consumption tracks the transaction load. % time in C1 state is inversely proportional to processor utilization at each load level. When the min and max heap sizes are the same, memory committed is constant across load line. Disk I/O has regular bursts of ~140K byte writes with overall average of ~3.3K bytes/sec for all load levels while disk reads are none. Network I/O is ~1.5K bytes/sec and is almost constant across load line. The basic OS events interrupts and context switches/sec have some unique behavior which requires further investigation.

Experiments using different JVM options, processor scaling, frequency scaling and platform generation scaling show that primary metric for SPECpower_ssj2008 and associated data fairly reflect configurations and OS settings for performance, power and overall ssj_ops/Watt.

All these results are specific to the platform and OS measured. We expect similar data from different architectures and OS(s) will be very valuable. This initial characterization is just a first look and more measurements are required to continue in-depth characterization.

In summary, we are just getting started!

Acknowledgment

Special appreciation to Christopher B. Jorgensen, a graduate student intern from Portland State University, for his work setting up, running and validating multiple series of measurements, collecting the bulk of the data shown in this paper.

Legal

Performance Data as of 30 January 2008.

Performance tests and ratings are measured using specific computer systems and/or components and reflect the approximate performance of Intel products as measured by those tests. Any difference in system hardware or software design or configuration may affect actual performance. Buyers should consult other sources of information to evaluate the performance of systems or components they are considering purchasing. For more information on performance tests and on the performance of Intel products, visit Intel Performance Benchmark Limitations (http://www.intel.com/performance/resources/limits.htm)

References

[1] SPEC, http://www.spec.org
[2] SPECpower_ssj2008 http://www.spec.org/power_ssj2008/
[3] SPECpower_ssj2008 User Guide http://www.spec.org/power_ssj2008/docs/SPECpower_ssj2008-User_Guide.pdf
[4] SPECpower_ssj2008 Hardware Setup Guide http://www.spec.org/power_ssj2008/docs/SPECpower_ssj2008-Hardware_Setup_Guide.pdf
[5] SPECpower_ssj2008 FAQ http://www.spec.org/power_ssj2008/docs/SPECpower_ssj2008-FAQ.html
[6] SPECpower_ssj2008 Run and Reporting rules http://www.spec.org/power_ssj2008/docs/SPECpower_ssj2008-Run_Reporting_Rules.pdf
[7] SPECpower_ssj2008 Result File Filed Description http://www.spec.org/power_ssj2008/docs/SPECpower_ssj2008-Result_File_Fields.html
[8] SPECpower_ssj2008 Design Overview http://www.spec.org/power_ssj2008/docs/SPECpower_ssj2008-Design_overview.pdf
[9] SPECpower_ssj2008 CCS Design http://www.spec.org/power_ssj2008/docs/SPECpower_ssj2008-Design_ccs.pdf
[10] SPECpower_ssj2008 PTD Design http://www.spec.org/power_ssj2008/docs/SPECpower_ssj2008-Design_ptd.pdf
[11] SPECpower_ssj2008 SSJ Design http://www.spec.org/power_ssj2008/docs/SPECpower_ssj2008-Design_ssj.pdf
[12] SPEC Power and Performance Methodology http://www.spec.org/power_ssj2008/docs/SPECpower-Methodology.pdf
[13] SPECpower_ssj2008 Release http://www.spec.org/power_ssj2008/press/SPECpower_ssj2008-Press%20Release.html
[14] SPECpower_ssj2008 publications http://www.spec.org/power_ssj2008/results/res2007q4/power_ssj2008-20071129-00015.html http://www.spec.org/power_ssj2008/results/res2007q4/power_ssj2008-20071129-00016.html http://www.spec.org/power_ssj2008/results/res2007q4/power_ssj2008-20071129-00023.html http://www.spec.org/power_ssj2008/results/res2007q4/power_ssj2008-20071129-00017.html
[15] Morin et al., IISWC 2005 “A multi-level comparative performance characterization of SPECjbb2005 versus SPECjbb2000” http://ieeexplore.ieee.org/iel5/10230/32621/01526002.pdf
[16] BEA JRockit 6 P27.4.0 JDK http://dev2dev.bea.com/jrockit/releaseupdate.html
[17] BEA JRockit Command Line Reference http://edocs.bea.com/jrockit/jrdocs/refman/index.html