Chauffeur: A framework for measuring Energy Efficiency of Servers
Masters Project

Jeremy Arnold
University of Minnesota
arnoldje@us.ibm.com

December 16, 2013

Abstract

Energy efficiency is an important consideration for servers, bringing about a need for benchmarks to fairly measure energy efficiency. These benchmarks must be relevant, reproducible, fair, verifiable, and easy to use. This paper describes these characteristics in detail and provides an assessment of existing energy efficiency benchmarks based on these criteria. Next it describes Chauffeur, a new framework developed to help benchmarks achieve these goals with relatively little effort. SPEC's Server Efficiency Rating Tool (SERT) builds upon the Chauffeur framework to deliver more meaningful energy efficiency data than is provided by the current generation of benchmarks and workloads. Experimental SERT data is used to highlight some of the features of Chauffeur and characteristics of SERT.

1 Introduction

Energy consumption of data centers has become an important issue for many companies, resulting in increased focus on the energy efficiency of servers [1].

There are many challenges in evaluating the energy efficiency of computer servers. Some traditional performance-oriented benchmarks have been updated to include energy efficiency metrics; while these have some value, the results of these measurements are not indicative of energy consumption under real-world conditions. Early benchmarks like SPECpower_ssj2008 that were designed specifically for measuring energy efficiency have their own limitations.

I have made the following contributions in the context of energy efficiency measurements for servers: 1) a detailed set of criteria that can be used to assess the quality of computer benchmarks, and energy efficiency benchmarks in particular, 2) development of the Chauffeur framework for creating high-quality energy efficiency benchmarks.

This paper begins by describing the need for new energy efficiency benchmarks. Next it outlines the design criteria that must be considered when developing energy efficiency benchmarks, and evaluates how current benchmarks fare against these criteria. Then it introduces Chauffeur, a framework designed to support the implementation of workloads that overcome the limitations of these existing benchmarks, and the Server Efficiency Rating Tool (SERT) implemented by the Standard Performance Evaluation Corporation (SPEC) which builds on this framework. Finally, SERT results are used to demonstrate how Chauffeur and SERT meet its design goals to produce a more complete measure of server energy efficiency.

2 The Need for Energy Efficiency Benchmarks

Computer performance benchmarks have been widely used in both industry and academia for decades. In recent years, energy efficiency metrics have been added to existing performance benchmarks, and new benchmarks have been created specifically to measure energy efficiency. This section explores the reasons why energy efficiency benchmarks are important, and why additional measures of energy efficiency are still needed.

2.1 Significance of Energy Efficiency

In the past, the acquisition cost of a server was the most significant component of its total cost of ownership. Over time, maintenance and operational costs have become more significant, and in recent years power and cooling costs have grown to be a critical factor in the equation, in some cases dominating the cost of the server itself [2].

Government agencies such as the United States Environmental Protection Agency (US EPA) have taken note of these trends and taken a more active role in establishing criteria for energy efficient servers. The US EPA has established an ENERGY STAR program for servers with basic requirements, and a future version of this program is expected to introduce more comprehensive criteria including measurements of active mode efficiency [3, 4]. Other governments are also in the process of developing similar programs. It is likely that in the future, both government and commercial entities will include ENERGY STAR or similar qualifications as part of their procurement processes.

System vendors have also noticed this trend and given increased focus to energy consumption as a criteria in the design of systems and their components. While this generally hasn't resulted in a decrease in peak power consumption of typical servers [5], it has slowed the pace of growth, resulting in improved efficiency. Hardware and operating system vendors have also improved power management technologies to the point where they can be enabled by default without impacting the perceived performance of the system [6].

2.2 Low Utilization Servers

Servers are typically sized to handle peak loads, but spend much of their time running at significantly lower utilizations [7, 8]. Even when operating at "peak" capacity, some resources may not be at full utilization. External storage can also consume significant amounts of power, sometimes exceeding the server's power consumption. Even if a server is at full capacity, the storage it is using may not be [9].

Different applications may have significantly different patterns of utilization. Servers used for corporate email may be heavily utilized during the daytime, moderately utilized during the evening, and have low utilization overnight. Web servers used primarily for entertainment may have their heaviest usage in the evening hours, with moderate utilization during the day and lower overnight. Servers used for payroll processing may be used almost exclusively during the evening and overnight hours during certain days of the month and virtually unused at other times.

Server consolidation and virtualization are commonly used to increase utilization of servers and reduce the amount of idle capacity, under the assumption that the peak times for one workload match the low utilization periods for another workload. However, some workloads are not good candidates for virtualization, and even a fully virtualized environment typically needs spare capacity to handle peak loads [10].

As a result, a typical server is operating at less than full capacity most of the time, and often far less. Power management technology can be used to take advantage of these periods to reduce power consumption when the full resources of the server are not needed. Common techniques include dynamic voltage and frequency scaling (DVFS) and putting idle processors in a low power sleep state until they are needed [8].

2.3 Estimating Power Consumption

Benchmarks are a tool for providing information and comparing alternatives, but customers aren't primarily interested in a system's benchmark results; instead, they want to understand how the system will behave in their own environment. In this sense, the benchmark is a proxy for the customer's application. The accuracy of an energy efficiency assessment based on a benchmark result depends on how closely the benchmark environment matches the actual application environment.

Detailed descriptions of the hardware, software, and tuning used in a benchmark result are critical for assessing whether a particular benchmark test system matches the environment of interest to the customer.

In many cases, a benchmark publication will not be available for the exact system configuration the customer is interested in. Most benchmarks stress a particular component of the system. In many cases this is the CPU, and minimal disk is required; a counter-example is TPC-C which uses far more disks than are required for typical server applications. Since vendors will optimize their hardware configuration to match the requirements of the benchmark, the resulting system configuration is often not typical of production servers. For performance benchmarks, this isn't generally a big issue – for example, a company might base their purchasing decision in part on SPECcpu results, but understand that their application will require more disks; these additional disks would not be expected to hurt the performance of the CPUs. With energy efficiency, every component of the system may contribute to the power consumption. This makes it difficult to estimate the impact of additional components on the energy efficiency of the system. Server vendors such as IBM[1] and HP[2] have provided tools to estimate

[1] http://www-947.ibm.com/systems/support/tools/estimator/energy/index.html

[2] http://h18004.www1.hp.com/products/solutions/power/advisor-online/HPPowerAdvisor.html

power requirements for specific configurations.

Using a suite of multiple workloads rather than a single application can make the results more representative of a variety of application types, and therefore relevant to a wider spectrum of customers. Requiring common tuning to be used for multiple applications also helps to limit "super-tuning" that may make the results less meaningful for real servers.

2.4 Inadequacy of Performance Benchmarks

Traditional performance-oriented benchmarks are ill-suited to measuring energy efficiency. While there is some value in measuring and reporting power consumption data for a performance benchmark, this will, at best, produce a measure of the energy efficiency at peak utilization. Such a value would provide little information about the efficiency at lower utilizations.

Additionally measuring the power at "Active Idle" (where a server is ready to accept work but no work is actually in progress) is an improvement, because these two values provide upper and lower bounds on the power consumption of the system. However, this still does not provide information about how the power consumption changes between these two points. A linear extrapolation between the two points used to provide a reasonably good model of the power consumption of a server [11], but in recent years the power curve has become nonlinear for many servers [6, 12, 13].

Attempts to retroactively calculate energy efficiency metrics for published benchmark results can be particularly problematic. When measured power data is not available, the power consumption must be estimated. This is typically done using the nameplate power rating of the system (or some subset of its components) along with some assumptions. However, the nameplate power rating for a server is a conservative maximum value intended for safely powering the system. Under real conditions, the system is unlikely to draw this much power even at peak load. The nameplate rating often overstates the actual peak requirements by a factor of two or more [11, 14]. Basing energy efficiency estimates on these nameplate values could significantly skew the results.

Despite these limitations, estimated power measurements can be used to draw historical trends where measured data is not available. Rivoire, Shah, Ranganathan, and Kozyrakis estimate energy efficiency of winners of the sort benchmarks for the ten years prior to their introduction of JouleSort [15]. Poess and Nambiar analyze the trends of estimated power consumption of past TPC-C results [4]. In this case, the accuracy of the data may be improved by the fact that TPC-C includes a price/performance metric, leading results to use carefully balanced systems to avoid any extra components that would increase cost without improving performance.

Another challenge is that performance benchmarks are highly tuned environments; vendors use a variety of tuning options to improve their results. In some cases these tuning options may increase power consumption disproportionately with the performance gain. If energy efficiency is not a standard metric, the submitter of the result will not have to balance performance with energy efficiency, and will likely tune their system for maximum performance at the expense of efficiency. This may not be a realistic measure of the energy efficiency a customer would experience with their own application.

This is also an argument against making power measurements an optional component of the benchmark. This will tend to result in results that have power data being tuned for energy efficiency, while results that do not include power data are tuned for performance, with no meaningful way to compare between the two. Furthermore, this division may scare vendors away from publishing energy data since they may compare unfavorably with performance-tuned results. This

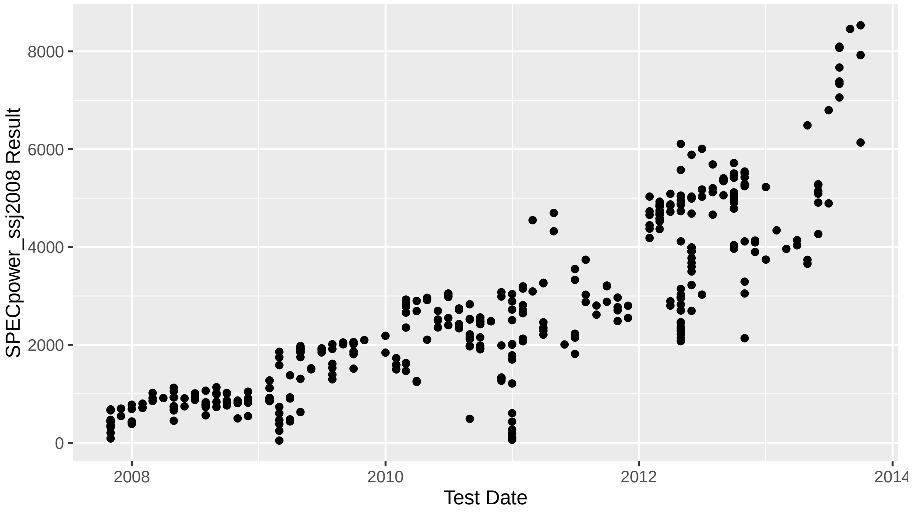


Figure 1: SPECpower_ssj2008 scores by Test Date, including all single-node results published at http://www.spec.org/power_ssj2008/results as of November 19, 2013

topic will be explored further in section 6.2.

2.5 Energy Efficiency Benchmarks Show Improvements

The current set of energy efficiency benchmarks (described in more detail in section 4) already demonstrate significant improvements in the energy efficiency of servers. While it is difficult to prove causation, it is likely that some portion of these improvements are due to the existence of these benchmarks, which force vendors to improve energy efficiency in order to keep up with their competitors.

SPECpower_ssj2008 has the greatest number of published results, with 461 results published as of November 19, 2013. This large body of data allows us to visualize trends in efficiency over time. Figure 1 shows all of the published single-node SPECpower_ssj2008 results by test date, showing an improvement of more than 12 times between the most efficient result published in 2007 compared to the most efficient result published in 2013.

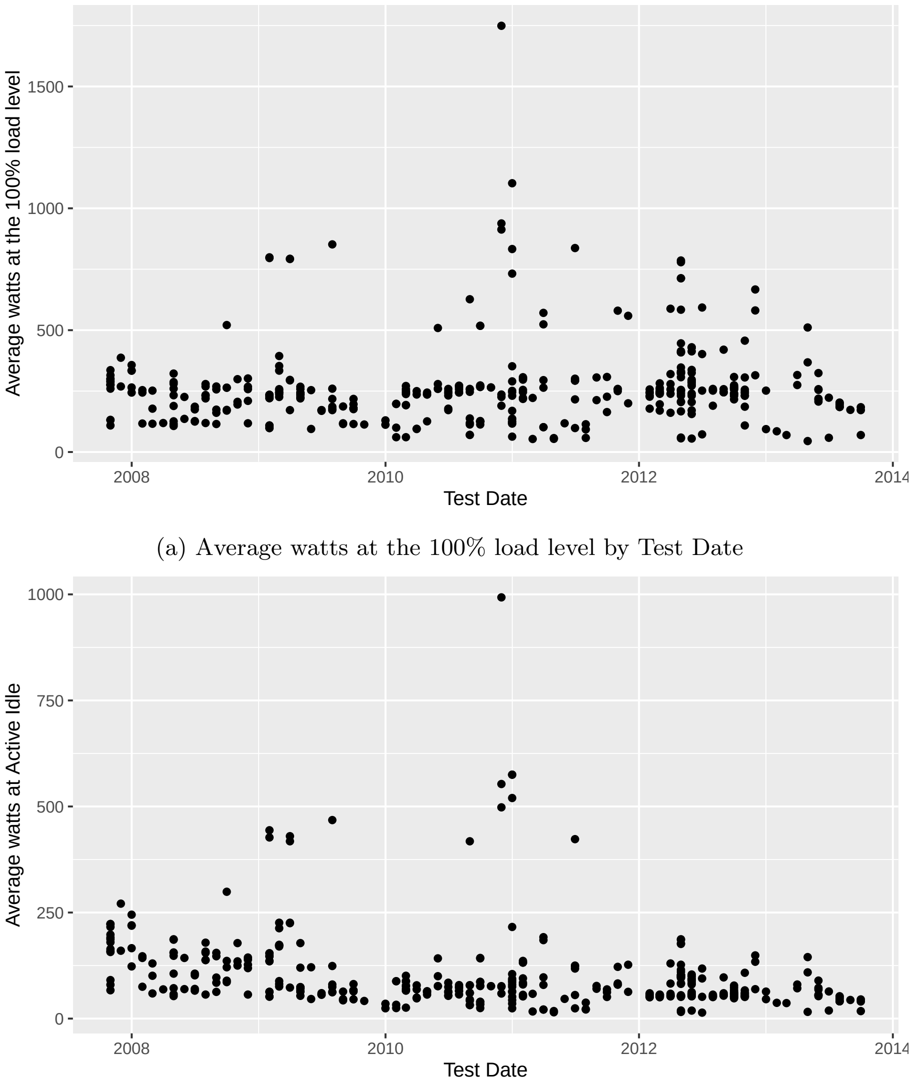
The graph of results tends to plateau at certain points, leading some to conclude that efficiency has reached its peak [16]. In reality, jumps in the results tend to occur with new processor releases, followed by periods of relatively little improvement as vendors make incremental improvements to their power management and tuning. For example, the jump in early 2009 was due to results using the Intel Xeon X5570 processor, the improvement in early 2010 corresponds with the introduction of the Intel Xeon X5670, and the increase at the beginning of 2012 came with the Intel Xeon E5-2660.

Figure 2(a) shows the average watts at the 100% load level by test date for single-node SPECpower_ssj2008 results. While there has been some reduction in the power consumption over time, the overall trend has been fairly flat – thus, newer servers generally don't consume less power than their predecessors at peak utilization, but they deliver improved performance without increasing power consumption.

The trend for power consumption at active idle, shown in Figure 2(b) shows that idle power has declined over time, indicating that power management technologies have improved to better reduce power consumption when a server isn't busy. This is further illustrated in Figure 3 which shows the idle-to-peak power ratio (IPR) for these results [12, 13]. The IPR values are trending downward over time, indicating a greater reduction in power from peak to idle in recent results.

2.6 Current Efficiency Benchmarks Have Shortcomings

While current energy efficiency workloads have been successful in demonstrating the relative efficiency of servers (and perhaps at driving improvements in efficiency), there is still ample need for further energy efficiency benchmarks. Specific limitations of existing energy efficiency benchmarks will be discussed in



(a) Average watts at the 100% load level by Test Date

(b) Average watts at active idle by test date

Figure 2: SPECpower_ssj2008 Average watts at the 100% load level and active idle by test date, including all single-node results published at http://www.spec.org/power_ssj2008/results as of November 19, 2013

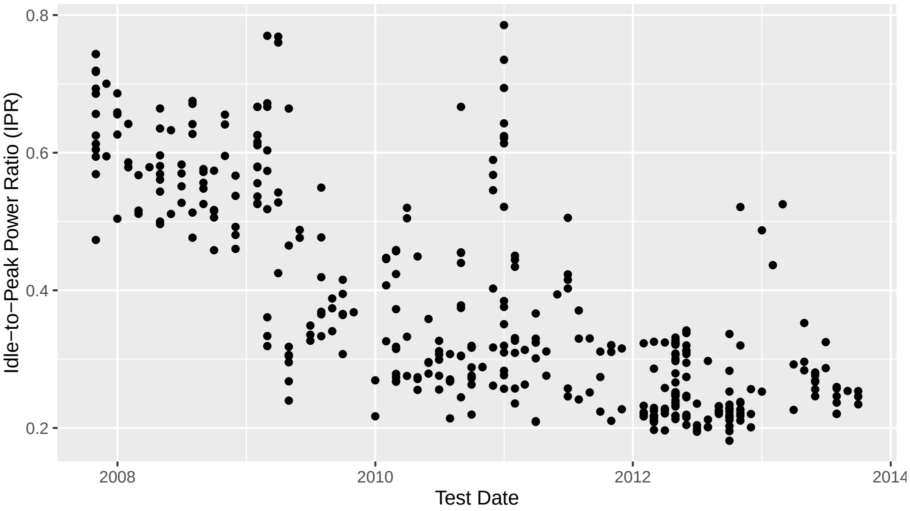


Figure 3: SPECpower_ssj2008 Idle-to-Peak Power Ratio (IPR) by Test Date, including all single-node results published at http://www.spec.org/power_ssj2008/results as of November 19, 2013

Section 4. This section addresses more general areas where additional benchmarks are needed.

2.6.1 Broader Measures of Efficiency

Different benchmarks suit different purposes. Fanara, Haines, and Howard categorize benchmarks into three levels of approximation: 1) a Generalized Benchmark gives a broad measure of a system's performance and efficiency characteristics, 2) an Application Proxy Benchmark provides more accurate data for a specific application type, and 3) Real Application Data uses the intended application directly to assess a system according to its planned usage. Each of these levels provides additional accuracy at the expense of generality [3].

Most current energy efficiency benchmarks fall into the second category. They provide efficiency data for a particular type of workload, but do not give a good general picture of the broader efficiency of the server across multiple workloads with different characteristics.

2.6.2 Representative Application-specific Benchmarks

In addition to broad measures of efficiency, there is a need for more representative application-specific benchmarks. Most of the existing benchmarks focus on specific hardware components (such as processors) without exercising a balanced system. Real servers tend to require some mixture of CPU, memory, storage, and network capabilities, often along with requirements for redundant components or other reliability features. Benchmarks that do not exercise these features will tend to penalize these real systems. No single benchmark can address the full variety of workloads that customers will run on these systems; a broader set of benchmarks allows customers to focus on the results that are most meaningful for their applications.

2.6.3 Dynamic Environments

Current benchmarks that measure power consumption at multiple utilizations do so in a controlled, orderly fashion. For example, SPECpower_ssj2008 runs at 100% of a server's capacity, then 90%, then 80% and so forth until it reaches active idle; at each load level it runs for 30 seconds to allow the system to reach steady state before measuring power and performance for 240 seconds. This methodology allows for accurate and repeatable measurements, but it is not particularly representative of many real-world environments, which tend to have more dynamic changes in utilization. Such environments require power management to act more aggressively to reduce power consumption when possible and allow the server to operate at full performance when necessary.

More generally, a benchmark intended to measure the efficacy of power management technologies should establish quality of service (QoS) constraints to ensure that transactions are processed in a timely manner at whatever utilization the server is running at. People expect that the server will execute transactions at least as quickly when the server is at a low utilization as when it is heavily utilized. If power management is too aggressive at reducing energy at low utilizations (e.g. putting the processor into a deep sleep state), it may be slow to “wake up” again when work enters the system. In particular, benchmarks that measure “Active Idle” should measure the response time of the first transactions to be processed after the Active Idle period has ended to ensure that these transactions are still processed in a reasonable amount of time.

2.6.4 Energy Efficiency of Virtualized Systems

Virtualization has become a popular way of managing server applications. There are many advantages to running applications in virtual machines, including the ability to consolidate applications on a smaller number of servers that can then run at a higher utilization. These environments have characteristics that are distinct from physical servers, and energy efficiency must be measured differently. For this reason, while SPECpower_ssj2008 is technically capable of running in these environments, run rules explicitly prohibit this for compliant runs.

When virtualization is used to balance a single workload across multiple virtual servers (possibly running on multiple physical servers), the infrastructure will typically scale up to meet demand by provisioning additional virtual servers, then scale back during periods of low utilization by shutting down some of these servers. This is a much different model than scaling for physical servers, where the utilization on that server will change with the application load. An energy efficiency benchmark for this type of virtualized environment would need to scale in a similar manner in order to produce an accurate measurement. Such a benchmark would demonstrate the ability of the virtualization infrastructure to make intelligent decisions about when to add or remove virtual servers and what physical servers new virtual machines should be deployed to.

In other cases, virtualization is used to deploy multiple heterogeneous applications to one or more physical servers. In this case, the different applications may each scale differently. Often, it is expected that periods of high utilization for one application may coincide with periods of low utilization for another application. In this type of environment, one of the goals of the virtualization infrastructure is to intelligently shift server resources among virtual machines to optimize performance according to the load for each virtual machine. An energy efficiency benchmark for this type of environment would be seeking to demonstrate the ability of the infrastructure to do this while maximizing energy efficiency.

3 Design Criteria for Energy Efficiency Benchmarks

Benchmark designers have to balance several, often conflicting, criteria in order to be successful. Different benchmarks balance these criteria differently, resulting in disparate strengths and weaknesses. Since no single benchmark can be strong in all of these areas, there will always be a need for multiple benchmarks [17]. But just because a benchmark is new doesn't mean that it is better than existing benchmarks [18]. It is important to understand the characteristics of a benchmark and determine whether or not it is applicable for a particular situation. When developing a new benchmark, the goals of the benchmark should be defined so that choices between competing design criteria can be made in accordance with those goals to achieve the desired balance.

Several researchers and industry participants have listed various desirable characteristics of benchmarks [17, 18, 19, 20, 21, 22, 23]. The contents of the lists vary based on the perspective of the author and their choice of terminology and grouping of characteristics, but most of the concepts are similar. The key characteristics can be organized in the following groups, which will be discussed in more detail in the next sections:

Relevance How closely the benchmark behavior correlates to behaviors that are of interest to consumers of the results

Reproducibility The ability to consistently produce similar results when the benchmark is run with the same test configuration

Fairness Allowing different test configurations to compete on their merits without artificial limitations

Verifiability Providing confidence that a benchmark result is accurate

Usability Avoiding roadblocks for users to run the benchmark in their test environments

Energy efficiency benchmarks are subject to these same criteria, but each category includes additional issues that are specific to measuring energy efficiency.

Existing literature regarding the design of benchmark applications is lacking, so the material in this section covers the criteria both for benchmarks in general as well as characteristics specific to energy efficiency benchmarks.

3.1 Relevance

"Relevance" is perhaps the most important characteristic of a benchmark. Even if the workload was perfect in every other regard, it will be of minimal use if it doesn't provide relevant information to its consumers. Yet relevance is as much a characteristic of the use of a benchmark as it is of the benchmark itself; benchmarks may be highly relevant for some scenarios and of minimal relevance for others. For the consumer of benchmark results, an assessment of a benchmark's relevance must be made in context of the planned use of those results. For the benchmark designer, relevance means determining the intended use of the benchmark and then designing the benchmark to be relevant for those areas [24].

A general assessment of the relevance of a benchmark or workload involves two dimensions: the breadth of its applicability, and the degree to which the workload is relevant in that area. For example, an XML parsing benchmark may be highly relevant as a measure of XML parsing performance, somewhat relevant as a measure of enterprise server application performance, and not at all relevant for graphics performance of 3D games. Conversely, a suite of CPU benchmarks such as SPEC CPU2006 may be moderately relevant for a wide range of computing environments. The behavior illustrated in these examples is generally true: benchmarks that are designed to be highly relevant in a specific area tend to have narrow applicability, while benchmarks that attempt to be applicable to a broader spectrum of uses tend to be less meaningful for any particular scenario [18].

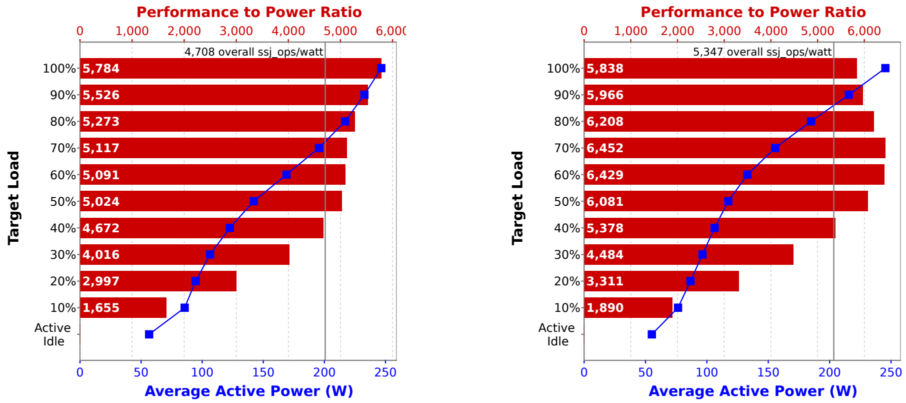
Scalability is an important aspect of relevance, particularly for server benchmarks. The most relevant benchmarks will be multi-process and/or multi-threaded in order to be able to take advantage of the full resources of the server [17]. Achieving scalability in any application is difficult; for a benchmark, the challenges are often even greater because the benchmark is expected to run on a wide variety of systems with significant differences in available resources. The benchmark designer must also strike a careful balance between avoiding artificial limits to scaling and behaving like real applications (which often have scalability issues of their own).

Energy efficiency benchmarks carry the same set of requirements for relevancy, but also include new challenges. Since most servers run at low utilizations much of the time [11, 24, 25], energy usage should be measured at different load levels. For many modern servers, energy consumption at low utilizations can be significantly lower than at peak utilization. The amount of savings at lower utilizations can vary depending on the hardware, operating system, and tuning used.

This is illustrated in the SPECpower_ssj2008 results shown in Figure 4. The result shown in Figure 4(a) achieved 1,426,130 ssj_ops and 247W @100% target load, and 56.6W at Active Idle. The result in Figure 4(b) had very similar numbers: 1,432,623 ssj_ops and 245W @100% target load, and 55.2W at Active Idle. Yet the first result had a SPECpower_ssj2008 metric of 4,708 overall ssj_ops/watt, while the second result was over 13% higher at 5,347 overall ssj_ops/watt. The difference between these results is that the second system was able to reduce power more dramatically at intermediate load levels, achieving peak efficiency at the 70% target load. For a server that spends much of its time running at low utilizations, the practical difference between these two systems could be significant.[3]

Run rules for energy efficiency benchmarks need to place limits on environmental factors (such as ambient temperature) to ensure that the results are obtained in realistic environments and will be applicable to real systems. Energy efficiency benchmarks also need to define which components of a system need to be included in power measurements. For example, SPECpower_ssj2008 does not require a video monitor to be included in power measurements for servers, but does require a video monitor to be included for systems classified as "personal

[3] Results obtained from http://www.spec.org/power_ssj2008/results/res2012q1/power_ssj2008-20120305-00428.html and http://www.spec.org/power_ssj2008/results/res2012q4/power_ssj2008-20120918-00544.html on 28 March 2013. SPEC® and the benchmark name SPECpower_ssj® are registered trademarks of the Standard Performance Evaluation Corporation. For more information about SPECpower, see http://www.spec.org/power_ssj2008/.



(a) Configuration 1

(b) Configuration 2

Figure 4: Sample SPECpower_ssj2008 Results

systems". These distinctions are less important for traditional performance benchmarks since these components don't contribute to the performance of the system. For some benchmarks it is appropriate to measure power separately for certain sub-components of the system under test; for example, SPECvirt_sc2010 allows power to be measured either for the server only or for both server and external storage devices.

Energy efficiency benchmarks benefit from support for runs on multiple systems. Many performance benchmarks are limited to running on a single system, while others support or require multiple systems for different tiers of the application, such as an application server and a database running on different physical systems. Some benchmarks also allow the benchmark to run across clusters of systems, but this can make it trivial to expand the cluster to get a higher result, turning the benchmark into a contest of which vendor is willing to spend the most money on their test configuration.

In energy efficiency benchmarks, however, support for multi-system configurations allows the benchmark to support systems such as blades that share power and cooling infrastructure. In these environments, it can be inaccurate or impractical to measure energy usage for a single system in isolation, but power consumption can be measured for a group of systems.

One way to increase the breadth of relevance is to support user-configurable parameters that change the behavior of the benchmark. This allows the workload to represent a larger class of applications than a single benchmark could otherwise achieve [17]. This typically isn't appropriate for industry standard benchmarks (where it is important that published results are comparable) but is useful for research-oriented workloads where settings can be chosen in order to perform some experiment. Even some industry standard benchmarks, like SPECpower_ssj2008, include user-configurable parameters that may be used for non-compliant runs.

3.2 Reproducibility

Reproducibility is the capability of the benchmark to produce the same results consistently for a particular test environment. It includes both run-to-run consistency and the ability for another tester to independently reproduce the results on another system.

Ideally, a benchmark result is a function of the hardware and software configuration, so that the benchmark is a measure of the performance of that environment; if this were the case, the benchmark would have perfect consistency. In reality, the complexity inherent in a modern computer system introduces significant variability in the performance of an application. This variability is introduced by several factors, including things such as the timing of thread scheduling, dynamic compilation, physical disk layout, network contention, and user interaction with the system during the run[18, 26]. Energy efficiency benchmarks often have additional sources of variability due to power management technologies dynamically making changes to system performance and temperature changes affecting power consumption.

Benchmarks can address this run-to-run variability by running for long enough periods of time to include representative samples of these variable behaviors. Some benchmarks require submission of multiple runs with scores that are near each other as evidence of consistency. Benchmarks also tend to run at steady state, unlike more typical applications which have variations in load due to factors such as the usage patterns of users.

The ability to reproduce results in another test environment is largely tied to the ability to build an equivalent environment. Industry standard benchmarks require results submissions to include a description of the test environment, typically including both hardware and software components as well as configuration options. Similarly, published research that includes benchmark results generally includes a description of the test environment that produced those results. However, in both of these cases, the description may not provide enough detail for an independent tester to be able to assemble an equivalent environment.

Hardware must be described in sufficient detail for another person to obtain identical hardware. Software versions must be stated so that it is possible to use the same versions when reproducing the result. Tuning and configuration options must be documented for firmware, operating system, and application software so that the same options can be used when re-running the test. Unfortunately, much of this information cannot be automatically obtained in a reliable way, so it is largely up to the tester to provide complete and accurate details. TPC benchmarks require a certified auditor to audit results and ensure compliance with reporting requirements. SPEC uses a combination of automatic validation and committee review to establish compliance.

Complete descriptions of the system configuration are especially important for energy efficiency benchmarks. Systems may include components that have little or no impact on performance but may consume energy. For example, the performance results of a CPU-focused benchmark may not be dependent on what video card is installed in the system, so the system description for such a benchmark might not include that detail. But if the same benchmark includes an energy metric, the choice of video card could be important since different cards may require different amounts of power; therefore, it may be necessary to disclose the video card in the system configuration details.

Even with a complete and accurate description of the system configuration, results may not be completely reproducible due to variations in power consumption of different physical parts [27, 28, 29].

Environmental constraints and other run rules can also improve reproducibility by putting limits on the test configuration. For example, the SPEC Power and Performance Methodology recommends a requirement that the minimum ambient temperature is no less than 20 degrees Celsius. Most data centers are kept at or above this range, so this allows results to be reproduced without introducing special cooling requirements.

Not all power analyzers are equivalent, and some may not be able to measure results with sufficient accuracy for a particular benchmark. The SPEC Power and Performance Methodology describes several requirements that analyzers must meet in order to produce acceptable results; SPEC has an acceptance process for verifying that analyzers meet these requirements and can therefore be used for benchmarks such as SPECpower_ssj2008.

A related issue is the calibration of power analyzers. These devices need to be calibrated periodically to ensure their continued accuracy. The SPEC Power and Performance methodology recommends that benchmarks require power analyzers to have been calibrated within the past year.

3.3 Fairness

Fairness ensures that systems can compete on their merits without artificial constraints. Because benchmarks always have some degree of artificiality, it is often necessary to place some constraints on test environments in order to avoid unrealistic configurations that take advantage of the simplistic nature of the benchmark.

Benchmark development requires compromises among multiple design goals; benchmarks developed by a consensus of experts is generally perceived as being more fair than a benchmark designed by a single company [20]. While "design by committee" may not be the most efficient way to develop an application, it does require that compromises are made in such a way that multiple interested parties are able to agree that the final benchmark is fair. As a result, benchmarks produced by organizations such as SPEC and the TPC (both of which are comprised by members from companies in the industry as well as academic institutions and other interested parties) are generally regarded as fair measures of performance.

Benchmarks require a variety of hardware and software components to provide an environment suitable for running the benchmark. It is often necessary to place restrictions on what components may be used. Careful attention must be placed on these restrictions to ensure that the benchmark remains fair.

Some restrictions must be made for technical reasons. For example, a benchmark implemented in Java requires a Java Virtual Machine (JVM) and an operating system and hardware that supports it. A benchmark that performs heavy disk IO may effectively require a certain number of disks to achieve acceptable IO rates, which would therefore limit the benchmark to hardware capable of supporting that number of disks.

Benchmark run rules often require hardware and software to meet some level of support or availability. While this restricts what components may be used, it is actually intended to promote fairness. Because benchmarks are by nature simplified applications, it is often possible to use simplified software to run them; this software may be quite fast because it lacks features that may be required by real applications. For example, enterprise servers typically require certain security features in their software which may not be directly excercised by benchmark applications; software that omitted these features may run faster than software that includes them, but this simplified software may not be usable for the customer base that the benchmark is targeted to. Rules regarding software support can be a particular challenge when using open source software, which is often supported primarily by the developer community rather than commercial support mechanisms.

Both of these situations require a careful balance. Placing too many or inappropriate limits on the configuration may disallow results that are relevant to some legitimate situations. Placing too few restrictions can pollute the pool of published results and, in some cases, reduce the number of relevant results because vendors can't compete with the "inappropriate" submissions.

Portability is an important aspect of fairness. Some benchmarks, such as TPC-C, only provide a specification and not an implementation of the benchmark, allowing vendors to implement the specification using whatever technologies are appropriate for their environment (as long as the implementation is compliant with the specification and other run rules). Other benchmarks, such as those from SPEC, provide an implementation that must be used. Achieving portability with benchmarks written in Java is relatively simple; for C and C++, it can be more difficult [23].

If the benchmark allows code to be recompiled, rules must be defined to state what compilation flags are allowed. SPEC CPU2006 defines Base results (with minimal allowed compilation flags) and Peak results (allowing the tester to use whatever compilation flags they would like). Similarly, Java benchmarks may put limits on what JVM command-line options may be used.

In some cases multiple implementations may be required to support different technologies. In this case, it may be necessary (as with SPECweb2009) for results with different implementations to be assigned to different categories so they cannot be compared with each other.

Benchmark run rules often include stipulations on how results may be used. These requirements are intended to promote fairness when results are published and compared, and often include provisions that certain basic information is included with every submission. For example, SPECpower_ssj2008 requires that if a comparison is made for the power consumption of two systems at the 50% target load level, the performance of each system at the 50% load level as well as the overall ssj_ops/watt value must also be stated.

SPEC has perhaps the most comprehensive fair use policy which further illustrates the types of fair use issues that benchmarks should consider when creating their run rules [30].

Energy efficiency benchmarks have fairness concerns similar to other benchmarks. Once again, having environmental constraints (such as a minimum temperature threshold) can help ensure that different results are comparable.

Optional energy efficiency metrics can create new challenges for fairness. When performance results are published without regard for energy efficiency, the environment is likely to be tuned for optimal performance even at the expense of a large increase in energy consumption. Results that include energy efficiency metrics are likely to take a more balanced approach, either optimizing for energy efficiency or making improvements to performance only when there isn't a disproportionate increase in energy consumption. Comparisons between results that have energy efficiency metrics and those that don't may not be accurate. Benchmarks that include optional energy efficiency metrics may need to prohibit or limit these comparisons.

3.4 Verifiability

Within the industry, benchmarks are typically run by vendors who have a vested interest in the results. In academia, results are subjected to peer review and interesting results will be repeated and built upon by other researchers. In both cases, it is important that benchmark results are verifiable so that the results can be deemed trustworthy.

Good benchmarks perform some amount of self-validation to ensure that the workload is running as expected, and that run rules are being followed. For example, a workload might include configuration options intended to allow researchers to change the behavior of the workload, but standard benchmarks typically limit these options to some set of compliant values which can be verified at runtime. Benchmarks may also perform some functional verification that the output of the test is correct; these tests could detect some cases where optimizations (e.g. experimental compiler options) are producing incorrect results.

Verifiability is simplified when configuration options are under the control of the benchmark, or when these details can be read by the benchmark. In this case, the benchmark knows the configuration and the details can be included with the results. Configuration details that must be documented by the user are less trustworthy since they could have been entered incorrectly (whether by accident or intentionally).

One way to improve verifiability is to include more details in the results than are strictly necessary to produce the benchmark's metrics. Inconsistencies in this data could raise questions about the validity of the data. For example, a benchmark with a throughput metric might include response time information in addition to the transaction counts and elapsed time.

SPEC PTDaemon includes several features designed to improve verifiability of power measurements. A checksum is used to ensure that only official SPEC PTDaemon binaries are used. PTDaemon sets current and voltage ranges itself (when allowed by the device) so that it knows what ranges are in use. Data is collected for voltage, current, and power factor in addition to power. For each measurement interval it reports the minimum and maximum readings in addition to the average.

The SPEC Power and Performance Benchmark Methodology recommends measuring the uncertainty of the energy measurements, which are a function of the analyzer being used, the range settings on the analyzer, and the actual measurements. Power analyzers typically have to be set to some current range; if this range is not appropriate for the actual current being measured, the measurements reported by the analyzer may be inaccurate. The technical specifications for each power analyzer typically include accuracy formulas based on the range setting, measured values, and other factors; these can be used to ensure that the measurements do not exceed the uncertainty required by the benchmark. SPEC PTDaemon includes support for automatically calculating the uncertainty of the results.

SPEC also recommends using a temperature sensor to verify that the ambient temperature doesn't fall below the limit specified by the benchmark [24].

3.5 Usability

Most users of benchmarks are technically sophisticated, making ease of use less of a concern than it is for more consumer-focused applications. There are, however, several reasons why ease of use is important.

One of the most important ease of use features for a benchmark is self-validation. This was already discussed in terms of making the benchark verifiable. Self-validating workloads give the tester confidence that the workload is running properly.

Another aspect of ease of use is being able to build practical configurations for running the benchmark. For example, the current top TPC-C result has a system under test with over 100 distinct servers, over 700 disk drives and 11,000 SSD flash modules (with a total capacity of 1.76 petabytes), and a system cost of over $30 million USD. Of the 18 non-historical accepted TPC-C results published between January 1, 2010 and August 24, 2013, the median total system cost was $776,627 USD. These configurations aren't economical for most potential users [18].

Accurate descriptions of the system hardware and software configuration are critical for reproducibility, but can be a challenge for usability due to the complexity of these descriptions. Benchmarks can improve ease of use by providing tools to assist with this process.

Energy efficiency benchmarks carry special ease of use challenges. The users of these benchmarks tend to be a cross-section of performance specialists and energy specialists. These groups have different skill sets, resulting in a more diverse set of users than a typical performance benchmark; therefore, fewer assumptions can be made about the user's understanding of running the benchmark.

In an energy efficiency benchmark, the energy usage of the system must be measured. Some systems may provide energy usage data from internal sources, but benchmarks typically require an external power analyzer that meets certain criteria for accuracy. Properly installing and configuring this device can be a challenge even for experienced users. For example, the team publishing the first TPC-Energy result relates an experience with overheating of power cables during some of their initial experiments due to exceeding the rating of those cables[31]. These types of issues can't be dealt with directly by benchmark code, but documentation for energy efficiency benchmarks should caution users to carefully plan their environments to avoid safety problems or inaccuracy in results.

Benchmarks can simplify collection of energy data by collecting data directly from the power analyzer and performing automatic validation that the results meet the accuracy requirements specified by the benchmark. SPEC has implemented the SPEC PTDaemon for automatically collecting data from power analyzers and temperature sensors; this tool has also been licensed to other groups for inclusion in their benchmarks.

3.6 Key Requirements for Energy Efficiency Benchmarks

While many of the design criteria for energy efficiency benchmarks are the same as the criteria for traditional performance benchmarks, the sections above identify several requirements that are either specific to energy efficiency benchmarks or have increased importance:

Multiple Load Levels Since servers rarely run at 100% utilization, it is important for energy efficiency benchmarks to measure power consumption at multiple levels of system utilization.

Multiple Workloads Different workloads use system resources differently, and no single workload can model energy efficiency for all types of applications. Suites of multiple workloads can exercise a wider range of behavior to produce more accurate results.

Multi-system Results Many servers, particularly blades, use shared power and cooling infrastructure that can make it difficult or impossible to accurately measure energy efficiency of a single system in isolation. Benchmarks that are able to exercise multiple systems in a coordinated fashion can produce meaningful results for these environments.

Accurate Power Measurements Energy efficiency benchmarks need to define power analyzer accuracy requirements, and should verify that these requirements are met. Requirements on environmental factors, such as ambient temperature, may be necessary to ensure results are relevant for real usage.

Full Disclosure System components and settings that have no impact on performance may change the system's power consumption. Energy efficiency benchmarks may require more complete descriptions of the system than are needed for many performance benchmarks.

Ease of Use Accurately measuring power consumption can be challenging; benchmarks that are easy to use and perform automatic validation will reduce the burden on users and result in more accurate results.

4 Existing Energy Efficiency Benchmarks

Several different benchmarks and workloads have been proposed for evaluating energy efficiency of computer servers [32, 33]. In some cases, power measurements have been added to existing performance benchmarks as optional or required metrics. In other cases, benchmarks have been designed specifically for measuring energy efficiency.

In this section I discuss several benchmarks that include power metrics and assess their quality against the design criteria discussed in the previous section.

4.1 TPC-Energy

The Transaction Processing Council (TPC)[4] has been producing database and transaction processing benchmarks for over 20 years. There are currently three active TPC benchmarks: TPC-C, TPC-E, and TPC-H. In 2010, the TPC released the TPC-Energy specification which defines how energy usage can be measured and reported for any TPC benchmark result [34]. Energy metrics for TPC benchmark results are optional, but when reported they must be in compliance with the TPC-Energy specification.

TPC-Energy results include a "watts per performance" value; lower values indicate less energy usage than higher values. In addition to the overall energy efficiency metric, energy requirements can be reported for independent subsystems such as the Database Server or the Storage Subsystem. Energy consumption can be measured on a subset of system components when multiple components are considered to be equivalent; the energy consumption of the measured components is extrapolated to estimate the total energy consumption of the system.

TPC-C is an On-Line Transaction Processing (OLTP) benchmark first introduced in 1992 by the TPC. It has a database containing nine tables and uses a mix of five transactions commonly found in order-entry systems (New Order, Payment, Delivery, Order Status, and Stock Level)[35]. As of August 24, 2013 there have been only 3 TPC-C results published with energy measurements, all by a single vendor.

TPC-E is another OLTP benchmark, released in 2007. It is intended to be more representative of modern OLTP applications than the TPC-C benchmark. In particular, it was designed to use more complex transactions with a significant reduction in I/O requirements. The workload models a brokerage firm, with 33

[4] `http://www.tpc.org/`

database tables and 10 transaction types. The database uses a more realistic set of data types, content, and constraints than TPC-C [35]. As of August 24, 2013, there have been 7 TPC-E results published with energy measurements.

TPC-H is a decision support benchmark exercising ad-hoc queries, released in 1999. It includes 22 queries with diverse characteristics, and the metric captures both single-user and multi-user results [36]. As of August 24, 2013 there are 6 published TPC-H results that include energy measurements.

Relevance The TPC-Energy specification provides relevant measurements of the energy consumption of the environment used to run the benchmark, but the relevance of the resulting data is a function of the relevance of the benchmark itself. It optionally includes separate measurements of different components of the environment (database server, application server, storage, and miscellaneous), which can provide additional information that is relevant to customers. The resulting energy efficiency metric is a "watts per transaction" ratio, where smaller values indicate better efficiency. Observers often find it easier to understand a "bigger is better" metric, but the TPC-Energy metric is consistent with the existing TPC pricing metrics, where the goal is to have maximum performance with minimum cost per transaction – similarly, the goal for energy is to have the maximum performance with the least energy per transaction.

At the time of its release, TPC-C was representative of applications common at that time, and was therefore highly relevant. In the past 20 years, technology and applications have changed; while TPC-C has undergone some revisions, it is largely the same application that it was at its initial release. Modern applications typically have more complex data models and queries, as well as more substantial business logic. TPC-C exercises many components of a system, including CPU, memory, disk storage, and network, though the balance of these components may not match real applications.

The longevity of TPC-C has proven its ability to scale; results published as of August 24, 2013 range in performance from 9,112 tpmC to 30,249,688 tpmC, a range of about three orders of magnitude over a period of about 15 years. On the other hand, much of this scalability is a result of super-tuned configurations which are increasingly unrealistic [35]. Due to the aging application model, TPC-C is not particularly relevant today, though it provides historical context and the energy component of the workload is a realistic measure of the peak energy required by the test environment.

TPC-E was designed to overcome many of the issues that made TPC-C increasingly unlike modern applications. It uses a more realistic database schema with diverse data types and enforced constraints. Transactions are more complex than those in TPC-C, and run rules are intended to limit extreme tuning in the implementation of the benchmark. TPC-E is less I/O-intensive than TPC-C, resulting in a more typical balance of CPU, memory, disk storage, and network activity [35, 37]. These improvements over TPC-C make TPC-E a more relevant measure of OLTP performance; TPC-E with TPC-Energy is a relevant measure of the energy efficiency of this type of application.

The TPC-H benchmark models a Decision Support System (DSS) rather than the OLTP model of TPC-C and TPC-E. This type of application typically consists of a large, mostly static dataset, with complex ad-hoc queries. TPC-H is closely based on the retired benchmark TPC-D, released in 1994. At the time of its release, the TPC-D benchmark pushed the limits of the available database systems, but subsequent optimizations reduced the utility of the benchmark. Modifications in TPC-H made it more relevant, but the industry has continued advancing since its release. The TPC has increased the longevity of TPC-H by introducing new "scale factors" which increase the data set size without changing the underlying application model [38]. Recognizing that a more substantial overhaul was needed to support modern DSSs, the TPC released a new Decision Support benchmark called TPC-DS in 2012; however, no results have yet been published, and it remains to be seen whether TPC-DS will be a viable replacement for TPC-H.

While the relevance of TPC-H may be waning, there has been relatively little work on energy efficiency of a DSS environment. Using TPC-H with TPC-Energy may provide valuable information about these types of systems [39]. Even if the workload is not a perfect simulation of a modern DSS, TPC-H more closely resembles these environments than other measures of energy efficiency do.

Reproducibility The specifications for TPC benchmarks include many constraints intended to ensure that the benchmark results are valid and that the required quality-of-service metrics are being met. These requirements help ensure consistent results. Each benchmark test must be audited by an approved auditor to ensure compliance with all of the benchmark requirements. The TPC-Energy specification adds requirements specific for power analyzers and environmental factors which must also be included in the TPC-C audit when reporting energy measurements.

Many TPC benchmark environments (including TPC-C, TPC-E, and TPC-H) are too large to allow all of the components to have their energy consumption measured practically; for example, a single TPC-C test could use a cluster of a dozen or more database servers and even larger numbers of storage arrays. The TPC-Energy specification allows representative subsets of the environment to have their energy measured, and then extrapolates these results to the remaining components. The specification requires two equivalent units to be measured (selected by the auditor), and that their energy measurements are within 10% of each other; the measurement from the higher of the two is used for the unmeasured components. These rules help to ensure that the energy measurements are consistent.

The Full Disclosure Report for a TPC results provides substantial detail about the hardware and software used for the result, allowing the result to be replicated by others.

The extensive run rules and auditing result in good consistency and reproducibility for results in all three of these TPC benchmarks.

Fairness The TPC benchmarks are developed and approved by members of the TPC. At the time TPC-C was written, the membership included 16 industry representatives [40], many of whom are competitors. The fact that these members were able to agree on the specification indicates a level of fairness in the benchmark definition.

Because TPC-C, TPC-E, and TPC-H are full system benchmarks and each includes a pricing component, it is expected that any hardware included in the test configuration is actually used during the test. As a result, power measurements for the full system are a fair measure of the energy used. In some cases, the benchmark run rules allow the tested configuration to be different than the priced configuration; for example, the priced configuration may have additional storage devices which are not needed during the test but would be required to hold data when running the application over a longer period of time. The TPC-Energy specification includes provisions for estimating the energy that would be used by the priced configuration.

The TPC benchmark specifications also include a number of requirements for service times and other quality of service metrics which must be met by compliant results.

TPC-C, TPC-E, and TPC-H are benchmark specifications, and do not include specific implementations. The implementation is provided by the benchmark sponsor, and must be disclosed and published with the result.

The TPC has a detailed fair use policy which restricts how results may be used and compared.

These factors all contribute to TPC-C, TPC-E, and TPC-H (with TPC-Energy) being very fair benchmarks.

Verifiability Results from TPC benchmarks must be audited before they are accepted as compliant. The auditor must be certified by the TPC and independent from the entity publishing the results. The auditor oversees the test and verifies that the benchmark application follows the requirements of the benchmark specification and that details of the environment are reported accurately.

TPC-Energy includes additional requirements for auditing. Auditors must receive separate certification from the TPC for TPC-Energy. The auditor must verify that power is measured following the requirements of the TPC-Energy specification and that results are reported accurately.

A Full Disclosure Report (FDR) is written by the test sponsor and reviewed by the auditor. The FDR includes a detailed description of the hardware and software components of the system as well as descriptions of how each of the benchmark and TPC-Energy requirements were met (or verified to be met).

The auditing process and extensive disclosure requirements provide a high level of verifiability in TPC-Energy results with TPC-C, TPC-E, and TPC-H.

Usability Getting compliant TPC benchmark results is a non-trivial task. Since the implementation is not provided by the TPC, a suitable implementation must be obtained or developed.

As described above, TPC-C configurations can be quite expensive, with system costs ranging from hundreds of thousands to millions of dollars. TPC-E and TPC-H configurations are only slightly cheaper. In addition to the costs of obtaining the necessary hardware and software, such large systems can require substantial effort to set up, configure, and tune.

The auditing process for TPC benchmarks can also be daunting, and adds to the cost. Full Disclosure Reports often run several hundred pages, largely due to requirements to disclose the implementation of the transactions and other application code.

The TPC provides an Energy Measuring System (EMS) package of software for measuring energy usage. It includes a licensed version of SPEC's PTDaemon software for interfacing with power analyzers and temperature sensors to automatically collect data from these devices during the run.

The use of a power analyzer adds complexity to running a benchmark; the complexity is often greater for the large system configurations that are typically used for TPC-C, TPC-E, and TPC-H publications. The TPC has worked to simplify this process as much as possible without sacrificing accuracy by allowing power to be measured for representative subsets of the system under test.

4.2 SPECvirt_sc2010

SPEC produces several benchmarks developed by its members. SPEC's first benchmark for evaluating virtualized environments in a server consolidation scenario, SPECvirt_sc2010[5], incorporates modified versions of other SPEC benchmarks (SPECjAppserver2004, SPECmail2008, and SPECweb2005) running in groups of virtual machines [41]. SPECvirt_sc2010 includes optional power metrics for either server power (SPECvirt_sc2010_ServerPPW) or total power for both server and storage (SPECvirt_sc2010_PPW). The power metrics measure power at full utilization and at idle. As of August 24, 2013, there are 31

[5] http://www.spec.org/virt_sc2010

SPECvirt_sc2010 results published on SPEC's website, but only two results each for SPECvirt_sc2010_ServerPPW and SPECvirt_sc2010_PPW.[6]

Relevance Server consolidation is one of the most common uses of virtualization technology, allowing multiple applications that often run at low utilization to be run on a single physical server at higher utilization. One of the drivers of consolidation is reduced energy consumption since most servers have the highest efficiency (performance per watt) at high utilization. A benchmark focusing on server consolidation therefore provides a useful measure of server energy efficiency.

The component workloads that make up SPECvirt_sc2010 represent fairly typical server applications, including an enterprise Java application with database access, mail serving, and web serving. While the actual applications being consolidated in real-world environments may be different, the behavior of these applications is representative of others.

SPECvirt_sc2010 power measurements are optional. When power data is included, it must be measured at full load and no load (active idle). This provides upper and lower bounds on the energy usage, but no intermediate load levels are included.

Reproducibility Compliant SPECvirt_sc2010 results must include a detailed description of the hardware and software configuration used. Reproducing a particular test result is non-trivial since these are complex environments with many components that could influence performance and/or energy usage.

Run rules for the benchmark define several quality of service (QoS) metrics which must be met in order to have a compliant result. These metrics indicate some level of consistency in the results and ensure that the component

[6] In May, 2013, SPEC released SPECvirt_sc2013 as a replacement for SPECvirt_sc2010. The new version is very similar, but includes more relevant workloads.

benchmarks were running properly.

Fairness Use of SPECvirt_sc2010 results is subject to the SPEC fair use policies, which promote full disclosure when using the results and attempt to disallow unfair comparisons of results.

Because power measurements are an optional component of the benchmark, run rules prohibit comparisons between results that include power measurements and those that do not.

SPEC requires that software used to run the benchmark meets certain standards (such as HTTP 1.1 compliance for the web server component) but does not dictate use of specific software packages, allowing individual testers to choose software packages that are relevant for their users. SPEC provides the implementation of the benchmark and client driver software, but leaves the tester with the freedom to configure the environment appropriately (within the confines of the run rules).

Verifiability SPECvirt_sc2010 results must be reviewed by SPEC before publication. This peer review process gives credibility to the results. The review also ensures that the environment is described in sufficient detail.

The SPECvirt_sc2010 harness performs a variety of validity checks which give credibility to the accuracy of the results. The report includes a variety of information about the performance of each virtual server; inconsistencies in this data would suggest possible non-compliance with the run rules.

SPEC PTDaemon is used to collect power and temperature data. This data is used in accordance with the SPEC Power and Performance Methodology, which includes requirements for the accuracy of the results (according to vendor specifications).

Usability Virtualized environments can be complicated to configure. The workloads that make up SPECvirt_sc2010 are also non-trivial, resulting in additional complexity to the configuration.

The harness provided by SPEC simplifies the actual running of the benchmark, and SPEC PTDaemon automates the collection and correlation of power and temperature data.

4.3 SPECweb2009

SPECweb2009[7] is a now-retired benchmark of static and dynamic web serving. It is based closely on its predecessor, SPECweb2005, but requires power measurements. The benchmark consists of three different test scenarios: Banking, Ecommerce, and Support. Three different scripting languages, ASP, JSP, and PHP, are supported for dynamic pages; results obtained with different scripting languages cannot be compared to each other. Only 8 results were published before SPECweb2009 was retired.

SPECweb2009 includes three distinct workloads: Banking, Ecommerce, and Support. Each of these three has different characteristics. For example, the Banking workload emulates a typical online banking site; SSL is used for all requests. The Ecommerce workload includes users searching for products, with some portion of the users choosing to buy a product. This workload uses a mix of SSL and non-SSL requests. The Support workload simulates a support website, and emphasizes downloads of large files (e.g. user's guides or firmware updates), with no usage of SSL [42]. All three workloads include a mix of dynamic and static requests.

The Ecommerce workload is used to measure power consumption at 6 different load levels: 100%, 80%, 60%, 40%, 20% and 0% of the activity measured during the Ecommerce workload.

[7] `http://www.spec.org/web2009`

Relevance The three applications represented in the benchmark are all examples of commonly used web applications. Each was developed in part from analysis of logs from real applications, leading to realistic workflows, mixes of static and dynamic content, and page sizes.

Power data is a required component of the benchmark, forcing testers to balance energy efficiency with peak performance when designing and tuning the configuration. Energy is measured at multiple levels of utilization, providing readers with energy efficiency data that is relevant for servers which run at less than full utilization.

Web serving performance appears to be of less interest than it was several years ago, perhaps contributing to the short lifetime of this benchmark before it was retired.

Reproducibility The SPECweb2009 report includes a detailed description of the hardware used for the test as well as software configuration and tuning.

Typical SPECweb2009 results require 64 or more client systems to drive a single server. This makes it difficult and expensive for an independent tester to reproduce a published result.

Fairness A variety of different technologies are used for generation of dynamic web pages. SPECweb2009 includes implementations for three of these (ASP, JSP, and PHP) in order to allow the use of a variety of web server software. The run rules include provisions for implementing support for additional scripting languages.

Use of SPECweb2009 results is subject to the SPEC fair use policies, which promote full disclosure when using the results and attempt to disallow unfair comparisons of results.

The SPECweb2009 run rules allow the use of open-source "Community Supported Applications" provided that they meet certain criteria that indicate that the application is stable and adequately supported.

Verifiability SPECweb2009 automatically validates compliance with many of its run rules, including meeting response time constraints. Many performance values are included in the report, allowing for some amount of independent verification of results.

At least one result from a test location must be reviewed and accepted as compliant by SPEC before any result obtained at that location may be publically disclosed. This rule is intended to demonstrate that the licensee is able to produce compliant results. SPEC encourages licensees to submit all of their results for review. SPEC members may request full disclosure reports for any publically disclosed SPECweb2009 result, whether or not it was submitted for review by SPEC.

Power and temperature data is verified in accordance with the SPEC Power and Performance Methodology.

Usability Setting up an environment for a SPECweb2009 run can be complicated. It often involves at least 64 client systems. Each component of the environment must be set up carefully in order to deliver compliant and optimal results.

The SPECweb2009 client software automates the actual run and collection of the required data. SPEC PTDaemon is used to record power and temperature data from the power analyzer(s) and temperature sensor(s).

4.4 SPECpower_ssj2008

SPECpower_ssj2008 was the first industry-standard benchmark specifically designed for measuring energy efficiency of servers.[8] Released in December 2007, the benchmark measures power at multiple levels of utilization to show the server's power consumption at full utilization, active idle, and several points in between. This spectrum of load levels shows the ability of the server to dynamically reduce power consumption when the full resources of the server are not needed [43].

SPECpower_ssj2008 is currently the most widely-published energy efficiency benchmark, with 461 results published on SPEC's website[9] as of November 19, 2013. This relatively large number of results provides value to those trying to identify an energy efficient server to purchase, as well as for researchers studying trends in the results.

The transactions executed by SPECpower_ssj2008 leveraged code from the SPECjbb2005 benchmark, which is in turn based on TPC-C transactions but maintains data in memory instead of using a database. Despite this history, SPECpower_ssj2008 results are not comparable to either of these predecessors. The primary metric for the benchmark is "overall ssj_ops/watt", which is the sum of the performance (measured in ssj_ops) at each of the 11 load levels (including at active idle) divided by the sum of the average watts at these 11 load levels.

Relevance The "SSJ" in SPECpower_ssj2008 stands for Server Side Java. The benchmark is intended to mimic a Java server application, but does not make use of Java EE (Java Platform, Enterprise Edition) middleware or a database like many such applications do. It is a multi-threaded application

[8] The author of this paper has been a participant in SPEC's power committee since 2006, and was the lead developer for SPECpower_ssj2008.

[9] `http://www.spec.org/power_ssj2008/results/`

and in most cases runs multiple Java Virtual Machines (JVMs). It stresses the CPU and moderate amounts of memory, but does not make any direct use of disk storage and includes only trivial network activity.

The business logic in SPECpower_ssj2008 is simple compared to most modern server applications. Because the benchmark implementation is based on the popular SPECjbb2005 benchmark (which was in turn based on SPECjbb2000), the critical code paths in SPECpower_ssj2008 are heavily optimized in major JVM implementations. While this makes the application less representative of real applications, it is not necessarily a bad thing for a benchmark focused on energy efficiency – since the major code paths are already optimized, systems can compete primarily on their ability to reduce power consumption at lower utilizations, and not so much by improvements in raw performance due to software optimizations.

When SPECpower_ssj2008 runs with multiple JVMs, each JVM runs independently of the others, with no shared data. This allows the application to scale almost perfectly to large systems, at the expense of some realism. SPECpower_ssj2008 also allows runs for multiple systems, as long as those systems are homogeneous and include some shared infrastructure. The intended use of this feature is for systems such as blades that have multiple systems in the same chassis. Power usage cannot be measured for individual blades, so the ability to run the benchmark on multiple systems to produce a single result is an important feature for measuring energy efficiency in these environments.

The most important feature of SPECpower_ssj2008 is its ability to run multiple load levels at different levels of utilization (100%, 90%, ..., 10% and Active Idle). This demonstrates the server's ability to reduce power consumption at low utilizations. While the official score weights each load level equally, the results for each load level are described in the report so consumers can weight the load levels differently for their specific purposes.

SPECpower_ssj2008 includes several customizable options that can change its behavior. These options are not allowed for compliant runs, but can be useful for research purposes [44]. For example, when running with multiple JVMs, each JVM can be configured to run a different sequence of load levels. This could be used to mimic environments that have multiple workloads that have different usage patterns.

Reproducibility Java applications tend to have more variable performance than native applications due to the impact of garbage collection, just-in-time (JIT) compilation, and timing differences between threads. As a result, run-to-run consistency of SPECpower_ssj2008 is a bit higher than some other benchmarks. Because each run only takes about 70 minutes, testers often do multiple runs and submit the best.

SPECpower_ssj2008 establishes the peak throughput for a system by running through a "calibration" process which runs transactions as fast as possible for three intervals; the calibrated throughput is the average of the last two of these intervals. The first interval serves as a warmup period to allow the benchmark to achieve steady state. Testers are allowed to increase the number of calibration intervals to as many as 10 in order to give it sufficient time to warm up; the calibrated throughput is always the average of the last two calibration intervals regardless of the total number [44].

After calibration, the workload runs transactions at the different load levels by inserting delays between batches of transactions. The run is automatically marked invalid if the actual throughput is not within 2% of the target throughput (or up to 2.5% less than the target at the 90% and 100% load levels). This ensures that any power management technologies are not preventing the system from keeping up with the intended work.

SPECpower_ssj2008 submissions are required to include a detailed description of the system under test, with sufficient detail that an independent tester should be able to build an equivalent system. Submitters are required to include any configuration details (such as power management settings and other BIOS options) which are required to reproduce the result. Results are peer-reviewed by SPEC members before they are accepted and published on SPEC's web page; however, not all details can be verified with 100% certainty.

SPEC maintains a list of "Accepted Power Analyzers" which are supported by SPEC PTDaemon and have been demonstrated to meet the requirements listed in the run rules. These include minimum requirements for accuracy which ensure that power data is accurate even when comparing data from runs with two different analyzers. Analyzers are required to have been calibrated within the past year to ensure the measurements are accurate.

Fairness SPECpower_ssj2008 was developed by a committee with representatives from several companies from the computer server industry, along with participants from academia [43]. Since the workload is written in Java, it requires a complaint JVM to run. SPEC supplies binaries (compiled Java class files packaged in jar archives) which must be used for the test; recompilation and any modification of the source code is not allowed.

SPEC has a comprehensive fair use policy for results obtained with SPEC benchmarks. SPECpower_ssj2008 does not allow estimated results to be published; it does, however, allow results to be published without review by SPEC as long as at least one result has been accepted by SPEC from the same test location. This is intended to ensure that the tester understands how to properly run the benchmark and configure a power analyzer. Any results published on SPEC's web site must be reviewed and accepted by SPEC.

Additional fair use requirements apply when comparing results. All comparisons must include both performance and power metrics, and comparisons of results at lower load levels must also reference the performance and power metrics at the 100% load level. These rules are intended to restrict misleading comparisons.

Verifiability SPECpower_ssj2008 includes a large number of self-validation checks to ensure that a result is compliant with the run-rules. These checks give the testers a high degree of certainty that a result that is not marked invalid is indeed compliant. SPEC's peer-review process adds greater confidence in the accuracy of results posted on SPEC's web page.

The initial release of SPECpower_ssj2008 did not include checks of power analyzer uncertainty. In general, the uncertainty values will be acceptable as long as the analyzer ranges are set appropriately; when they are not set correctly, the measured values may not have sufficient accuracy to meet the criteria specified in the benchmark's run rules. The range settings can be confusing to new users, and in the initial release of the benchmark the accuracy could only be validated through complex calculations which depended on the specific power analyzer being used. This was a frequent source of early SPECpower_ssj2008 results being rejected. A later update of SPECpower_ssj2008 (including a new version of the SPEC PTDaemon) included support for calculating the uncertainty of the measurements automatically, allowing the run to be marked invalid if the uncertainty is higher than allowed by the benchmark. After this and other enhancements, the main issues with results submitted to SPEC are editorial in nature and can be addressed without re-running the benchmark.

SPEC allows benchmark submission files to be edited after the run to correct descriptive information such as the processor characteristics and operating system name, but does not allow the results themselves to be modified. Checksums are computed to confirm that non-editable portions of the results file are not modified. While this could be subverted by a determined cheater, it does prevent inadvertent edits or casual attempts to "improve" a result.

Another characteristic of SPECpower_ssj2008 that aids in verifiability is that an extensive amount of performance and power data is included in the full disclosure report. If a particular result appears to be unrealistic, reviewers can examine these details to confirm that the data is consistent with the result [44].

Overall, the verifiability of SPECpower_ssj2008 is quite good.

Usability SPECpower_ssj2008 is relatively easy to run. A quick-start guide provided with the benchmark (and available on the Internet[10]) outlines the steps in sufficient detail for new users to make their first run. Further details are available in a more extensive user's guide[11]. A minimum of two systems is required for running the benchmark; in addition to the system under test, a controller system is required for running the Control and Collect System (CCS) and PTDaemon.

Most of the complexity of getting a compliant run is related to the configuration of the power analyzer. SPEC's Power Measurement Setup Guide[12] attempts to simplify the process with details for setting up each type of power analyzer and temperature sensor such that PTDaemon will be able to communicate with it. Details for connecting the system under test to the power analyzer are left to the documentation provided with the power analyzer itself. The Power Measurement Setup Guide also describes how appropriate current range settings can be determined and configured.

As described above, SPECpower_ssj2008 performs an extensive set of validity checks which give the tester confidence that the workload is running properly.

This benchmark was designed for volume servers, so suitable servers are

[10] http://www.spec.org/power/docs/SPECpower_ssj2008-Quick_Start_Guide.pdf

[11] http://www.spec.org/power/docs/SPECpower_ssj2008-User_Guide.pdf

[12] http://www.spec.org/power/docs/SPEC-Power_Measurement_Setup_Guide.pdf

moderately priced and readily available. A variety of power analyzers have been accepted by SPEC as being capable of producing compliant results; the cost of these devices varies based on their capabilities, but reasonably-priced options exist. SPEC PTDaemon also includes support for non-accepted devices which may not deliver sufficient accuracy for compliant runs but may be suitable for experimental runs.

SPECpower_ssj2008 includes a variety of configuration options that can be used by expert users to run experiments in non-compliant configurations, but these options are not needed for typical users. These options are described in a `SPECpower_ssj_EXPERT.props` file which can be ignored by those who don't need them.

Overall, the usability of SPECpower_ssj2008 is reasonably good given the complexities inherent in measuring energy usage. Experienced users can quickly configure the benchmark for new systems, and SPEC provides ample documentation to get new users running.

4.5 JouleSort

The JouleSort benchmark was proposed as an energy efficiency benchmark with a balanced use of system components (CPU, memory, and disk storage) [15]. At the time it was introduced in mid-2007 it was the earliest of the benchmarks discussed here to be available. The goal of JouleSort is to sort a fixed amount of data (10 GB, 100 GB, or 1 TB) with as little energy as possible. New sizes are added with powers of 10 as they become interesting; in 2012, a 100 TB result was reported. For each of the dataset sizes there are two different categories of results: Daytona (commercially supported, off-the-shelf components) and Indy (no restrictions). JouleSort is a complement for performance-oriented sort benchmarks[13], but with its focus on energy efficiency, JouleSort effectuates

[13] `http://sortbenchmark.org/`

different hardware and software optimizations than is seen in pure performance benchmarks [45].

Like the other sort benchmarks, JouleSort is an informal benchmark, with new winners announced yearly, and the current and past winners listed on the web page. Only the winning configurations (and in some years a close second) are listed.

Relevance Sorting is a relatively common activity for servers, but it was chosen primarily because it is a well understood operation that stresses CPU, memory, and I/O, along with the operating system and filesystem [15]. Its strength is in its balanced use of all of these system components.

In practice, the workload balance may not be the right balance for most servers. In the paper introducing JouleSort [15], the winning 100 GB system had a single Intel Core 2 Duo processor (2 cores, 2 total threads), 2 GB memory, and 13 disk drives. This configuration may have more disks and less CPU and memory capacity than are required for many server applications. This is, of course, a concern for any benchmark: its use of system components may not match the usage of other applications.

The processor in the original JouleSort paper was a mobile processor and the disk drives were laptop disks. While these components are more energy efficient than their server-class counterparts, they may not meet the performance and reliability requirements for many servers. A benchmark that encourages the use of these components may not be particularly relevant for servers. More recent JouleSort winners have made use of Solid State Disks (SSDs), which may be more relevant for servers, though still not representative of most systems in use today.

For the smaller dataset sizes, each run is quite short – the 2012 winner for the 10 GB size completed in just 8.47 seconds. With the 100 GB size, the runtime was 133.0 seconds [46]. These short run times may not be particularly relevant for server workloads.

Due to these limitations, JouleSort is more useful as a demonstration of technology that can be used to build more energy efficient systems than it is a benchmark for measuring the energy efficiency of a particular system.

Reproducibility Run-to-run consistency of sorting is generally good; the current JouleSort rules use the average of 5 consecutive runs. For the 2012 winners for 10 GB, 100 GB, and 1 TB, the difference between the slowest and fastest of these 5 runs was less than 2.5% of the average runtime, and the difference between the lowest and highest energy of the 5 runs was less than 2%.

The run rules require a description of the system, but don't dictate what level of detail must be included. In practice, the winning reports typically contain reasonable levels of detail, though less than what is required by some other benchmarks.

Fairness The sort benchmarks were initially defined by one individual. In recent years they have been maintained by a committee representing three companies; all of the current committee members are past winners of the sort benchmarks. JouleSort itself was proposed in a paper written by individuals from Stanford University and HP Labs. Despite the limited breadth of the committee, none of the benchmark rules appear to favor any particular vendors.

The benchmark rules define the required behavior of a sort implementation, but do not provide the implementation itself (though a program is provided to generate suitable input data). Any implementation that meets the requirements is allowed, so no particular hardware and software are excluded.

Verifiability A validation program is provided for confirming that the output of the sort is indeed in sorted order. The input generation program provides a checksum which is also used in the validation program to ensure that the same records are included in both the input and output files. JouleSort doesn't provide any specific verification of energy measurements, but researchers are expected to describe the process used.

As described above, the benchmark rules have few specific reporting requirements. Since most winning results are submitted by researchers from academia, the reports typically describe any novel features in detail.

Usability Running a sort program is simple, and the program for generating suitable input data is provided. Measuring the energy usage during the test can be more complicated. The JouleSort definition references the SPECpower_ssj2008 requirements for power analyzers, but does not provide a program to interface with the power analyzer. For the small datasets, the short duration of the test can make it difficult to get an accurate power measurement.

The largest datasets may be processed by multiple systems working in parallel; since these systems may have a large number of power supplies, directly measuring the power for the entire set of systems may be impractical. The run rules allow for measuring subsets of the system and making reasonable extrapolations to estimate the total power consumption.

The current JouleSort run rules require a "fairly accurate" power analyzer with ±1% accuracy. It is up to the tester to determine whether or not this level of accuracy is actually met.

4.6 Related Energy Efficiency Benchmarks

In addition to the energy efficiency benchmarks for servers listed above, there are a number of other energy efficiency benchmarks that may be relevant for server environments even though they don't target servers themselves. These benchmarks are described briefly below.

4.6.1 Green500

The Green500 List[14] ranks supercomputers according to their energy efficiency [47, 48]. It is based on the Linpack benchmark used for the TOP500 rankings of supercomputer performance[15].

The TOP500 list has established itself as the premier ranking of supercomputers. While Linpack may not be the best possible measure of performance [16], it is an adequate measure of the performance of supercomputers. As a result, it makes sense to build on this success as the basis for a measure of energy efficiency for supercomputers. The creators of the Green500 list acknowledge that other workloads may behave differently, and have stated an intention of adapting Green500 to include other workloads in the future.

Green500 only measures energy efficiency at near-peak utilization. This may be appropriate for supercomputers since they tend to be more heavily utilized than typical servers. Since it only includes systems on the TOP500 list, Green500 focuses on a narrow subset of systems typically designed for peak performance, so it may not be representative of more general systems.

4.6.2 SPC

The Storage Performance Council (SPC)[16] produces benchmarks to measure the performance of storage products. Since 2009, the SPC has added energy extensions to each of their benchmarks in order to report energy efficiency as well as performance of these products.

The SPC-1 benchmark focuses on mostly random I/O operations including both reads and writes. The SPC-2 benchmark stresses large sequential I/Os that are common for some types of commercial applications. The SPC-1C and

[14] `http://www.green500.org`

[15] `http://www.top500.org`

[16] `http://www.storageperformance.org/`

SPC-2C benchmarks use the same workloads as SPC-1 and SPC-2, but focus on smaller components of a full storage subsystem.

Power consumption is measured at multiple levels of utilization. Energy measurements are optional for all of the SPC benchmarks, and only a small percentage of published results currently include energy data.

Storage is an important component of many enterprise servers. The SPC benchmarks complement system-level and server energy efficiency benchmarks.

4.7 Summary of Existing Energy Efficiency Workloads

The complexity and diversity of benchmarks makes it difficult to summarize their quality at a high level. Nevertheless, Table 1 provides the author's assessment of each of the benchmarks described in previous sections for the qualities described in Section 3. The basis for these assessments is provided in previous sections. The author's work with SPEC may introduce some bias, and assessments from an independent observer could be different.

5 Chauffeur and the Server Efficiency Rating Tool

In 2009, the EPA announced Energy Star requirements for computer servers which included a limit on idle power consumption. The EPA also indicated their intention for a more robust Tier 2 specification which (among other changes) would incorporate energy measurements with the server under load. The EPA has been working with SPEC to define a method for collecting these measurements. To that end, SPEC developed the Server Efficiency Rating Tool (SERT)[17] [49].

[17] `http://www.spec.org/sert/`

Table 1: Summary of assessments of energy efficiency benchmark quality

	TPC-C w/	TPC-E w/	TPC-H w/
	TPC-Energy	TPC-Energy	TPC-Energy
Relevance	Poor	Good	Fair
Reproducibility	Fair	Fair	Fair
Fairness	Good	Good	Good
Verifiability	Good	Good	Good
Usability	Poor	Poor	Poor
# Results	3[a]	7[a]	6[a]
	SPECvirt_sc2010	SPECweb2009	SPECpower_ssj2008
Relevance	Good	Fair	Fair
Reproducibility	Fair	Good	Good
Fairness	Good	Good	Good
Verifiability	Good	Good	Good
Usability	Fair	Fair	Good
# Results	2 / 2[b]	8[c]	444[d]
	JouleSort		
Relevance	Poor		
Reproducibility	Fair		
Fairness	Good		
Verifiability	Poor		
Usability	Fair		
# Results	23[e]		

[a] Results published at `http://www.tpc.org` as of Aug 24, 2013

[b] 2 SPECvirt_sc2010_ServerPPW, 2 SPECvirt_sc2010_PPW results published at `http://www.spec.org` as of Aug 24, 2013

[c] Results published at `http://www.spec.org`. SPECweb2009 is now retired; no additional results will be published by SPEC

[d] Results published at `http://www.spec.org` as of Aug 24, 2013

[e] Annual winners in each category published at `http://sortbenchmark.org`

SPEC recognized that the SERT would have several requirements that differed from a conventional benchmark. Like SPECpower_ssj2008, the SERT would require the ability to put controlled but variable loads on the system; however, the SERT would need to be more highly automated and exercise a variety of workloads.

As a participant in SPEC's power committee, I designed and implemented the Chauffeur framework which is the foundation of the SERT. I also assisted with the design and implementation of many of the SERT "worklets". This section will outline the design goals for the SERT, then describe the design for Chauffeur and how it allows SERT to achieve these goals. The next section will present results and give a preliminary evaluation of the SERT using the criteria established in the first portion of this paper.

5.1 Design Goals for the SERT

SPEC has made the design document for the SERT available for public review [50]. This document includes the following goals for the SERT:

Stress multiple system components In most traditional benchmarks, the system will be configured with sufficient memory, disk, and other components so that the benchmark is CPU-bound. Thus, the benchmark defines a particular balance of system components that will be used in the test; reducing the amount of memory or IO capabilities would hurt the performance (or fail to run at all), while adding memory or IO would consume additional power without providing improved results.

The EPA's draft 2 ENERGY STAR Version 2.0 Computer Server specification[18] defines a product family with a range of system components. Qualification of the product family is made using a representative set of configurations

[18] http://www.energystar.gov/index.cfm?c=revisions.computer_servers

for the family, including minimum, typical, and maximum configurations. As a result, the SERT must be able to produce reasonable results with a wide variety of configurations with different balances between CPU, memory, and IO capabilities.

The SERT is intended to show better performance results for systems with faster components or larger capacities. Where possible, the tests will directly measure the benefits of additional hardware components added to the system. In some cases, artificial adjustments may be required to compensate for the additional power consumed by these components. Adjustments may also be necessary to account for systems with more room for expansion – for example, servers with a larger number of DIMM slots may consume more power than servers with a smaller number of slots even if the extra slots are not populated, due to the infrastructure required to support those slots.

The initial target for SERT is to measure servers with 1-8 processors and between 8 GB and 1 TB of memory. Disks contained within the system enclosure will be tested; external storage is excluded. The SERT can run on a single system or a group of up to 64 homogeneous servers (such as blades).

While the initial SERT design document indicated that both storage and network IO would be included, subsequent updates indicated that network IO would not be included in the first release. This was due to the challenges of measuring network performance without requiring significant numbers of driver systems, as well as the relatively small and static power consumption of network cards.

Multiple synthetic worklets In order to support the necessary range of systems, multiple workloads are used to exercise different components of the system, including CPU, memory, and disk storage, as well as a hybrid workload intended to provide a more balanced use of the system components. Each of these workloads is composed of one or more “worklets”. Each worklet uses different application code to exercise the system differently. By using multiple distinct worklets, a more balanced measurement of the system capabilities is obtained.

Because the worklets are intended to stress individual components of the system, they are somewhat synthetic in nature, although they do perform functions that are often used on computer servers. This is suitable for a general assessment of a system’s energy efficiency, but does not replace the need for other benchmarks that focus on specific application areas [3]

SPEC and the EPA are expecting SERT results to be obtained for a large number of systems and configurations. As a result, it was important to limit the runtime in order for it to be practical for systems vendors to run SERT for these systems. However, it is also important for each worklet to run long enough to produce accurate data. SPEC set a 4-5 hour runtime target for SERT to balance these factors.

Multiple load levels Since most servers rarely run at full utilization, it is important to collect energy efficiency data at multiple utilizations in order to understand how the system is able to reduce power consumption at lower utilizations. Different patterns of system usage may result in different power management behavior, so for SERT it is important to run each worklet at multiple load levels. The 4-5 hour target runtime limits the number of load levels that can be run. SPEC has chosen different numbers of load levels to run for different types of worklets in order to maximize the amount of useful information obtained during the run. For example, the Disk Storage worklets run fewer load levels than other worklets because current systems show relatively small contribution to power consumption due to disk (for internal disks). A larger number of load levels is used for the Hybrid SSJ worklet (based on SPECpower_ssj2008) to provide more granular data for one worklet that stresses both CPU and Memory.

The SERT memory worklets run multiple load load levels, but rather than varying the throughput (amount of work per unit of time), the memory worklets vary the amount of memory they are using. This measures the ability of the system to reduce power consumption for memory that is not actively in use.

Cross-platform support The SERT is intended to run on a variety of platforms. Only 64-bit platforms are supported; while most of the worklets run properly on 32-bit systems, the memory worklets are designed for large memory capacities that are common on modern servers, and 64-bit operating systems are required for addressing these memory capacities. Support for specific hardware architectures and operating systems is defined primarily by the commitments for testing by SPEC members participating in SERT development; while nothing is being done to preclude support for other platforms, they can't be supported until adequate testing is performed.

The initial version of SERT is written primarily in Java, both because of its popularity for server applications as well as for ease of portability. Since Java applications tend to have different performance characteristics than those written in other languages like C and C++, a future version of SERT is likely to include worklets written in other languages.

Test "as-shipped" As its name implies, the SERT is intended to be a rating tool and not a conventional benchmark. In particular, while most benchmarks encourage fully optimized results under optimal conditions, the SERT is designed to run under more typical conditions with limited optimization [50].

Since a key goal of the ENERGY STAR program is to improve energy efficiency over time, it is likely to require that default system power management settings be used in order to encourage vendors to make their default settings as efficient as possible.

Due to the variety of worklets that are part of the SERT, some amount of JVM tuning must be allowed in order for the tool to run effectively. The memory worklets are a particular challenge, since they allocate the majority of the system memory to the Java heap. The heap sizes themselves are under the control of SERT, but the JVM may require other command-line options to be able to effectively use that amount of heap space. In addition, some amount of performance optimization is necessary in order to ensure that a wide variety of systems can compete fairly – any particular set of tuning flags (including the default flags for the JVM) will tend to favor some system configurations over others. Allowing some tuning also provides flexibility for adapting to new hardware as well as future versions of operating systems and JVMs that are released during the lifetime of the SERT.

In order to avoid "super-tuning", SPEC publishes a list of tuning options that must be used for compliant results for a particular combination of operating system, JVM, and processor micro-architecture. A common set of options must be used for all of the worklets in each workload (CPU, Memory, Storage, and Hybrid) to help ensure that these options are applicable to a variety of applications.

5.2 Introducing Chauffeur

The Chauffeur framework was designed to meet the SERT goals outlined in the previous section. Chauffeur implements common code for running worklets at multiple load levels and reporting their results, allowing the implementation of individual worklets to focus on application logic rather than infrastructure. This infrastructure is significant; while Chauffeur contains over 25,000 lines of code (not including comments and blank lines), the combined total of all of the SERT worklets is only about 13,500 lines, and most of the worklets are implemented with fewer than 1500 lines of code. Thus, the implementation of new worklets based on Chauffeur is far simpler than implementing them from scratch.

In addition to supporting SERT, Chauffeur is intended to provide a flexible framework for future benchmarks and support experimentation for research purposes.

This section lists some of the key design features of Chauffeur and describes how they support the design goals of the SERT and meet the requirements for energy efficiency benchmarks.

5.2.1 Multiple Processes and Multiple Threads

Chauffeur is designed to make use of multiple processes, each with multiple threads, to make it possible to scale from small single-processor servers to much larger systems with multiple nodes. The major components of a SERT environment are shown in Figure 5.

The Chauffeur "Director", SERT UI, and SPEC PTDaemon instances run on a controller system that is separate from the System Under Test (SUT). Running these components on an external system ensures that they are not contributing to the load on the SUT, particularly at low utilizations where extra tasks could prevent the server from reducing power as much as possible [49].

Each server in the SUT runs an instance of the Chauffeur "Host" JVM. The Director communicates with each Host to control the run. The Host launches a set of Client JVMs to run the actual application code. A new set of Client JVMs is typically used for each worklet to reduce the influence of different worklets on each other. Chauffeur also supports configuration options to launch new Client JVMs only once for the whole suite, for each workload, or at finer granularities including for every interval of a particular worklet.

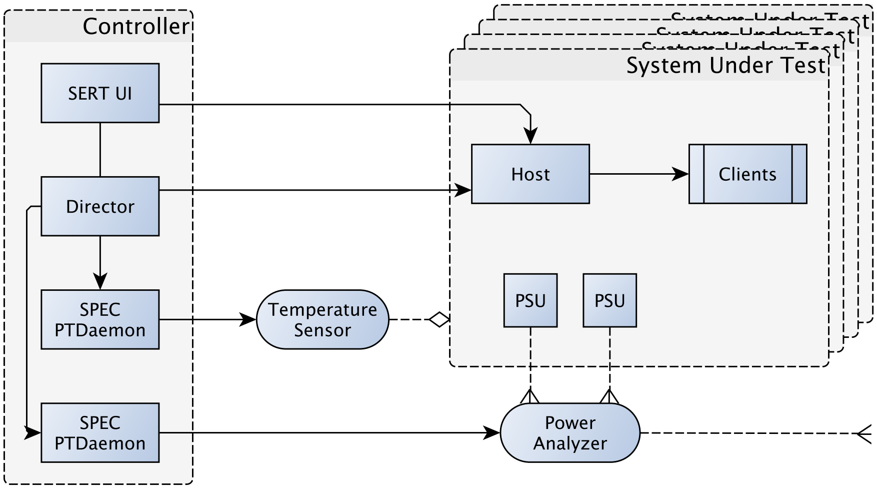


Figure 5: SERT Components

Communication among the JVMs uses a command-driven model, where the Director initiates transitions to different stages of the run. Some other benchmarks, such as SPECpower_ssj2008, include some automatic state transitions within each client (for example, as each client moves from pre-measurement to measurement to post-measurement). This causes difficulty in synchronizing collection of power data with the performance measurements, and can also create challenges for virtualized systems which sometimes have clock skew issues. Chauffeur's implementation avoids these issues since the Director controls the run.

The Client JVMs are usually launched with platform-specific affinity commands to define which processors each JVM will use. While not all real-world applications are able to use affinity as effectively as Chauffeur, using affinity settings is necessary to scale on large systems; placing affinity under Chauffeur's control allows it to be used consistently and the commands to be documented accurately. Affinity support can be disabled for worklets where it is not appropriate.

5.2.2 Energy Measurements

While Chauffeur can be used for traditional performance workloads, it has been designed for measuring energy efficiency. It includes support for communicating with SPEC PTDaemon, which has support for collecting power and temperature data from a variety of power analyzers and temperature sensors.

Unlike most traditional benchmark frameworks, Chauffeur also has the ability to drive workloads at various load levels – for example, at 67% and 33% of the system's capacity. This support is independent of the workload itself, so any Chauffeur-based workload can be calibrated and then run at any desired load level.

It is important to note that these load levels do not necessarily correspond to CPU utilization. Different hardware platforms and operating systems may calculate CPU utilization differently, particularly in the presence of multiple processors and hyperthreading, so CPU utilization is not a reliable metric for determining what portion of the system's capacity is in use [51]. Furthermore, some workloads may not be CPU-bound. For example, the SERT Storage worklets operate at low CPU utilization but are gated by I/O performance. Chauffeur's calibration mechanism operates independently of CPU utilization.

Another feature of Chauffeur that is important for measuring energy usage is the ability to run workloads simultaneously on multiple operating system instances. Many modern systems, such as blades, include shared power and/or cooling infrastructure which make it inaccurate or impractical to measure energy usage for a single system in isolation. Running workloads across multiple systems at the same time in a coordinated fashion allows energy usage to be measured accurately in these environments. This support may also prove to be useful for measuring performance and energy usage in virtualized environments, though such environments are not currently supported by the SERT.

5.2.3 Multiple Workload Support

Chauffeur is intended to make it easy to implement new workloads and worklets. The SERT includes a dozen worklets, and others are in development for possible inclusion in future releases. By utilizing the Chauffeur framework, the workload code consists almost entirely of the actual code being tested, while Chauffeur itself takes care of configuration, driving the load, inter-process communication, reporting, and monitoring. New workloads can be created and "plugged in" without modifications to Chauffeur itself.

Workloads can be executed individually or together in a single run. Support for multiple workloads is important for energy efficiency measurements since different types of applications can have different energy efficiency characteristics. Chauffeur can run different workload in the same Java Virtual Machine (JVM) or launch a separate JVM (with varying configurations) for each workload. In many cases it is also possible to run transactions from multiple workloads concurrently.

5.2.4 Easy to Run

Chauffeur includes several features intended to make it easy to produce valid results. Workloads are self-calibrating. Client JVMs are launched automatically and can automatically make use of operating system-specific affinity settings. Appropriate Java heap settings can be calculated automatically based on the amount of memory available in the system. While the initial release of the SERT requires some amount of system-specific configuration, one goal for future updates is to allow users to initiate a run with little or no manual configuration.

Chauffeur also includes support for the collection of system configuration data which can be used to automatically populate some of the descriptive fields in the full disclosure report, or to validate some of the user-entered data. This support is in its infancy, and the initial release only gathers small subsets of the configuration data needed to be reported. Future versions should be able to offer expanded support to gather more data automatically, improving usability and verifiability.

5.2.5 Flexible Configuration and Data Collection

Chauffeur offers a highly flexible configuration for defining and running workloads. Many aspects of the workload execution can be modified through an XML configuration file, including the number of client JVMs, the length of each measurement interval, and the sequence of load levels. Chauffeur also provides the capability to set workload-specific configuration options in the configuration file.

Most components of Chauffeur can be replaced with different implementations specified in the configuration file. For example, a researcher could be interested in mimicking the utilization patterns of a real server. Chauffeur would allow the researcher to provide a custom "Sequence" implementation which would read the current CPU utilization from a remote system and then use that value as the load level target for the next Chauffeur measurement interval. This type of behavior is clearly too esoteric to be part of the standard Chauffeur implementation, but can be used with Chauffeur due to the ability to plug in custom implementations (without modification to Chauffeur itself).

A generic "Listener" interface is used to support custom data collection or to interface with other components. Listeners are notified at various stages of the run. Chauffeur itself uses Listeners for writing results files, gathering data from SPEC PTDaemon, and recording garbage collection statistics from client JVMs. New listeners can be plugged in without modifying Chauffeur itself. Custom listeners could be used to run platform-specific performance tools, collect data from power analyzers that are not supported by SPEC PTDaemon, or to initiate system configuration changes at different stages of a run.

The current Chauffeur implementation reads the entire configuration at the beginning of the run and then executes the tasks defined in that configuration; however, the interprocess communication protocol is designed to run a single command at a time. It is likely that a future version of Chauffeur will allow runs to be controlled dynamically and interactively.

5.2.6 Self-validation

Chauffeur includes a framework for validating the results of a run to ensure that it is compliant with run rules. While not all run rules can be verified automatically, self-validation is valuable for the people running a benchmark, people reviewing results, and the people reading published results. For those running the benchmark, self-validation gives confidence that the benchmark is running properly and that the results they are getting are compliant. Self-validation also simplifies the results review process, since the reviewer can be confident that the result passes these tests, and can focus their review on aspects of the result that cannot be validated automatically. These tests also give credibility to published results, since they reduce the likelihood that a result will later be found to be non-compliant.

The validation framework in Chauffeur allows validation checks to be performed at the beginning of the run, when appropriate, to provide the user with immediate feedback if the run is launched in a non-compliant manner. More extensive validation is performed at the end of the run, notifying the user of any non-compliant aspects of the run that were detected. The same set of checks is performed when the reporter runs; repeating the validation when the report is generated allows new validation checks to be added after the initial release to help catch common issues which are identified in early results – the new version of the reporter can be used with existing results to perform these validation checks.

In many cases, validators can detect that a run is definitely invalid. In other cases, warnings can be issued when there it is likely that a compliance issue exists but the validator can't know for sure. Chauffeur validators can also produce informational messages, such as notifying users of possible editorial issues.

Architecturally, Chauffeur's validation framework is independent of SERT, and new validation rules could be plugged in at run time. In the initial release, the set of validation rules is hard-coded for the SERT, but this limitation will be resolved in a future release.

Some sample validation checks that are included in SERT are:

- Checking that the workload code has not been modified or recompiled.
- Ensuring that configuration options are consistent with the run rules.
- Monitoring temperature measurements to confirm that the minimum ambient temperature requirements are met.
- Using checksums to detect manual modification of results after a run is complete.
- Confirming that performance requirements are met (e.g. that a 50% load level measurement has approximately 50% of the calibrated throughput)

These validation checks cannot completely prevent intentional cheating, they can prevent accidental mistakes by users and give a high degree of confidence that a result is valid.

5.2.7 Customizable Reporting

At the conclusion of a Chauffeur run, the results need to be summarized in a meaningful way to the tester as well as to reviewers and those looking at published results. These reports need to include information about the system configuration and test environment as well as the results themselves.

All of the results of a Chauffeur run are stored in an XML file. Even in a small environment this file can contain over 3 MB of data for a single run. This is far more data than can be usefully included in the standard report. Most users won't need to see data that is not part of the standard report, but for research purposes it is often useful to focus on non-default data.

The Chauffeur Reporter can process the results file from a run and produce a report in HTML and/or plain text formats. It also supports generation of CSV files (comma-separated values) to make it easy to import summarized results into spreadsheets or other applications.

The Chauffeur Reporter processes the results file using an XSL transform to either plain text (when generating CSV files) or to an XML-based representation of the report, which is then used to generate HTML and text reports. Custom XSL stylesheets can be passed to the reporter to generate custom reports including whatever data the researcher is interested in. For example, the SERT results shown in the next section were extracted from the raw results files using the reporter with a custom transformation and CSV output, which was then imported into R for statistical analysis. Customized reports will also be useful for future Chauffeur-based benchmarks which will likely summarize data differently than SERT does today.

The XML-based report representation includes high-level definitions of tables, graphs, and text sections. This generic report representation is independent of the final output format, and can be used to generate either HTML or plain text. It is likely that a future version of Chauffeur will also be able to generate PDF reports as well.

5.2.8 Portability

The Chauffeur framework is written in Java, and most of the current Chauffeur worklets are also written in Java to simplify portability.

Applications written in Java may have different performance and energy efficiency characteristics than applications written in other languages. Chauffeur-based benchmarks (or a future version of SERT) may include worklets written in other languages. The Chauffeur framework is designed to allow worklets (or portions of worklets) to be implemented in other programming languages. The storage worklets in SERT, for example, use native code to ensure that IO actually goes to disk and is not cached within the operating system or JVM [49].

Calls to native code can be implemented using the Java Native Interface (JNI). This approach results in both Java and native code executing during every transaction. An alternative approach would be to plug in an alternate implementation of Chauffeur's Interval interface which executes the entire interval in native code. Similarly, the transition to native code could be made at the Sequence, Phase, Worklet, or Workload level. A final approach would be to re-implement the entire Chauffeur Client in native code, while keeping the Java-based implementations of the Chauffeur Director and Host. The communications protocol used among the Chauffeur processes intentionally avoids dependencies on Java functionality (such as Java Serialization) in order to allow non-Java implementations to fully replace the Java Client implementation.

While these alternative approaches would require significant portions of Chauffeur code to be rewritten in another programming language, these native implementations would still be able to take advantage of large portions of the Chauffeur infrastructure, including the general control mechanisms, configuration support, communication protocols, listener infrastructure, and the reporter.

Chauffeur's support for launching client JVMs with affinity commands does require platform-specific code. If support for a particular platform is not available, Chauffeur will launch the JVM without affinitization, which may put that platform at a disadvantage. Affinity commands can be specified manually, and affinity support for new platforms can be added without modifications to Chauffeur itself.

6 Results and Analysis

The SERT was only recently released and there is not yet adequate public data to fully evaluate the quality of SERT. In this section, SERT results are presented to assess how well the SERT meets its design goals. Additional results are shown from experiments demonstrating how changes to the system configuration can affect the energy efficiency of the system as measured by SERT.

Results shown in this paper were obtained using the initial public release of SERT (1.0.0). The results are not fully compliant with SERT run rules, because power data was not collected using an accepted power analyzer, and temperature data was not collected. Some results may have been invalid for other reasons as well. Note that SPEC's fair use rules for SERT prohibit competitive comparisons between results obtained for research purposes (such as those in this paper) and results associated with an official energy efficiency program (such as ENERGY STAR).

6.1 Baseline Results

Baseline results were measured on an IBM x3550 M2 with two 4-core 2.93 GHz Intel Xeon X5570 processors, 32 GB RAM, and a single 300 GB 10K RPM SATA hard drive. Default UEFI firmware settings were used. The server was running Red Hat Enterprise Linux (RHEL) 6.3. The 64-bit IBM J9 7.0 SR3 JVM was used with no JVM command-line options apart from the heap settings made by SERT itself.

The system uses two redundant 675 watt power supplies. Because the power supplies are redundant, the total power consumption of the system is never greater than what one power supply can provide. During these tests, both power supplies were connected to separate circuits of an IBM DPI C13 PDU+ power distribution unit (PDU). This PDU can measure power consumption for each circuit. This device is not supported by SPEC PTDaemon, so a custom Chauffeur listener was written to collect data from the PDU during the run. The accuracy of the measurements may not be as good as a calibrated external power analyzer, but is sufficient for the relative comparisons shown here.

The graphs produced by the SERT reporter for one run in this configuration is shown in Figure 6. Each of the three graphs includes rows for each of the SERT worklets, grouped by workload – CPU worklets are highlighted in blue, memory worklets in salmon, storage worklets in green, and the hybrid worklet in pink. The Watts and Efficiency Score graphs include data points for each of the load levels. Since different worklets use different numbers of load levels, some worklets may show only a single data point while others may show as many as eight.

The Watts graph shows the average power used during each load level. The Normalized Performance shows a relative measure of the performance at the 100% load level for each worklet. The performance values are divided by the corresponding value from a reference system; this gives a common scale to the disparate worklets (whose raw performance results can vary by multiple orders of magnitude).

The Efficiency Score for each load level is calculated as the ratio of the

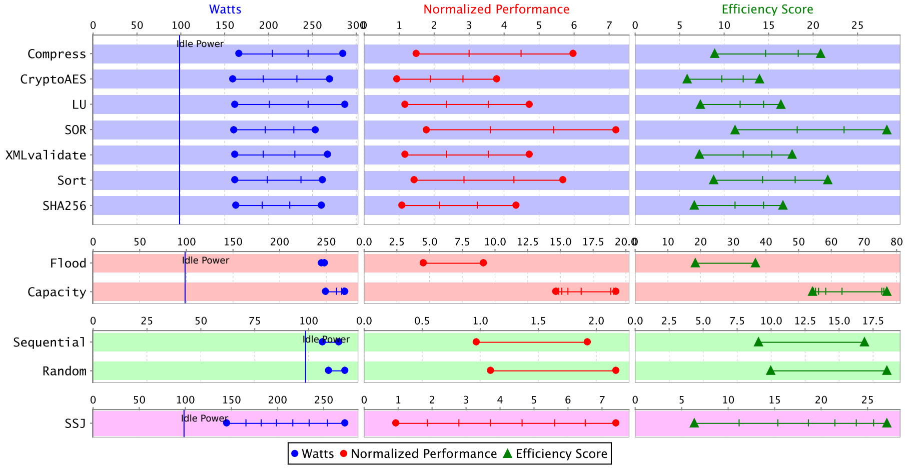


Figure 6: Baseline SERT Result

Normalized Performance value to the Watts value for that load level (converted to kilowatts). This is a measure of the efficiency (performance per watt).

6.1.1 Consistency

To assess the run-to-run consistency of the SERT results, 20 runs were performed on the same system with the baseline configuration. Figure 7(a) shows the distribution of the Performance results relative to the arithmetic mean result for each of the worklets in these runs. For the memory worklets each interval is shown separately, since these worklets vary the amount of memory used for each interval, resulting in possible differences in behavior in each interval. The interquartile range of the relative performance for all of the worklets other than Hybrid SSJ is within ±1% of the mean; aside from a few outliers, the overall range is within ±2% of the mean. The range for Hybrid SSJ is higher, at about ±2.5% for the interquartile range and ±3% overall.

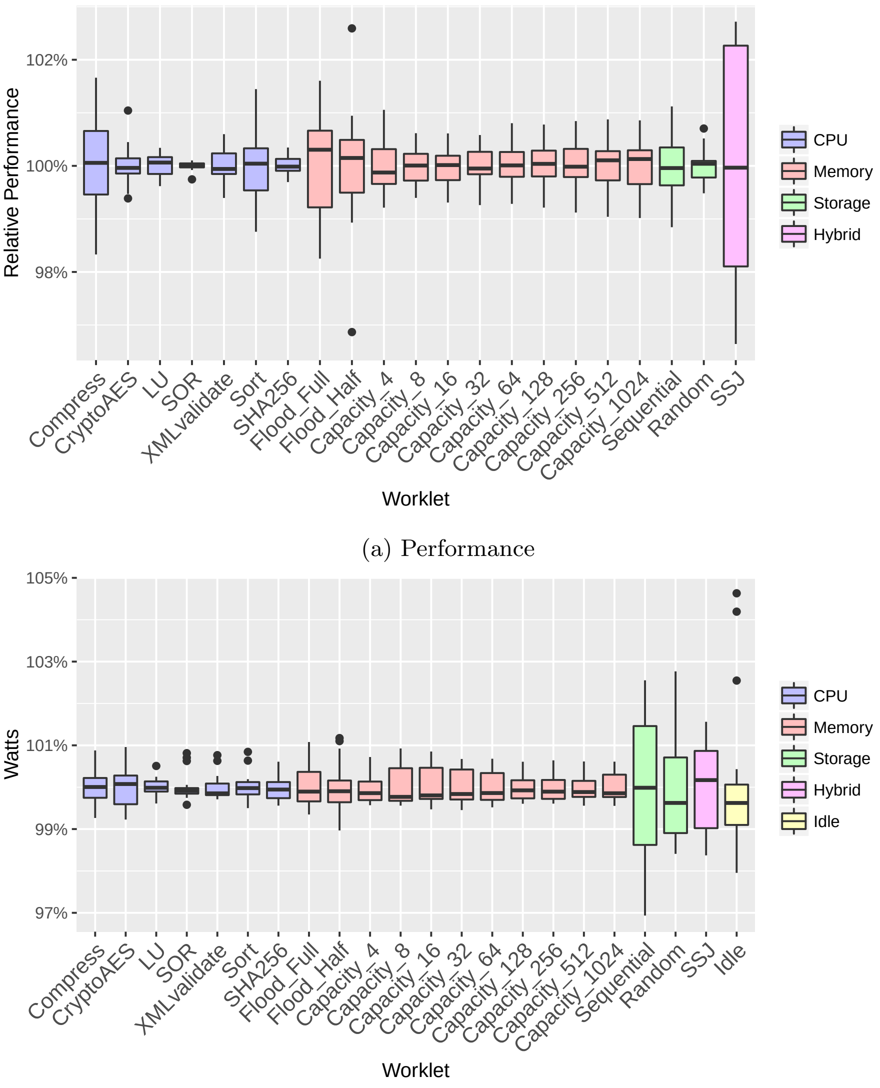
Similarly, Figure 7(b) shows the distribution of the watts values for each worklet relative to the mean values. These measurements have less variability, with the overall range within ±1% of the mean for most worklets. Worklets that have lower absolute power results (Storage and Idle) have a somewhat higher relative range of around ±3% of the mean.

Both performance and power consumption at the 100% load level are shown in Table 2, along with the coefficient of variation of these values among runs.

Workload	Worklet	100% Perf	CV Perf	100% Watts	CV Watts
CPU	Compress	9,472	0.94	286	0.38
CPU	CryptoAES	10,831	0.35	271	0.45
CPU	LU	16,456	0.21	288	0.22
CPU	SOR	21,488	0.08	254	0.33
CPU	XMLvalidate	5,294	0.30	268	0.29
CPU	Sort	27,764	0.71	262	0.32
CPU	SHA256	1,403	0.17	259	0.33
Memory	Flood_Full	519	1.03	247	0.49
Memory	Flood_Half	253	1.11	247	0.61
Memory	Capacity_4	804,489	0.47	250	0.41
Memory	Capacity_8	823,684	0.34	250	0.47
Memory	Capacity_16	817,705	0.32	252	0.44
Memory	Capacity_32	707,432	0.35	262	0.41
Memory	Capacity_64	665,176	0.41	266	0.40
Memory	Capacity_128	645,106	0.42	268	0.33
Memory	Capacity_256	634,773	0.46	269	0.34
Memory	Capacity_512	629,147	0.47	269	0.31
Memory	Capacity_1024	626,105	0.50	270	0.33
Storage	Sequential	157	0.53	118	1.74
Storage	Random	231	0.32	119	1.40
Hybrid	SSJ	479,384	2.36	276	0.99
Idle	Idle			100	1.79

Table 2: Mean and coefficient of variation (CV) of performance and power consumption at the 100% load level for each SERT worklet (n = 20 runs), running in an out-of-the-box configuration (default UEFI options and Baseline JVM tuning options).

Performance results and power consumption at the 100% load level are approximately Normally distributed, as shown by the Normal quantile plots in Figures 8 and 9. The Hybrid SSJ performance results showed bimodal behavior; this is discussed further in Section 6.2.1. It is worth noting that in an earlier set of data collected for this paper, the quantile plots showed bimodal behavior



(a) Performance

(b) Power Consumption

Figure 7: Distribution of the performance and power consumption at the 100% load level for each SERT worklet ($n = 20$ runs) relative to the mean for that worklet, running in an out-of-the-box configuration (default UEFI options and Baseline JVM tuning options).

for the Storage worklets as well. This was caused by leaving the disk controller cache enabled (in opposition to SERT requirements); some runs benefited more from the cache than others. Disabling the controller cache eliminated these bimodal results.

6.1.2 Multiple Workloads

The SERT uses multiple workloads to stress different components of the system, and most workloads consist of multiple worklets that exhibit different runtime behavior. Figure 10(b) shows the mean power consumption for each worklet at the 100% load level for 20 runs in the baseline configuration.

The CPU worklets had the highest power consumption in these tests, ranging from 254.5 to 287.8 watts.

The memory worklets use slightly less power than most of the CPU worklets. These worklets are intended to exercise the memory system; they also use CPU, but not as fully as the CPU worklets.

The Storage worklets consume less than half of the power of the other worklets, with power consumption 135.5 - 170 watts lower than the CPU worklets.

The Hybrid SSJ worklet is intended to exercise both CPU and Memory, and push the system to higher power consumption. In these baseline results, the Hybrid CPU power is among the highest of any worklet.

Idle power consumption is about 18.2 watts less than the Storage worklets, and 154.8 - 188.1 watts less than the CPU worklets.

6.1.3 Multiple Worklets

Each of the SERT workloads consists of one or more worklets that execute different business logic. Figure 10 shows the performance (normalized to the SERT reference system) and power consumption at the 100% load level for each of the worklets in the baseline runs.

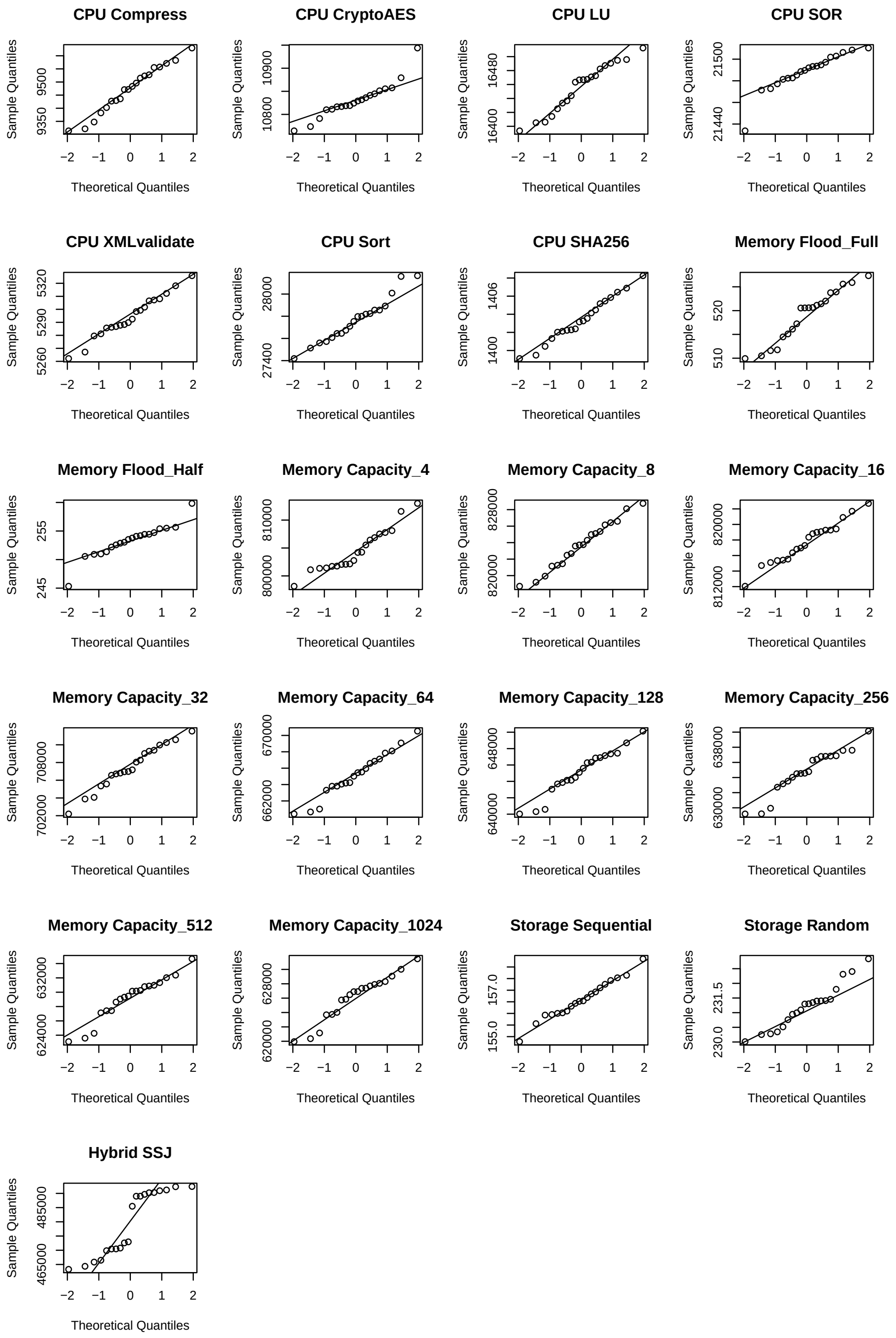


Figure 8: Normal quantile plots for the performance at the 100% load level for each of the worklets with out-of-box tuning options ($n = 20$ runs).

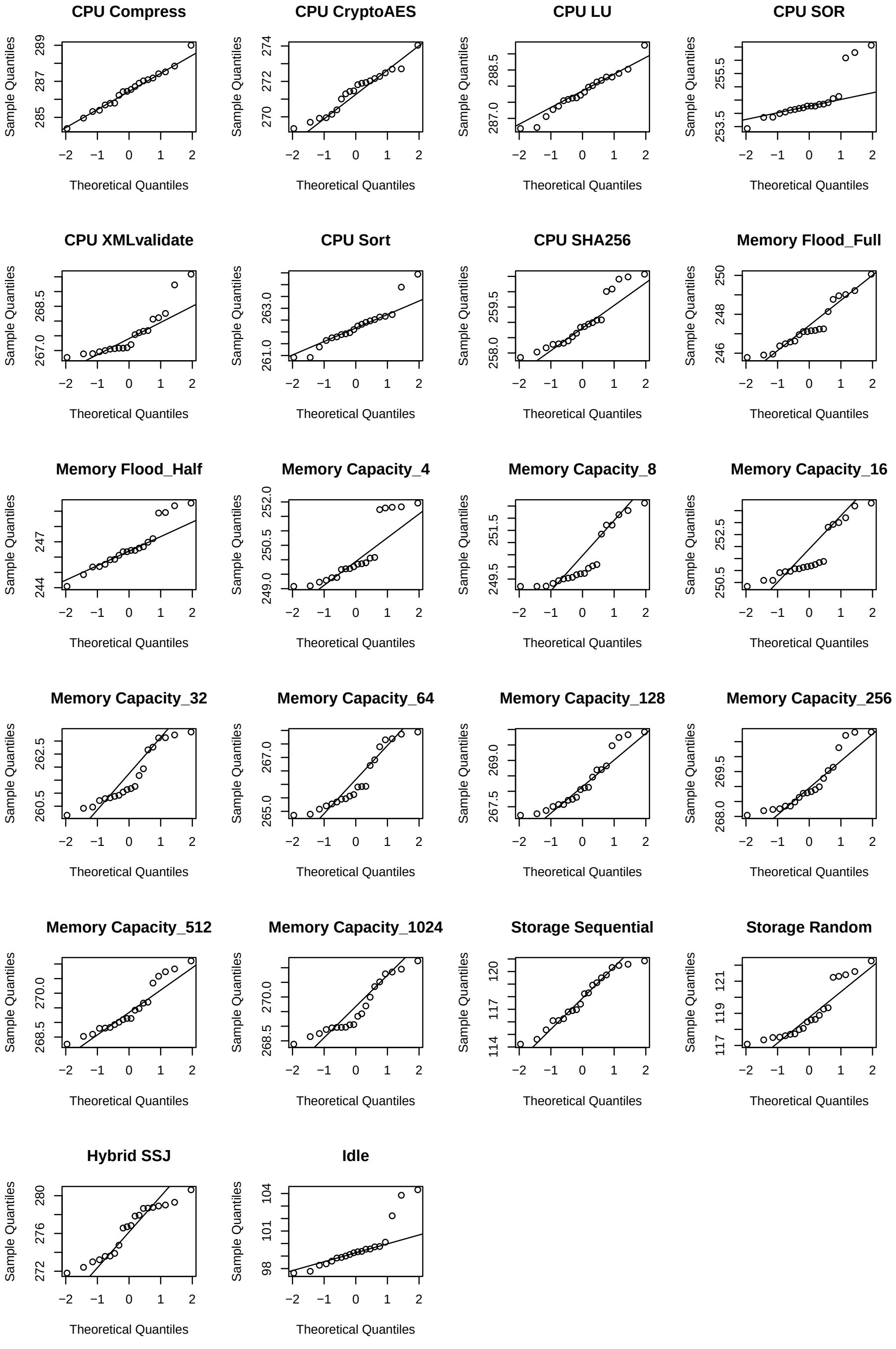
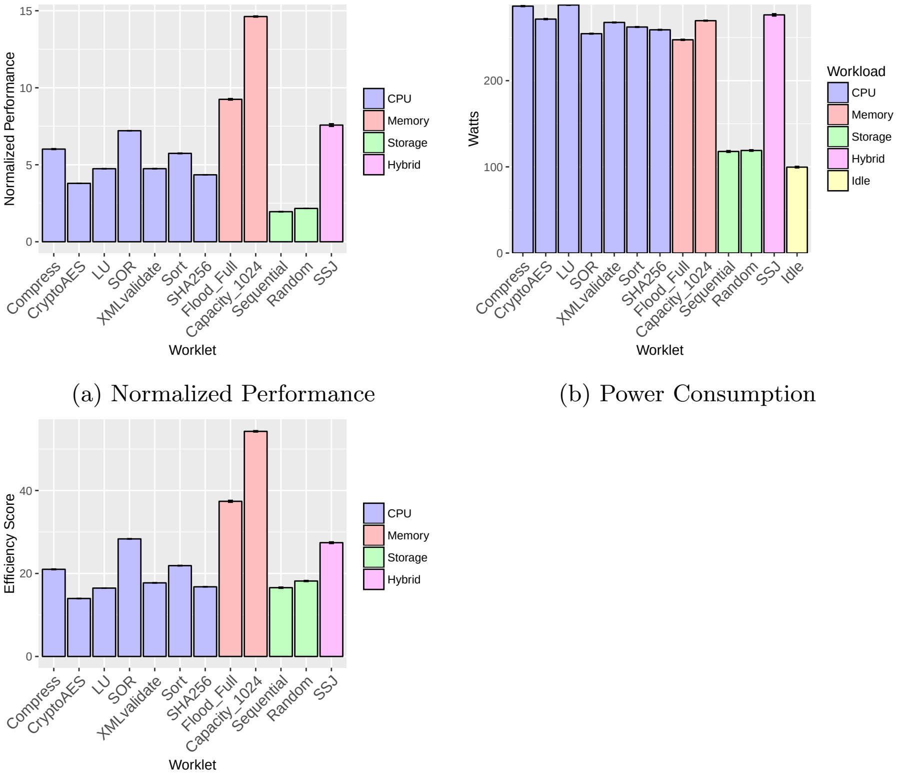


Figure 9: Normal quantile plots for the power consumption at the 100% load level for each of the worklets with out-of-box tuning options ($n = 20$ runs).



(a) Normalized Performance

(b) Power Consumption

(c) Efficiency Score

Figure 10: SERT results showing the power consumption and normalized performance at the 100% load level for each worklet in an out-of-the-box configuration (default UEFI options and Baseline JVM tuning options). Results are the mean of $n = 20$ runs; error bars show the 95% confidence intervals.

The need for multiple worklets is illustrated by the significant variation in the normalized performance scores and power consumption of each worklet within a workload. If the worklets in a workload exercised the system in the same way, they would be expected to have an equivalent performance score relative to the reference system, and the power consumption of each worklet would be the same. If this were the case, there would only be a need to measure one of these worklets.

The baseline data shows variation of 3.4 in the normalized performance scores and 33.3 watts in power at the 100% load level for the CPU worklets. These ranges are significant and demonstrate the importance of basing energy efficiency metrics on multiple worklets.

6.1.4 Multiple Load Levels

Each worklet is run at multiple load levels to show the ability of the system to reduce power consumption when the system is running at lower utilizations. The number of load levels for each worklet was chosen by SPEC to balance the value of the data against the run time required for each measurement interval. The CPU worklets run 4 load levels (at 100%, 75%, 50%, and 25% of the maximum throughput), the Storage worklets run 2 load levels (at 100% and 50% of the maximum throughput), and the Hybrid worklet runs 8 load levels (100% down to 12.5% of the maximum throughput, in 12.5% increments). The memory worklets vary the amount of memory used rather than the rate of memory usage: Flood runs at "Full" and "Half" levels, using nearly all of the available memory and about half of the available memory, respectively. The Memory Capacity worklet uses 9 data set sizes ranging from 4 GB to 1 TB; the system is able to make use of available memory to cache results so they don't have to be recomputed on demand.

SERT validity checks verify that the target throughput is achieved during the measurement interval, within a threshold of ±2% (or up to −4% for the 100% load level, since minor fluctuations in throughput can cause the target to be missed at high utilizations). The target and actual percentage for each load level in one of the baseline runs is shown in Table 3. All of the load levels are well within the limits required by the SERT validation. The memory worklets are not included in this table since they vary memory usage rather than transaction rates.

Workload	Worklet	100.0%	87.5	75.0	62.5	50.0	37.5	25.0	12.5
CPU	Compress	99.3		75.0		50.1		25.0	
	CryptoAES	99.9		74.9		49.9		25.0	
	LU	99.6		75.1		50.0		25.0	
	SOR	99.9		75.0		50.0		25.0	
	XMLvalidate	100.0		75.0		50.0		25.0	
	Sort	99.6		74.9		50.0		25.1	
	SHA256	99.7		74.8		50.0		25.1	
Storage	Sequential	99.2				49.8			
	Random	99.2				49.7			
Hybrid	SSJ	99.2	87.2	75.1	62.5	49.8	37.6	25.0	12.4

Table 3: Actual percentage throughput for each load level for one SERT run in an out-of-the-box configuration (default UEFI options and Baseline JVM tuning options).

On the configuration used for the baseline results, power consumption changed roughly linearly with the throughput for the CPU and Hybrid worklets. This can be most clearly seen in the Hybrid SSJ results in Figure 11, where the values at each load level are evenly spread between the maximum (100%) and minimum (12.5%) load levels. There is, however, a more significant reduction in power consumption between the 12.5% load level and the Idle Power. The idle measurement period allows the system to enter a lower performance state, and some system components may be in a sleep state while the application is not actively executing transactions.

This linearity of power consumption is a characteristic of the system under

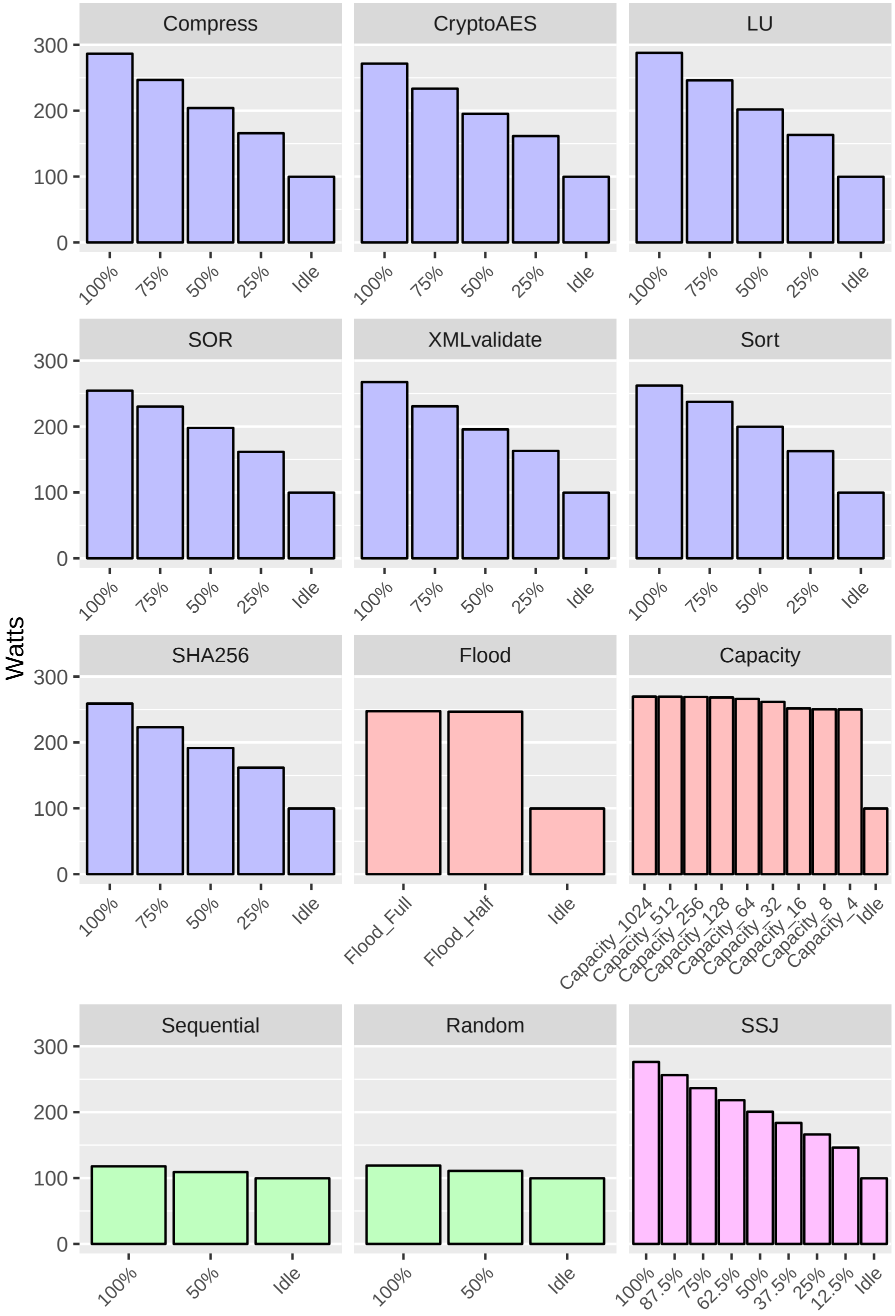


Figure 11: Power consumption at each load level for each SERT worklet with out-of-box tuning options. Values are the mean of $n = 20$ runs.

test and its configuration. Other systems may show larger changes in power consumption at some load levels than others. This was already exhibited in the two SPECpower_ssj2008 results shown in Figure 4. The configuration in Figure 4(a) has a fairly linear power consumption curve (shown in the blue line), while Figure 4(b) has a more concave curve where the system power changes more drastically at higher load levels than at low utilizations.

Each of the CPU worklets has very similar power consumption at low utilizations, and these match closely to the corresponding load levels for Hybrid SSJ as well. There is somewhat more variation at the 75% and 100% load levels.

The memory worklets show very little variability in power consumption in this environment. Because different measurement intervals of Flood access different amounts of memory, the main opportunity for systems to reduce power consumption in Flood_Half is to reduce the power of the memory that is not in use. In Capacity, there is some variation in the amount of memory accessed, but the main difference is in the percentage of transactions which can be satisfied using the in-memory cache of results. This results in better performance for smaller data set sizes which can be kept completely or mostly in the system's physical memory.

The Storage worklets in this environment already run near the idle power, so there is little opportunity for power reduction. This is the main reason why the storage worklets run fewer load levels than the other worklets. Systems with larger numbers of disks may have higher power requirements to exercise those disks, so the power savings at low utilizations may be greater in these environments.

6.2 Effects of Tuning

The baseline results above used default UEFI settings, operating system configuration, and JVM tuning parameters. In benchmark results, changes to these settings are usually made in order to optimize performance or energy efficiency. In the SERT, tuning is limited in order to limit "super-tuning", promote comparability among results, and more closely measure out-of-the-box configurations. This section shows the impact of changing optimization parameters on both performance and power consumption.

UEFI Firmware and the JVM both provide a large number of options, and an exhaustive search for the best combination of options is impractical. Appropriate settings were identified by finding SPECjbb2005 and SPECpower_ssj2008 results from similar configurations[19].

6.2.1 JVM Tuning Options

The baseline results described in Section 6.1 used no JVM tuning options aside from those set automatically by SERT (e.g. the initial and maximum heap size). The SERT run rules require compliant runs to use a pre-defined set of options depending on processor, operating system version, JVM version, and the amount of memory available. The list of options only includes current processor versions, so the system used in these tests has no defined options. However, at the time of this writing all of the Intel processors have identical options defined when running in Linux using IBM J9. These "Standard" options are listed in Table 4.

The results referenced above for both SPECjbb2005 and SPECpower_ssj2008 used the same JVM tuning options. These "Optimized" options are also listed

[19] See `http://www.spec.org/osg/jbb2005/results/res2009q2/jbb2005-20090512-00722.html` and `http://www.spec.org/power_ssj2008/results/res2009q2/power_ssj2008-20090519-00164.html`

Table 4: JVM Tuning Options Used

Configuration	Options
Baseline	*none*
Standard	`-Xaggressive -Xcompressedrefs -Xgcthreads2`

Optimized	`-Xaggressive -Xcompressedrefs -XlockReservation`
	`-Xnoloa -XtlhPrefetch -Xgcthreads4`

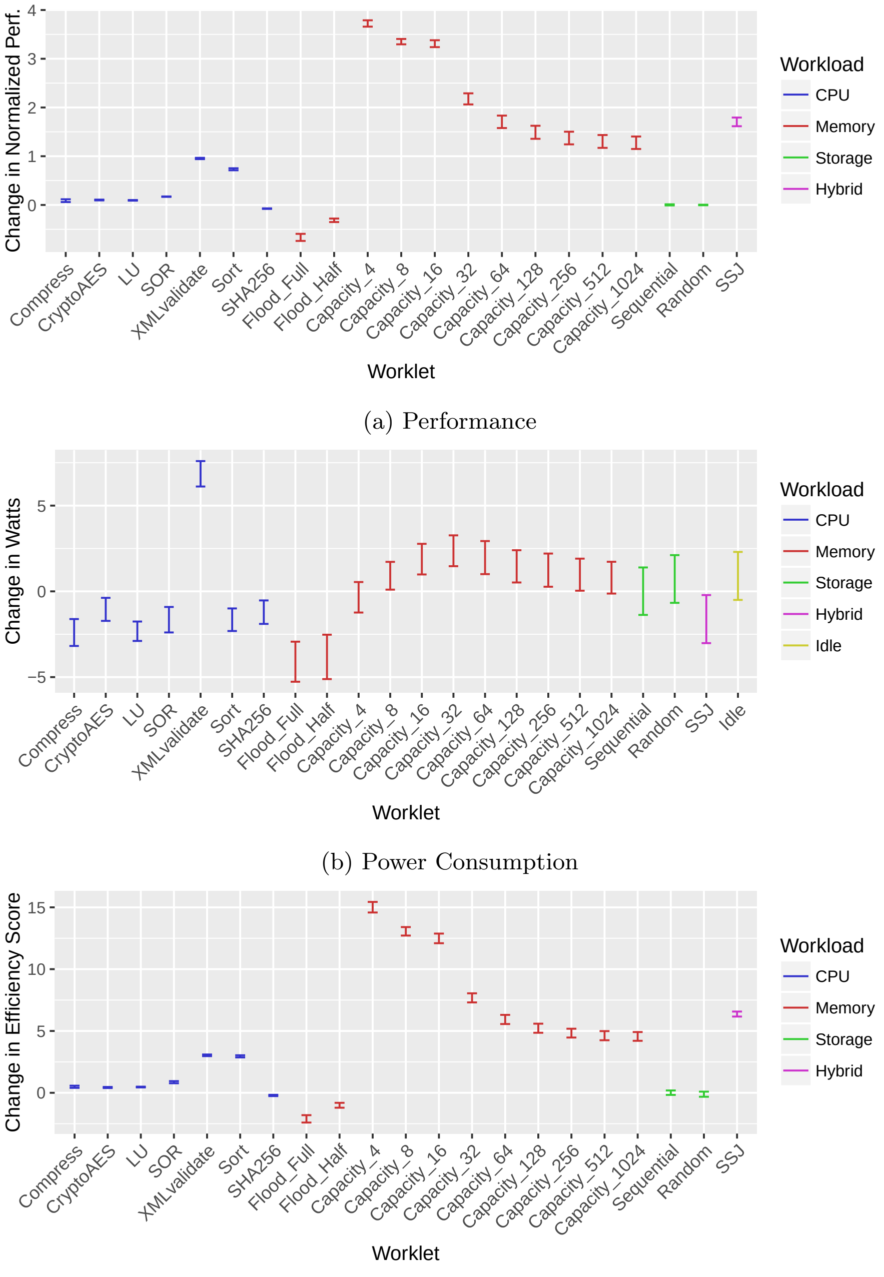
in Table 4. The options used by these published results included `-Xlp` to enable large-page support. The "Optimized" options used in this paper do not explicitly enable large page support because RHEL 6.3 includes support for Transparent Huge Pages which has a similar effect without manual configuration.

To assess the impact of JVM tuning on SERT results, runs were performed using both the Standard and Optimized options on the same system that was used for the baseline results. Following SERT conventions, no tuning options were used for the Storage or Idle workloads.

Figure 12 shows the difference in the mean performance, watts, and efficiency score at the 100% load level for each worklet as a result of using the Standard JVM options instead of the Baseline options. Each graph shows the 95% confidence interval from a two-sided t test.

The Standard JVM tuning parameters resulted in substantial performance improvements to the XMLvalidate, Sort, Capacity, and SSJ worklets, as can be seen in Figure 12(a). Flood performance declined, while other worklets had little or no change. The performance improvement was most significant for the small sizes of the Capacity worklet, where most results can be returned from an in-memory cache rather than being recomputed.

Most of the CPU worklets consumed slightly less power when using the Standard options, as shown in Figure 12(b). The largest change in power consumption was for the XMLvalidate worklet, which increased by 6.9 watts. Power



(a) Performance

(b) Power Consumption

(c) Efficiency Score

Figure 12: SERT results showing the change in performance, power consumption, and efficiency score at the 100% load level for each worklet using SERT Standard JVM options compared to the Baseline configuration. Results are the 95% confidence intervals from a t test comparing 10 runs with the Standard configuration to 20 runs in the baseline configuration.

usage declined slightly for Flood and increased slightly for some intervals of Capacity. Other worklets had little or no change.

Figure 12(c) shows that the efficiency score calculated by SERT for the 100% load level (the ratio between the Normalized Performance and the Watts at that point, multiplied by 1000) changed with a shape similar to the performance scores, due to the large change in performance relative to the change in power.

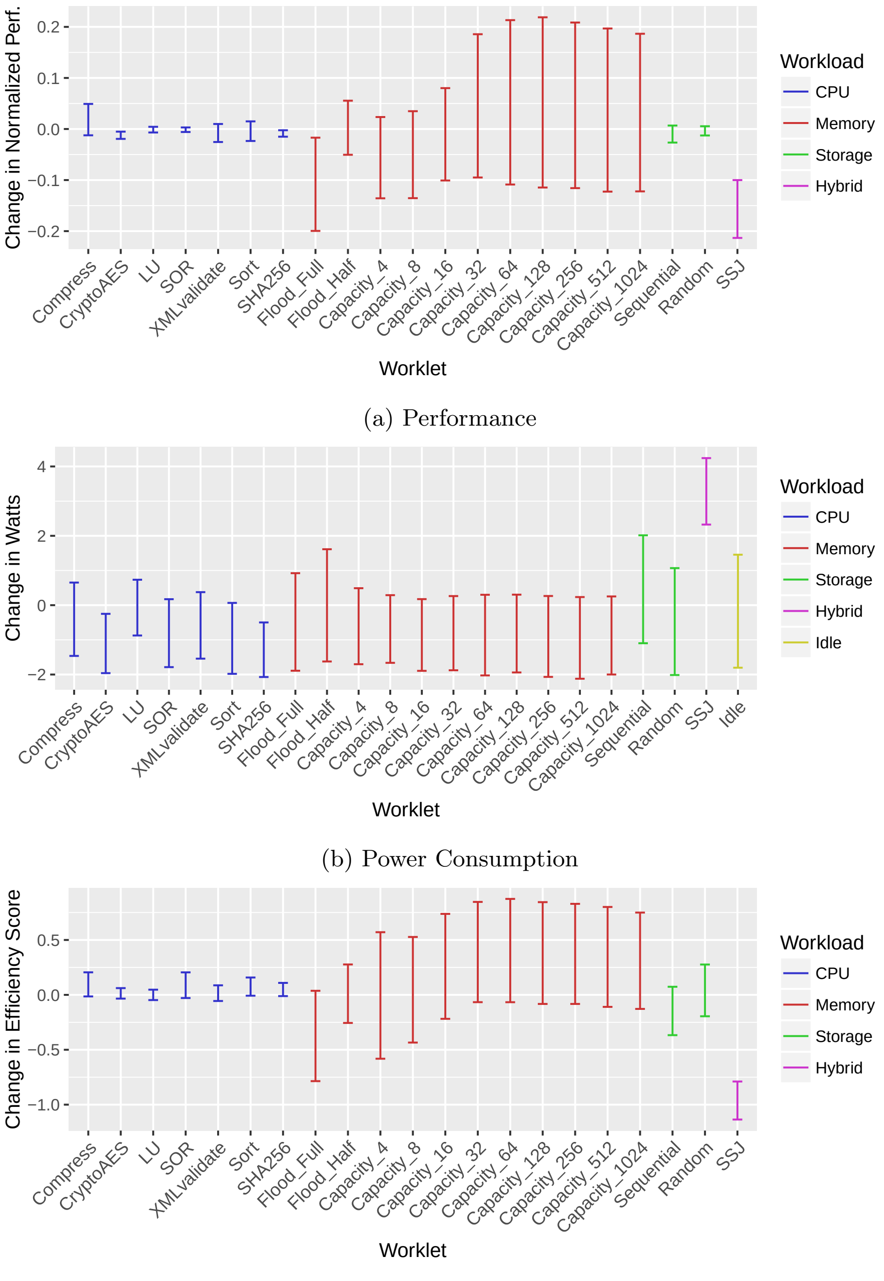
Using the Optimized JVM settings from Table 4 resulted in very little change compared to the Standard options. Figure 13 compares these two configurations. Even where there were statistically significant changes in the performance, the differences were too small to be important. There was also little change in the power consumption, except in SSJ where the power rose by 3.3 watts. This resulted in a small decrease in the Efficiency Score for SSJ, while the other worklets had no statistically significant difference. Therefore, it appears that the SERT Standard options provide good performance for this configuration.

The SERT Standard JVM Options were chosen to provide run-to-run consistency as well as performance. In Section 6.1.1 we saw that with baseline CPU options the results of the Hybrid SSJ worklet were bimodal, with some runs having performance significantly better than others. When using the Standard options the results were more Normally distributed, as shown in Figure 14.

6.2.2 UEFI Settings

The SPECjbb2005 and SPECpower_ssj2008 runs referenced above used different UEFI firmware settings. It is assumed that the SPECjbb2005 configuration was tuned for performance, while the SPECpower_ssj2008 environment was tuned for energy efficiency. The settings used for Default (baseline), Performance-Optimized, and Energy-Efficient configurations are shown in Table 5. Results in this section used the Baseline JVM options.

Using performance-optimized UEFI settings yielded small but statistically

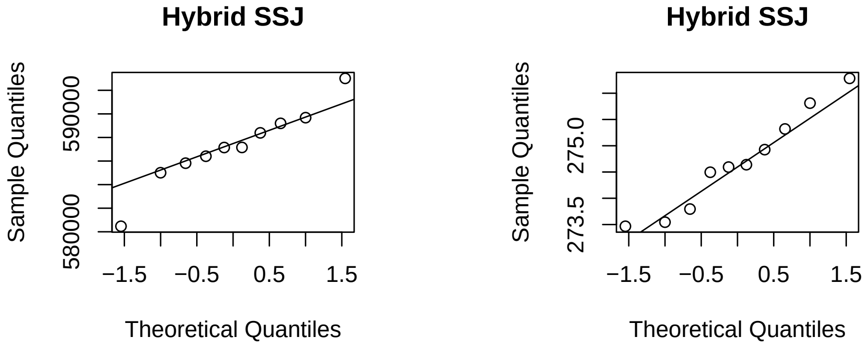


(a) Performance

(b) Power Consumption

(c) Efficiency Score

Figure 13: SERT results showing the change in performance, power consumption, and efficiency score at the 100% load level for each worklet using Optimized JVM options compared to the SERT Standard options. Results are the 95% confidence intervals from a t test comparing 10 runs with the Optimized configuration to 10 runs in the Standard configuration.



(a) Performance

(b) Power Consumption

Figure 14: Normal quantile plot for the performance and power consumption of Hybrid SSJ at the 100% load level with SERT Standard JVM options ($n = 10$ runs).

Table 5: UEFI Tuning Options Used

Option	Default	Performance	Energy Efficient
Turbo Mode	Disable	Enable	Disable
Power C-States	Disable	Disable	Enable
C1 Enhanced Mode	Enable	Disable	Enable
Cache Data Prefetch	Enable	Disable	Disable
QPI Link Speed Select	Max Perf.	Max Perf.	Min Power
Memory Speed	Max Perf.	Max Perf.	Min Power
Demand Scrub	Enable	Disable	Enable
Onboard Slot 1	Enabled	Enabled	Disabled
Onboard Slot 2	Enabled	Enabled	Disabled

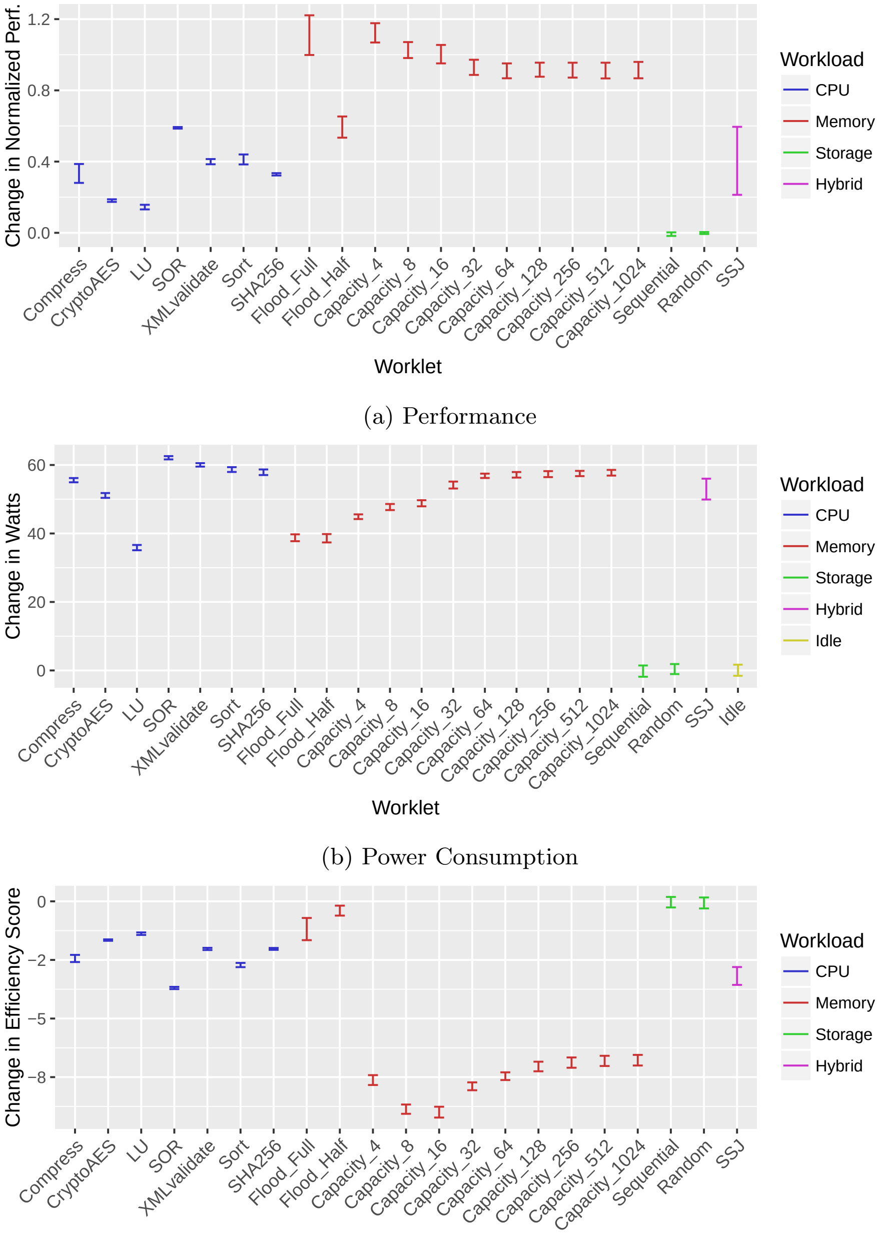
significant improvements to performance for the CPU and Hybrid worklets, with slightly larger improvements to the memory worklets, as shown in Figure 15(a). However, Figure 15(b) shows that this improvement came with an increased power consumption of 35.9 to 62.1 watts for these worklets. In Figure 15(c) we see that this resulted in a net loss in efficiency at the 100% load level.

The high power consumption at the 100% load level could be a reasonable trade off if it is only significant at the 100% load and the power consumption at lower utilizations is similar to the baseline. This might be the case if the system entered a "turbo mode" to increase performance when the system is fully utilized but operated similar to the default configuration at lower utilizations. However, Figure 16 suggests that this is not the case, since the power consumption is significantly higher than the baseline at all utilizations.

The use of Energy Efficient UEFI options had almost no impact on the performance of the CPU and Storage worklets relative to the Default settings (Figure 17). There was a small but statistically significant reduction in performance of the Hybrid and Memory Capacity worklet, and a more significant reduction (24.1%) for the Memory Flood worklet.

These options resulted in broad reductions in power consumption. The Idle and Storage worklets (which run with near-idle power) improved by around 8.1 watts. The Memory Capacity and most of the CPU worklets improved by 12.1 - 13.8 watts, while CPU Compress had a slightly larger reduction of 16.4 watts. Hybrid SSJ improved by 19.2 watts. The greated reduction in power was for the Flood worklet (21.4 watts).

The reduction in power consumption resulted in a small improvement in the Efficiency Score for most worklets. The significant improvement in power usage in Flood was insufficient to overcome the decline in performance, resulting in a lower Efficiency Score.



(a) Performance

(b) Power Consumption

(c) Efficiency Score

Figure 15: SERT results showing the change in performance, power consumption, and efficiency score at the 100% load level for each worklet using Performance-Optimized UEFI settings compared to the Default settings. Results are the 95% confidence intervals from a t test comparing 10 runs with the Performance-Optimized configuration to 20 runs in the Default configuration.

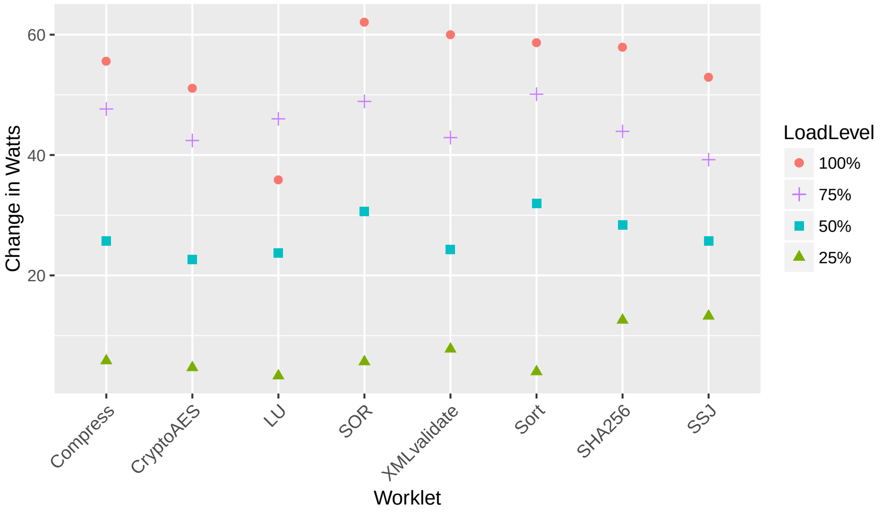
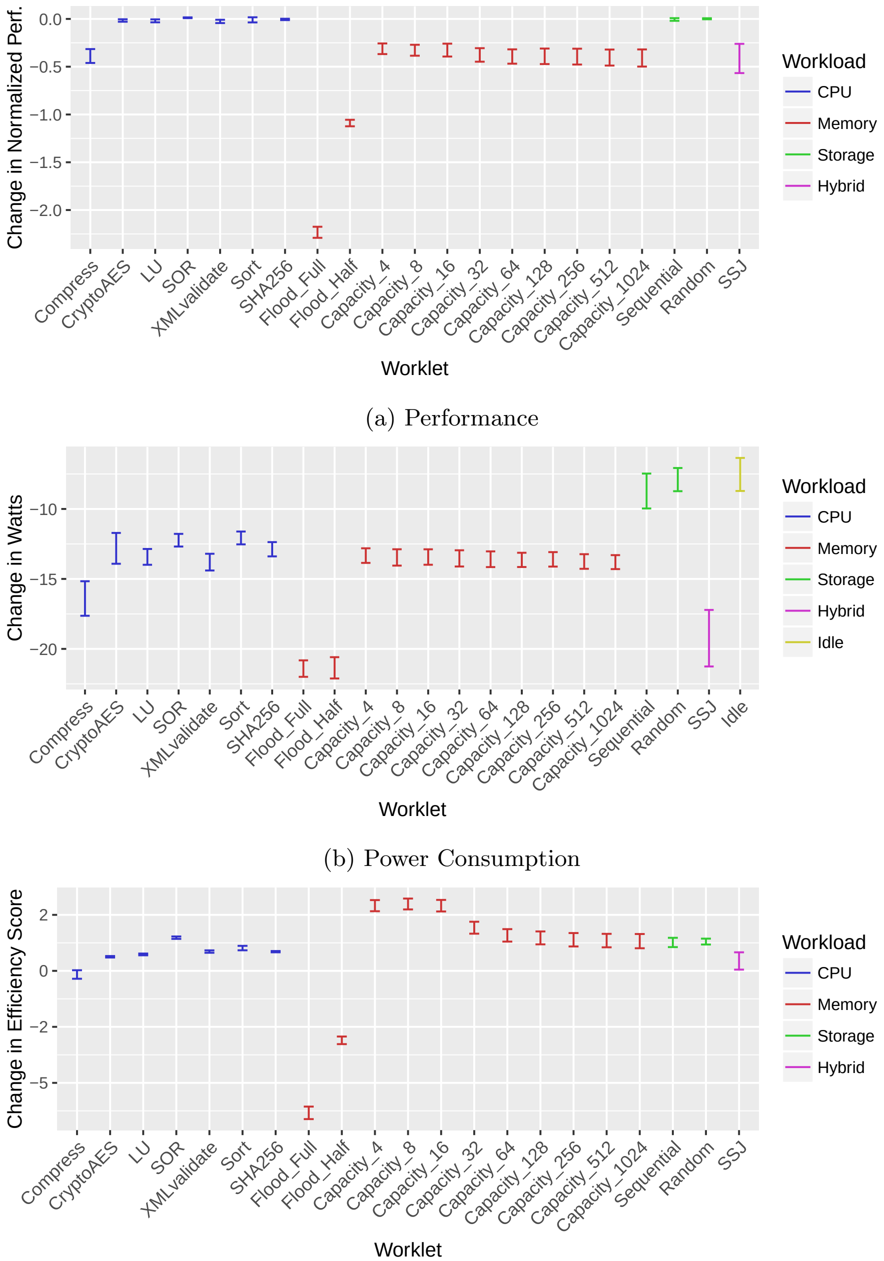


Figure 16: SERT results showing the change in power consumption at multiple load levels for CPU and Hybrid worklets using Performance-Optimized UEFI settings compared to the Default settings. Results are the mean difference from 10 runs with the Performance-Optimized configuration and 20 runs in the Default configuration.

6.2.3 Combined Effects

Use of the SERT Standard JVM options improved energy efficiency for most worklets by improving performance with minimal impact on power consumption. Using Energy Efficient UEFI options improved efficiency by reducing power consumption with minimal impact to performance for most worklets. Figure 18 shows the combined effect of using both the Standard JVM options and Energy-Efficient UEFI settings.

This combination of tuning yields performance improvement relative to the baseline configuration for nearly all worklets, with the most notable exception being Flood. The performance was slightly less than what was achieved with the Standard JVM options using the Default UEFI settings. Power consumption was reduced relative to the baseline configuration for all worklets; in many cases the reduction in power was greater than when the Energy Efficient UEFI

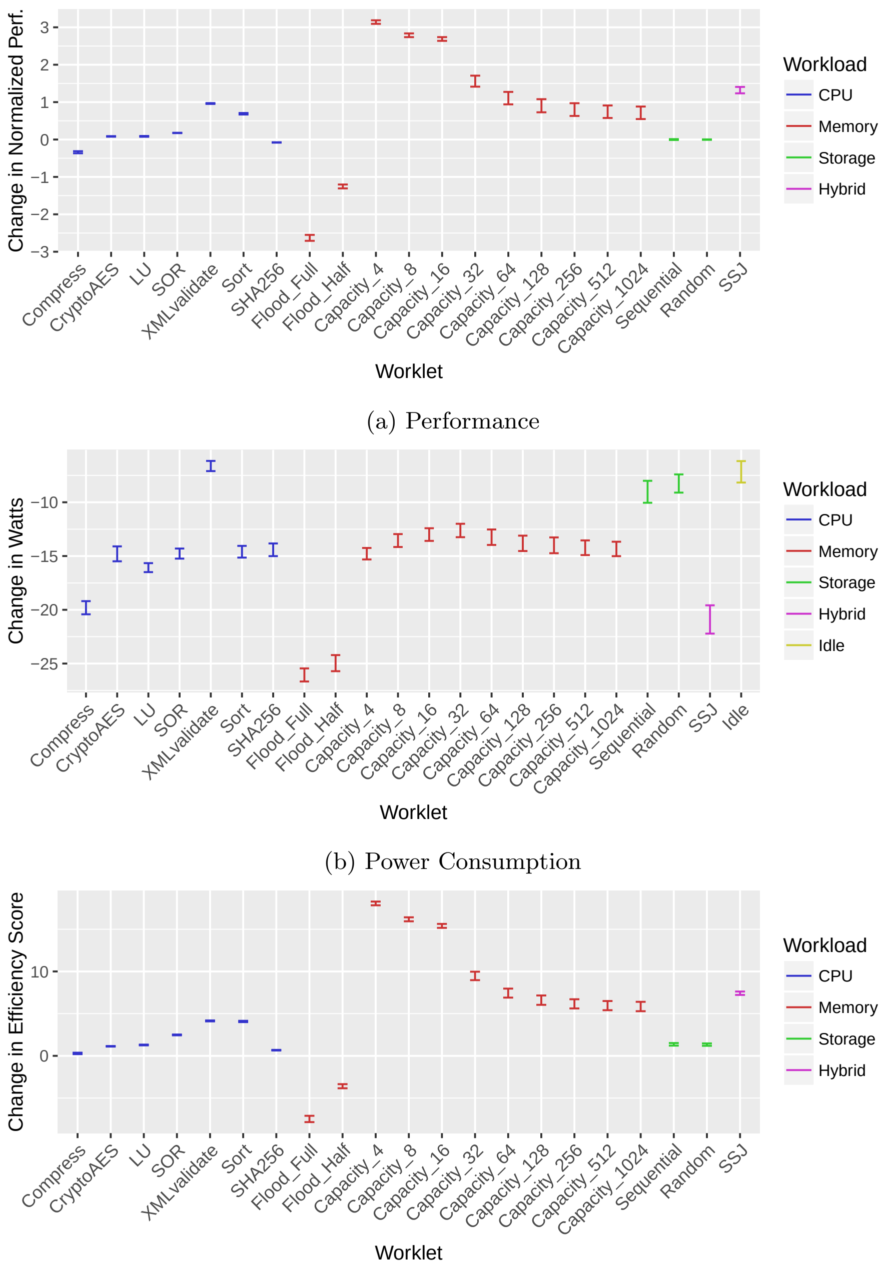


(a) Performance

(b) Power Consumption

(c) Efficiency Score

Figure 17: SERT results showing the change in performance, power consumption, and efficiency score at the 100% load level for each worklet using Energy Efficient UEFI settings compared to the Default settings. Results are the 95% confidence intervals from a t test comparing 10 runs with the Energy Efficient configuration to 20 runs in the Default configuration.



(a) Performance

(b) Power Consumption

(c) Efficiency Score

Figure 18: SERT results showing the change in performance, power consumption, and efficiency score at the 100% load level for each worklet using Energy-Efficient UEFI settings and SERT Standard tuning options compared to the Baseline configuration. Results are the 95% confidence intervals from a t test comparing 10 runs with the Energy Efficient/Standard configuration to 20 runs in the Baseline configuration.

settings were used with the Baseline JVM options. Overall, this combination of tuning resulted in better energy efficiency than the baseline configuration. In most cases, the improvement in the Efficiency Score was greater than when any configuration change was made in isolation.

6.2.4 Evaluation of the Impact of Tuning

The previous sections have shown that tuning can affect worklets differently. Options that help one worklet may have little or no impact on others, and in some cases may hurt the performance or power consumption of other worklets. SERT encourages generally-applicable tuning options by requiring the same set of options to be used for all of the worklets in a workload.

Performance-only benchmarks encourage highly tuned environments; vendors use a variety of tuning options to improve their results. In some cases, particularly with UEFI settings, these performance tuning options may increase power consumption disproportionately with the performance gain. For this reason, performance tuning options should not be set blindly without assessing their impact on the applications that will actually be running on the server; they may increase power consumption without improving performance.

This also illustrates a challenge for benchmarks that have an “optional” energy efficiency benchmark. This often results in energy usage not being reported for highly tuned environments that yield the best performance, while environments tuned for energy efficiency have worse performance. Requiring energy measurements for all submissions of a benchmark still allows testers to optimize their system for either performance or energy, but provides both metrics so that the consumers of the results can balance the two factors appropriately depending on their needs. For similar reasons, an energy efficiency benchmark should report the performance as well as the efficiency so that users can determine whether or not the system can deliver the required performance.

6.2.5 Nameplate Power

Section 2.4 described the difficulties of estimating power consumption of servers based on their nameplate power ratings. The server used in these tests has a nameplate rating of 675 watts; however, the maximum average power within an interval for any worklet in all of these tests was only 342.9 watts. Clearly, using the nameplate power would have been a poor substitute for measuring the power consumption, even when looking only at the peak power.

6.3 Quality of the SERT

While the SERT is not intended to be a benchmark, its design goals are consistent with most of the design criteria for energy efficiency benchmarks described in section 3. It is appropriate, therefore, to evaluate the SERT based on these criteria. Since the SERT has only been available for a short time, a full evaluation is premature, but a preliminary evaluation can be made based on the results above, as well as the SERT Design Document's [50] descriptions of the tool. The quality of SERT should be reevaluated after there has been sufficient use to assess the quality of the results.

Relevance The SERT contains a suite of worklets, including some designed to stress CPU, memory, and disk storage. Each worklet is simple, often repeating a small set of operations on randomly-generated data; therefore, the relevance in any specific application area is limited, but the SERT does give a broad overview of the system's energy efficiency.

Scalability was an important design goal for the SERT. This scalability is largely achieved through spreading work across multiple JVMs and using platform-specific affinity commands to pin each JVM to a particular set of processors to minimize shared resources. Each client JVM runs multiple threads, but there is little interaction among the threads, resulting in low rates of contention. The memory worklets were designed to take advantage of up to 1 TB of memory. The storage worklets use a separate client JVM for each physical disk, and is designed to scale to virtually any number of disks contained within the system enclosure.

A key feature of the SERT is that it runs each type of worklet at multiple load levels, providing opportunities for systems to reduce power consumption at low CPU utilizations as well as low memory and disk utilizations.

The SERT run rules have many similarities to SPECpower_ssj2008 run rules; in particular, it includes detailed requirements for acceptable power analyzers and the system configuration.

Reproducibility Section 6.1.1 discussed the run-to-run consistency of SERT results. These results show that (at least in this environment) SERT delivers consistent results in each valid run for both performance and power.

Descriptions of the hardware and software environment used for the SERT are similar to SPECpower_ssj2008, which has proven itself to be sufficient for reproducing results. The ability to automatically collect certain environmental details from the system at runtime helps improve the accuracy of this data in SERT results.

Because SERT launches client JVMs directly and uses automatic heap size calculations and affinity commands, many details about the client JVM configuration are known and can be reported directly without user involvement.

One aspect of the SERT that may prove to be problematic for reproducibility is the requirement to disable any cache on disk controllers used by the Storage worklets. Different disk controllers provide different interfaces for configuration and for detecting whether any cache was disabled, making it difficult for the SERT to automatically determine whether this requirement has been followed.

Fairness The SERT was designed and implemented by a SPEC committee consisting of several server vendors as well as participation by individuals with no current corporate affiliation. Input was obtained from the US EPA and other organizations, who in turn collected feedback from other interested parties. A draft SERT Design Document [50] was published publically early during the development process, and many aspects of the design were presented to the research community during development as well [52, 49]. The Beta-2 release was made available to interested parties and feedback was obtained from these early users. These actions were taken in order to give ample opportunity for interested parties to influence the design of the SERT in order to produce a tool that was a fair measure of server energy efficiency.

Few technical restrictions are placed on the servers that can run the SERT, though the EPA and other organizations may place their own restrictions on what results they will accept. The SERT does require a 64-bit platform, a compliant Java Virtual Machine implementation, and at least one storage device. The hardware is generally required to be in an "as-shipped" configuration, limiting the ability for testers to publishing results with "benchmark special" hardware that would not be used in real-world environments.

Currently, only Windows, Linux, and AIX operating systems are supported. Support for additional operating systems and hardware platforms may be added in the future if sufficient testing can be performed to validate these configurations. A small amount of porting effort would be required in order to support automatic processor affinity and for the storage worklets to run properly.

The SERT application may not be modified or recompiled, and SPEC has defined a process to restrict the JVM command-line options. The intent of these restrictions is to avoid favoring particular hardware architectures while limiting the ability to use super-tuned configurations which are not representative of real-world environments.

Verifiability The SERT is predicted to be of much broader interest to systems vendors than past energy efficiency benchmarks due to its inclusion in the ENERGY STAR program. This may lead to a greater number of results obtained by smaller vendors who may not have extensive skills with configuring and running benchmarks. This makes verifiability even more important for SERT than it is for most benchmarks.

A number of validation checks are performed at the conclusion of each run in order to ensure that the run rules were followed. While these tests cannot fully validate compliance with SERT run rules, a result that passes all of the validity checks is highly likely to be an accurate result. The validity checks generally cannot, however, confirm that the system configuration was described properly, or that system tuning was performed in accordance with the SERT run rules. The raw results files produced by the SERT do include some amount of automatically discovered system configuration data; while most of this data is not used to generate the report, a reviewer can use this data to ensure it is consistent with the manually-reported system configuration details.

The most basic SERT validity checks ensure that a compliant configuration file was used, and that the results file was not modified after the run was complete (aside from system configuration details which are allowed to be edited). Other tests confirm that target load levels were met within a suitable tolerance level, and that there is good consistency in the results from multiple Client JVMs and among multiple hosts.

Data collected from power analyzers is validated in accordance with the guidelines established in the SPEC Power and Performance Benchmark Methodology [24], using checks very similar to those in SPECpower_ssj2008. For example, analyzers must be on SPEC's list of approved analyzers, following a review of the technical specifications and testing to confirm that the device meets the accuracy requirements defined by SPEC. The device is required to have been calibrated within one year prior to the test. PTDaemon calculates the level of uncertainty in the measurements according to manufacturer specifications, and checks are performed to ensure that the uncertainty in the measurements is within established limits.

While the standard reports produced by SERT show a small amount of summarized data, the raw results include more extensive details about the environment. These details can be used to detect any inconsistencies; a lack of inconsistencies in this data gives additional credibility to the results.

Usability The SERT is a complex suite of workloads, but SPEC has taken several steps to make it as easy to run as possible. Feedback from Beta-2 users has been incorporated into the final release to resolve common challenges through changes to the behavior, improved error handling, and more detailed documentation. Additional improvements to usability are likely to be made in future updates.

The graphical user interface provides a simpler way for both new and experienced users to run the tool. It guides the user through a sequence of steps to ensure that the configuration is correct before beginning the run. It can automatically launch the SERT Director and SPEC PTDaemon instances; the user must start a SERT Host process on the system under test, but this requires very little configuration, and the Host can be reused in multiple test runs. For advanced users, the SERT can also be launched manually via the command-line.

Validity checks give the user confidence that their runs are configured properly and meeting the run rules established by SPEC.

Power consumption is measured using an external power analyzer communicating with SPEC PTDaemon. Once the connection to PTDaemon is established, Chauffeur ensures that measurements are started and stopped at the proper times.

A common challenge in collecting accurate data from a power analyzer is ensuring that the analyzer is set to an appropriate range. If the measured values are outside of this range, the analyzer may not be able to measure them at the level accuracy required by SERT. Manually setting ranges can be quite difficult with SERT, since the correct range can change for each load level, and different worklets may require different range settings. To simplify this process, the SERT includes a procedure for doing an abbreviated run which can be used to detect the proper range settings for each interval in each worklet. The result of this procedure is a group of settings that should be used for the full runs. While this procedure may not work perfectly in all environments, it appears to work well for most systems and requires little knowledge or effort from the user.

7 Future Work

The initial development of Chauffeur has been focused on meeting the needs of SERT. Run rules for SERT intentionally limit flexibility in how SERT and Chauffeur are used for compliant results; this is necessary for meeting the goals for SERT of giving a broad measure of server efficiency with minimal tuning. However, this paper describes the potential for utilizing Chauffeur-based workloads for future research.

Obvious extensions to Chauffeur which have already been mentioned include the introduction of new worklets (including worklets not written in Java) and new Listeners to automatically collect additional data during a run.

Ease of use is another area for additional improvement. For users, selecting the correct tuning options for their configuration can be a challenge; additional enhancements to Chauffeur and SERT can help with selecting appropriate options automatically. For researchers, additional documentation is needed to provide information about how Chauffeur can be adapted to their needs.

Other future enhancements could include improved support for heterogeneous workloads. This could include a “commingled” worklet that mixes CPU, memory, and storage transactions running concurrently. Alternatively, each client JVM on a system could run a different set of transactions, or a different sequence of load levels. This could be extended to multi-system environments by running different transactions or sequences of load levels on different hosts; this could be of particular interest for emulating a virtualization consolidation environment where multiple virtual machines are running on the same host, each running independently.

The SERT runs an orderly sequence of load levels in order to maximize consistency, but real applications tend to be more dynamic. Chauffeur could be used to study these environments by using a more random sequence of load levels with more frequent changes. This could be useful in assessing the impact of power management technologies on performance and power consumption in dynamic environments. The sequence of load levels could be modeled after utilization information from production systems.

Chauffeur currently focuses on application throughput, but supplies minimal information about response time. Maintaining adequate application response time can be an important factor for power management technologies. Future Chauffeur enhancements could verify that transactions meet some threshold for quality of service, particularly at low utilizations when power management may hurt response time.

8 Conclusion

Energy efficiency has become a significant concern for servers. Over the past few years, several energy efficiency benchmarks have been released. These are intended both to drive product improvements and to enable fair comparisons between servers. While these benchmarks have been a good start, additional benchmarks are needed to overcome some of the limitations of the earlier attempts and provide workloads that mimic the behavior of real environments.

Like all benchmarks, energy efficiency benchmarks need to be relevant, reproducible, fair, verifiable, and easy to use. In order to meet these goals, energy efficiency benchmarks in particular should measure power consumption at multiple utilizations, and use multiple workloads to exercise a variety of behavior. Support for multiple systems allows energy benchmarks to produce meaningful results for systems with shared power and cooling infrastructure. Power measurements should be obtained automatically from a power analyzer capable of producing results with sufficient accuracy. Complete and accurate descriptions of the system hardware and software configuration are critical for reproducibility. Due to the inherent complexity of collecting power data, ease of use features are important to allow users to collect the data accurately and with as little effort as possible.

SPEC has developed the SERT as its next-generation tool for evaluating energy efficiency. While SERT is not intended to be a benchmark, it does provide valuable data regarding the energy efficiency of a server. The Chauffeur framework was designed to help SERT and future benchmarks to meet the design criteria for energy efficiency benchmarks. Chauffeur implements functionality common to most benchmarks and includes a variety of features intended to simplify collection of meaningful energy data.

Early SERT results show that it produces consistent results, and demonstrate the necessity of measuring multiple workloads at multiple utilizations. Further experiments demonstrate the impact of system tuning on energy efficiency, and show the influence that default tuning options can have on energy consumption in customer environments.

While the SERT is a significant improvement in the ability to measure energy efficiency, it will certainly not be the last tool or benchmark to do so. Future benchmarks are still needed to provide more relevant results for specific usage patterns and more balanced usage of system components. Additional benchmarks can also provide energy efficiency data for specialized areas such as virtualization. Chauffeur provides a solid foundation that these future benchmarks can take advantage of.

9 Acknowledgements

Special thanks to current and former members of the SPECpower committee, including Klaus-Dieter Lange, Hansfried Block, Greg Darnell, Mike Tricker, Nathan Totura, John Beckett, Christian Koopman, Sanjay Sharma, Van Smith, Karl Huppler, Karin Wulf, Paul Muehr, and David Ott. I have had the pleasure of working with these professionals for nearly 7 years, and each has influenced my interest in and understanding of energy efficiency benchmarking from the time of my initial "weekend project" (which eventually became SPECpower_ssj2008) through the development of SERT. Thanks also to my managers at IBM, Paul Wadehra and Terry Thomas, who supported my educational opportunities as well as my split role between my "day job" and my work with SPEC.

References

[1] James Laudon. Performance/watt: the new server focus. *SIGARCH Comput. Archit. News*, 33(4):5–13, November 2005.

[2] Luiz André Barroso. The price of performance. *Queue*, 3(7):48–53, September 2005.

[3] A. Fanara, E. Haines, and A. Howard. The state of energy and performance benchmarking for enterprise servers. *Performance Evaluation and Benchmarking*, pages 52–66, 2009.

[4] Meikel Poess and Raghunath Othayoth Nambiar. Energy cost, the key challenge of today's data centers: a power consumption analysis of TPC-C results. *Proc. VLDB Endow.*, 1:1229–1240, August 2008.

[5] E.R. Masanet, R.E. Brown, A. Shehabi, J.G. Koomey, and B. Nordman. Estimating the energy use and efficiency potential of us data centers. *Proceedings of the IEEE*, 99(8):1440–1453, 2011.

[6] Chung-Hsing Hsu and S.W. Poole. Power signature analysis of the SPECpower_ssj2008 benchmark. In *Performance Analysis of Systems and Software (ISPASS), 2011 IEEE International Symposium on*, pages 227 –236, April 2011.

[7] P. Bohrer, E.N. Elnozahy, T. Keller, M. Kistler, C. Lefurgy, C. McDowell, and R. Rajamony. The case for power management in web servers. *Power aware computing*, 78758, 2002.

[8] C. Lefurgy, K. Rajamani, F. Rawson, W. Felter, M. Kistler, and T.W. Keller. Energy management for commercial servers. *Computer*, 36(12):39 – 48, December 2003.

[9] K.D. Lange. The next frontier for power/performance benchmarking: Energy efficiency of storage subsystems. *Computer Performance Evaluation and Benchmarking*, pages 97–101, 2009.

[10] Akshat Verma, Gargi Dasgupta, Tapan Kumar Nayak, Pradipta De, and Ravi Kothari. Server workload analysis for power minimization using consolidation. In *Proceedings of the 2009 conference on USENIX Annual technical conference*, USENIX'09, pages 28–28, Berkeley, CA, USA, 2009. USENIX Association.

[11] Xiaobo Fan, Wolf-Dietrich Weber, and Luiz Andre Barroso. Power provisioning for a warehouse-sized computer. In *Proceedings of the 34th annual international symposium on Computer architecture*, ISCA '07, pages 13–23, New York, NY, USA, 2007. ACM.

[12] G. Varsamopoulos and S.K.S. Gupta. Energy proportionality and the future: Metrics and directions. In *Parallel Processing Workshops (ICPPW), 2010 39th International Conference on*, pages 461 –467, September 2010.

[13] G. Varsamopoulos, Z. Abbasi, and S.K.S. Gupta. Trends and effects of energy proportionality on server provisioning in data centers. In *High Performance Computing (HiPC), 2010 International Conference on*, pages 1 –11, December 2010.

[14] J. Mitchell-Jackson, J.G. Koomey, B. Nordman, and M. Blazek. Data center power requirements: measurements from silicon valley. *Energy*, 28(8):837 – 850, 2003.

[15] Suzanne Rivoire, Mehul A. Shah, Parthasarathy Ranganathan, and Christos Kozyrakis. JouleSort: a balanced energy-efficiency benchmark. In *Proceedings of the 2007 ACM SIGMOD international conference on Man-*

agement of data, SIGMOD '07, pages 365–376, New York, NY, USA, 2007. ACM.

[16] Chung-Hsing Hsu, Jeffery A. Kuehn, and Stephen W. Poole. Towards efficient supercomputing: searching for the right efficiency metric. In *Proceedings of the third joint WOSP/SIPEW international conference on Performance Engineering*, ICPE '12, pages 157–162, New York, NY, USA, 2012. ACM.

[17] K. Skadron, M. Martonosi, D.I. August, M.D. Hill, D.J. Lilja, and V.S. Pai. Challenges in computer architecture evaluation. *Computer*, 36(8):30 – 36, August 2003.

[18] K. Huppler. The art of building a good benchmark. *Performance Evaluation and Benchmarking*, pages 18–30, 2009.

[19] J.L. Gustafson and Q.O. Snell. HINT: A new way to measure computer performance. In *System Sciences, 1995. Proceedings of the Twenty-Eighth Hawaii International Conference on*, volume 2, pages 392–401. IEEE, 1995.

[20] R. García-Castro and A. Gómez-Pérez. Benchmark suites for improving the RDF (S) importers and exporters of ontology development tools. *The Semantic Web: Research and Applications*, pages 155–169, 2006.

[21] F. Stefani, D. Macii, A. Moschitta, and D. Petri. FFT benchmarking for digital signal processing technologies. In *17th IMEKO World Congress*, 2003.

[22] Susan Elliott Sim, Steve Easterbrook, and Richard C. Holt. Using benchmarking to advance research: a challenge to software engineering. In *Proceedings of the 25th International Conference on Software Engineer-*

ing, ICSE '03, pages 74–83, Washington, DC, USA, 2003. IEEE Computer Society.

[23] J.L. Henning. SPEC CPU2000: measuring CPU performance in the new millennium. *Computer*, 33(7):28–35, 2000.

[24] SPEC power and performance benchmark methodology. `http://www.spec.org/power/docs/SPEC-Power_and_Performance_Methodology.pdf`.

[25] L.A. Barroso and U. Holzle. The case for energy-proportional computing. *Computer*, 40(12):33 –37, December 2007.

[26] D. Skinner and W. Kramer. Understanding the causes of performance variability in HPC workloads. In *Workload Characterization Symposium, 2005. Proceedings of the IEEE International*, pages 137 – 149, October 2005.

[27] James Donald and Margaret Martonosi. Power efficiency for variation-tolerant multicore processors. In *Proceedings of the 2006 international symposium on Low power electronics and design*, ISLPED '06, pages 304–309, New York, NY, USA, 2006. ACM.

[28] Shekhar Borkar, Tanay Karnik, Siva Narendra, Jim Tschanz, Ali Keshavarzi, and Vivek De. Parameter variations and impact on circuits and microarchitecture. In *Proceedings of the 40th annual Design Automation Conference*, DAC '03, pages 338–342, New York, NY, USA, 2003. ACM.

[29] Radu Teodorescu and Josep Torrellas. Variation-aware application scheduling and power management for chip multiprocessors. In *Proceedings of the 35th Annual International Symposium on Computer Architecture*, ISCA '08, pages 363–374, Washington, DC, USA, 2008. IEEE Computer Society.

[30] SPEC fair use rule. `http://www.spec.org/fairuse.html`.

[31] Erik Young, Paul Cao, and Mike Nikolaiev. First TPC-Energy benchmark: Lessons learned in practice. In Raghunath Nambiar and Meikel Poess, editors, *Performance Evaluation, Measurement and Characterization of Complex Systems*, volume 6417 of *Lecture Notes in Computer Science*, pages 136–152. Springer Berlin / Heidelberg, 2011.

[32] Doron Chen, Ealan Henis, Ronen I. Kat, Dmitry Sotnikov, Cinzia Cappiello, Alexandre Mello Ferreira, Barbara Pernici, Monica Vitali, Tao Jiang, Jia Liu, and Alexander Kipp. Usage centric green performance indicators. *SIGMETRICS Perform. Eval. Rev.*, 39(3):92–96, December 2011.

[33] Meikel Poess, Raghunath Othayoth Nambiar, Kushagra Vaid, John M. Stephens, Jr., Karl Huppler, and Evan Haines. Energy benchmarks: a detailed analysis. In *Proceedings of the 1st International Conference on Energy-Efficient Computing and Networking*, e-Energy '10, pages 131–140, New York, NY, USA, 2010. ACM.

[34] TPC-Energy specification. `http://www.tpc.org/tpc_energy/`.

[35] Trish Hogan. Overview of TPC Benchmark E: The next generation of OLTP benchmarks. In Raghunath Nambiar and Meikel Poess, editors, *Performance Evaluation and Benchmarking*, volume 5895 of *Lecture Notes in Computer Science*, pages 84–98. Springer Berlin / Heidelberg, 2009. 10.1007/978-3-642-10424-4_7.

[36] Meikel Poess and Chris Floyd. New TPC benchmarks for decision support and web commerce. *SIGMOD Rec.*, 29:64–71, December 2000.

[37] Shimin Chen, Anastasia Ailamaki, Manos Athanassoulis, Phillip B. Gibbons, Ryan Johnson, Ippokratis Pandis, and Radu Stoica. TPC-E vs.

TPC-C: characterizing the new TPC-E benchmark via an I/O comparison study. *SIGMOD Rec.*, 39(3):5–10, February 2011.

[38] Raghunath Othayoth Nambiar and Meikel Poess. The making of TPC-DS. In *Proceedings of the 32nd international conference on Very large data bases*, VLDB '06, pages 1049–1058. VLDB Endowment, 2006.

[39] Justin Meza, Mehul A. Shah, Parthasarathy Ranganathan, Mike Fitzner, and Judson Veazey. Tracking the power in an enterprise decision support system. In *Proceedings of the 14th ACM/IEEE international symposium on Low power electronics and design*, ISLPED '09, pages 261–266, New York, NY, USA, 2009. ACM.

[40] W. Kohler, A. Shah, and F. Raab. Overview of TPC Benchmark C: The order-entry benchmark. *Transaction Processing Performance Council, Technical Report*, 1991.

[41] Hui Lv, Yaozu Dong, Jiangang Duan, and Kevin Tian. Virtualization challenges: a view from server consolidation perspective. In *Proceedings of the 8th ACM SIGPLAN/SIGOPS conference on Virtual Execution Environments*, VEE '12, pages 15–26, New York, NY, USA, 2012. ACM.

[42] Rema Hariharan and Ning Sun. Workload characterization of SPECweb2005. In *SPEC Benchmark Workshop. SPEC*. Citeseer, 2006.

[43] K.-D. Lange. Identifying shades of green: The SPECpower benchmarks. *Computer*, 42(3):95 –97, March 2009.

[44] L. Gray, A. Kumar, and H. Li. Workload characterization of the SPECpower_ssj2008 benchmark. *Performance Evaluation: Metrics, Models and Benchmarks*, pages 262–282, 2008.

[45] A. Beckmann, U. Meyer, P. Sanders, and J. Singler. Energy-efficient sorting using solid state disks. In *Green Computing Conference, 2010 International*, pages 191 –202, August 2010.

[46] P. Pillai, M. Kaminsky, M.A. Kozuch, and D.G. Andersen. FAWNSort: energy-efficient sorting of 10GB, 100GB, and 1TB.

[47] Wu-chun Feng and K.W. Cameron. The Green500 list: Encouraging sustainable supercomputing. *Computer*, 40(12):50 –55, December 2007.

[48] Wu-chun Feng, Xizhou Feng, and Rong Ce. Green supercomputing comes of age. *IT Professional*, 10(1):17 –23, January-February 2008.

[49] Klaus-Dieter Lange, Mike G. Tricker, Jeremy A. Arnold, Hansfried Block, and Christian Koopmann. The implementation of the Server Efficiency Rating Tool. In *Proceedings of the third joint WOSP/SIPEW international conference on Performance Engineering*, ICPE '12, pages 133–144, New York, NY, USA, 2012. ACM.

[50] Server Efficiency Rating Tool (SERT) design document. `http://www.spec.org/sert/docs/SERT-Design_Doc.pdf`, August 2012.

[51] Dimitris Tsirogiannis, Stavros Harizopoulos, and Mehul A. Shah. Analyzing the energy efficiency of a database server. In *Proceedings of the 2010 international conference on Management of data*, SIGMOD '10, pages 231–242, New York, NY, USA, 2010. ACM.

[52] Klaus-Dieter Lange and Michael G. Tricker. The design and development of the Server Efficiency Rating Tool (SERT). In *Proceedings of the second joint WOSP/SIPEW international conference on Performance engineering*, ICPE '11, pages 145–150, New York, NY, USA, 2011. ACM.