Energy Efficiency of Servers in Data Centers

Wentai Wu, Peng Cheng Laboratory, Shenzhen, China
Weiwei Lin, South China University of Technology, Guangzhou, China and Peng Cheng Laboratory, Shenzhen, China
Keqin Li, State University of New York, New Paltz, NY, United States

Abstract

High efficiency of energy usage is a key indicator of sustainable data centers. This article focuses on the assessment of energy efficiency (EE) at the server level. First, state-of-the-art metrics for quantifying server energy efficiency are introduced with details on how they are formulated and parameterized. Second, practical examples based on two benchmark suites are provided to help the reader better understand the mechanism of server EE assessment.

Key Points

- The definition and formulation of three energy efficiency (EE) metrics at the server level.
- The mechanism and procedure of calculating server EE using the SPECpower_ssj2008 benchmark (Example 1).
- The mechanism and procedure of calculating server EE using the SERT benchmark (Example 2).

Introduction

The rapid growth of data center (DC) energy consumption and the inefficient usage of power are two widely surveyed issues in data center energy management. Among all the information and communications technology (ICT) sectors, data centers are playing a critical role in the process of digitalization and also growing rapidly in terms of energy consumption, which makes energy management one of the top priorities for data center operators and Cloud Service Providers (CSPs).

Any modern data center is a complicated system that involves the interplay of a variety of facilities from power distribution units and lighting to ICT infrastructures and cooling subsystems. Also, the power demand varies between data centers that embrace different software technology stacks and application types (e.g., data processing can account for 45% of the total energy in DCs specifically designed for big data applications (Wu *et al.*, 2016)). Some studies investigated the energy efficiency (EE) of data centers from a holistic standpoint (Reddy *et al.*, 2017; Belady and Malone, 2007), whilst some focus on individual subsystems such as heating, ventilation and air conditioning (HVAC) (Lee and Chen, 2013; Pakbaznia and Pedram, 2009). Among all these factors, the energy efficiency of physical servers is broadly reckoned as the most critical one when it comes to the development of sustainable DCs (von Kistowski *et al.*, 2019). With a clear up-going trend for future DCs in size and power density, the measurement and rating for servers in energy efficiency have become increasingly important not only in the deployment stage but also throughout the operation cycles.

Servers are the key infrastructure in any data center. Statistics show that about 40% of the electricity supply of a modern data center is used for powering the servers and IT devices. On the other hand, it is reported that roughly 20%∼30% of the servers in data centers are actually idling. The average server utilization in production environment, even in a well-managed data center of Google, is usually lower than 40% (Tirmazi *et al.*, 2020). This leads to a great percentage of energy waste and low energy efficiency of the entire data center. Therefore, how to improve energy efficiency via server management has been a critical aspect in the development of green data centers.

On the roadmap of developing sustainable data centers, two primary issues stand out. On the one hand, managing a large cluster of servers of diverse models in a fine-grained and scalable manner is a systematic challenge. It requires a comprehensive solution that embraces both bottom-up and top-down design. On the other hand, the measurement and monitoring of server energy efficiency is critical in terms of providing useful information for the management of the entire data center, but it is challenging to design practical and flexible energy efficiency metrics at the granularity of servers. General metrics like Power Usage Effectiveness (PUE) reflect energy efficiency at data center level but offer little insight for server-level management. Dimensionless ratios, such as Deployed Hardware Utilization Ratio (DH-UR) and Deployed Hardware Utilization Efficiency (DH-UE) (Stanley *et al.*, 2007), cannot specify the impact of any individual server or quantify the efficiency differences between servers. Without sufficiently informative metrics for server energy efficiency and power efficiency, the effort on reducing energy consumption may compromise the actual performance of the servers and leads to Service Level Agreement (SLA) violations. Therefore, the energy efficiency of servers needs to be studied and evaluated well in advance to the deployment of any energy-preserving strategies.

The main purpose of this section is to introduce important metrics for measuring energy efficiency at the server level. To demonstrate step by step the basic pipeline from data sampling to the calculation of EE metrics, this section also illustrates how server energy efficiency is measured with two practical examples based on *SPECpower_ssj2008* and *SERT*. These two benchmarks are widely used for evaluating server power efficiency. The content of this section can provide guidance for the researchers and practitioners in the field of data center engineering and management.

 doi:10.1016/B978-0-323-90386-8.00054-1

Metrics of Server Energy Efficiency

By definition, a metric of server energy efficiency (EE) is supposed to measure the efficiency of servers in terms of performance output and energy or power usage. Without loss of generality, the energy efficiency metrics discussed in this article are not restricted by the definition of *energy* (as the consumption over a period of time). We are also interested in instant metrics associated with the *power* (as a spot measurement) of a server, in which case the basic definition of server EE can be formulated as the performance that the server yields over the power it consumes:

$$Server\ EE = \frac{performance}{power\ consumption} \tag{1}$$

This definition is straightforward and apparently easy to implement. In fact, most of the EE metrics in the literature are derived from this Performance per Watt (PpW) or performance-to-power ratio (Lin *et al.*, 2018, 2022). However, there are two outstanding issues that complicate the practice of this formula. On the one hand, both performance and power consumption vary as the server operates at different level of capacity, i.e., utilization. It is important to model server EE as a function of the server's current utilization or a comprehensive score that considers various load levels of the server. On the other hand, the performance of a server, as the numerator in Eq. (1), is a measure that may vary largely given different computing work (e.g., big data processing, web serving, and content caching) and can be quantified in many ways such as million instructions per second (MIPS), floating-point operations per second (FLOPs), or application-specific throughput measures. This diversity offers great flexibility whilst poses challenges to making fair comparison in different use cases.

Server-Level EEUI

To measure the energy efficiency (EE) of a particular server, a typical approach is to break down the server as a whole system into individual components and to further define EE metrics at the component level. Based on this rationale, it is assumed that the maximum energy efficiency is achieved when the capacity of all major components (CPU, memory, storage and network component) is optimally utilized. On the contrary, the server works at its lowest level of energy efficiency when idling. Specifically, considering CPU, memory, storage and network as the major components, the server energy-efficient utilization indicator (EEUI) (Abaunza *et al.*, 2018) boils down to four corresponding terms:

$$Server_{EEUI} = Server_{EEUI}^{CPU} + Server_{EEUI}^{Memory} + Server_{EEUI}^{Network} + Server_{EEUI}^{Disk} \tag{2}$$

where $Server_{EEUI}^{X}, X \in \{CPU, Mem, Net, Disk\}$ denotes the EEUI of component. For example, the EEUI of CPU is defined as follows:

$$Server_{EEUI}^{CPU} = \frac{associated\ EE\ level\ of\ CPU\ mapped\ by\ CPU\ utilization}{maximum\ EE\ that\ CPU\ can\ operate} \times \frac{power\ consumed\ by\ CPU}{total\ power\ consumption} \tag{3}$$

In a more general form, the EEUI of any component X is defined as follows:

$$Server_{EEUI}^{X} = \frac{EE(X)}{EE_{\max}(X)} \times \frac{power\ consumed\ by\ X}{total\ power\ consumption} \tag{4}$$

where $EE(X)$ represents the associated EE level at which component X operates and $EE_{\max}(X)$ is the maximum EE value that component X can attain. For a CPU, $EE(X)$ is the speed in GHz that it is operating at over its power consumption. For network, $EE(X)$ is the ratio of the network I/O rate to the power consumed by associated network component.

The definition of EEUI is intuitive. The EEUI of a server is additively attributed to the EEUI of its power-consuming components, and the EEUI of each individual component is proportional to its utilization and power consumption. It is also worth mentioning that server EEUI is not designed for comparing energy efficiency of different server hardware or architectures. Instead, it is typically used to monitor the level of energy efficiency when the servers are operating under different load levels. In summary, EEUI measures server energy efficiency by combining component-level EE as a function of the utilization level at which the component is operating and further weighting the component-level EE proportionally to the associated power usage.

Workload-Dependent Metric of Server EE

The energy efficiency of a server can be traced down to the specific workload that the server is running, and further, down to the EE of each worklet that comprises the workload (Polverini and Tosoratti, 2018). This perspective comes up with a bottom-up modeling of server EE, as shown in **Fig. 1**. First, the (energy) efficiency of each worklet is formulated in Eq. (5) where a set of predefined utilization levels are considered. The efficiency score of a worklet is defined as the sum of worklet performance divided by the sum of worklet power consumption over all the utilization levels:

$$worklet\_efficiency = \frac{\sum performance\_at\_each\_utilization\_level}{\sum power\_at\_each\_utilization\_level} \tag{5}$$

Once the worklet efficiency is obtained, the second step is to define the energy efficiency at workload level. The approach presented here aggregates, for each workload, the associated worklet efficiency values in the form of harmonic mean, as in the following equation:

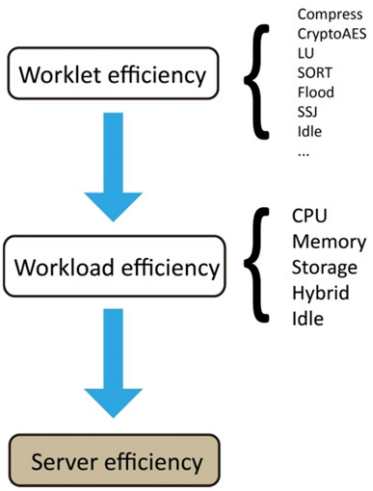

Fig. 1 The logical structure of workload-dependent energy efficiency metric.

$$workload\_efficiency = n * \left(\sum \frac{1}{worklet\_efficiency} \right)^{-1} \tag{6}$$

Where n is the number of worklets associated to the specific workload.

The third step is modeling energy efficiency at the server level, which boils down to the energy efficiency of the most relevant workloads associated to the individual components of a server such as CPU, RAM, and hard disks. An approach to combine the workload efficiency into an overall efficiency metric (Polverini and Tosoratti, 2018) is presented as follows:

$$server\_efficiency_{high} = \left(\frac{0.6}{CPU\_eff} + \frac{0.4}{memory\_eff} \right)^{-1} \tag{7}$$

$$server\_efficiency_{low} = DR \left(\frac{0.6}{CPU\_eff} + \frac{0.4}{memory\_eff} \right)^{-1} \tag{8}$$

with

$$DR = \left(\frac{idle\_power}{\max\_power} + 1.5 \right) \tag{9}$$

For servers, the classification of user patterns proves to be very difficult in practice. To simplify, the working status of a server can be inspected from the perspective of load level by considering two cases. In the first one, the server, e.g., within a highly efficient data center at scale providing services to a relatively fixed number of customers, is likely to run consistently at a high utilization level (typically 60% or above). In the second case, a low utilization scenario is considered, wherein the server is typically deployed in a small-scale data center/server room with long periods of inactivity (e.g., at night or weekend). In this case, the static power consumption in idle mode accounts for the majority of the total. Therefore, at low utilization, the server energy efficiency is mainly influenced by the *DR* (dynamic range) factor, which in turn is dependent on the idle power of the server.

Eqs. (7) and (8) formulate the energy efficiency of a server at high utilization and low utilization, respectively. *CPU_eff* denotes the CPU workload efficiency and *memory_eff* denotes the memory workload efficiency. *DR* is a multiplicative factor for calibrating the efficiency value in the low utilization case. *idle_power* represents the power consumption of the server in idle state, and max_*power* represents the server power consumption corresponding to the highest load level. In this specific case, a weighted harmonic mean is used, and the weights for the workloads (60% for CPU and 40% for Memory) represent the pattern of real-life applications, i.e., a typical 60:40 CPU-memory ratio. It is worth noting that workloads on other components are not considered in this metric as contributing factors to the overall server efficiency.

Quadratic Approximation-Based Server Power Efficiency Metric

As the most intuitive metric, Floating-point operations per second (FLOPS)/Watt can be used to measure a server's power efficiency as the ratio indicates the equivalent performance output of the server given a unit (Watt) of power input:

$$server\ power\ efficiency = \frac{performance\ in\ FLOPs}{power\ consumption\ in\ Watts} \tag{10}$$

The problem here is that, for some server models, the accurate, real-time acquisition of performance and power consumption is not natively supported. Inspired by the studies on server behavior, both the performance and the power consumption have a strong correlation with the server utilization and thus can be approximated using polynomial functions. Based on the accuracy

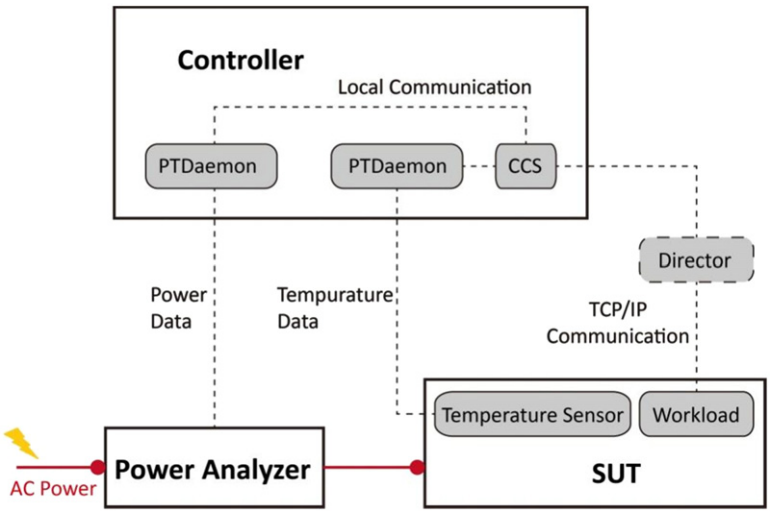

Fig. 2 An overview of SPECpower_ssj2008 benchmarking system components.

requirement, quadratic functions are usually adopted (Lin *et al.*, 2018). In this case, Eq. (10) can be further extended as the following:

$$server\ power\ efficiency = \frac{f_1(u_{cpu})}{f_2(u_{cpu})} = \frac{c_0 + c_1 * u_{cpu} + c_2 * {u_{cpu}}^2}{d_0 + d_1 * u_{cpu} + d_2 * {u_{cpu}}^2} \tag{11}$$

where the parameters c_0, c_1, c_2, d_0, d_1, and d_2 are coefficients that can be obtained by fitting measured data of server power consumption and performance at various resource utilization levels. The variable u_{cpu} is the utilization of CPU.

Examples of Server Energy Efficiency Measurement

Specpower_Ssj2008

SPECpower_ssj2008 is a benchmark suite developed by the Standard Performance Evaluation Corporation (SPEC), a not-for-profit group of computer vendors, system integrators, universities, research organizations, publishers, and consultants. It is the first industry-standard SPEC benchmark designed to provide a holistic view of a server system's power consumption running Java server applications. The general approach is to generate samples of measured performance with measured power consumption. The benchmark suite provides a variety of workload configuration to emulate the workload environment of servers in a data center running under various levels of transaction rate relative to its maximum throughput.

Configuration overview

The basic SPECpower_ssj2008 test bed implementation is illustrated in **Fig. 2**. Typically, the architecture of the benchmark entails four roles:

- A system under test (SUT), with a group of JVMs running specified workload
- A power analyzer, controlled by an instance of Power Temperature Daemon (PTDaemon)
- A temperature sensor, controlled by a separate instance of PTDaemon
- A controller system, with a JVM running the Control and Collect System (CCS)
- (optional) A director system, with a JVM running the director. The director JVM can also run in the SUT or the controller system.

CCS handles the hardware function of measuring and recording the power consumption and inlet temperature of the SUT. It also controls the software installed on both the SUT and the Controller, communicating via the TCP/IP transport protocol. CCS communicates with the *Director*, which instructs the SUT to execute the workload while the CCS collects the power and temperature data via the PTDaemon. *The Temperature Sensor* must be placed no more than 50 mm in front of (upwind of) the main airflow inlet of the SUT. SPECpower_ssj2008 measures the inlet temperature of the SUT and marks the results "valid" only if the temperature measured is 20 °C or higher. A stable temperature value is not required during warm-up or measurement phases. *The Power Analyzer* must be located between the AC power supply and the SUT. Each analyzer and sensor interact with their dedicated instance of *SPEC PTDaemon*, which gathers their readings while the worklets are being executed (SPEC, 2012a).

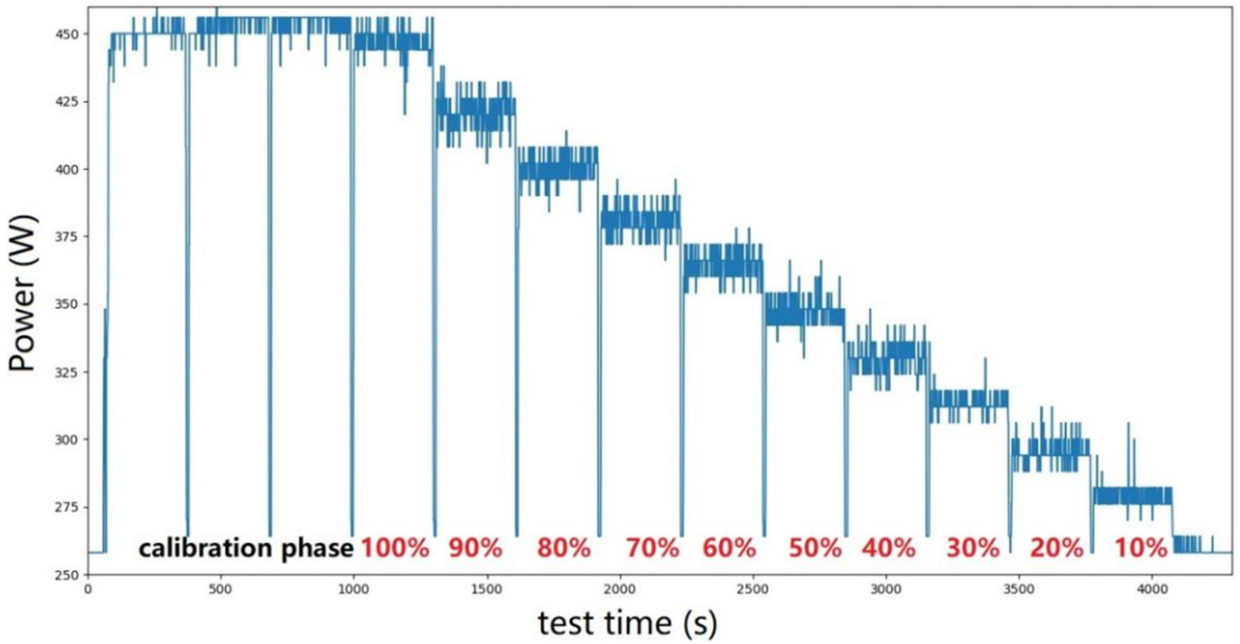

Fig. 3 The change of Load Levels and power consumption throughout a compliant SSJ2008 run.

The SUT and workload

For the SPECpower_ssj2008 (SSJ2008 for short) benchmark, the SUT is a single-address-space server system or a homogeneous collection of such systems which share resources that are electrically relevant. The target loads are measured in terms of the performance under a fixed class of workload that runs on the SUT. The workload is programmed as a Java application. It generates and completes 6 different transaction types, with the approximate distribution shown below:

- New Order (30.3%) – insert a new order into the system
- Payment (30.3%) – record a customer payment
- Order Status (3.0%) – request the status of an existing order
- Delivery (3.0%) – process orders for delivery
- Stock Level (3.0%) – find recently ordered items with low stock levels
- Customer Report (30.3%) – create a report of recent activity for a customer

Input data for each transaction is randomly generated. The transactions modify in-memory data structures representing Warehouses, Customers, Orders, Orderlines, etc. Most of the transactions apply change to the local warehouse only and thus incur minimal interaction between the warehouse threads during benchmarking (SPEC, 2012b).

An SSJ2008 test consists of two main phases: Calibration and running a series of Target Loads. The calibration phase is used to determine the maximum throughput that a system is capable of. Once this calibrated throughput is established, the system runs a series of target loads. Each workload runs at some percentage load level of the calibrated throughput. Through a compliant test run of SPECpower_ssj20008, the level of workload is supposed to decrease from 100% to 0% with an interval of 10%, as shown in **Fig. 3**. Progressively turning down the workload and limiting each change to 10% at each level guarantees stable and reliable power measurement. An increasing load sequence, otherwise, will result in a jump from 100% to 10% when moving from the final calibration interval to the first target load and another jump from 100% to Active Idle at the end of the run.

For each target Load Level (e.g., 90%, 80%), the test controller records the actual transaction rate and the corresponding mean timespan from the start of one transaction to the start of the next transaction. During the measurement interval, randomized delays are inserted into the worklet execution; these delays follow an exponential distribution that statistically converges to the desired transaction rate. As a result, lower target loads consist of short bursts of activity separated by relatively long periods of inactivity. **Fig. 4** shows two examples of the distribution of transactions by running with a single processor thread at 20% and 80% target load level using this technique (von Kistowski *et al.*, 2018).

Specpower_ssj2008 metric

SPECpower_ssj2008 was first released in December 2007 and continues to facilitate innovations in the evaluation of server efficiency. Rather than trying to approximate all the typical workloads across organizations, SPECpower_ssj2008 is focused on transactional server-side Java workloads that simulate a warehouse-based customer ordering, supply and replenishment model. This synthetic workload simulates many aspects of business-oriented Java implementations, together with the underlying server

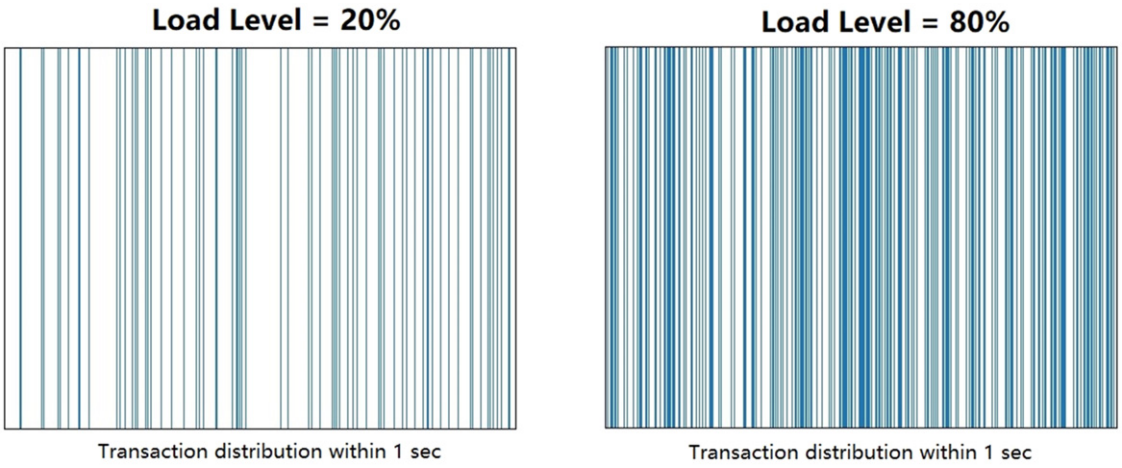

Fig. 4 Transaction rate of SPECpower_ssj2008 at different Load Levels.

hardware including processors (with support of multiple cores per processor), memory hierarchies (including caches), and the system Symmetric Multiprocessing (SMP) scalability (von Kistowski *et al.*, 2018).

The performance-to-power ratio under the target load is a measurement of the number of *ssj_ops* (throughput) processed per watt of power consumed. The more *ssj_ops* the SUT can produce with one watt of power, the more efficient the SUT is in terms of power usage. The SPECpower_ssj2008 metric is calculated as the sum of all *ssj_ops* scores for all target loads, divided by the sum of all power consumption averages (in watts) for all target loads, including the active idle measurement interval. The unit of the SPECpower_ssj2008 metric is "overall ssj_ops / watt" (SPEC, 2012c).

$$\text{SPECpower\_ssj2008 metric} = \frac{\sum_{i=1}^{n} \text{ssj\_opsfor target load } i}{\sum_{i=1}^{n} \text{watts for target load } i + \text{watts for active idle}} \tag{12}$$

Since SPECpower_ssj2008's initial release in December 2007, the SPEC community has received constant updates of fully documented results (https://www.spec.org/power_ssj2008/results/power_ssj2008.html). **Fig. 5** shows a benchmark results summary released in November 2019 for Fujitsu Server PRIMERGY TX1320 M4.

As an example, the following SPECpower_ssj2008 metric is derived from the data demonstrated in **Fig. 5**.

$$\text{SPECpower} - \text{ssj 2008 metric} = \frac{\sum Performance}{\sum Power} = \frac{79,687 + 159,095 + \ldots + 797,225 \text{ ssj\_ops}}{18.8 + 21.2 + \ldots + 56.0 + 16.3\text{watt}}$$

$$= \frac{4,368,669 \text{ ssj\_ops}}{353.4\text{watt}} = 12,364\text{overall ssj\_ops/watt} \tag{13}$$

SERT

The Server Efficiency Rating Tool (SERT) is a next-generation tool set for measuring and evaluating the energy efficiency of servers. SERT was developed by the Standard Performance Evaluation Corporation (SPEC) at the request of the U.S. Environmental Protection Agency (EPA) as part of the Energy Star program. The SPECpower Committee designed, implemented, and released the SERT suite with professional engagement by leaders of various global energy-efficiency programs and their stakeholders in order to accommodate for their regional program requirements. The technical content in this section is based on SERT 2.0 which is the latest version of the SERT suite.

SERT has a lot in common with the SPECpower_ssj2008 benchmark but offers stronger flexibility. SPECpower_ssj2008 is a benchmark that emulates a transactional business application. It features different load levels that scale with the application's throughput, yet its method is limited to a single workload with a fixed, pre-defined number of load levels. By contrast, the SERT provides a single score for multiple different workloads with varying numbers of load levels. In a sense, SERT is a comprehensive extension of the SPECpower_ssj2008 benchmark.

Sert workloads and worklets

The goal of the SERT suite is to include all major aspects of server architecture avoiding any preference towards any specific architectural features, which may result in strong variation of server performance under different workloads. SERT evaluates the capacity of a server comprehensively by using various load patterns that are intended to stress all major components of a server.

Fujitsu Server PRIMERGY TX1320 M4			
Load level	Performance (ssj_ops)	Power (W)	Performance-to-power Ratio
100%	797,225	56.0	14,229
90%	712,828	48.1	14,834
80%	635,321	42.5	14,935
70%	555,428	37.7	14,746
60%	477,700	33.6	14,230
50%	395,875	29.6	13,374
40%	317,519	26.1	12,151
30%	237,991	23.5	10,132
20%	159,095	21.2	7,519
10%	79,687	18.8	4,237
Active Idle	0	16.3	0

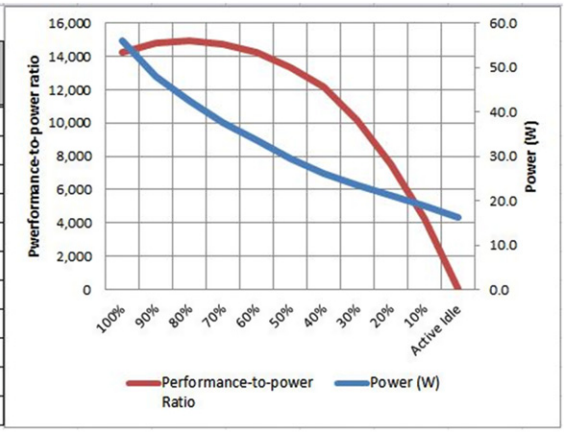

Fig. 5 Power and performance at multiple target load levels for Fujitsu Server PRIMERGY TX1320 M4 (Data from SPEC power_ssj2008 results https://www.spec.org/power_ssj2008/results/res2019q4/power_ssj2008–20191008–01009.html).

The SERT test suite consists of several workloads which are designed to stress the major components of the system under test (SUT) including CPU, memory, and storage. Each workload includes one or several worklets that execute specific code sequences to focus on one of these system components. **Fig. 6** shows the relationship between the overall suite, workloads, and worklets for SERT (SPEC, 2022).

Each of the three types of workload (CPU, Memory, and Storage) is comprised to at least two different worklets (the CPU workload covers up to 7 unique worklet programs). Each worklet supports a variety of configurable parameters for customized experiments. Each worklet runs at various (at least two) load levels. For example, the *SSJ* worklet can be set to run at pre-defined load levels from 100% to 12.5% with a minimum interval of 12.5%. **Table 1** lists the built-in SERT worklets and their load levels.

These workloads can be classified based on resource intensity:

- CPU: Data compression, encryption/decryption, complex number arithmetic, matrix factorization, floating point array manipulation, sorting algorithm, string manipulation, and a CPU-intensive workload derived from ssj2008, which simulates an on-line transaction processing workload;
- Memory: XML document manipulation and validation using pre-computed and cached data lookup, and array manipulation with read/write operations across four major classes of data transformation;
- Storage: Two individual transaction pairs combining sequential and random read/write.
- Idle: A steady state in which the server is ready to execute any worklet but is not actually doing so (von Kistowski *et al.*, 2018).

Sert metric

The SERT metric is computed in three steps:

1. Calculate worklet efficiency scores from the recorded data. 2. Calculate workload efficiency based on the worklet efficiency scores. 3. Calculate the SERT metric based on the workload efficiency scores.

Through these steps the SERT metric is computed in a bottom-up manner. The worklet efficiency scores are first calculated using the measured power consumption and performance of the worklet at various load levels. Then, the scores of all worklets are combined into a workload efficiency score for the corresponding workload. Finally, the SERT metric is an aggregate of all the workload scores. **Fig. 7** illustrates the overall workflow for calculating the SERT metric (SPEC, 2021).

Worklet Efficiency Calculation

All SERT worklets, except *Idle*, run at various load levels. For each of these load levels, the associated efficiency is calculated separately. The energy efficiency Eff_{load} for a worklet at a certain load level is defined in Eq. (14):

$$Eff_{load} = \frac{Normalized\ performance}{Avg.power\ consumption} \tag{14}$$

where the performance is normalized over all but memory worklets. Power consumption in this context is the average measured power in Watts at each load level.

The worklet efficiency score $Eff_{worklet}$ is calculated using the geometric mean of the load level scores, as formulated in Eq. (15). The geometric mean is used here rather than the arithmetic mean or sum in consideration that the arithmetic mean favors load levels with higher efficiency scores.

Usually, higher load levels feature higher efficiency scores for most systems. As a result, a change in energy efficiency at a higher load level has a greater impact on the SERT metric than it would at a lower load level. The geometric mean, on the other hand,

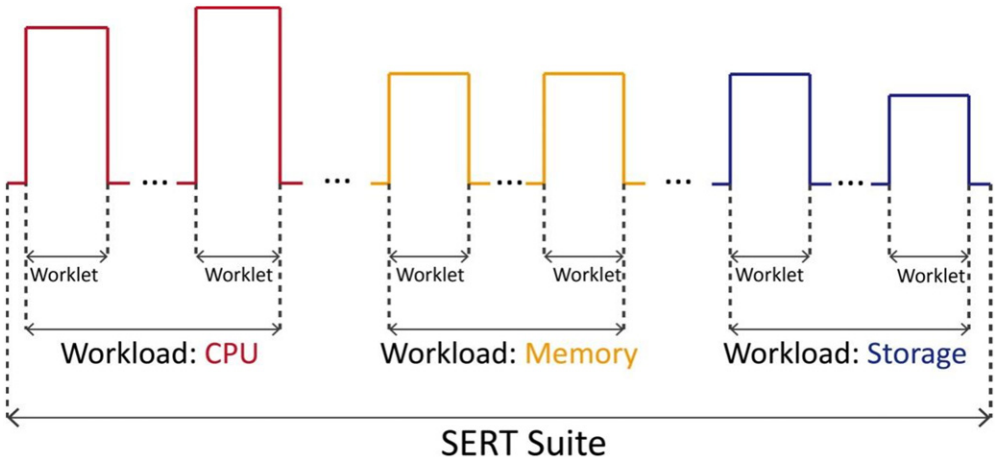

Fig. 6 Composition of the SERT suite test process.

Table 1 Pre-configured load levels for SERT worklets and their associated workload classes

Workload	*Worklet name*	*Load level*
CPU	Compress	100%, 75%, 50%, 25%
	CryptoAES	100%, 75%, 50%, 25%
	LU	100%, 75%, 50%, 25%
	SHA256	100%, 75%, 50%, 25%
	SOR	100%, 75%, 50%, 25%
	SORT	100%, 75%, 50%, 25%
	SSJ	100%, 87.5%,75%, 62.5%, 50%, 37.5%, 25%, 12.5%
Memory	Flood3	Full, Half
	Capacity3	Max, Base
Storage	Random	100%, 50%
	Sequential	100%, 50%
Idle	Idle	Idle

treats relative changes in efficiency equally at each load level. The resulting geometric mean is multiplied with *a factor of 1000* in order to rescale the resulting score into a more intuitive range.

$$Eff_{worklet} = \exp\left(\frac{1}{n} * \sum_{i=1}^{n} \ln\left(Eff_{load\,i}\right)\right) * 1000 \tag{15}$$

In Eq. (15), n represents the number of load levels per worklet and Eff_{load_i} the energy efficiency for load level i.Example:The *CryptoAES* worklet is designed to run at four load levels (25%, 50%, 75%, and 100% load, see **Table 2**). The power and normalized performance are measured directly from the system. In **Table 2**, the efficiency for each load level is calculated according to Eq. (14):

Using the energy efficiency scores for each load level, the worklet efficiency score for *CryptoAES* is computed by plugging these values into Eq. (15), which results in

$$Eff_{worklet} = \exp(1/4 * (\ln(0.281) + \ln(0.337) + \ln(0.365) + \ln(0.425))) * 1000 = 348.33 \tag{16}$$

Workload Efficiency Calculation

Workload efficiency is calculated by aggregating the efficiency scores of all worklets within the workload using the geometric mean (Eq. 17). The worklet efficiency scores to be aggregated are the results of the score calculation in Eq. (15).

$$Eff_{workload} = \exp\left(\frac{1}{n} * \sum_{i=1}^{n} \ln\left(Eff_{worklet\,i}\right)\right) \tag{17}$$

where n represents the number of worklets per workload and $Eff_{worklet\,i}$ is the energy efficiency for each specific worklet. Example: As an example, we calculate the workload energy efficiency for the *storage* workload which consists of two worklets: *Random* and

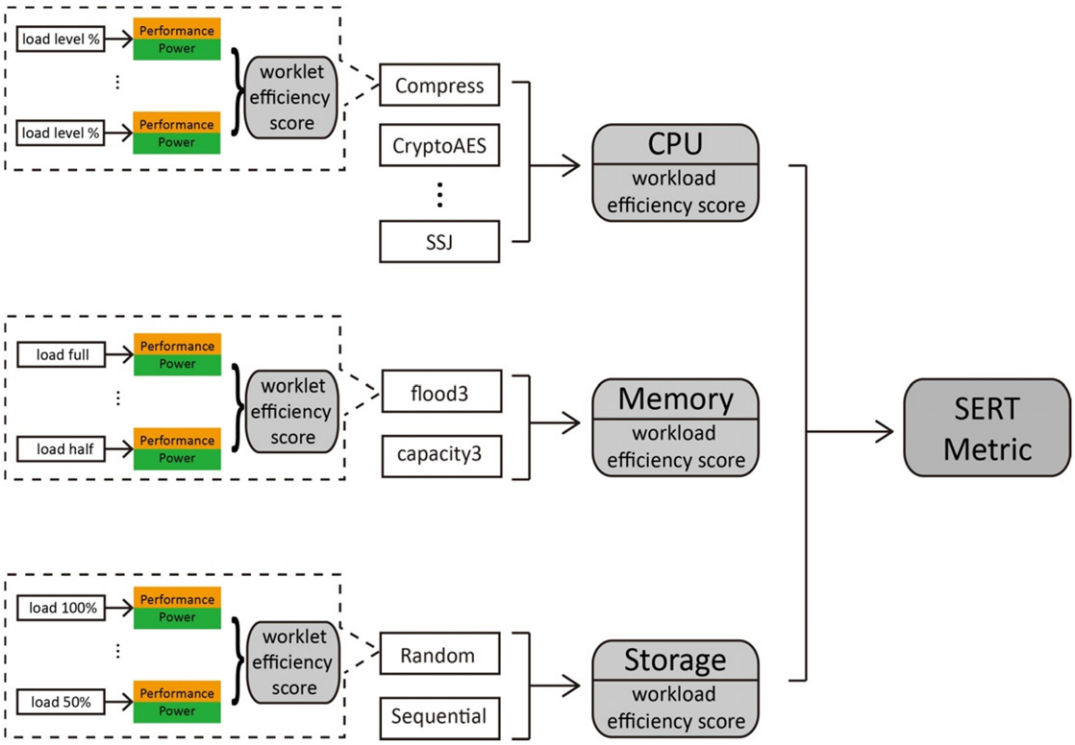

Fig. 7 The calculation procedure of the SERT metric.

Table 2 Example data and efficiency for *CryptoAES* worklet

Load level	*Normalized performance*	*Power (Watts)*	*Efficiency*
25%	49.616	176.583	0.281
50%	99.221	294.086	0.337
75%	148.823	407.581	0.365
100%	199.476	469.005	0.425

Sequential. For brevity, we assume that efficiency scores for both worklets have been calculated as we do in the previous step. Listed in **Table 3** are the corresponding Worklet Efficiency Scores:

As defined, the Workload Efficiency Score for *Storage* is the geometric mean of the associated Worklet Efficiency Score for *Random* and *Sequential*. Using the data in **Table 3**, one can derive the workload efficiency score according to Eq. (17):

$$Eff_{workload} = \exp(1/2 * (\ln(14.36) + \ln(33.86))) = 22.05 \tag{18}$$

Sert Metric Calculation

The SERT metric (or SERT Efficiency Score) (SPEC, 2021) is the final aggregate of all the associated workload scores. In contrast to other workload-dependent metrics, the SERT metric does not consider all workloads equally important. Instead, it uses a weighted geometric mean to differentiate the importance of different workload scores. The pre-defined workload weights are as follows:

- CPU weight: 65% (High)
- Memory weight: 30% (Medium)
- Storage weight: 5% (Low)

These weights are determined by expert groups and represent a realistic importance distribution of the different workloads when comparing SERT to real-world data center workloads. Note that the workloads do not exclusively apply to the specific hardware component after which they are named. More specifically, the *CPU* workloads also incur a certain level of memory usage, whilst the *Memory* workloads also cause an increase in CPU utilization. This means that the weighted efficiency for CPU workloads

Table 3 Example worklet efficiency scores pre-calculated

Worklet	*Worklet efficiency score*
Random	14.36
Sequential	33.86

Table 4 Example Workload Efficiency Scores pre-calculated

Worklet	*Workload efficiency score*
CPU	85.75
Memory	36.50
Storage	22.05

is to some extent influenced by the efficiency of memory and storage components, and vice versa. This is essentially the rationale behind the weight settings. As a result, the SERT metric is calculated as follows (Eq. 19):

$$SERT\ Efficiency\ Score = \exp\left(0.65 * \ln\left(Eff_{cpu}\right) + 0.3 * \ln\left(Eff_{\text{memory}}\right) + 0.05 * \ln\left(Eff_{\text{storage}}\right)\right) \tag{19}$$

Example: Wecalculate the SERT metric of our example system based on the efficiency scores of the three individual workloads, i.e., *CPU*, *Memory* and *Storage*. **Table 4** lists the workload efficiency scores for our example:

Using these workload efficiency scores, the SERT Efficiency Score is calculated using the weighted geometric mean over the workload efficiency scores:

$$SERT\ Efficiency\ Score = \exp(0.65 * \ln(85.75) + 0.3 * \ln(36.50) + 0.05 * \ln(22.05)) = 62.01 \tag{20}$$

Summary

As the scale and power density of data centers keep on growing, the management of physical servers is reckoned as the one of the most important aspects in the path to next-generation sustainable data centers. Server management in data centers involves a comprehensive set of sustainable technologies where the methodology for measuring and rating the energy efficiency of servers is the very foundation. In this article, we first introduce representative metrics of server energy efficiency, illustrate the design rationale, and discuss how they are formulated and parameterized. Next, we present two use cases in which we explain the definition of server energy efficiency in the widely-recognized SPECpower_ssj2008 benchmark and SERT benchmark. Step by step we elaborate the procedure for calculating the energy efficiency at the server level based on sample data from real servers.

The development of energy efficiency metrics plays an important role in the research and practice of server management. The content of this article is expected to provide guidance for the assessment and monitoring of energy efficiency at the server level in data center environments and further, insights for energy efficiency-aware server management for sustainable clouds.

Acknowledgments

This work is supported by the Key-Area Research and Development Program of Guangdong Province (2021B0101420002), National Natural Science Foundation of China (62072187), Guangdong Major Project of Basic and Applied Basic Research (2019B030302002), the Major Key Project of PCL (PCL2021A09) and Guangzhou Development Zone Science and Technology (2021GH10, 2020GH10).

References

Abaunza, F., Hameri, A.P., Niemi, T., 2018. EEUI: A new measure to monitor and manage energy efficiency in data centers. International Journal of Productivity and Performance Management 67, 111–127.

Belady, C.L., Malone, C.G., 2007. Metrics and an infrastructure model to evaluate data center efficiency. In: Proceedings of the ASME InterPACK Conf. Collocated ASME/JSME Thermal Eng. Heat Transfer Summer Conf., pp. 751–755.

Lee, K., Chen, H., 2013. Analysis of energy saving potential of air-side free cooling for data centers in worldwide climate zones. Energy and Buildings 64, 103–112.

Lin, W., Wang, W., Wu, W., *et al.*, 2018a. A heuristic task scheduling algorithm based on server power efficiency model in cloud environments. Sustainable Computing: Informatics and Systems 20, 56–65.

Lin, W., Wu, W., He, L., Li, K., 2022. An on-line virtual machine consolidation strategy for dual improvement in performance and energy conservation of server clusters in cloud data centers. IEEE Transactions on Services Computing 15 (2), 766–777.

Pakbaznia, E., Pedram, M., 2009. Minimizing data center cooling and server power costs. In: Proceedings of the ACM/IEEE International Symposium on Low Power Electronics and Design, pp. 145–150.
Polverini, D., Tosoratti, P., 2018. Towards a metric for the energy efficiency of computer servers. Computer Standards & Interfaces 55, 116–125.
Reddy, V.D., Setz, B., Rao, G.S., Gangadharan, G.R., Aiello, M., 2017. Metrics for sustainable data centers. IEEE Transactions on Sustainable Computing 2, 290–303.
SPEC, 2012a. Design Overview: SPECpower_ssj2008. Available at: http://www.spec.org/power/docs/SPECpower_ssj2008-Design_Overview.pdf (accessed 28.06.22).
SPEC, 2012b. Design Document: SSJ Workload. Available at: http://www.spec.org/power/docs/SPECpower_ssj2008-Design_ssj.pdf (accessed 28.06.22).
SPEC, 2012c. User Guide: SPECpower_ssj2008. Available at: http://www.spec.org/power/docs/SPECpower_ssj2008-User_Guide.pdf (accessed 28.06.22).
SPEC, 2021. The SERT® Metric and Impact of Server Configuration. Available at: https://www.spec.org/sert2/SERT-metric.pdf (accessed 28.06.22)
SPEC, 2022. The SERT® Suite Design Document 2.0.x. Available at: https://www.spec.org/sert2/SERT-designdocument.pdf (accessed 28.06.22).
Stanley, J.R., Brill, K., Koomey, J., 2007. Four metrics define data center greenness. White Paper, Uptime Institute.
Tirmazi, M., Barker, A., Deng, N., *et al*., 2020. Borg: The next generation. In: Proceedings of the fifteenth European Conference on Computer Systems, pp.1–14, New York, NY: ACM.
von Kistowski, J., Lange, K.D., Arnold, J.A., *et al*., 2019. Measuring and rating the energy-efficiency of servers. Future Generation Computer Systems 100, 579–589.
von Kistowski, J., Lange, K.D., Arnold, J.A., *et al*., 2018. Measuring and benchmarking power consumption and energy efficiency. In: Proceedings of the Companion of the 2018 ACM/SPEC International Conference on Performance Engineering, pp. 57–65, New York, NY: ACM.
Wu, J., Guo, S., Li, J., Zeng, D., 2016. Big data meet green challenges: Greening big data. IEEE Systems Journal 10 (3), 873–887.

Further Reading

Bose, R., Roy, S., Mondal, H., Chowdhury, D.R., Chakraborty, S., 2021. Energy-efficient approach to lower the carbon emissions of data centers. Computing 103 (8), 1703–1721.
Energy Efficiency in Data Centers: Recommendations for Government-Industry, 2008. National Data Center Energy Efficiency Strategy Workshop. Available at: https://www1.eere.energy.gov/manufacturing/tech_assistance/pdfs/data_center_wrkshp_report.pdf.
Fryer, E., 2013. Data Centres and Power: Fact or Fiction? Available at: https://doczz.net/doc/3885943/data-centres-and-power–fact-or-fiction.
IBM Cloud Education, 2021. Containerization: Explore the history of containerization technology, the benefits and advantages of utilizing the technology, and how it related to virtualization. Available at: https://www.ibm.com/cloud/learn/containerization.
IEA, 2017. Digitalization and Energy, International Energy Agency. Available at: https://www.iea.org/reports/digitalisation-and-energy.
Madhavapeddy, A., Scott, D.J., 2014. Unikernels: The rise of the virtual library operating system. Communications of the ACM 57 (1), 61–69.
Sharkey, J., Buyuktosunoglu, A., Bose, P., 2007. Evaluating design tradeoffs in on-chip power management for CMPs. In: Proceedings of the 2007 international symposium on Low power electronics and design, pp. 44–49, New York, NY: ACM.
Wu, Q., 2014. Making Facebook's software infrastructure more energy efficient with Autoscale, facebook Engineering. Available at: https://engineering.fb.com/2014/08/08/production-engineering/making-facebook-s-software-infrastructure-more-energy-efficient-with-autoscale/.

Relevant Websites

http://www.spec.org/power/docs.
SPEC power benchmark documentation.
https://www.spec.org/sert2.
SERT 2.xdocumentation.