# Towards a Benchmark for Software Resource Efficiency*

Norbert Schmitt
University of Würzburg
Germany
norbert.schmitt@uni-wuerzburg.de

Richard Vobl
RETIT GmbH
Germany
vobl@retit.de

Andreas Brunnert
RETIT GmbH
Germany
brunnert@retit.de

Samuel Kounev
University of Würzburg
Germany
samuel.kounev@uni-wuerzburg.de

## ABSTRACT

Data centers already account for over 250TWh of energy usage every year and their energy demand will grow above 1PWh until 2030 even in the best-case scenarios of some studies. As this demand cannot be met with renewable sources as of today, this growth will lead to a further increase of CO2 emissions. The data center growth is mainly driven by software resource usage but most of the energy efficiency improvements are nowadays done on hardware level that cannot compensate the demand. To reduce the resource demand of software in data centers one needs to be able to quantify its resource usage. Therefore, we propose a benchmark to assess the resource consumption of data center software (i.e., cloud applications) and make the resource usage of standard application types comparable between vendors. This benchmark aims to support three main goals (i) software vendors should be able to get an understanding of the resource consumption of their software; (ii) software buyers should be able to compare the software of different vendors; and (iii) spark competition between the software vendors to make their software more efficient and thus, in the long term, reduce the data center growth as the software systems require less resources.

## CCS CONCEPTS

• **Computer systems organization** → **Cloud computing**; • **Software and its engineering** → *Software performance*.

## KEYWORDS

resource efficiency, benchmark, cloud, data center, resource utilization, sustainability, software development, software engineering

**ACM Reference Format:**
Norbert Schmitt, Richard Vobl, Andreas Brunnert, and Samuel Kounev. 2021. Towards a Benchmark for Software Resource Efficiency. In *Companion of the 2021 ACM/SPEC International Conference on Performance Engineering (ICPE '21 Companion), April 19–23, 2021, Virtual Event, France.* ACM, New York, NY, USA, 4 pages. https://doi.org/10.1145/3447545.3451176

*Vision Paper


*ICPE '21 Companion, April 19–23, 2021, Virtual Event, France*

ACM ISBN 978-1-4503-8331-8/21/04.
https://doi.org/10.1145/3447545.3451176

## 1 INTRODUCTION

In 2012 the New York Times and Andrae et al. estimated that data centers consume about 30 billion watts of power [1, 10], resulting in about 263 TWh[1] of energy usage per year. Andrae et al. also estimate that the energy consumption will rise to 1137 TWh until 2030 in the best case scenario [1]. The data center growth and the resulting energy consumption is mainly driven by the computing resource usage (i.e., processor, memory, network, and storage) of deployed applications. This growing energy demand also increases the amount of CO2 emissions as it is not possible to operate all data centers at all times with renewable energy. Assuming a CO2-intensity of 200g CO2 per kWh in a rather optimistic scenario[2], a consumption of 264TWh will result in about 53 billion tons of CO2 produced to operate data centers and 1137TWh will get us close to 30 billion tons of CO2.

While computing resource and energy efficiency are not identical, resource efficiency entails energy efficiency. As the instructions of an application control the hardware, it indirectly controls energy consumption. For example, a higher CPU utilization will prohibit the CPU from going into sleep states. The resource and energy efficiency is also not identical to performance as it first seems. Capra et al. showed that two different ERP software systems have little performance differences (+5%) but deviate widely in energy consumption (+50%) [6], resulting in a different efficiency. Hence, it is important to reduce the resource usage of data centers as technological advances cannot fully compensate for their growth [18].

To reduce the resource usage of data centers, operators can try to find optimal deployment configurations for their software and use software that is, in general, more resource-efficient than others when fulfilling the same tasks. While deployment and runtime resource optimizations have been and are still extensively researched, software resource efficiency is still a very challenging topic. Therefore, we need to raise awareness of developers and data center operators about the resource efficiency of applications to tackle the towering problem of climate change due to excessive wastage.

To achieve awareness among everybody involved, from designing, writing, and operating software products; we envision a new benchmark that is capable of assessing the resource consumption of data center software (i.e., cloud applications) and make the resource usage of standard application types (e.g., CRM, ERP) comparable.

[1] $30 \cdot 10^9 W \cdot 365 \cdot 24h = 262.8 \cdot 10^{12} Wh$
[2] https://www.electricitymap.org, accessed on September 30th, 2020

Our benchmark aims to support three main goals (i) software vendors should be able to get an understanding of the resource consumption of their software; (ii) software buyers should be able to compare the software of different vendors in terms of their resource efficiency; and (iii) spark competition between the software vendors to make their software more efficient and thus, reduce the data center growth as the software systems require fewer resources.

## 2 SETTING THE CONTEXT

Energy saving is mostly achieved through more energy-efficient hardware, including improved manufacturing processes, granular sleep states, and dynamic voltage and frequency scaling (DVFS). Runtime optimizations try to minimize resource wastage by consolidating services to larger instances, place services optimally inside a data center [16], and only have as many servers or service instances running to satisfy current demand (auto-scaling) [12]. These optimizations are concerned with the hardware the software is running on and where and when the software is running. We argue that for the software to operate energy-efficient, the infrastructure is not the only factor; the software must in itself be resource-efficient as resource-efficiency entails energy-efficiency. However, the software that provides a service and controls the underlying hardware is, in our experience, less often the center of attention.

Developers can use compiler optimizations to increase performance, a topic related to resource and energy efficiency but not identical [6]. Several efforts reduce the energy consumption of computing devices by leveraging different performance optimizations of compilers [8, 19]. Developers could also incorporate resource and energy efficiency in the software design instead of optimizing the binary at compile-time. For instance, with different choices of API calls [21]. The programming language could also play a role [23] by aiding the developer in writing efficient code or be more easily optimized for resource and energy use. These options could also increase the resource efficiency of the software. Yet, they are not independent. A monolithical software design could favor resource efficiency and extensive batch processing, but might be less suitable for auto-scaling and optimal placement decision making.

One reason that hardware has gotten more energy-efficient over time than software is the availability of suitable benchmarks. Many benchmarks measuring energy-efficiency or power exist. For example, the Transaction Processing Performance Council (TPC) released the TPC-Energy Specification [25] to augment existing TPC benchmarks with energy measurements. The Standard Performance Evaluation Corporation (SPEC) adds power measurements directly to its benchmarks, such as the SPECpower_ssj 2008 [14] and SPEC CPU 2017 [7]. Both approaches by SPEC and TPC measure the hardware power consumption. Therefore, a benchmark to describe the resource demand of a software system itself is missing to make it comparable with other software systems to optimize them.

The majority of literature on benchmarking [17] is not focused software itself but rather on hardware. Some works focus on cloud computing but address solely the performance [2], or measure hardware performance properties with synthetic benchmarks, like STREAM or Linpack, and not the resource efficiency of deployed cloud software [9]. The German Environment Agency has also identified the need to measure the software's resource and energy efficiency. Therefore, they have published a study on how software efficiency of typical desktop software can be determined [11]. Our benchmark, on the other hand, aims at measuring the resource-efficiency of software running in cloud environments. It aims at providing a comparable metric of resource-efficiency for different products (e.g., SAP S/4) of the same software type (e.g., ERP software) similar to the current energy-efficiency ratings of fridges or cars in Europe (e.g., A++, A+, B). A fundamental assumption of our benchmark is therefore, that more resource-efficient software will lead to less resource consumption and as such to less energy consumption in data centers.

## 3 OPEN CHALLENGES

In order to provide the foundation for more resource efficient software multiple open challenges have to be addressed.

Challenge 1: *How to describe the resource demand of software?* To the best of our knowledge, there is currently no commonly accepted standard to express the resource demand of software. While resource profiles are an approach to describing a software system's resource demand, they are dependent on the corresponding workload [5]. For client-side software, vendors typically try to define minimum hardware requirements, whereas, for server-side software capacity planning, trial and error is a commonly accepted norm to deal with software resource requirements [24, 28]. However,the results of capacity planning only describe the implications of a specific workload on that software, not the actual software resource demands. One of the main consequences of this approach is that even today, there are a lot of underutilized servers because the resource requirements have been estimated using wrong assumptions [15]. Therefore, it would be good to make the resource demand more transparent for the users, not only at run time but also during purchasing decisions. This transparency would allow the software users to consider the actual resource demands during their purchasing decisions and would create pressure on the vendors to reduce these numbers to convince the buyers to choose their product [5]. Eventually, this would drive down the resource and energy demand of all software systems.

Challenge 2: *How to specify and standardize workloads?* In previous work [4], the authors presented an approach based on [3] to describe the resource demand of individual transactions of a software system, but the workload needs to be standardized to make it comparable. The workload of software systems is very dependent on their type and can only standardized to a certain degree. According to [26], a workload is defined by behavior models combined into a behavior mix, which then has a certain workload intensity. Behavior models describe a typical usage flow of a user type (e.g., an employee vs. customer on an e-commerce page). The behavior mix specifies which percentage of the overall user population follows which behavior model. The workload intensity defines how many users with a given behavior mix are active in the system at a time. These workload parts will differ per application type (e.g., ERP, HR, CRM, Industry 4.0 software), and software vendors need to agree on a standard workload model for each application type in order to make resource demands comparable.

Challenge 3: *What are possible incentives to increase awareness and acceptance?* As the experience with other devices such as cars

or fridges shows, even if we would have the technical means and standardized workloads, the awareness of the stakeholders (e.g., software buyers, operators, and providers) needs to be increased to make the overall concept work. Therefore, this is a socio-technical problem that cannot be solved solely by providing a technical solution. Thus it might be possible to adapt working approaches from similar incentive-based mechanisms. Examples for this are the adoption of labels like the Energy Star [22] or motivation for environmental protection measures as a whole [20].

The research to address this is, therefore, twofold. On the one hand, software engineering research needs to agree on techniques and tools to measure and describe the resource demand of software. On the other hand, information system research needs to take the socio-technical aspects into account to ensure that the software engineering research results are used in practice.

## 4 VISION

We want to address the aforementioned challenges by providing an easy to use means to describe and assess the resource demand of software. Based on this description, we can define a resource-efficiency metric that allows the classification of and comparison between the resource efficiency of software for users and vendors.

To approach challenge 1, we will define a benchmark framework, shown in Figure 1, that can be used to collect resource demand data for different software types and will define:

**A format to describe the software resource demand.** In order to assess and compare the software resource demand, it should be possible to store and process it in a common way. Therefore, a format will be introduced to describe the software resource demand.

**A measurement adapter.** Resource demand can be collected using various tools. In order to produce the same output format, the benchmark framework needs to provide a measurement adapter used by the individual tools to store the measurements.

**A workload specification format.** The workload for comparable resource demand measurements needs to be defined in two layers: (i) in an abstract way for the workload of a particular application type and (ii) in a specific way for the workload for the particular software of an application type. The vendors should provide specific workload scripts. However, the abstract description format should be provided by the benchmark framework.

**An execution environment description.** It should be possible to clearly specify where the application is executed as part of an execution environment description. This description should be standardized by the benchmark framework. It is important that specific characteristics (e.g., CPU, network, or disk types) are outlined in detail in order to make the results comparable.

**An execution engine.** The benchmark should be able to execute the resource demand measurement based on the workload specification, measurement adapters, and execution environments.

**A central storage repository.** The benchmark framework should be able to store all results in a central repository. This repository allows a comparison of results over time as an important aspect to spark competition. Here it is crucial to find a commonly accepted way that can be agreed on by all stakeholders (e.g., software providers and consumers) to make the comparison easily accessible for everyone.

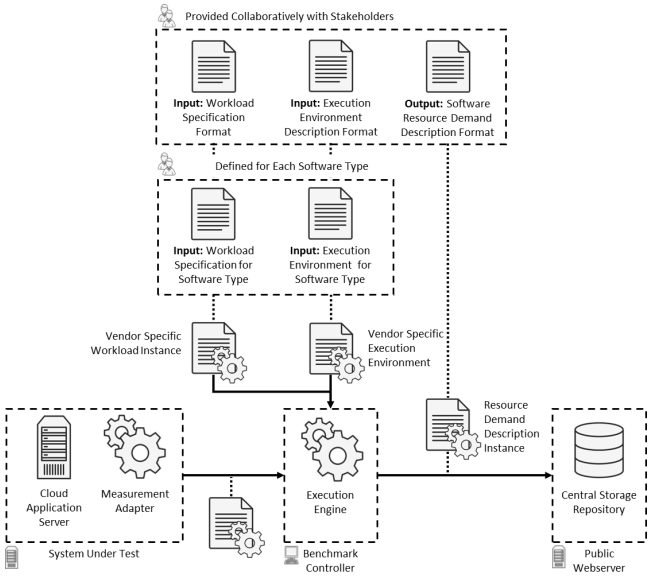


**Figure 1: Benchmark Vision with input, output, component, and specification artifcats**

In order to facilitate the standardization and acceptance of its results, this benchmark framework will be developed collaboratively with other stakeholders. Therefore, the Standard Performance Evaluation Corporation (SPEC) Research Group (RG)[3] has been chosen as a platform for its development. SPEC *is a non-profit corporation formed to establish, maintain and endorse standardized benchmarks and tools to evaluate performance and energy efficiency for the newest generation of computing systems.* SPEC's server efficiency rating tool (SERT) is now the *required test method for [the] ENERGY STAR for Computer Servers specification* and their SPECpower_ssj® 2008 benchmark has helped to improve the average operations-per-wattserver efficiency for servers by 19 times[4] by creating transparency through published results and thus creating incentives for vendors to improve their results. This is the kind of impact we envision for a benchmark built upon the presented framework that gives software vendors an incentive to increase their resource efficiency by using the benchmark results to attract potential customers.

Members of SPEC include several soft- and hardware vendors, which can help facilitate the process of defining standardized workloads from challenge 2, as they can be addressed, e.g., via surveys, to help determine the workloads for the different application types.

In addition to the technical work within SPEC, we will continuously share our results in the software engineering and information systems communities to increase the awareness about the issue and steer the development of more resource-efficient software (e.g., by automatic comparisons in the delivery pipelines). As more resource-efficient software can give a competitive edge, this is a suitable way to address challenge 3. Once endorsed by SPEC as an official research benchmark, the benchmark itself is able to draw attention and further increase awareness on this topic.

[3] https://research.spec.org/
[4] https://www.spec.org/30th/power.html

With software as the main driving force for the provisioning of data centers, solving the mentioned challenges and laying the foundation for more resource efficient software can play its role in collectively lowering the ecological footprint of the IT sector.

## 5 LIMITATIONS

While we envision our benchmark to be as generic and easy to use as possible, limitations exist. In this section, we present an explanation of the most limiting factors.

*Limited abstraction from hardware.* Abstracting the relationship of hard- and software can be difficult or even not entirely possible. So the influence of hardware on resource efficiency must be minimized. One solution might be to measure with a reference machine to achieve comparability with the downside that it might not be available for everyone or is no longer in production. Switching the reference machine would also make newer results of our benchmark incompatible with older ones and conflicting with the goal of comparability in benchmarks [13, 27]. Hence, possible hardware influences must be acknowledged and ensured to be minimal.
*Impact of resource efficiency on operational costs.* Even though we would achieve a workload standardization and inclusion of the resource demands into buying decisions, it does not mean that software is operated efficiently. It is, of course, very well possible to run a resource efficient software quite inefficiently. However, as only resource efficiently designed and implemented software can be operated efficiently, we will focus on the implementation and design time aspects of the resource demands. The reoccurring cloud hosting cost will likely influence the hosting process into a money-optimized format anyway.

*Custom build software artifacts and customer specific features.* While we aim for a generic workload specification, it is unfeasible to consider every additional feature or software built to order for a specific customer. These features and software artifacts are only convenient to said customer and are not widely used. Therefore, our envisioned benchmark limits itself to the common denominator of features available of the application type.

## 6 CONCLUSION

As software is the main reason for the provisioning of data centers, their increasing energy demand and ecological footprint, software professionals need an understanding of the resource efficiency of software, and the impact resource-efficient software can have on reducing energy usage and waste. We, therefore, envision a new benchmark, outlining three essential challenges: i) describing and measuring software resource demand in a comparable manner, ii) standardizing workloads for data center software with similar functionality, and iii) incentivizing possible users. With our vision in Section 4, we sketch out possible solutions to the introduced challenges with a benchmark framework and potential collaborations.

## REFERENCES


[1] Anders Andrae and Tomas Edler. 2015. On Global Electricity Usage of Communication Technology: Trends to 2030. *Challenges* 6, 1 (Apr 2015), 117–157.

[2] David Bermbach, Erik Wittern, and Stefan Tai. 2017. *Cloud Service Benchmarking* (1 ed.). Springer International Publishing.

[3] Reinhard Brandl, Martin Bichler, and Michael Ströbel. 2007. Cost accounting for shared IT infrastructures. *Wirtschaftsinformatik* 49, 2 (2007), 83–94.

[4] Andreas Brunnert and Helmut Krcmar. 2017. Continuous performance evaluation and capacity planning using resource profiles for enterprise applications. *Journal of Systems and Software* 123 (2017), 239 – 262.

[5] Andreas Brunnert, Kilian Wischer, and Helmut Krcmar. 2014. Using Architecture-Level Performance Models as Resource Profiles for Enterprise Applications. In *Proceedings of the 10th International ACM Sigsoft Conference on Quality of Software Architectures (QoSA '14)*. Association for Computing Machinery, New York, NY, USA, 53–62.

[6] Eugenio Capra, Chiara Francalanci, and Sandra A. Slaughter. 2012. Is software "green"? Application development environments and energy efficiency in open source applications. *Information and Software Technology* 54, 1 (2012), 60 – 71.

[7] Bob Cramblitt. 2019. SPEC releases new version of CPU benchmark suite. Press Release http://www.spec.org/cpu2017/press/v1_1_release.html.

[8] Stefan Valentin Gheorghita, Henk Corporaal, and Twan Basten. 2005. Iterative compilation for energy reduction. *Journal of Embedded Computing* 1, 4 (2005), 509–520.

[9] Lee Gillam, Bin Li, John O'Loughlin, and Anuz Pratap Singh Tomar. 2013. Fair benchmarking for cloud computing systems. *Journal of Cloud Computing: Advances, Systems and Applications* 2, 1 (2013), 1–45.

[10] James Glanz. 2012. Power, Pollution and the Internet. *New York Times* (September 23rd 2012).

[11] Jens Gröger, Andreas Köhler, Stefan Naumann, Andreas Filler, Achim Guldner, Eva Kern, Lorenz Hilty, and Yuliyan Maksimov. 2018. Entwicklung und Anwendung von Bewertungsgrundlagen für ressourceneffiziente Software unter Berücksichtigung bestehender Methodik-Abschlussbericht. *UBA TEXTE* 105 (2018).

[12] Nikolas Herbst. 2018. *Methods and Benchmarks for Auto-Scaling Mechanisms in Elastic Cloud Environments.* Ph.D. Dissertation. University of Würzburg, Germany.

[13] Karl Huppler. 2009. The Art of Building a Good Benchmark. In *Performance Evaluation and Benchmarking*, Raghunath Nambiar and Meikel Poess (Eds.). Springer Berlin Heidelberg, Berlin, Heidelberg, 18–30.

[14] Karl Huppler, Klaus-Dieter Lange, and John Beckett. 2012. SPEC: Enabling Efficiency Measurement. In *Proceedings of the 3rd ACM/SPEC International Conference on Performance Engineering (ICPE '12)*. Association for Computing Machinery, New York, NY, USA, 257–258.

[15] J. Jang, M. Jeon, H. Kim, H. Jo, J. Kim, and S. Maeng. 2011. Energy Reduction in Consolidated Servers through Memory-Aware Virtual Machine Scheduling. *IEEE Trans. Comput.* 60, 4 (2011), 552–564.

[16] Yichao Jin, Yonggang Wen, and Qinghua Chen. 2012. Energy Efficiency and Server Virtualization in Data Centers: An Empirical Investigation. In *2012 IEEE Conference on Computer Communications Workshops*. 133–138.

[17] Samuel Kounev, Klaus-Dieter Lange, and Jóakim von Kistowski. 2020. *Systems Benchmarking* (1 ed.). Springer International Publishing.

[18] Eric Masanet, Arman Shehabi, Nuoa Lei, Sarah Smith, and Jonathan Koomey. 2020. Recalibrating global data center energy-use estimates. *Science* 367, 6481 (2020), 984–986.

[19] Ricardo Nobre, Luís Reis, and João M. P. Cardoso. 2018. Compiler Phase Ordering as an Orthogonal Approach for Reducing Energy Consumption. *CoRR* abs/1807.00638 (2018). arXiv:1807.00638 http://arxiv.org/abs/1807.00638

[20] Stuart Oskamp. 2000. A sustainable future for humanity? How can psychology help? *The American psychologist* 55 5 (2000), 496–508.

[21] G. Rocha, F. Castor, and G. Pinto. 2019. Comprehending Energy Behaviors of Java I/O APIs. In *2019 ACM/IEEE International Symposium on Empirical Software Engineering and Measurement (ESEM)*. 1–12.

[22] Andrew R Sanderford, Andrew P. McCoy, and Matthew J. Keefe. 2017. Adoption of Energy Star certifications: theory and evidence compared. *Building Research and Information* (6 Jan. 2017), 1–13.

[23] Norbert Schmitt, James Bucek, Klaus-Dieter Lange, and Samuel Kounev. 2020. Energy Efficiency Analysis of Compiler Optimizations on the SPEC CPU 2017 Benchmark Suite. In *Proceedings of the 11th ACM/SPEC International Conference on Performance Engineering (ICPE 2020)*. ACM, New York, NY, USA, 4.

[24] Connie U. Smith. 2007. *Introduction to Software Performance Engineering: Origins and Outstanding Problems.* Springer Berlin Heidelberg, Berlin, Heidelberg, 395–428.

[25] Transaction Processing Performance Council (TPC) 2009. *TPC-Energy Specification.* Transaction Processing Performance Council (TPC). Version 1.0.0.

[26] Christian Vögele, André van Hoorn, Eike Schulz, Wilhelm Hasselbring, and Helmut Krcmar. 2018. WESSBAS: extraction of probabilistic workload specifications for load testing and performance prediction–a model-driven approach for session-based application systems. *Software & Systems Modeling* 17, 2 (2018), 443–477.

[27] Jóakim von Kistowski, Jeremy A. Arnold, Karl Huppler, Klaus-Dieter Lange, John L. Henning, and Paul Cao. 2015. How to Build a Benchmark. In *Proceedings of the 6th ACM/SPEC International Conference on Performance Engineering (ICPE 2015) (ICPE '15)*. ACM, New York, NY, USA.

[28] C. Murray Woodside, Greg Franks, and Dorina C. Petriu. 2007. The Future of Software Performance Engineering. *Future of Software Engineering (FOSE '07)* (2007), 171–187.