Green Software Metrics

Andreas Brunnert
Munich University of Applied Sciences HM
Munich, Germany
brunnert@hm.edu

ABSTRACT

Efficiency has always been at the core of software performance engineering research. Many aspects that have been addressed in performance engineering for decades are gaining popularity under the umbrella of Green IT and Green Software Engineering. Engineers and marketers in the industry are looking for ways to measure how green (in terms of carbon dioxide emissions) their software products are. Proxy measures are proposed, such as hosting cost or the power consumption of the hardware environment on which the software is running. In environments where a software system runs on a dedicated server instance, this may make sense, but in virtualised, containerised or serverless environments, it is necessary to find ways of allocating the energy consumption of the entire server to software components that share the same infrastructure. This paper proposes the use of resource demand measurements as a basis for measuring how green a given software actually is.

CCS CONCEPTS

• **Software and its engineering** → **Software performance**.

KEYWORDS

Green IT, Green Software Engineering, Resource Demand

ACM Reference Format:
Andreas Brunnert. 2024. Green Software Metrics. In *Companion of the 15th ACM/SPEC International Conference on Performance Engineering (ICPE '24 Companion), May 7–11, 2024, London, United Kingdom.* ACM, New York, NY, USA, 2 pages. https://doi.org/10.1145/3629527.3652883

1 INTRODUCTION

In order to meet global carbon dioxide (CO2) reduction targets, the IT industry needs to be able to quantify its emissions. Without quantifiable emissions, it is difficult to identify improvements and assess their impact. While it is common to use the energy consumption of servers or entire data centers to derive their CO2 emissions [1], there is no comparable metric for software [4]. Modern software systems run in virtualised, containerised or serverless infrastructures, for which we need to find ways of allocating the CO2 emissions of entire servers to individual software components or transactions [5].

To achieve this goal, several proxy approaches are currently being used in the industry [3, 4]. One proxy for carbon emissions is the hosting cost of a software system [3] (e.g., when renting instances from a cloud provider). The disadvantage of this approach is that the price of the runtime environment does not correlate exactly with the resource or CO2 emissions of the software. Just looking at the price differences for the same runtime environment on a cloud environment when choosing different payment options (e.g., up-front or on-demand) shows that there is no direct correlation between price and resource use.

Another proposal from the Green Software Foundation is the Software Carbon Intensity (SCI) specification[1]. SCI defines the carbon emissions of a software for a given unit of work (e.g., a request to the system). The SCI specification combines the carbon emissions of all components and transactions of a software system into a single rate value. The advantage of this approach is the simplicity of the result but it makes it difficult to understand the carbon emissions of specific transactions or components of a system.

As an alternative approach, the Green Software Measurement Model (GSMM) [4] attempts to describe a reference model for assessing the resource and energy efficiency of software products and components. GSMM focuses mainly on the individual components of a software system, without considering their interrelationships while processing individual units of work (e.g., transactions). Therefore, the authors [4] also note that the current GSMM methods are not fully applicable to complex architectures or distributed systems.

To overcome the limitations of the above approaches, this work proposes the use of resource demand measurements at the level of individual components and transactions as a basis for measuring how green a software is.

2 RESOURCE DEMAND MEASUREMENTS AS GREEN SOFTWARE METRICS

Measuring or calculating [6] the resource demands (i.e., CPU, memory, storage, network) of software systems is common in the software performance engineering community, as such data is required for capacity planning and performance modeling techniques. We propose to use the same data to quantify the emissions of a software system on a given runtime environment.

When measuring resource demand for a specific transaction, typical metrics collected are CPU time, bytes allocated in memory, or bytes written to/from storage or the network. Many of these metrics, such as the amount of memory consumed by a transaction or the amount of bytes written to or read from storage or the network, are independent of the underlying hardware, as they are primarily influenced by the parameters of a particular transaction. The only metric that is tied to the actual processor used during the measurements is CPU time.

In a previous work we have already shown and evaluated the ability to measure all these resource demands of a software system

[1]https://sci.greensoftware.foundation

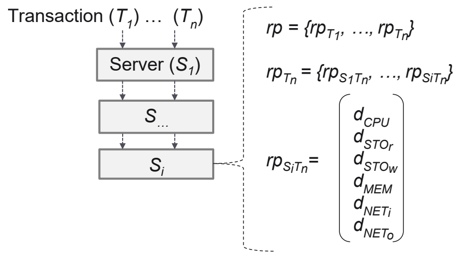
Transaction (T_1) … (T_n)
Server (S_1)
$S_{...}$
S_i
$rp = \{rp_{T_1}, …, rp_{T_n}\}$
$rp_{T_n} = \{rp_{S_1T_n}, …, rp_{S_iT_n}\}$
$rp_{S_iT_n} = (d_{CPU}, d_{STO_r}, d_{STO_w}, d_{MEM}, d_{NET_i}, d_{NET_o})$

Figure 1: Resource Profile [2]

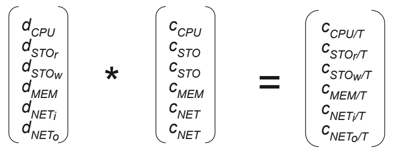
$$\begin{pmatrix} d_{CPU} \\ d_{STO_r} \\ d_{STO_w} \\ d_{MEM} \\ d_{NET_i} \\ d_{NET_o} \end{pmatrix} * \begin{pmatrix} c_{CPU} \\ c_{STO} \\ c_{STO} \\ c_{MEM} \\ c_{NET} \\ c_{NET} \end{pmatrix} = \begin{pmatrix} c_{CPU/T} \\ c_{STO_r/T} \\ c_{STO_w/T} \\ c_{MEM/T} \\ c_{NET_i/T} \\ c_{NET_o/T} \end{pmatrix}$$

Figure 2: Calculating Carbon Emissions per Transaction

[2]. Therefore, we propose to use this data as a basis for assessing the carbon intensity of a software system. To structure the data we use so-called resource profiles, which can store the data for each transaction of a given software system separately by server (see Figure 1), software component or even at the level of individual operations [2]. A resource profile (*rp*) is a set of vectors (i.e., rp_{T_n}) that describe the resource demand (*d*) for individual transactions (*T*, numbered from *1* to *n*) for a specific workload and a certain set of servers (*S*, numbered from *1* to *i*). Resource profiles contain resource demands for the following resource types: CPU (d_{CPU}), storage (differentiated by read d_{STO_r} and write d_{STO_w} operations), memory (d_{MEM}), and network (differentiated by incoming d_{NET_i} and outgoing d_{NET_o} traffic).

A resource demand vector ($rp_{S_i T_n}$) of a transaction on a given server (virtual machine, container or serverless component) can now be used to derive carbon emission metrics based on the carbon intensity of the underlying hardware components as shown in Figure 2. For this calculation we need to be able to quantify the carbon emissions (*c*) of the different resource types (CPU: c_{CPU},storage: c_{STO}, memory: c_{MEM}, network: c_{NET}) in the same unit as the resource demand is stored in the vector. For CPU, we need to know how much carbon is emitted when a core is running at a certain utilisation level; for memory, the carbon emissions for a given size (e.g., GB); and for storage and network, how much carbon is emitted when a given amount of data is processed. This data can be collected using services such as climatiq[2] for CPU, storage and memory for all common cloud environments, instance types and regions. Based on the measured resource demand data and carbon emissions of the individual resources the carbon emissions for individual transactions of a software system can be calculated as shown in Figure 2 by mutliplying the resource demand data with the carbon emissions. The use of both data sources allows the carbon intensity of software to be derived in more detail than is currently possible in industry.

To evaluate the feasibility of the aforementioned proposal, we are currently implementing a prototype based on the architecture

[2]https://www.climatiq.io

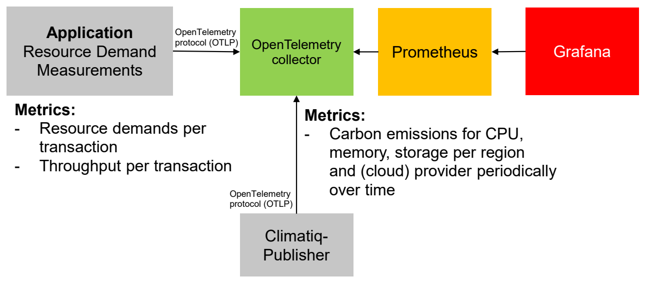
Application
Resource Demand
Measurements
OpenTelemetry protocol (OTLP)
OpenTelemetry collector
Prometheus
Grafana
Metrics:
- Resource demands per transaction
- Throughput per transaction
Metrics:
- Carbon emissions for CPU, memory, storage per region and (cloud) provider periodically over time
OpenTelemetry protocol (OTLP)
Climatiq-Publisher

Figure 3: Prototype Architecture

in Figure 3. We are using OpenTelemetry[3] as a standard for transferring the required metrics from the application and climatiq to Prometheus[4] as a time series database. We need to store both sets of data (resource demand and emission data) correlated by time as emissions vary over time depending on the amount of green energy used to power the runtime environment (e.g., lack of solar power at night). Grafana[5] is used to visualize the results of this calculation to show the carbon emissions per transaction as well as per server, container, or virtual machine involved in processing a transaction over time.

3 CONCLUSION & FUTURE WORK

The use of resource demand measurements helps to provide a more detailed insight into how much carbon a software system emits (in other words, how green it is). We are currently building a prototype that links all the different parts together (resource demand data collection, carbon emission data collection and calculation). Once this work is done, we plan to run software experiments to evaluate our proposal against other approaches on the market, such as the SCI or the GSMM[4]. We also do not currently include the carbon emissions of network traffic in our prototype as we do not have the necessary emissions data, but we are looking for suitable data sources to fill this gap in the future.

REFERENCES

[1] L. Belkhir and A. Elmeligi. Assessing ict global emissions footprint: Trends to 2040 & recommendations. *Journal of Cleaner Production*, 177:448–463, 2018.
[2] A. Brunnert and H. Krcmar. Continuous performance evaluation and capacity planning using resource profiles for enterprise applications. *Journal of Systems and Software*, 123:239–262, 2017.
[3] A. Currie, S. Hsu, and S. Bergman. *Building Green Software*. O'Reilly Media, Inc., 2024.
[4] A. Guldner, R. Bender, C. Calero, G. S. Fernando, M. Funke, J. Gröger, L. M. Hilty, J. Hörnschemeyer, G.-D. Hoffmann, D. Junger, T. Kennes, S. Kreten, P. Lago, F. Mai, I. Malavolta, J. Murach, K. Obergöker, B. Schmidt, A. Tarara, J. P. De Veaugh-Geiss, S. Weber, M. Westing, V. Wohlgemuth, and S. Naumann. Development and evaluation of a reference measurement model for assessing the resource and energy efficiency of software products and components—green software measurement model (gsmm). *Future Generation Computer Systems*, 155:402–418, 2024.
[5] A. Katal, S. Dahiya, and T. Choudhury. Energy efficiency in cloud computing data centers: a survey on software technologies. *Cluster Computing*, 26(3):1845–1875, June 2023.
[6] F. Willnecker, M. Dlugi, A. Brunnert, S. Spinner, S. Kounev, W. Gottesheim, and H. Krcmar. Comparing the accuracy of resource demand measurement and estimation techniques. In M. Beltrán, W. Knottenbelt, and J. Bradley, editors, *Computer Performance Engineering*, pages 115–129, Cham, 2015. Springer International Publishing.

[3]https://opentelemetry.io
[4]https://prometheus.io
[5]https://grafana.com