Mirroring Complex Software Systems for In Vitro Experimentation on Computational Resource Waste

Vincenzo Stoico, Ivano Malavolta and Patricia Lago

1Vrije Universiteit Amsterdam, De Boelelaan 1111, 1081 HV Amsterdam, The Netherlands

Abstract

Context. Software systems can often be inefficient and may waste the resources provided by the computing infrastructure. Waste of computing resources can have a significant impact on the environment, especially nowadays, where software has become ubiquitous, and the need to compute large volumes of data is increasing rapidly. Additionally, the complexity of modern software systems makes it difficult to optimize them at a fine-grained level, which can help reduce resource waste and, in turn, minimize the carbon footprint of computing.

Solution. We propose a novel approach to studying complex software systems in vitro, i.e., in a research laboratory setting. Our approach involves a model, namely the mirror, that reflects the resource usage of existing software systems. The mirror system is built upon traces retrieved from a real software system in production and semi-automatically refined until it approximates the resource usage of the source system. As a result, the mirror system can be adjusted to test various (architectural) tactics designed to minimize resource waste.

Relevance. This project aims to help the developers of software systems to minimize the resources wasted by their software during their execution and their optimization process. The mirror system allows researchers to study complex software systems and explore various software optimization techniques, without impacting the system in production.

Keywords

Software Engineering, Green Software, Energy Consumption, Software Performance

Problem Formulation

Computing has improved our quality of life, but at a significant environmental cost. Terabytes of data can be relentlessly collected from remote devices and, then processed and stored in data centers distributed across the globe. DOMO estimates that a person produces 102 MB of data every minute 1, while Statista 2 provides a global-scale estimation of 97 zettabytes (i.e., 9.7e + 13 gigabytes) of data manipulated in 2022. Computing great volumes of data offers important innovations, such as wider access to education, identification of intricate patterns in complex processes, and improved patient care in healthcare. However, the increasing demand for computation has a significant carbon footprint, particularly in terms of the resources consumed during the computation process and by the computing infrastructure. Large Language Models (LLMs) are among the most prominent examples of the negative impact of computation on the...
Training Llama and Code Llama, two LLMs made by Meta, produced, respectively, nearly 1,1015 tons of carbon emission (tCO2eq) [1] and 63.5 tCO2eq [2]. The environmental impact of computing becomes dramatic when considering the resources required to maintain and dispose of the computing infrastructure, such as the water needed to cool data centers, and the waste produced by the decommissioning of billions of devices [3].

Software optimization can be fruitful in reducing the environmental impact of computation, but it can be problematic due to the complexity of modern software systems. Modern software systems may involve multiple layers of abstractions, external libraries, programming languages, and can be deployed on various configurations of distributed infrastructures. As an example, Spotify is composed of more than 14000 software components³. Complexity hampers fine-grained monitoring of software systems and increases the probability of performance bottlenecks, which might lead to resource wastage. Inefficient data structures and redundant operations are just two examples of software inefficiencies impacting resource utilization [4, 5]. Cloud computing enables the efficient provisioning and scaling of computational resources on demand [6], but this is not enough to reduce the waste of resources. As indicated by Forbes⁴ and Common⁵, 30% of global Cloud spending is wasted. Everman et al. [7] confirm resource waste in the Cloud by analyzing 235 GB of data created by 6687 users on Azure Cloud retrieved by 2.7 million virtual machines. The study shows that the average CPU utilization during the lifetime of the considered virtual machines does not exceed 24%. In this poster proposal, we describe an approach to study the resource utilization of complex software systems in a controlled environment, called the mirror system.

The Mirror System

The mirror is a software system that approximates the resource usage of a subset or the entire software under study. Our goal is to use the mirror system to identify performance- and energy hotspots in existing software systems and optimize them through the definition and application of so-called tactics. A tactic is a "design decision that influence the achievement of a quality attribute response" [8]. In the context of our research, the mirror system represents a surrogate of a software system that can be easily modified for discovering the design decisions that led to resource wastage and investigating the impact of several design alternatives (i.e., tactics) to reduce resources underutilization, and finally reduce the impact of computation on the environment. As shown in Figure 1, the mirror system is built upon traces of the original system collected at runtime. Traces can include information at different granularity according to the purpose of the investigation. For example, to quantify the impact of various architectural designs on the performance and energy consumption of a software system, used traces should include the topology of the modules that make up the software, as well as the resource usage of the modules involved in the architecture. After creating and verifying the accuracy of the mirror system, the researchers can use it to observe the impact of the tactics on resource usage. The analysis results, including applied tactics, expected optimization, and instructions

³https://engineering.atspotify.com/opensource/
⁵https://www.cloudzero.com/blog/cloud-waste/
Figure 1: Use case scenario for the mirror system.

Approximating the resource usage of a software system to optimize it is a common practice in software engineering. Queuing models are widely used to test autoscaling policies [9], evaluate architectural alternatives [10], and autonomic managers [11]. In many cases, these models are used at design time from the specification of the software system. We build the mirror system from an existing software system running in its production environment, rather than a specification. Thus, we refine the mirror system in a bottom-up fashion as we replicate the resource usage of a software system in production. In addition, we extensively study how to approximate the energy consumption of software systems under different conditions, e.g., workload. To the best of our knowledge, approximating software energy consumption remains an open challenge in software engineering, as multiple and often conflicting design and implementation decisions influence software energy consumption. Our first attempt to replicate the resource usage of software involves a Layered Queuing Network (LQN), which approximates the CPU usage and energy consumption of embedded software and a microservices-based application [12]. In order to maintain the error between the LQN predictions and the data collected from the real systems under 10%, we assumed a linear relationship between energy consumption and performance. We define the mirror, instead, as a software system rather than a graphical model or a simulator, to control and empirically evaluate the effect of a subset of factors influencing energy consumption, such as CPU frequency scaling.

A mirror system is primarily beneficial for the developers of the software system under study that, as seen in Figure 1, can use the results provided by the researchers to optimize the software under study without interrupting its execution and save experimentation time. Researchers, in turn, can study and validate their findings on a surrogate of a real system that is usually complex and deployed in a production environment. This study would be also valuable for the researchers of the ICT4S community since they can provide suggestions and collaboration for realizing the mirror system, as well as industrial partners, who can provide real-world use cases to validate the approach.
References


