UNIVERSITÄT
LEIPZIG

Green Configuration

Energy Consumption of Configurable Software Systems

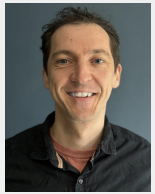
Max Weber

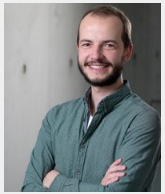
UNIVERSITÄT LEIPZIG

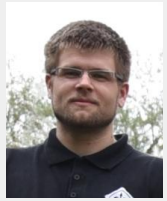
SE SIC Saarland Informatics Campus


Alina Mailach


Johannes Dorn


Norbert Siegmund


Max Weber


Florian Sattler


Sven Apel

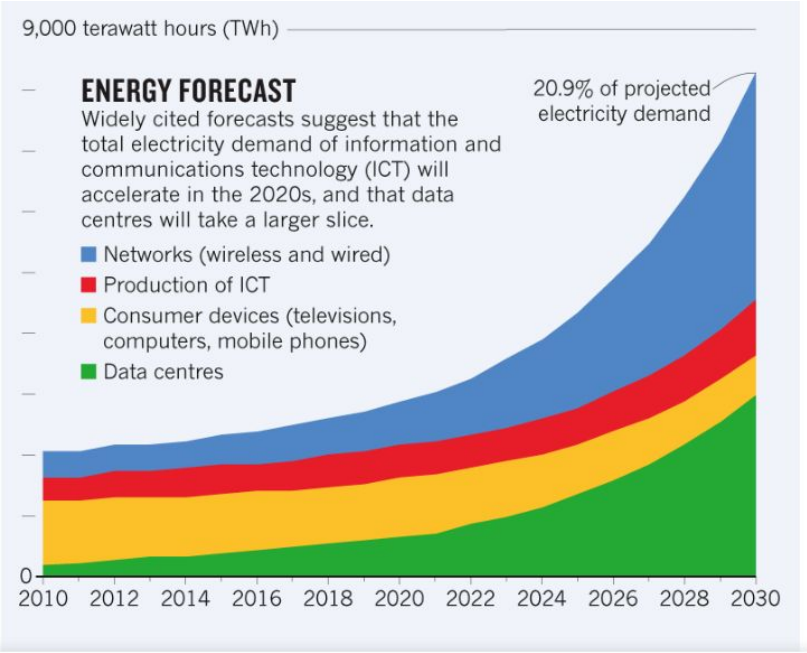

Christian Kaltenecker

DFG Deutsche Forschungsgemeinschaft

Projet goal: Assessing and reducing energy consumption of configurable software systems

The climate is warming

Energy consumption is rising



Energy Consumption at ecoCompute

CarbonDB - Real Time carbon emissions mapping to heterogenous IT infrastructure 🇬🇧

Arne Tarara

Green Coding Solutions

Embodied Carbon

Workloads, Use Cases

Software Systems

OS, Languages

Hardware

Rerouting the Cloud: Cutting 90% CO_2 and Cloud Costs with Smarter Workload Shifting 🇬🇧

Dryden Williams

CarbonRunner

Coding Smarter for a Greener Future | A comparison between Go & Java 🇬🇧

Moritz Bölter

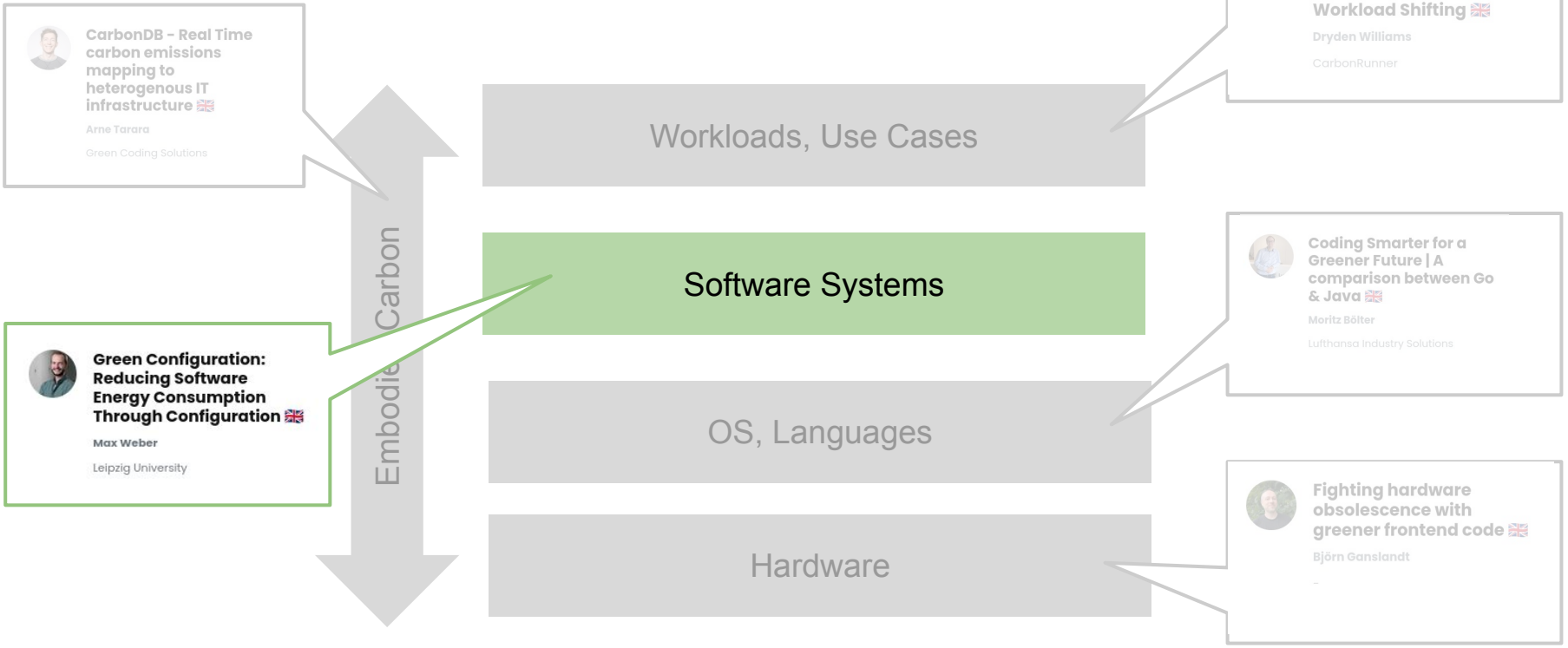
Lufthansa Industry Solutions

Fighting hardware obsolescence with greener frontend code 🇬🇧

Björn Ganslandt

-

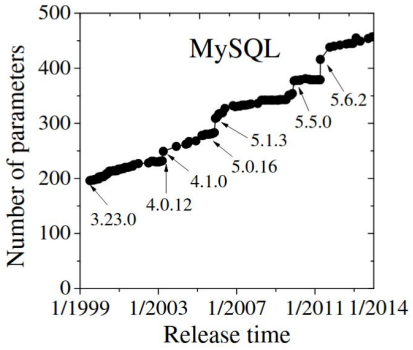
Energy Consumption at ecoCompute

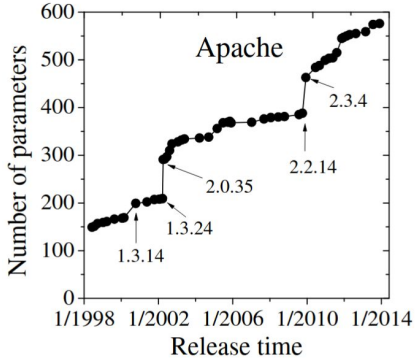


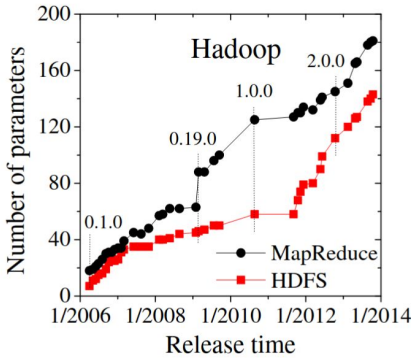
How can we reduce the energy consumption of software systems?

By tailoring software to its workload and usage scenario.

Configuration is everywhere!







Today's available configuration decisions:

MySQL – 700+

Apache – 900+

Hadoop – 400+

Most users stick to default configurations, leaving much potential unused.

1

Assessing Energy Consumption

How can we **measure** software energy consumption?

Energy Consumption of Configurable Software Systems

2

Debugging Energy consumption

3

Practitioners' perspective

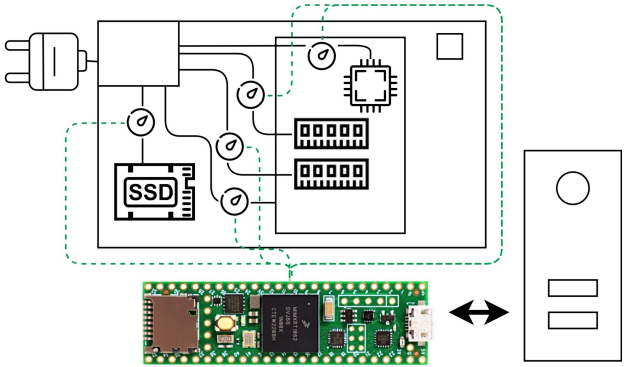
Measuring Energy Consumption at the Code Level

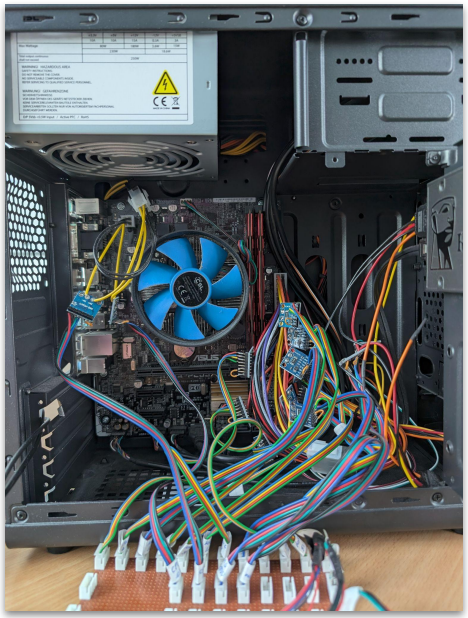
component-wise measurements
(adding up to the whole system)

High sampling rate:
3.5 kHz

no energy measurement bias

Accuracy: < 1.2%
measurement error





Hardware Components

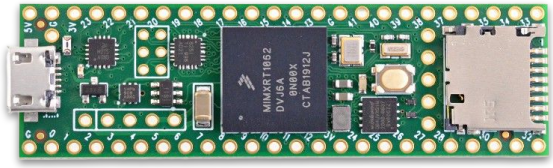
INA 226
Sample Time 280 us
Sample Rate 3.5 kHz

Raspberry Pi 4B

60€

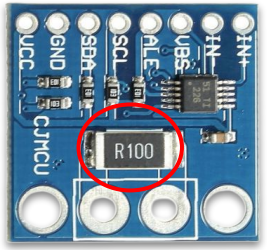
USB

Teensy 4.1



30€

I²C



5€

3,500 measurements per second
≙ 1 sample every 0.3 ms (for all components)

175€

Hardware Components

INA 226
Sample Time 280 us
Sample Rate 3.5 kHz

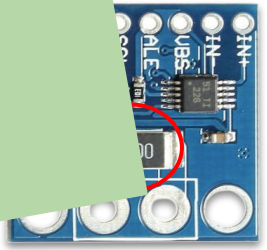
Raspberry Pi 4B

60€

USB

Teensy 4.1

Hours it took until it worked reliably:
Priceless.

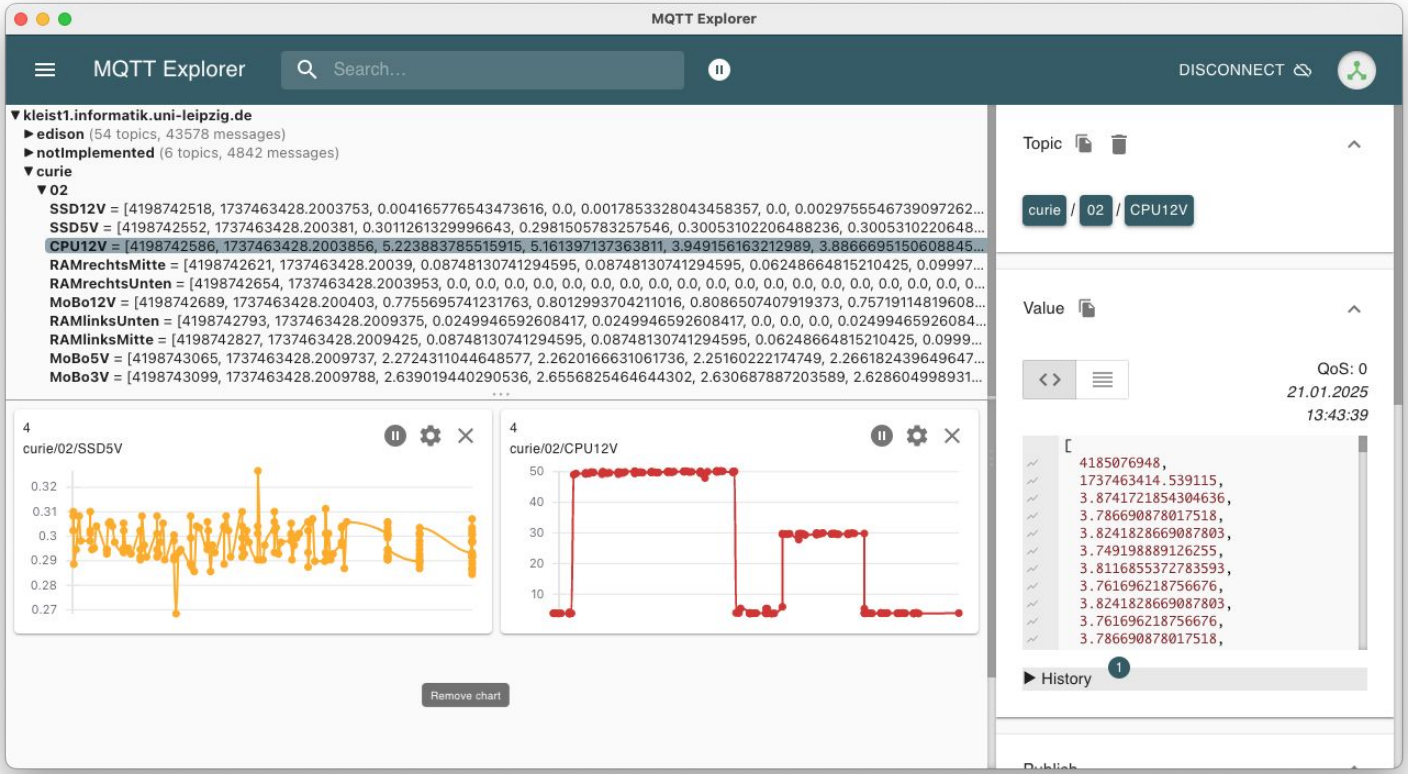


5€

3,500 measurements per second
≙ 1 sample every 0.3 ms (for all components)

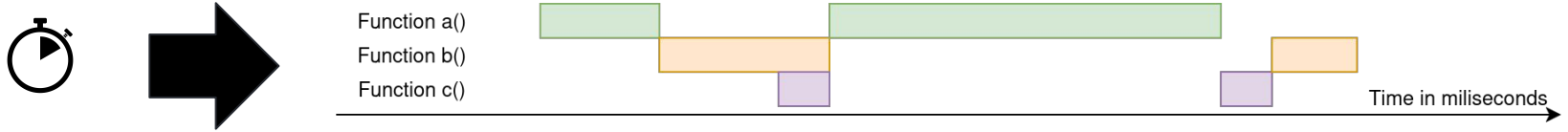
175€

Power Measurement

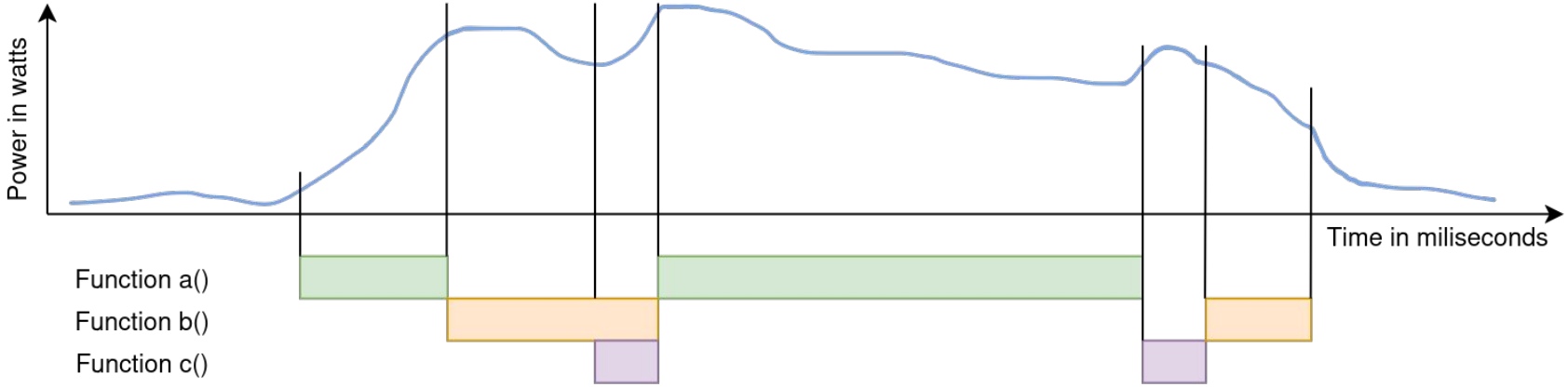


Performance Profiling

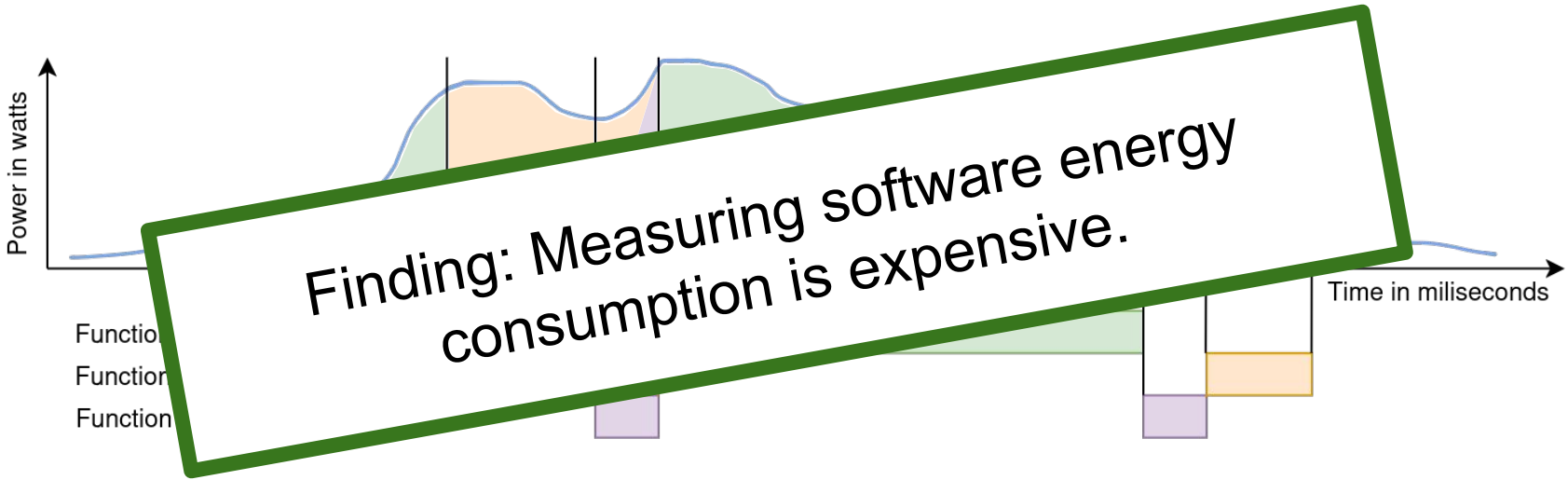
Linux *perf*



Synchronizing Energy and Performance Measurements



Synchronizing Energy and Performance Measurements



1

Assessing Energy Consumption

How can we **measure** software energy consumption?

Can we find a **reliable proxy**?

Energy Consumption of Configurable Software Systems

2

Debugging Energy consumption

3

Practitioners' perspective

Energy Metering is expensive, time consuming and tool intensive.

Infeasible in many projects

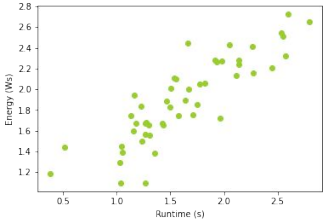
Solution: Using performance as a proxy for energy consumption

System-Level Correlation

Option-Level Correlation

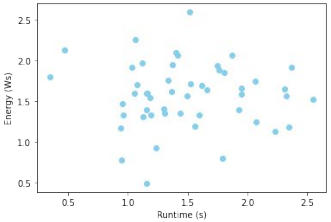
Literature Study Statements

positive correlation



"**perform significantly better** and **consume less energy** with only a small loss in QoS" [1]

no / absent correlation

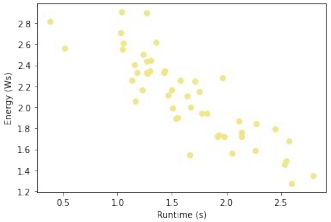


"the involved physical components **may not consume energy proportional to time**" [2]

"**caching** has an effect **on performance but not on energy consumption**" [2]

"no correlation if some **communication** comes into play" [4]

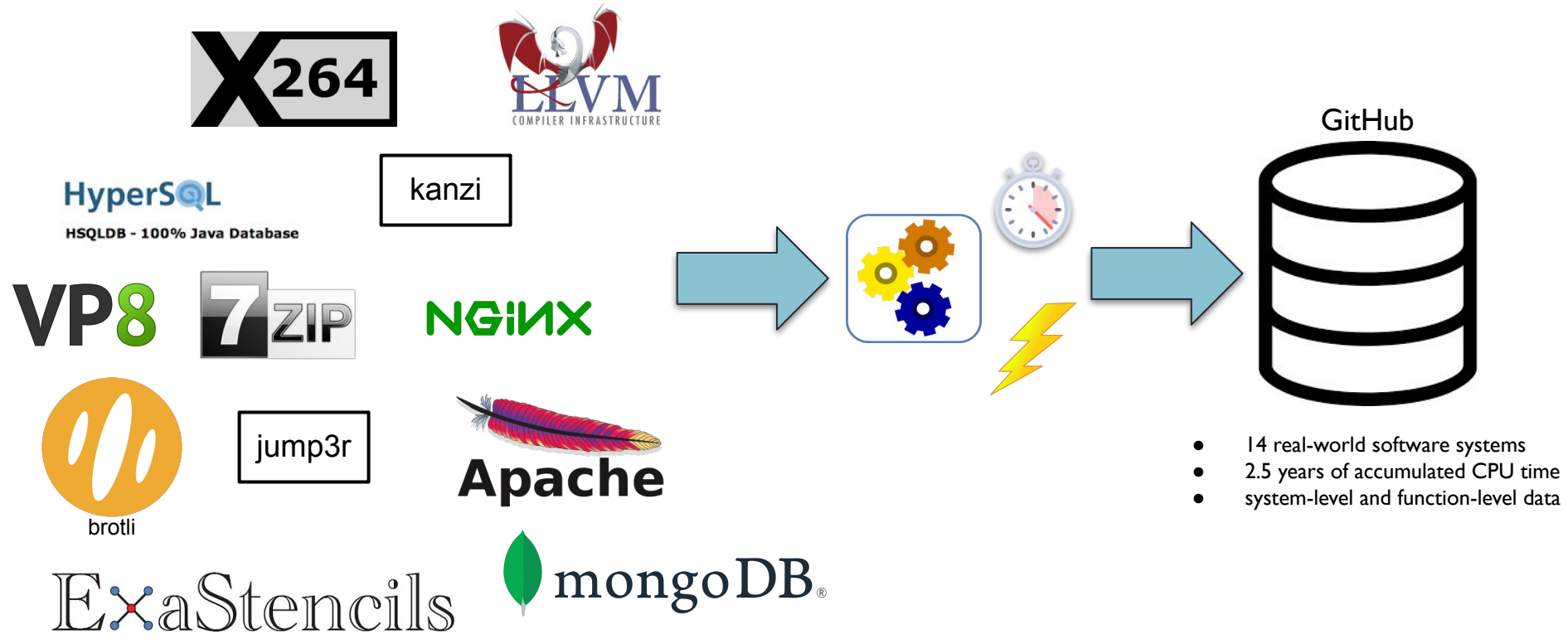
negative correlation



"computational offloading brings **better performance but not always better energy efficiency**" [3]

[1] Green: a framework for supporting energy-conscious programming using controlled approximation, 2020
[2] Evaluating the Impact of Caching on the Energy Consumption and Performance of Progressive Web Apps, 2020
[3] Analysis of Performance and Energy Consumption of Wearable Devices and Mobile Gateways in IoT Applications, 2019
[4] Quantifying energy use in dense shared memory HPC node, 2016

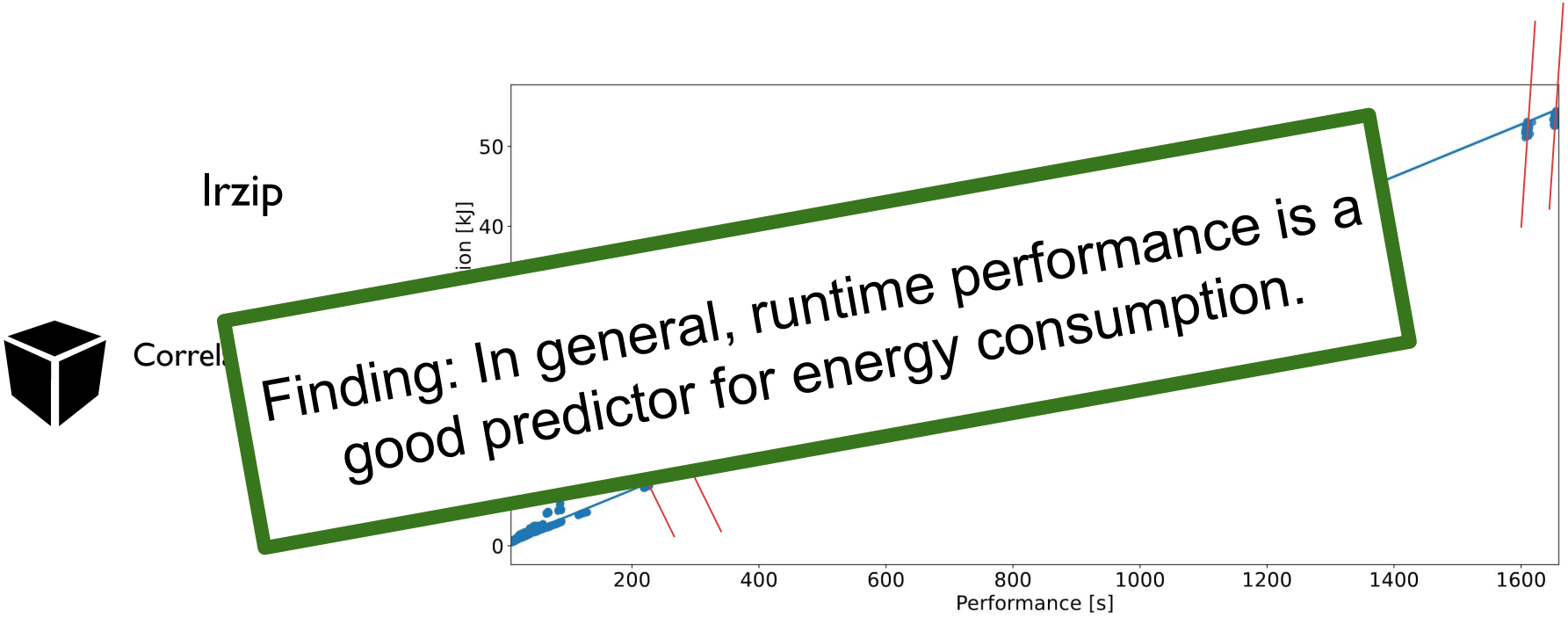
Empirical Study to Investigate Performance–Energy Correlation



- 14 real-world software systems
- 2.5 years of accumulated CPU time
- system-level and function-level data

System-Level Correlation

lrzip

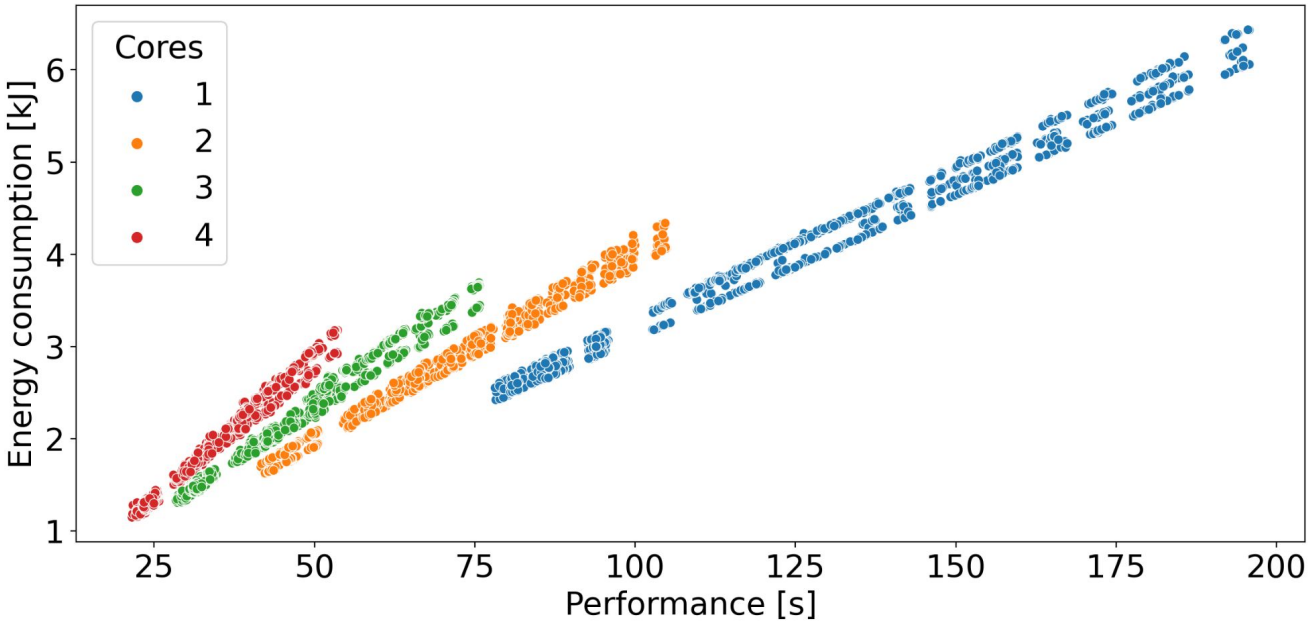


Finding: In general, runtime performance is a good predictor for energy consumption.

Option-Level Correlation

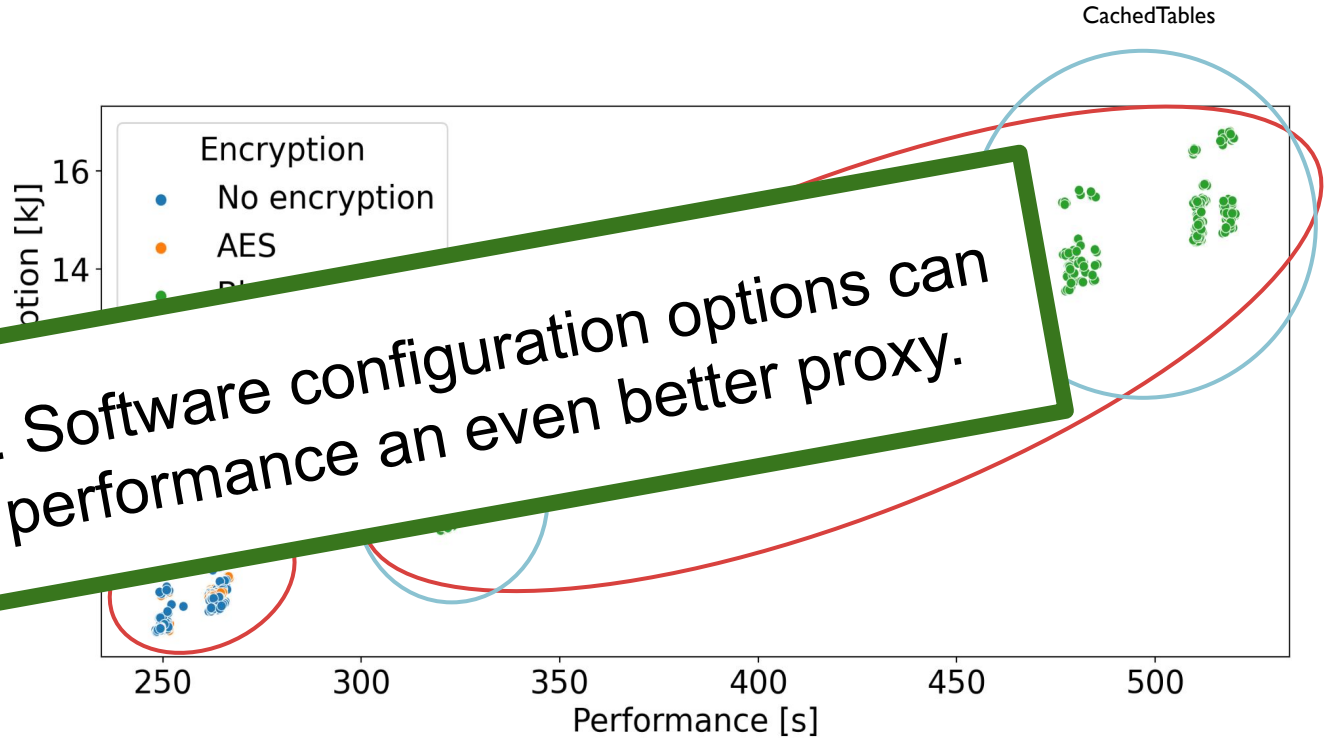
Correlation: 0.99
MAPE: 10.3%

Correlation: 0.99
MAPE: 2.5%



Option-Level Correlation

HyperSQL
HSQLDB - 100% Java Database



Finding: Software configuration options can make performance an even better proxy.

Correl
MAPE:

Blowfish & Memory Tables

Correlation: 0.13
MAPE: 213.5%

1

Assessing Energy Consumption

How can we **measure** software energy consumption?

Can we find a **reliable proxy**?

How can we get **code-level insights**?

Energy Consumption of Configurable Software Systems

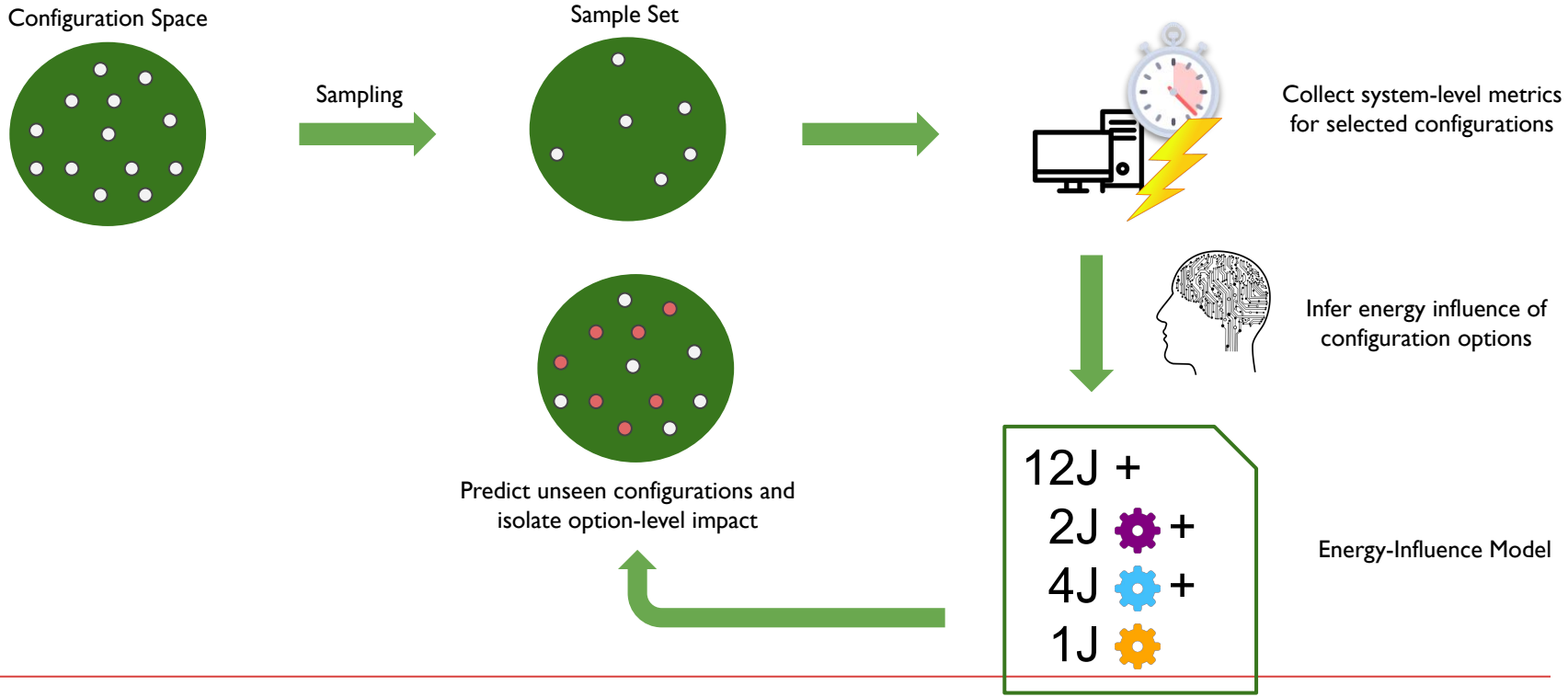
2

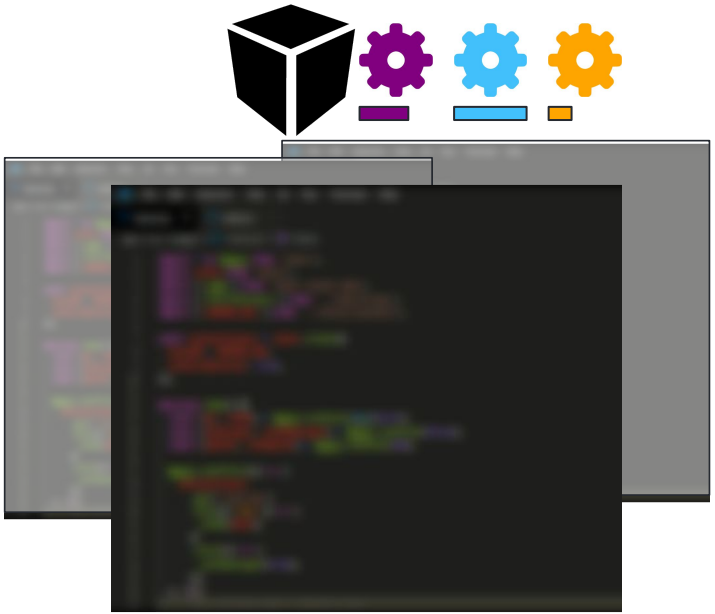
Debugging Energy consumption

3

Practitioners' perspective

Building Energy-Influence Models for Configurable Systems





Black-Box Models

As a **user**, I can optimize configurations to reduce energy consumption 💪

… but I can't see **where** in the code the energy goes.

As a **developer**, how do I know which part of my code I need to change?

How can we **efficiently** pin point energy hotspots in the code base with minimal measurement effort?

Black-Box Models

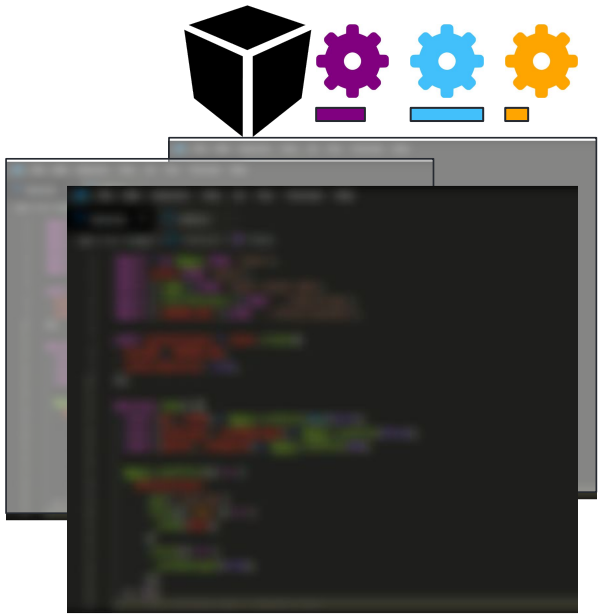
As a **user**, I can optimize configurations to reduce energy consumption 💪

… but I can't see **where** in the code the energy goes.

Established White-Box Models

We know exactly **which** code statements consume energy.

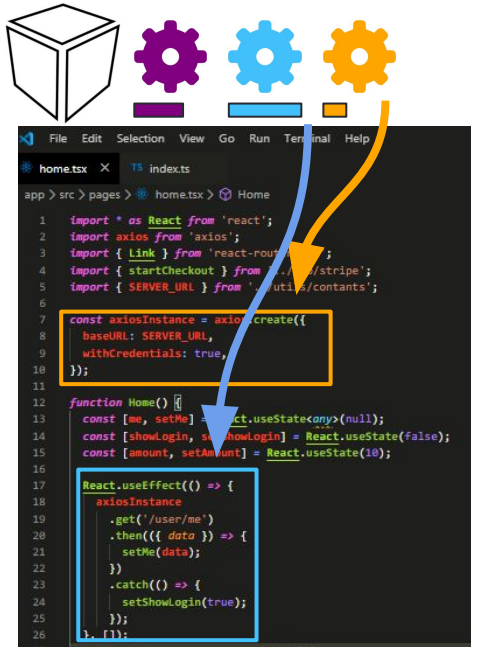
…but measuring this across a full system is often **infeasible** 😔



Black-Box Models

As a **user**, I can optimize configurations to reduce energy consumption 💪

… but I can't see **where** in the code the energy goes. 😔



Method-Level White-Box Models

As a user, I still benefit from efficient configurations that save energy ⚡.

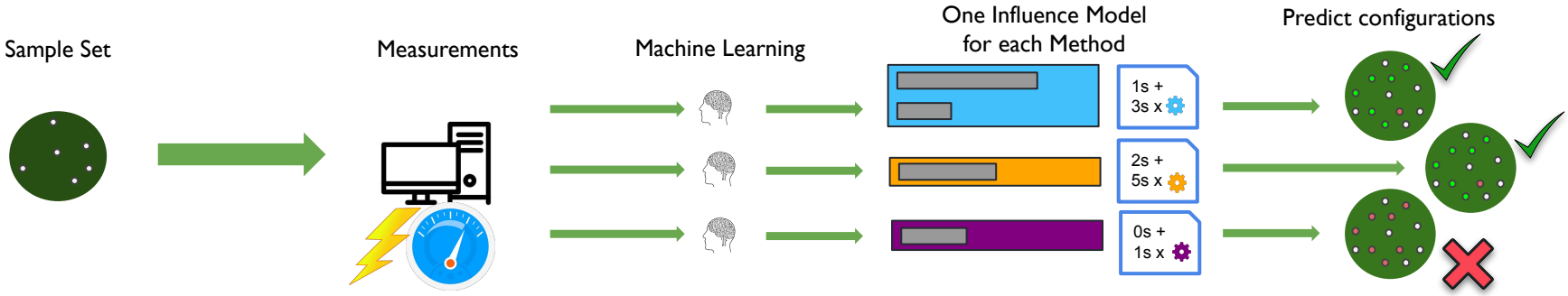
As a developer, I can identify which methods consume energy without measuring every single line of code.

Established White-Box Models

We know exactly **which** code statements consume energy. 💪

…but measuring this across a full system is often **infeasible.** 😔

White-Box Energy Influence Models via Profiling

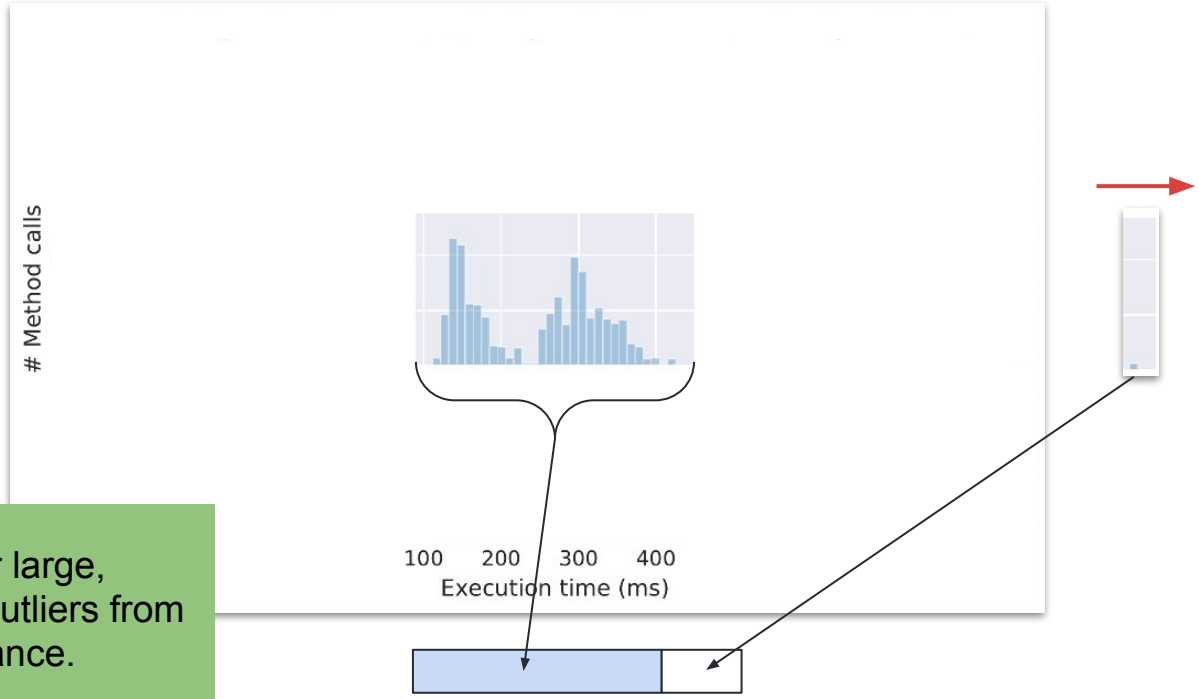


Challenge: Handling Variance in Measurement Data

Context variance

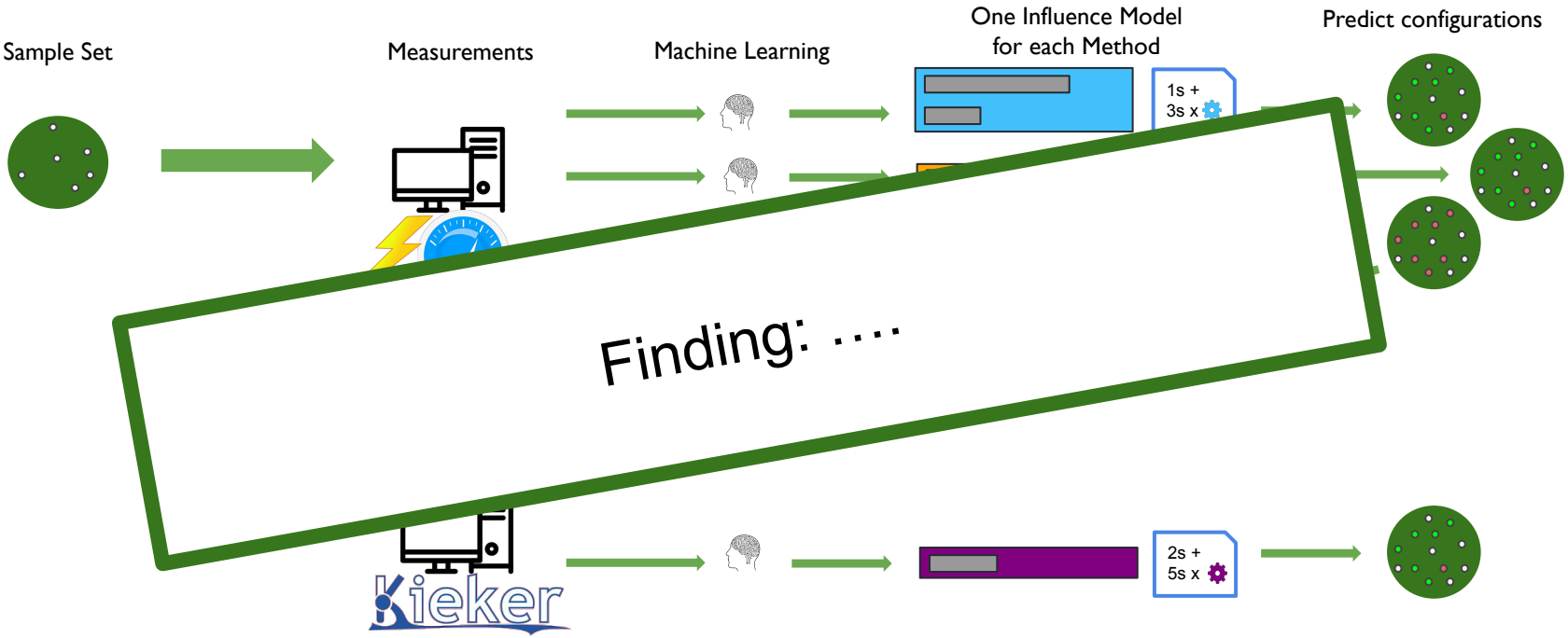
Configuration variance

Measurement variance

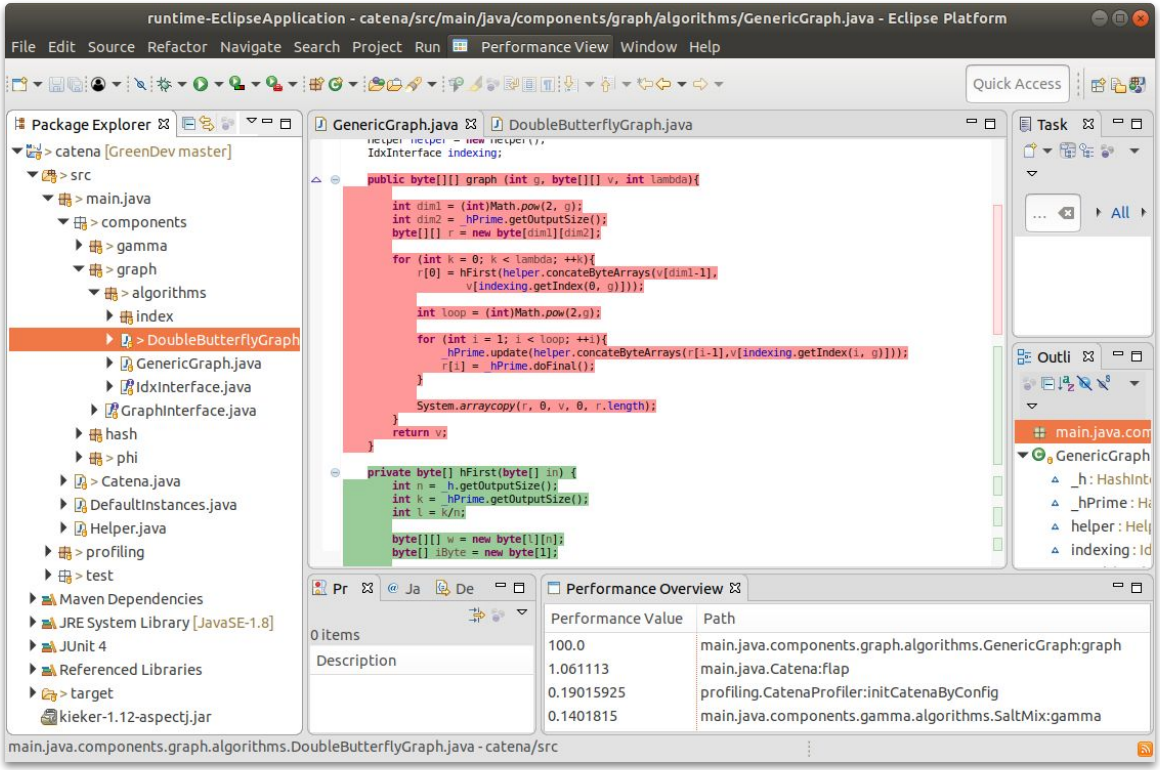


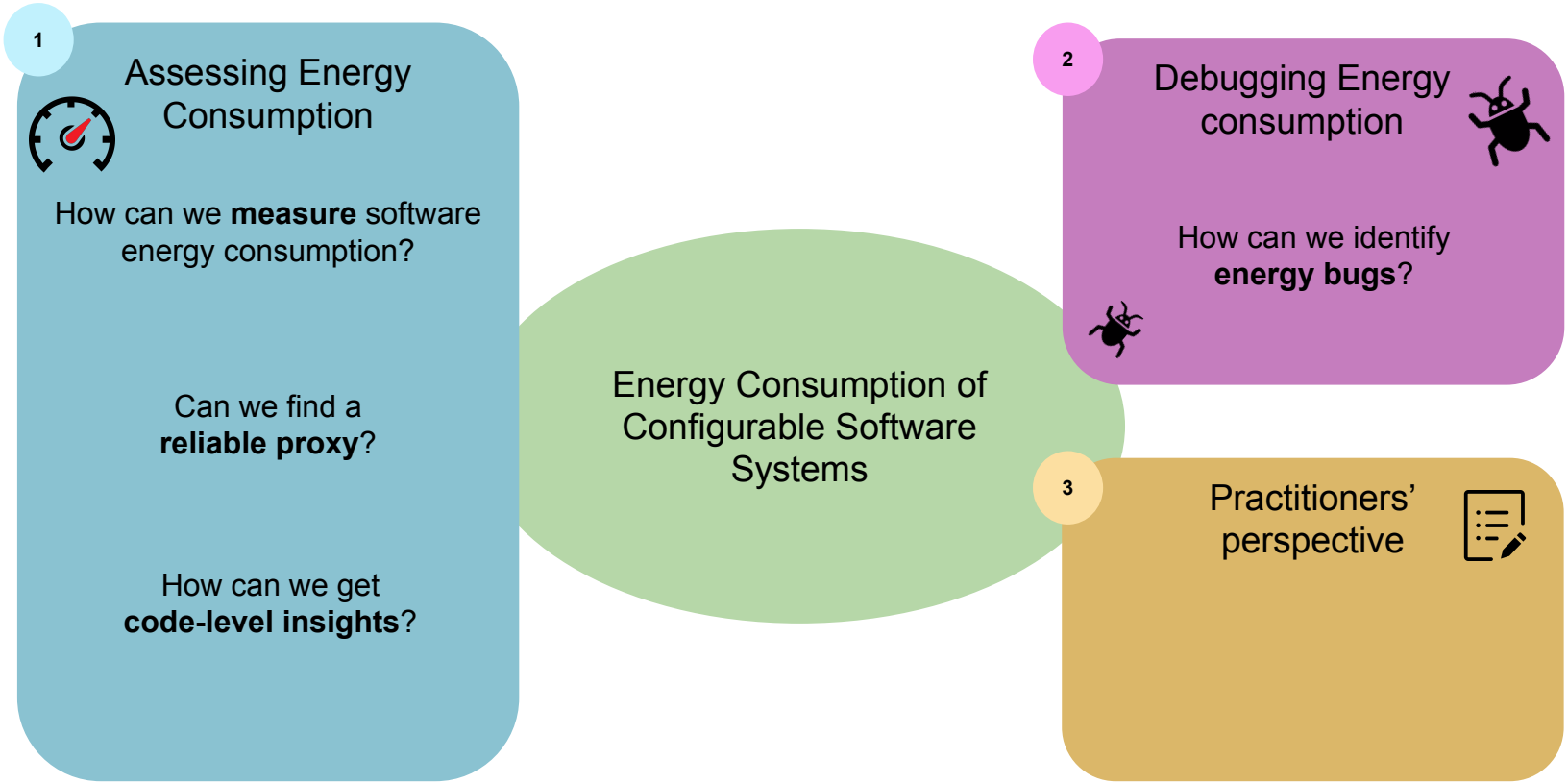
Solution: Filter large, non-deterministic outliers from context variance.

White-Box Energy Influence Models via Profiling



IDE Integration

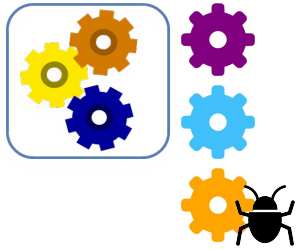



1
Assessing Energy Consumption
How can we **measure** software energy consumption?
Can we find a **reliable proxy**?
How can we get **code-level insights**?
Energy Consumption of Configurable Software Systems
2
Debugging Energy consumption
How can we identify **energy bugs**?
3
Practitioners' perspective

Performance Bugs in Configurable Systems

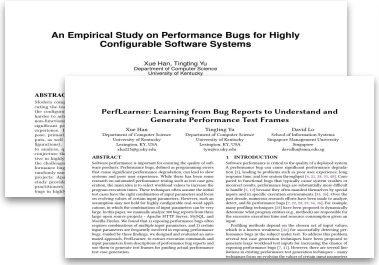
Functional bug

Configuration-dependent bugs



Performance bugs

Requires fixing code



An Empirical Study on Performance Bugs for Highly Configurable Software Systems

Xue Han, Tingting Yu
Department of Computer Science
University of Kentucky

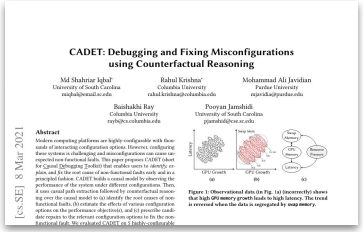
PerfLearner: Learning from Bug Reports to Understand and Generate Performance Test Frames

Xue Han
Department of Computer Science
University of Kentucky
Lexington, KY, USA

Tingting Yu
Department of Computer Science
University of Kentucky
Lexington, KY, USA

David Lo
School of Information Systems
Singapore Management University
Singapore

Fixes are time consuming



CADET: Debugging and Fixing Misconfigurations using Counterfactual Reasoning

Md Shahriar Iqbal
University of South Carolina

Rahul Krishna
Columbia University

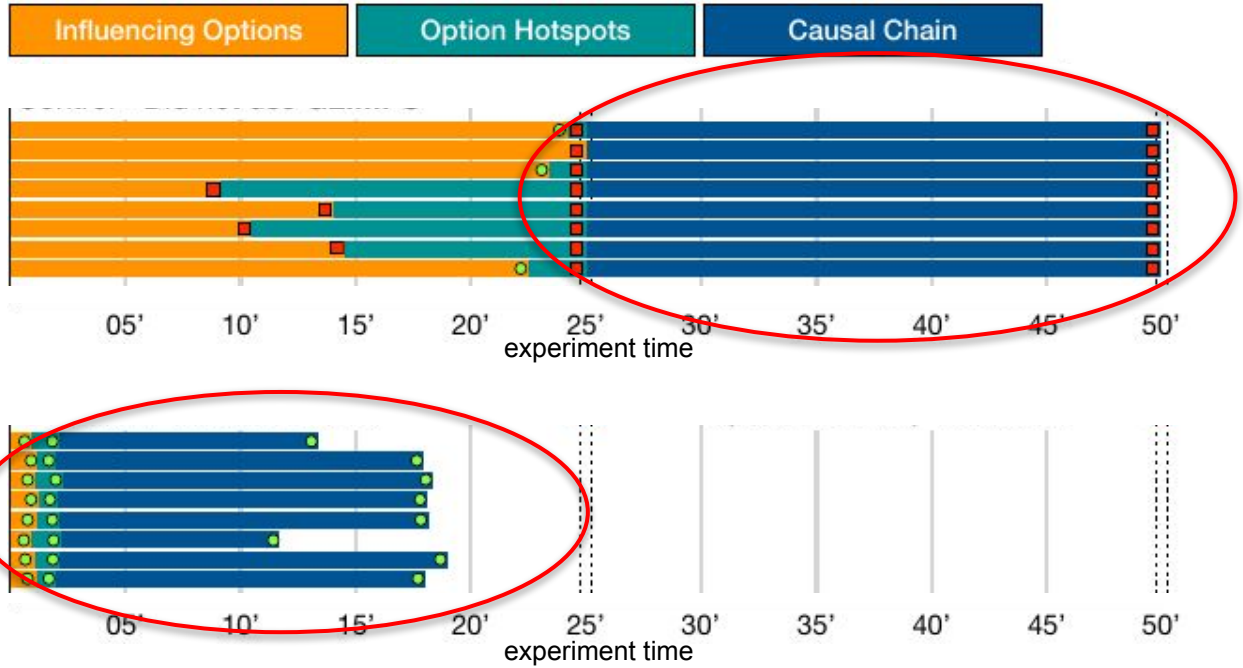
Mohammad Ali Javidian
Purdue University

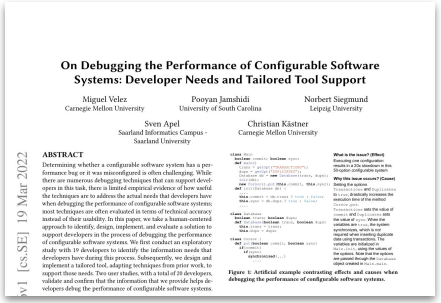
Baishakhi Ray
Columbia University

Pooyan Jamshidi
University of South Carolina

Abstract

Why Debugging Performance Bugs Takes So Much Time




On Debugging the Performance of Configurable Software Systems: Developer Needs and Tailored Tool Support

Miguel Velez, Carnegie Mellon University
Pooyan Jamshidi, University of South Carolina
Norbert Siegmund, Leipzig University
Sven Apel, Saarland Informatics Campus - Saarland University
Christian Kästner, Carnegie Mellon University

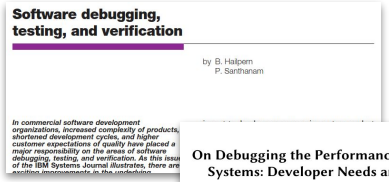
ABSTRACT

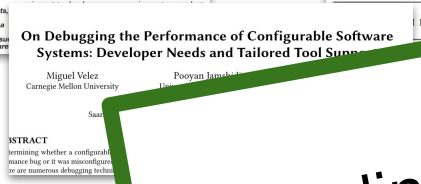
Determining whether a configurable software system has a performance bug or it was misconfigured is often challenging. While there are numerous debugging techniques that can support developers in this task, there is limited empirical evidence of how useful the techniques are to address the actual needs that developers have when debugging the performance of configurable software systems; most techniques are often evaluated in terms of technical accuracy instead of their usability. In this paper, we take a human-centered approach to identify, design, implement, and evaluate a solution to support developers in the process of debugging the performance of configurable software systems. We first conduct an exploratory study with 10 developers to identify the information needs that developers have during this process. Subsequently, we design and implement a tailored tool, adapting techniques from prior work, to support those needs. Two user studies, with a total of 20 developers, validate and confirm that the information that we provide helps developers debug the performance of configurable software systems.

Figure 1: Artificial example contrasting effects and causes when debugging the performance of configurable software systems.

[Miguel Velez, Pooyan Jamshidi, Norbert Siegmund, Sven Apel, and Christian Kästner. On debugging the performance of configurable software systems: Developer needs and tailored tool support]

Debugging Strategies in the Literature





overview of debugging strategies

Title	Description	Actions	References
Program comprehension	Developers gain comprehensive and high-level knowledge of the system by browsing		[Murphy et al. 2008; Romero et al. 2007; Velez et al. 2022; Vessey 1985]
Inspect source code			[Böhme et al. 2017; Hirsch and Hofer 2021; Murphy et al. 2008; Peng et al. 2016; Subrahmaniyan et al. 2008]
			[Subrahmaniyan et al. 2008]
			[Eisenstadt 1997; Romero et al. 2007]
			[Eisenstadt 1997; Gould 1975; Perscheid et al. 2017; Romero et al. 2007]
			[Böhme et al. 2017; Murphy et al. 2008]
	understand how transforms within the system, thereby identifying potential issues.	follow data flow	[Murphy et al. 2008; Peng et al. 2016; Romero et al. 2007; Subrahmaniyan et al. 2008]
Follow control flow	Tracking the program's execution path to understand the sequence of operations and identify deviations from expected behavior.	follow control flow	[Murphy et al. 2008; Romero et al. 2007]

Finding: Literature is not consolidated!

- ❖ reviewed re… strategies
- ❖ forward and … ward snowballing from seven seed publications
- ❖ → 74 publications, 12 different debugging strategies

- ❖ different levels of abstraction
- ❖ contradictory definitions
- ❖ strategies that often apply only in specific study setups
- ❖ strategies that resemble other strategies, which makes them hard to distinguish

Which debugging strategies are helpful, which hinder debugging performance bugs?

User Study

12 professional software developers
debugging a configuration-dependent performance bug

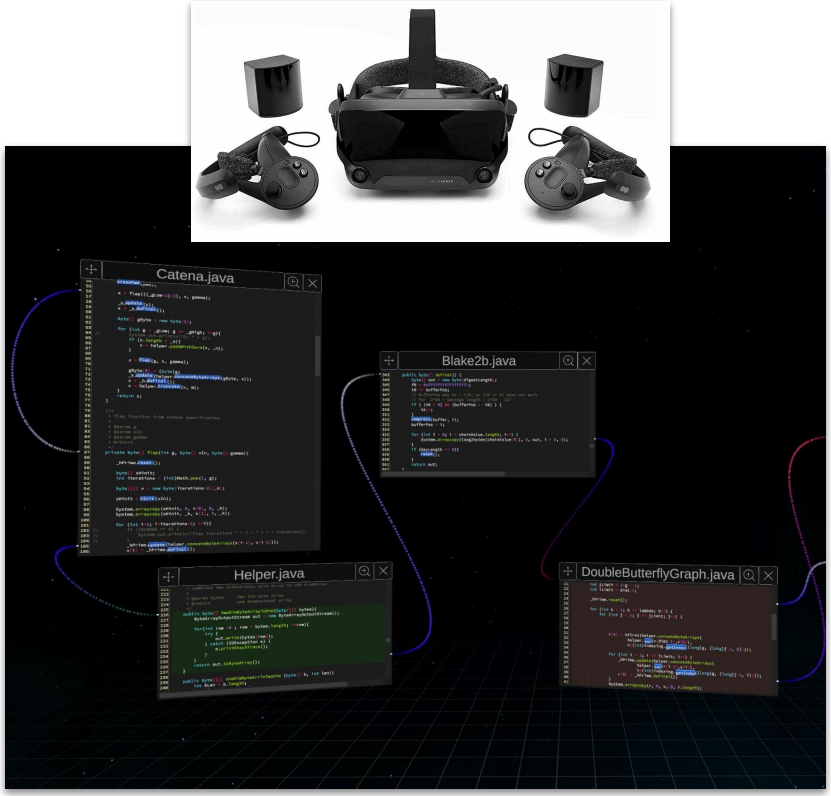
Density Converter
(multi platform image density conversion)

```
public final class Convert {
      public static void main(String[] rawArgs) {
        float fraction = 0.5f;
```

```
private static <T extends DensityDescriptor> Map<T, Dimension> getDensityBucketsWithFractionScale
          java.util.List<T> densities, Dimension srcDimension, Arguments args, float fraction) {
        double baseWidth = (double) srcDimension.width / fraction;
        double baseHeight = (double) srcDimension.height / fraction;
```

```
private void verticalFromWorkToDst(byte[][] workPixels, byte[] outPixels, int start, int delta) {
    // ...
    for (int x = start; x < dstWidth; x += delta) {
      final int xLocation = x * nrChannels;
      for (int y = dstHeight - 1; y >= 0; y--) {
        final int max = verticalSubsamplingData.arrN[y];
        for (int j = max - 1; j >= 0; j--) {
          int valueLocation = verticalSubsamplingData.arrPixel[index];
          float arrWeight = verticalSubsamplingData.arrWeight[index];
          // ...
```

SoftVR, Immersive Virtual Reality Debugging Tool



- position unlimited code windows with profiling information
- visualize control flow
- users can move through code-window setup

VR as novel study method:

- directly observe which code sections developers focus on
- analyze code-window patterns developers create

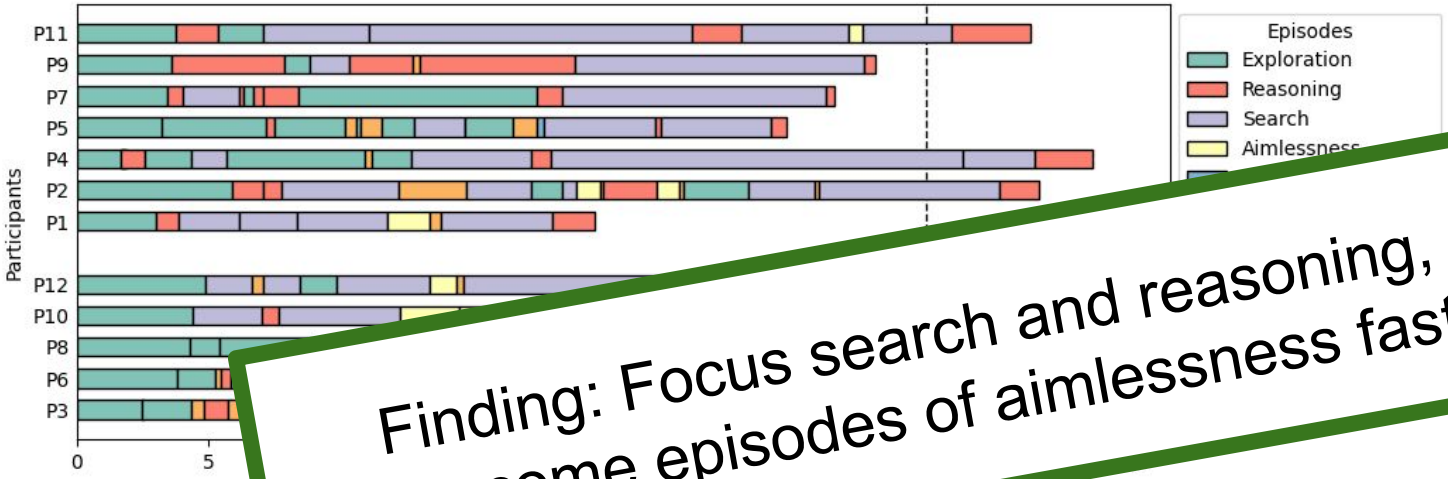
General Results

SoftVR is usable for performance bug detection (confirmed by SUS)

All developers reached the location of the bug in the code

7 developers successfully identified and reasoned about the bug
5 developers overlooked the bug or drew incorrect conclusions

Debugging as Episodes



Reasoning appears substantially more often in successful debugging.

...essness seem ...a possible ...tor for failure.

Finding: Focus search and reasoning, overcome episodes of aimlessness fast.

P11, reasoning

"I need to understand what is going on in order to be able to formulate hypotheses".

P4, (problem-specific) exploration:

"Usually I try to get a fairly complete picture of something first."

P9, aimlessness:

"[...] there is no point in staying in such a frustrating moment."

1

Assessing Energy Consumption

How can we **measure** software energy consumption?

Can we find a **reliable proxy**?

How can we get **code-level insights**?

Energy Consumption of Configurable Software Systems

2

Debugging Energy consumption

How can we identify **energy bugs**?

3

Practitioners' perspective

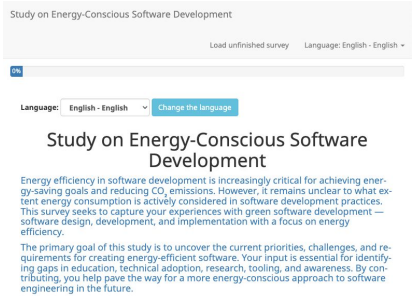
What is the **current state** of green software engineering?

Optimizing Software Energy Consumption in Practice

What is the **current state** of software energy consumption in practice?

How do practitioners **relate software energy consumption to other metrics** such as runtime performance?

134 developer

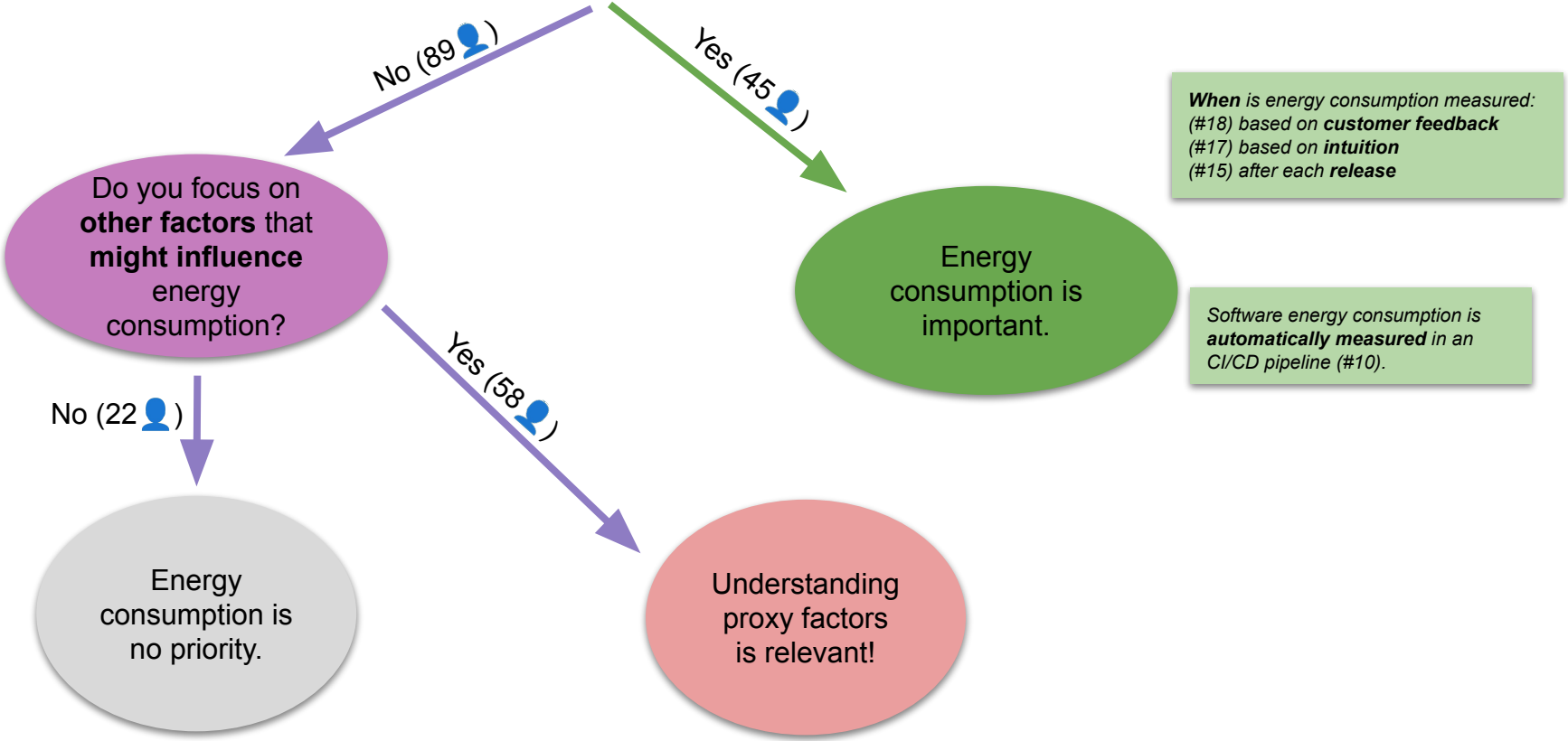


online survey

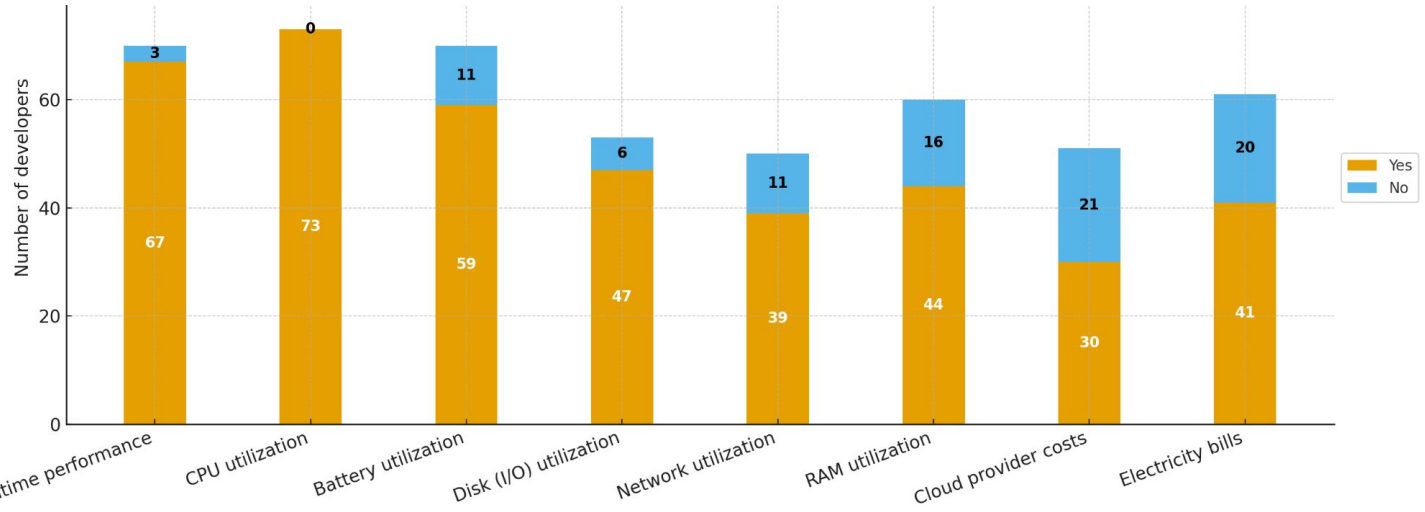
> 480 open text answers

>16 hours manual analysis

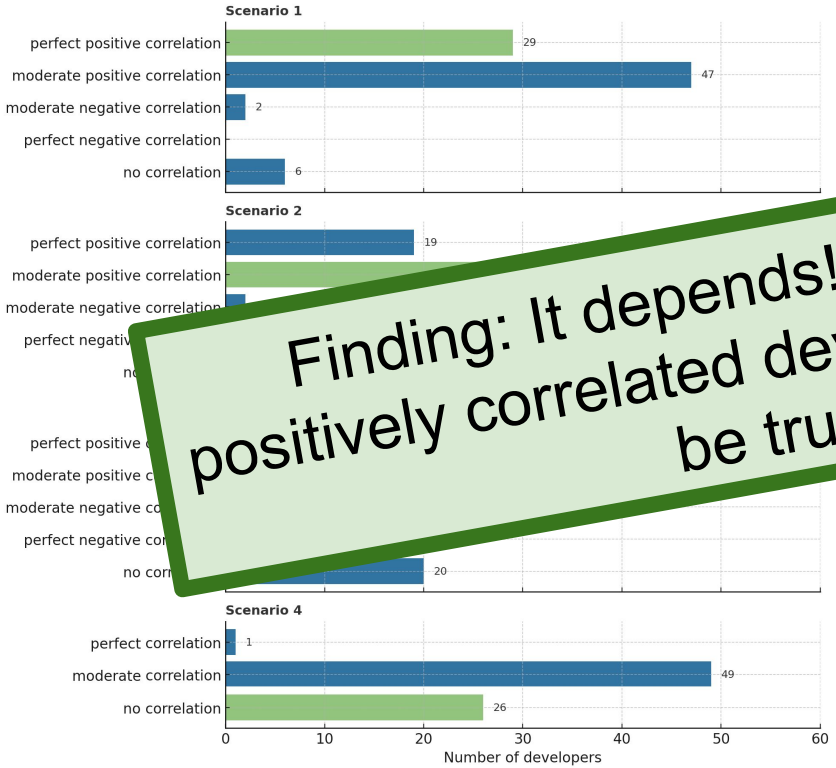
Is energy consumption relevant in participants' projects?



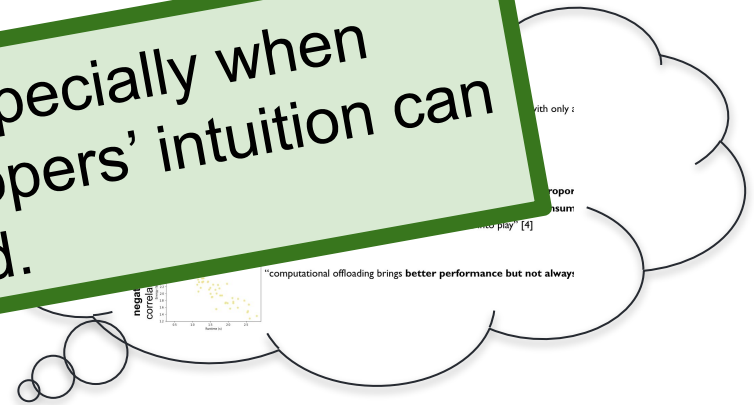
What metrics are considered good proxies for energy?



Do Developers Intuitively Know when Performance is a good Proxy for Energy?



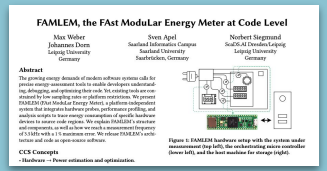
Finding: It depends! Especially when positively correlated developers' intuition can be trusted.



How to Support Practitioners

1

Assessing Energy Consumption



FAMLEM, the FAst ModuLar Energy Meter at Code Level

Max Weber, Johannes Dorn, Leipzig University, Germany
Sven Apel, Saarland Informatics Campus, Saarland University, Saarbrücken, Germany
Norbert Siegmund, ScaDS.AI Dresden/Leipzig, Leipzig University, Germany

Abstract

The growing energy demands of modern software systems calls for precise energy-assessment tools to enable developers understanding, debugging, and optimizing their code. Yet, existing tools are constrained by low sampling rates or platform restrictions. We present FAMLEM (FAst ModuLar Energy Meter), a platform-independent system that integrates hardware probes, performance profiling, and analysis scripts to trace energy consumption of specific hardware devices to source code regions. We explain FAMLEM's structure and components, as well as how we reach a measurement frequency of 3.5 kHz with a 1 % maximum error. We release FAMLEM's architecture and code as open-source software.

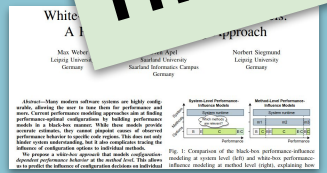
CCS Concepts

• Hardware → Power estimation and optimization.

Figure 1: FAMLEM hardware setup with the system under measurement (top left), the orchestrating micro controller (lower left), and the host machine for storage (right).

Twins or False Friends? A Study on Energy Consumption and Performance of Configurable Software

Max Weber*, Christian Kaltenecker†, Florian Sattler†, Sven Apel†, Norbert Siegmund*
* Leipzig University, Germany
† Saarland University, Saarland Informatics Campus, Germany

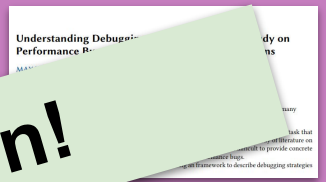


Norbert Siegmund
Leipzig University
Germany

Fig. 1: Comparison of the black-box performance-influence modeling at system level (left) and white-box performance-influence modeling at method level (right), explaining how

2

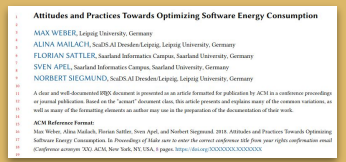
Debugging Energy consumption



Understanding Debuggi… Performance …dy on …ns

3

Practitioners' perspective



Attitudes and Practices Towards Optimizing Software Energy Consumption

MAX WEBER, Leipzig University, Germany
ALINA MAILACH, ScaDS.AI Dresden/Leipzig, Leipzig University, Germany
FLORIAN SATTLER, Saarland Informatics Campus, Saarland University, Germany
SVEN APEL, Saarland Informatics Campus, Saarland University, Germany
NORBERT SIEGMUND, ScaDS.AI Dresden/Leipzig, Leipzig University, Germany

A clear and well-documented LaTeX document is presented as an article formatted for publication by ACM in a conference proceedings or journal publication. Based on the "acmart" document class, this article presents and explains many of the common variations, as well as many of the formatting elements an author may use in the preparation of the documentation of their work.

ACM Reference Format:
Max Weber, Alina Mailach, Florian Sattler, Sven Apel, and Norbert Siegmund. 2018. Attitudes and Practices Towards Optimizing Software Energy Consumption. In *Proceedings of Make sure to enter the correct conference title from your rights confirmation email (Conference acronym 'XX)*. ACM, New York, NY, USA, 6 pages. https://doi.org/XXXXXXX.XXXXXXX

Thank you for your attention!

UNIVERSITÄT
LEIPZIG

Feel free to contact me!

max.weber@cs.uni-leipzig.de

https://github.com/AI-4-SE

linkedin.com/in/max-weber-84a83415b