

Automated measurement and green optimizations for software

SCI, Blue Angel for Software, CI / CD

 GREEN CODING;

Who are we

Your workshop facilitators

- Arne - Green Coding Solutions
- Didi - Green Coding Solutions
- Michelle - Green Coding Solutions
- Aydin - bluehands



**GREEN
CODING;**



</> GREEN CODING;

Our entry question

What is the goal of this workshop?

We want to make software sustainable

Through technical means - Measurement, Automation, Optimization

- In this workshop we want to
 - share successful **experiences and techniques**
 - share current **struggles and hurdles**
 - brainstorm new **ideas and approaches**
- To get the workshop started we will present
 - Our approach for an **open source automated measurement system**
 - Current possible routes for **certificates / standards**

We want to make software sustainable

Our definition of a sustainable software and our approach

- A sustainable software knows about it's **energy** and **CO2 emission**
- A sustainable software is constantly **monitored** and **optimized** for savings
- A sustainable software uses the **least amount of resources** for a given task (compared with similar applications)
- A sustainable software is lean and does **combat software bloat**
- => Be sure to bring your view of a "sustainable software" in later!

Our follow-up question

What technical parts do we need to get to a sustainable software?

Let's start with some requirements

What are our goals and what do we need to achieve them

- **Measurement:** We want to look at software and determine it's energy and CO₂ consumption
- **Reproducibility:** We want to quantify changes and thus runs must be reproducible and have low StdDev.
- **Optimization:** We want to make changes and compare their effect
- **DevOps / CI/CD integration:** We want to look at changes over time as incremental changes might be negligible
- **Standards:** We need standards that the measurements output to execute on all of the above - Actually ... this is the first!

Stage #1

Standards

Standards - Part #1: SCI

Green Software Foundation's SCI

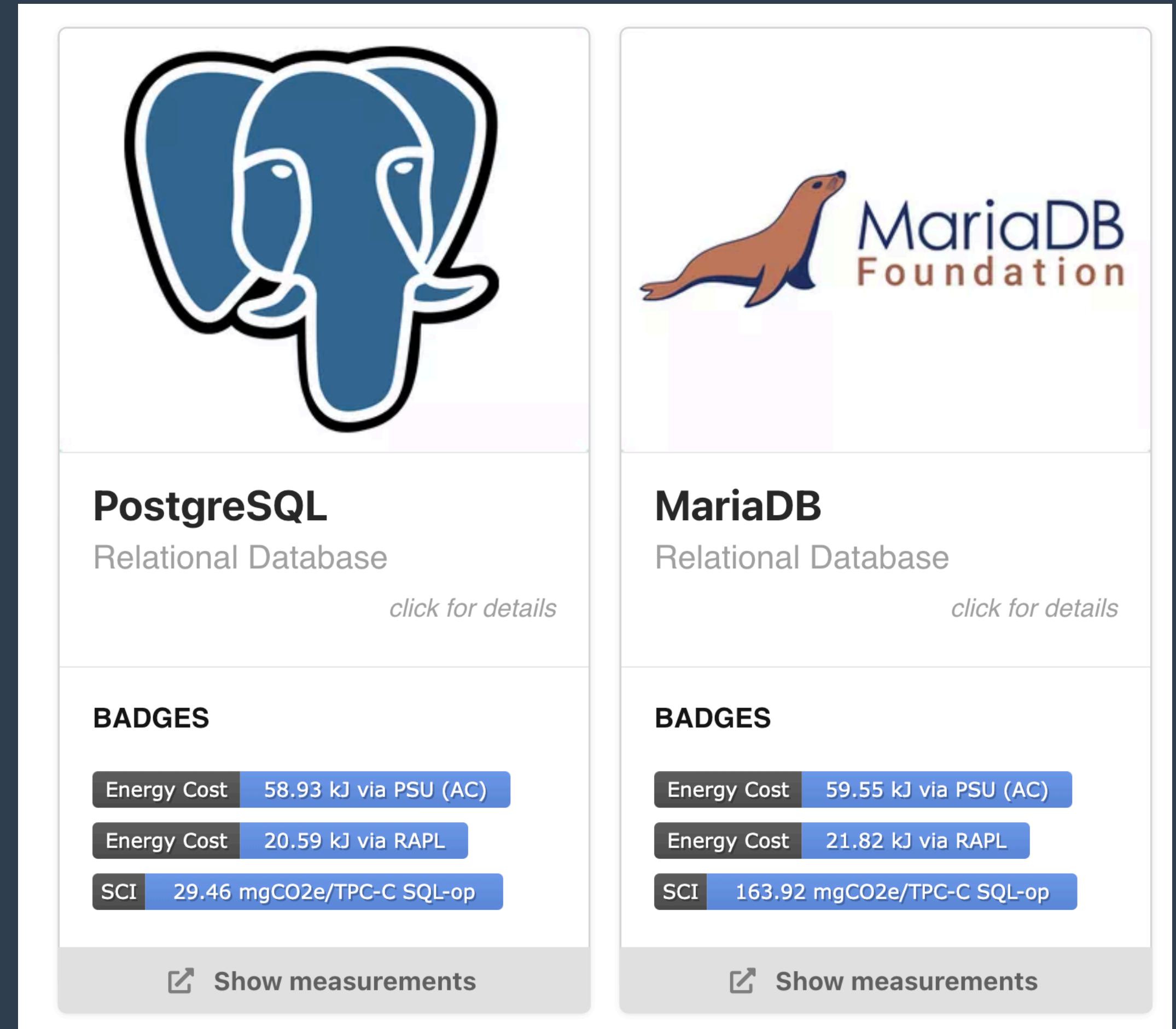
$$\text{SCI} = (\text{E} * \text{I}) + \text{M per R}$$

- (E) - Energy consumption (kilowatt hours) for different components:
 - Ex. CPU/GPUs, Data storage, Memory, Network
- (I) - Emissions factors
- (M) - Embodied emissions
 - Ex. data for servers, mobile devices and laptops
- (R) - Unit of work / use case

Standards - Part #1: SCI

Green Software Foundation's SCI

- By using standardised or at least identical Benchmarks software becomes comparable through the SCI
- What we want is typically multiple SCIs for one software
- Example **Grafana**
 - Case #1: Importing data
 - Case #2: Displaying 5 graphs
 - Case #3: Making aggregation



Standards - Part #2: Blue Angel

Blue Angel for Software by Umweltbundesamt

$$SCI = (E * I) + M \text{ per } R$$

- (E) - Energy consumption (kilowatt hours) for different components:
 - Ex. CPU/GPUs, Data storage, Memory, Network
- (I) - Emissions factors
- (M) - Embodied emissions
 - Ex. data for servers, mobile devices and laptops
- (R) - Unit of work / use case

Standards - Part #2: Blue Angel for Software

Ressource- and energy-efficient software products (DE-UZ 215)

- Goals:
 - Reduce hardware obsolescence
 - Increase transparency about software energy consumption
 - Increase user autonomy
 - Increase modularity and software re-use



Good for me.
Good for the environment.

Umwelt
Bundesamt

Standards - Part #2: Blue Angel for Software

Ressource- and energy-efficient software products (DE-UZ 215)

- Requirements:
 - Specification of intended use-case(s)
 - Measurement incl. utilization, energy etc.
- Formal declaration of no advertisements, security updates, etc.
- Must run on older hardware
 - Reference system or
 - OS Version compatibility



Good for me.
Good for the environment.

Umwelt
Bundesamt

Standards - Part #3: ISO 14001 / GHG Protocol

Environmental management

- ISO 14000 family is focused on life cycle assessment
- ISO 14067 provides framework of looking into software as phases
- GHG protocol provides the execution standard
- Result is a life-cycle-assessment for a digital product.
Methodology has international accreditation
- => Talk also to our speaker Moritz Bölter for experiences!



Stage #2

Infrastructure reproducibility

Reproducability - Orchestration

Using infrastructure files and git versioning

- Container Files
 - Docker Files
 - Kubernetes Files
 - Impact Framework
- Versioning of all resources and files integrated with Github repository

```
compose.yml
.
.
.
services:
  gcb-wordpress-apache:
    build:
      context: ..
      dockerfile: Dockerfile-wordpress
      container_name: gcb-wordpress-apache
      image: gcb_wordpress_apache
    ports:
      - 9875:9875
    restart: always
    environment:
      WORDPRESS_DB_HOST=gcb-wordpress-mariadb
    depends_on:
      - gcb-wordpress-mariadb
    .
    .
    .

```

Reproducability

Orchestrating - Tricky topic!

- Orchestrating docker containers
- Orchestrating Kubernetes k3s for smaller setups in deeper docker container
- Orchestrating full K8s with network plane in internal cluster (Not implemented!)
 - => We have seen this very unuseful for actual software optimizations. It is nice for reporting. But not for optimizations as normal observability does give no insights about actions on the system
 - => Share your insights!
- How to handle external networking? Mirror resources locally? Spoof?

Reproducability

Alternatives to orchestration - System auto discovery

- We have seen this from softwares like Kepler
- We call it: **Monitor mode**
 - Difference to Kepler: Triggers to start / end a measurement frame
 - This allows to have reproducible measurements even in live systems
- => Do you think of any other relevant resources for modern systems? Put it on a note later!

Reproducability - Orchestration

Going through all phases of the Blue Angel for Software

The screenshot shows the Blue Angel software interface for managing software configurations and runtime flows.

Configuration Details:

- name:** Nextcloud Base (mod_php-SQLite-Chromium)
- uri:** <https://github.com/green-coding-solutions/nextcloud-docker>
- commit_hash:** e4246c5e11bbad808f3e71c4eb120244328b63c7
- filename:** energy-tests/usage_scenario-sqlite-event-chromium.yml

Click here for more data ...

Phase Flow:

Baseline Installation Boot Idle Runtime Remove

Runtime can contain multiple flows. By default all runtime flows are aggregated. Please select a separate flow if needed.

Flow Selection:

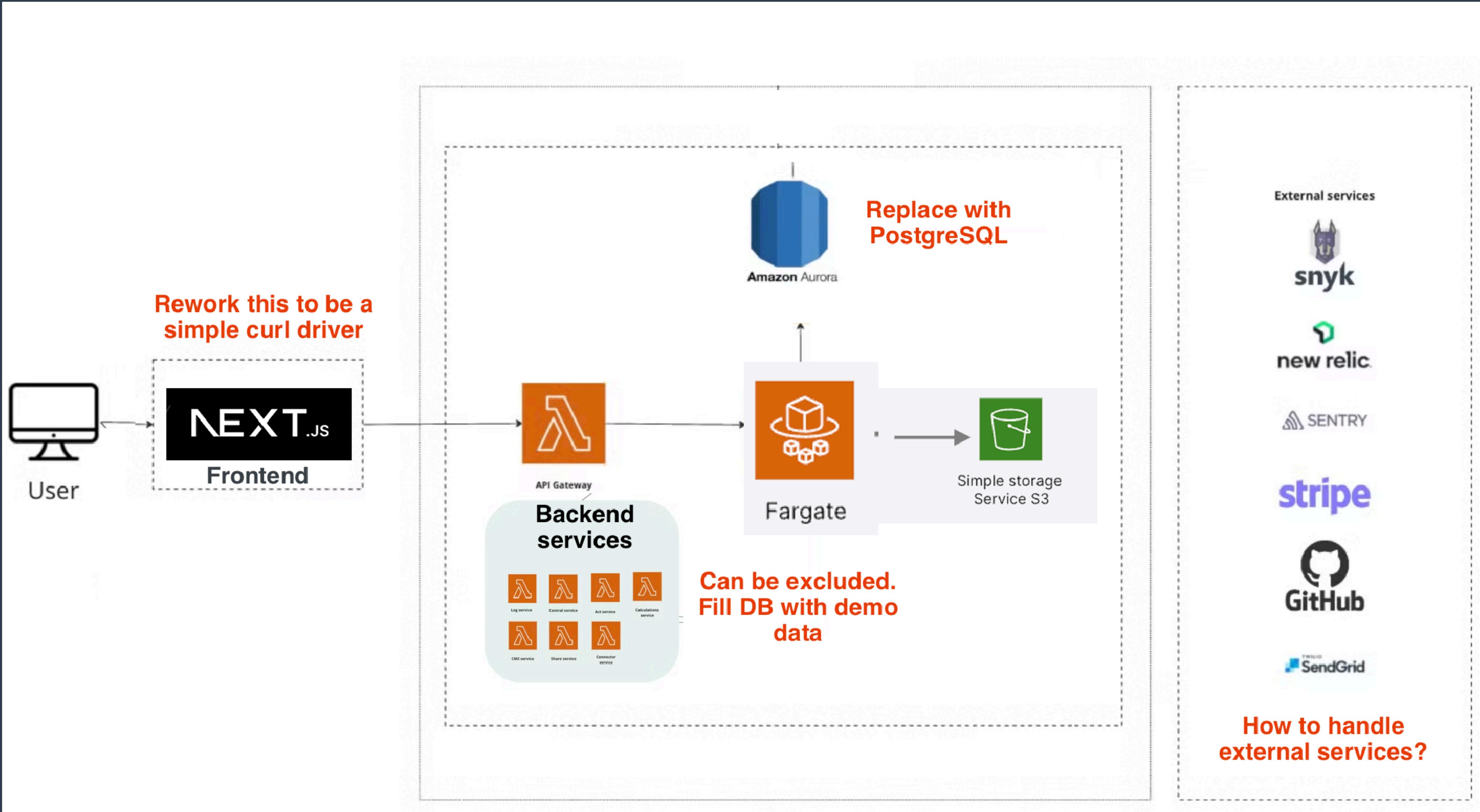
- All Flows
- Install Nextcloud
- Login and create event

Thinking of software in phases. It adheres to upcoming standards AND It will come in handy later!

Orchestration - Open problems

When looking at cloud architectures for instance

- Keep in mind that optimization can only happen in domains you control or can benchmark
- Even cloud applications can be effectively optimized with slight changes.
- Replace with equivalents (has bias, but will behave comparable)
- Be mindful of what needs to part for minimal use case to keep system error influence low.
- Example ->



</> GREEN CODING;

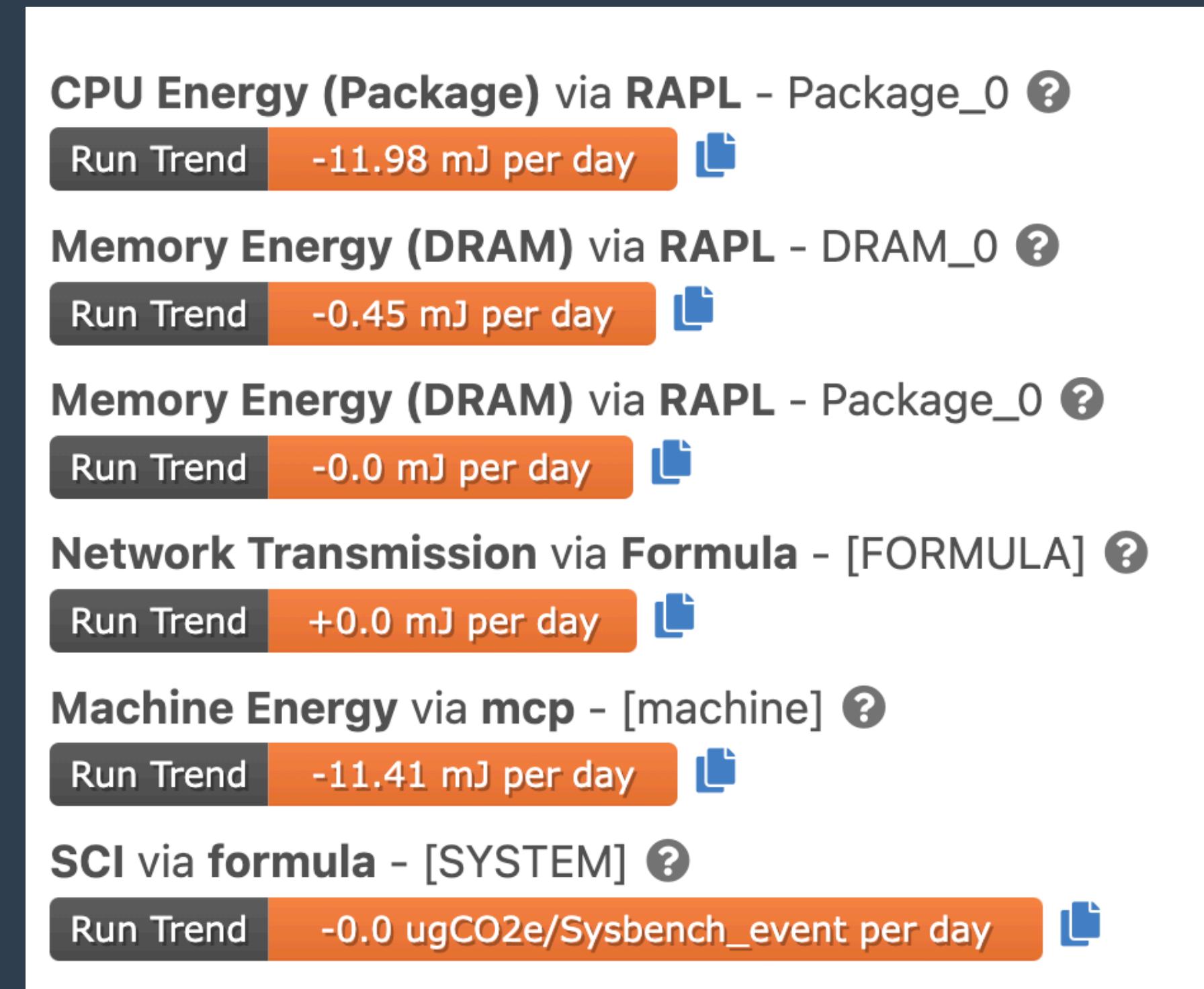
Stage #3

Measurement

Measurement

There is so much to measure. How to select?

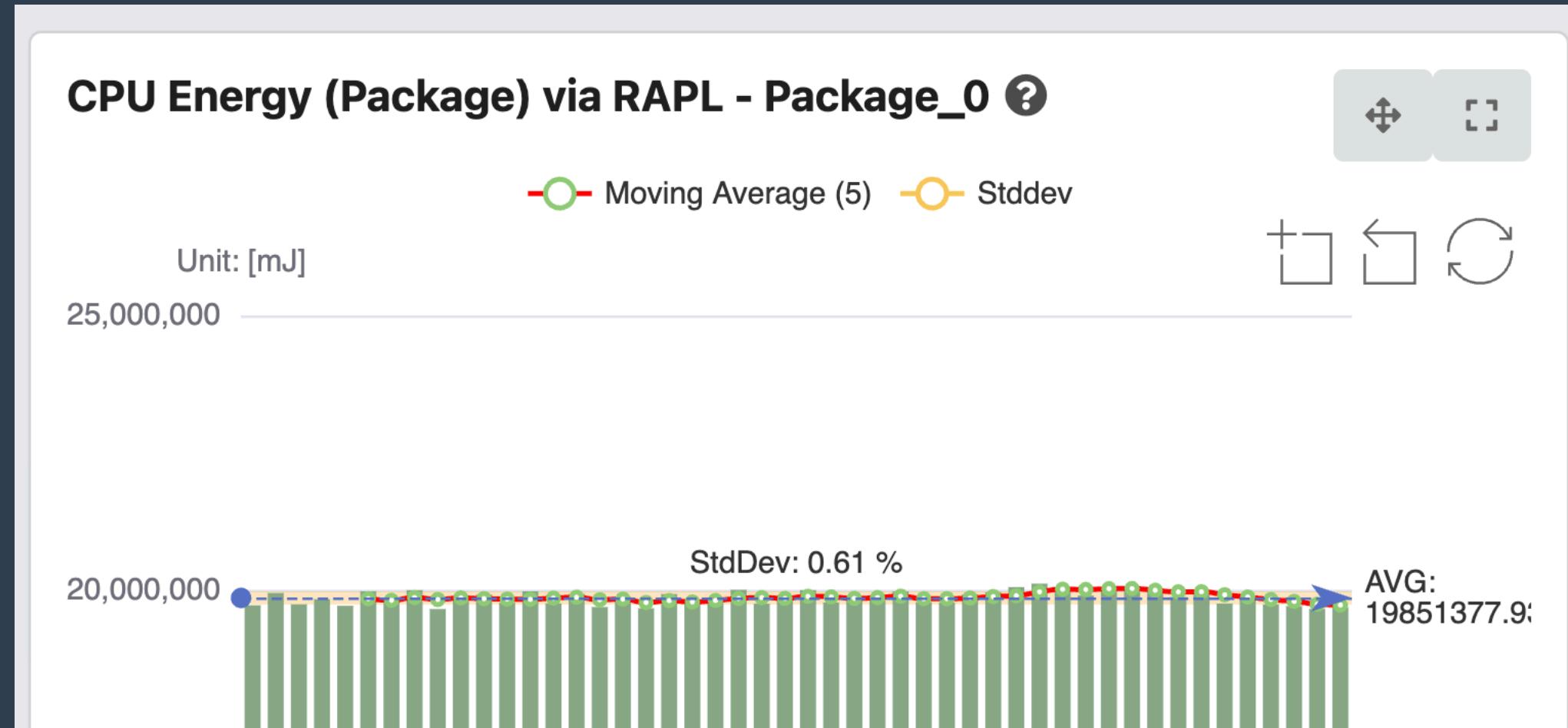
- We opted for a modular reporter architecture with sane defaults



Measurements - Reproducability

Maintaining measurement accuracy

- Temperature Control, Resource utilization control
- Off-the shelf OS with timers removed



Control workloads every 6h - StdDev < 1% between measurements

Running System Checks

```
Checking db online : OK
Checking single energy scope machine provider : OK
Checking tmpfs mount : OK
Checking < 5% CPU utilization : WARN (Your system
    seems to be busy. Utilization is above 5%. Consider terminating
    some processes for a more stable measurement.)
Checking single energy scope machine provider
Checking tmpfs mount for low filesystem latency : OK
Checking 1GB free hdd space : OK
Checking docker daemon : OK
Checking running containers : OK
Checking current temperature" : OK
```

Guard clauses and sanity checks



GREEN CODING;

Measurement-Reproducibility

Running test workloads and setting best practices

List of best practices

1. Never compare between machines to judge your software
2. An application should NEVER come to the bounds of its resources
3. The application you want to test must run at least twice as long as the minimal resolution
4. When running tests your disk load should not go over 50%
5. Limit amount and resolution of Metric Providers to what you absolutely need

6. Always check STDDEV

7. Design representative
Standard Usage Scenarios

8. Pin your dependencies

9. Use temperature control and validate measurement std.dev.

10. Trigger test remotely or keep system inactive

11. Your system should not overheat

12. Mount your /tmp
on /tmpfs

13. Turn logging off

14. Use --docker-prune

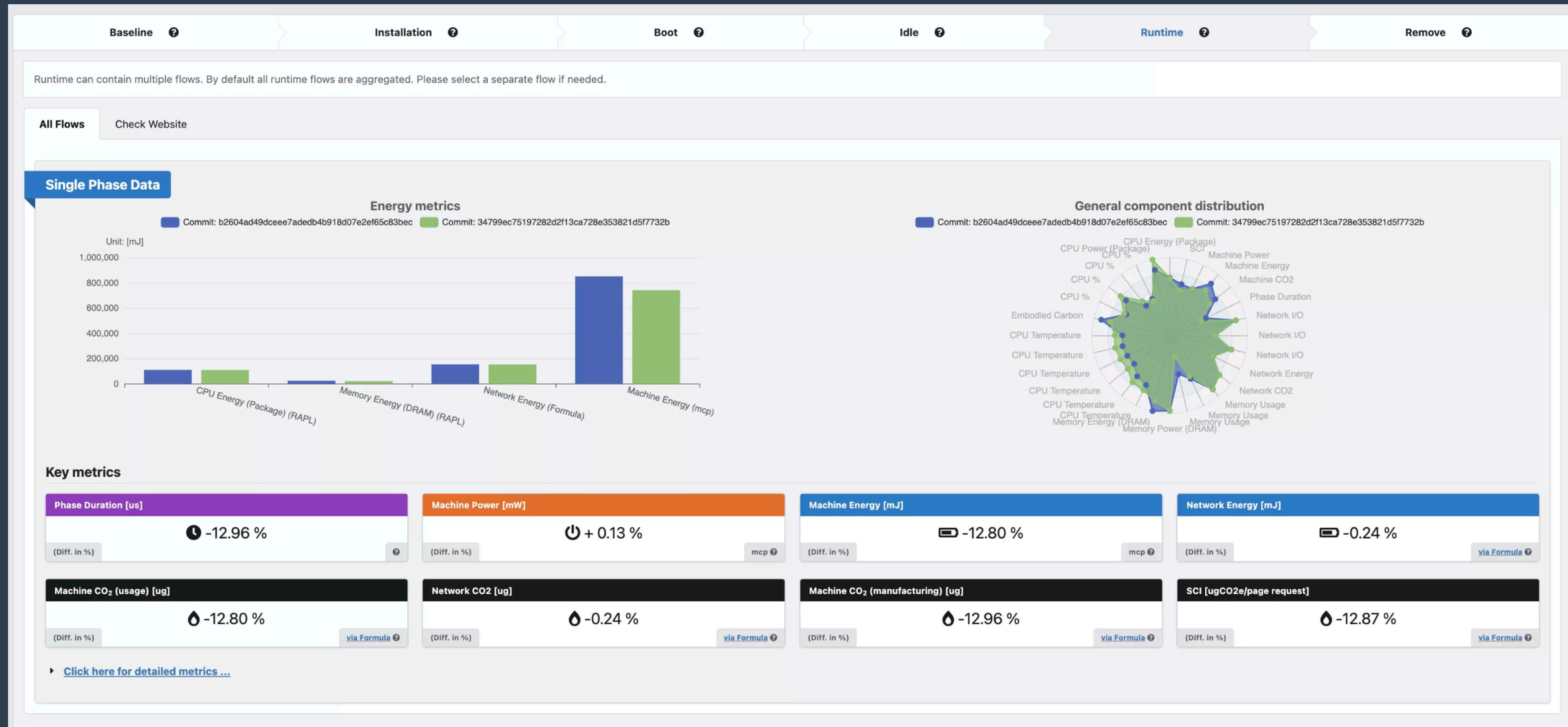
15. Use non standard sampling intervals and avoid undersampling

16. System Check Threshold

- Not be read on this slide! But it is a long list and growing ... Contribute!
- <https://docs.green-coding.io/docs/measuring/best-practices/>

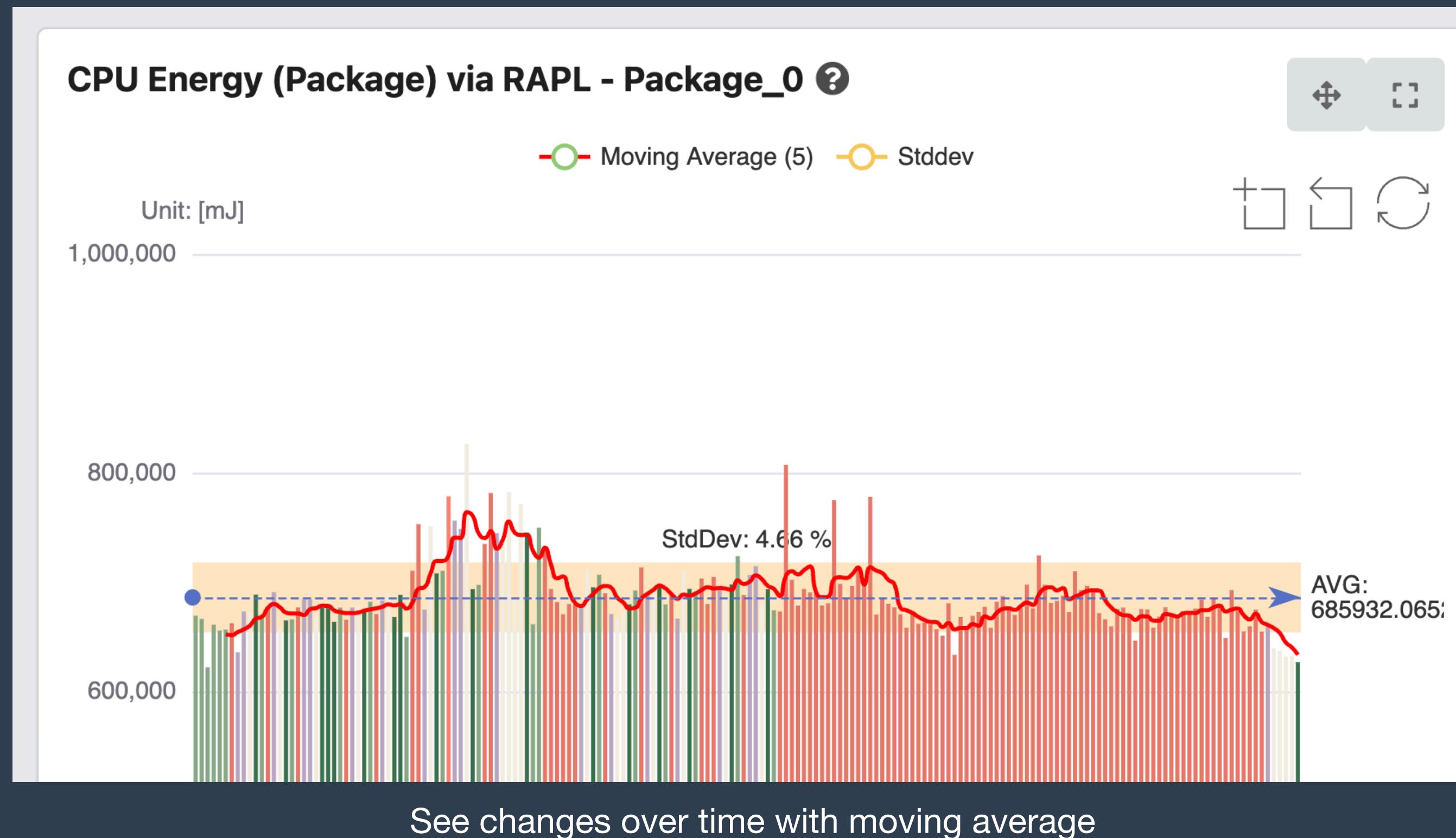
Measurements - Comparison

Comparing between Machines, Between repositories, Between Runs ...



Measurements - Comparison

Behaviour of software over time



CPU Energy (Package) via RAPL - Package_0 ?

Run Trend -1.33 mJ per day

Showing regression trend

Stage #4

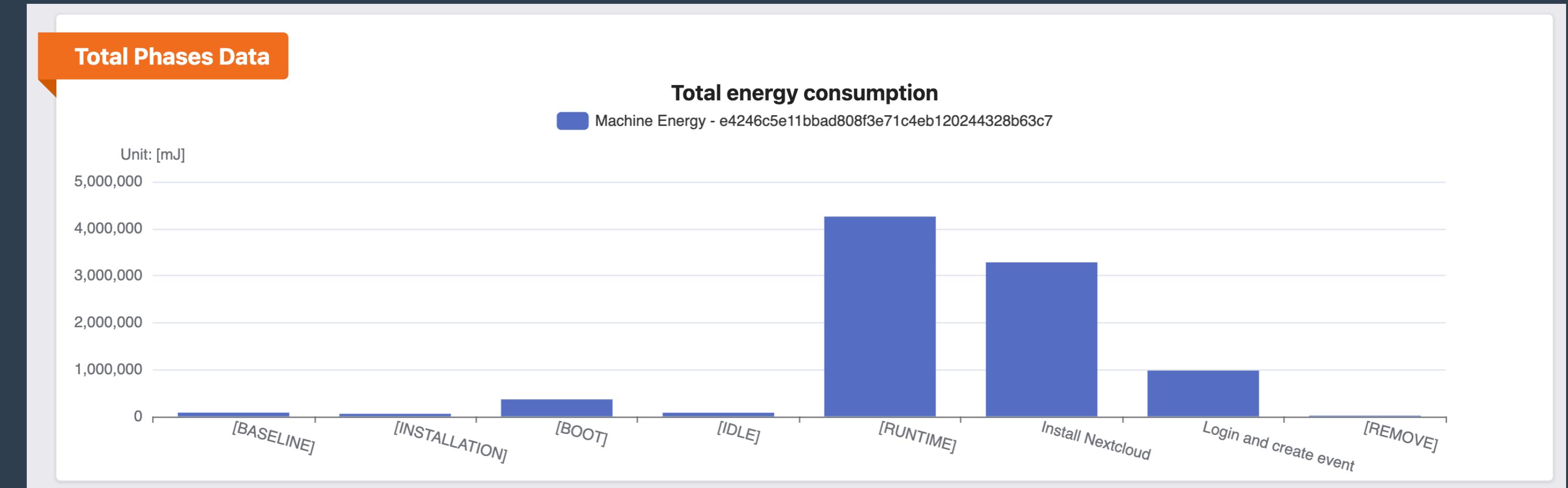
Optimizations

Optimizations

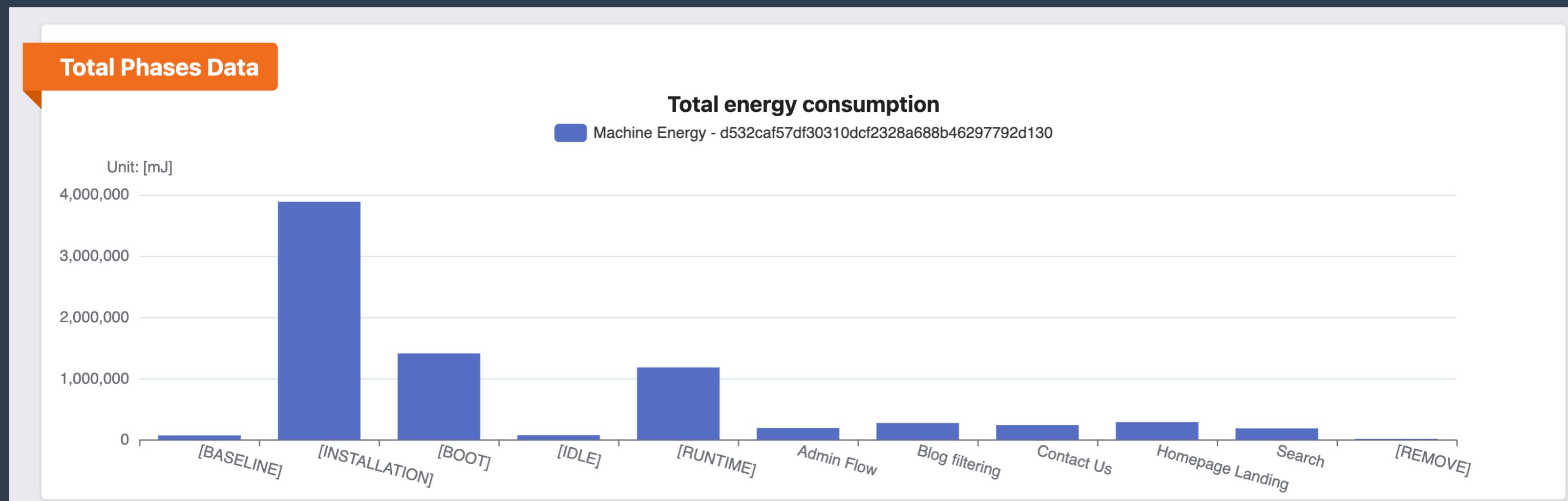
Split into four technical domains of optimizations

- **Part #1:** Algorithmical observability based optimizations
 - Using metrics that the OS already provides like CPU%, Memory, time etc.
- **Part #2:** Algorithmical profiling based optimizations
 - Using PMUs or deep profiling like IPC, Energy, access-times etc.
- **Part #3:** Changes over time
- **Part #4:** Targeted Code quality / review based optimizations through AI
 - Identifying code quality of specific parts of the application

Phases interplay already reveals a lot info



Boot and Build compared to Runtime is already low



Boot compared to runtime very bad. Problematic in FAAS and CI/CD cases where caching is possible

Optimizations - Part #1

Algorithmical observability based optimizations

- Once we orchestrate the application, we can tune it's resource consumption behaviour

The screenshot shows two alert cards from the Green Metrics Tool:

- Docker container boot time**: The container takes very long to become usable ~ 456s. Containers are meant to be started and killed quite fast. (Docker, docker-boot-time)
- Cpu container resource allocation**: Container 'app' is maybe overprovisioned. CPU utilization was '2%'. Max was '31%'. (Docker, docker-cpu-allocation)

Example from Green Metrics Tool for Container Boot and CPU resource metrics

Optimizations - Part #1

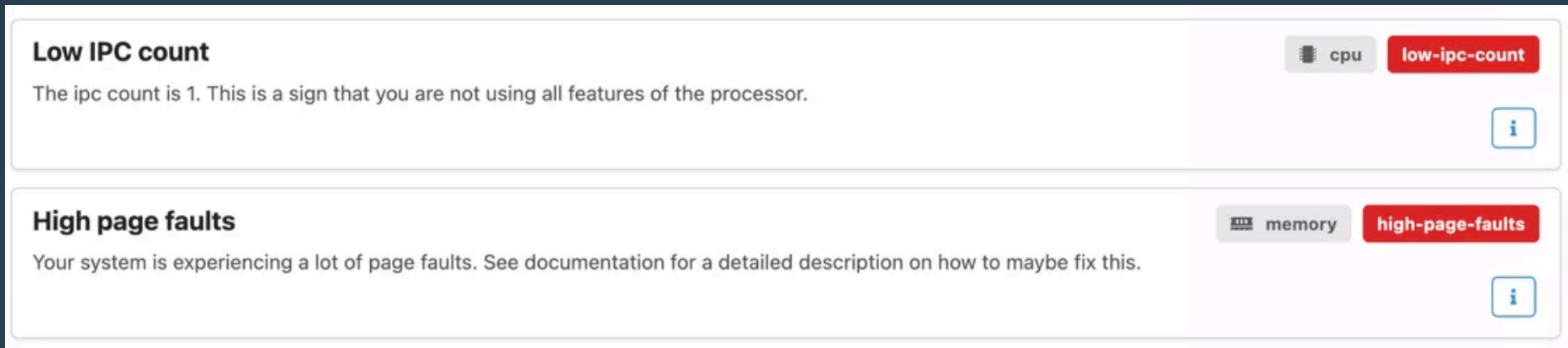
Algorithmical observability based optimizations

- We can determine over-provisioning by testing with lower limits and comparing results
- We can determine boot vs. runtime duration because we know an average request length (FaaS)
- We can look at image sizes, image layer architecture that breaks caching
- And many more from best practice lists like the Green Software Foundation, Digital Sustainability Center and many more ...

Optimizations - Part #2

Algorithmical profiling based optimizations

- We need to instrument an application by capturing tracepoints, patching the runtime, polling PMUs etc. to make this work.
- Also we must change characteristics of the OS and see how the application behaves



- But the benefits are high! In example: Utilizing processor in full can bring 8x improvement

Optimizations - Part #3

Changes over time

An important domain of Green Coding is also fighting software bloat



Docker image size between different releases of a software

Optimizations - Part #3

Changes over time

Run Data

General Measurement Badges Machine Usage Scenario Logs Network Optimizations 1

Container Image Size growing

Since your last run the container image size has increased by 20%. To combat software bloat every increase in resource consumption should be engaged with efficiency measures. Try pruning your image or re-running optimization with 'atime-full' option to auto detect unused files.

GMT core docker >

Network traffic size increased

Since your last run your container pulled 35.25 MB more data (66.21 MB in total). 22.10 MB of the pulled data was already known from a previous run. Try pulling this data only once during image build if possible

GMT core network >

Network traffic duplicated

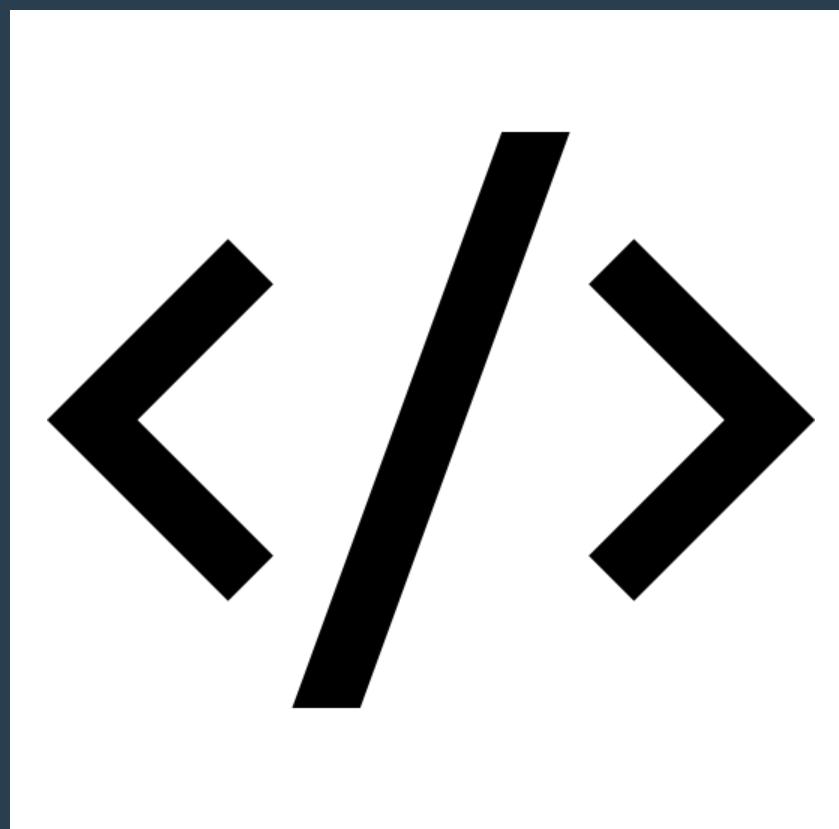
10.01 MB was pulled multiple times during the benchmark. If possible try to cache data and / or reduce polling frequency. If possible move to a socket based push connection instead of using polling for status.

GMT core network >

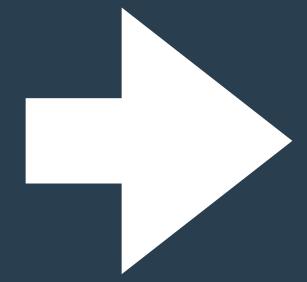
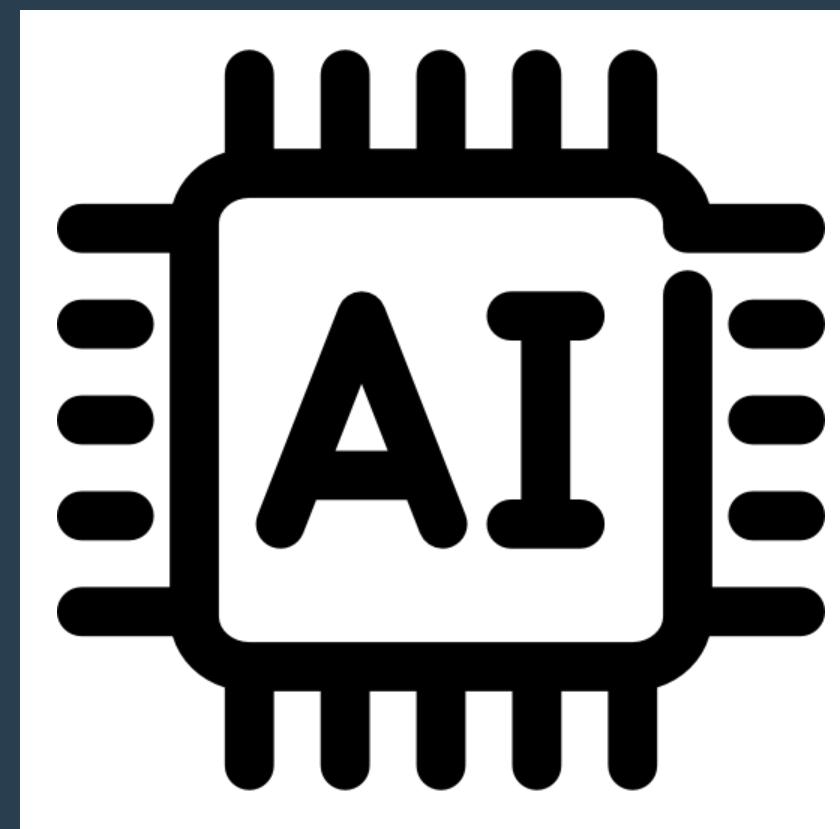
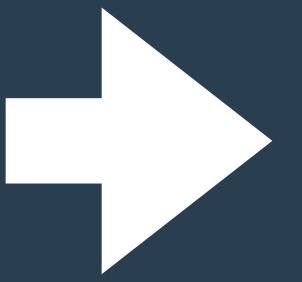
Example for intercepted and analysed network traffic

Optimizations - Part #4 - AI

AI Optimizations - What everyone wants

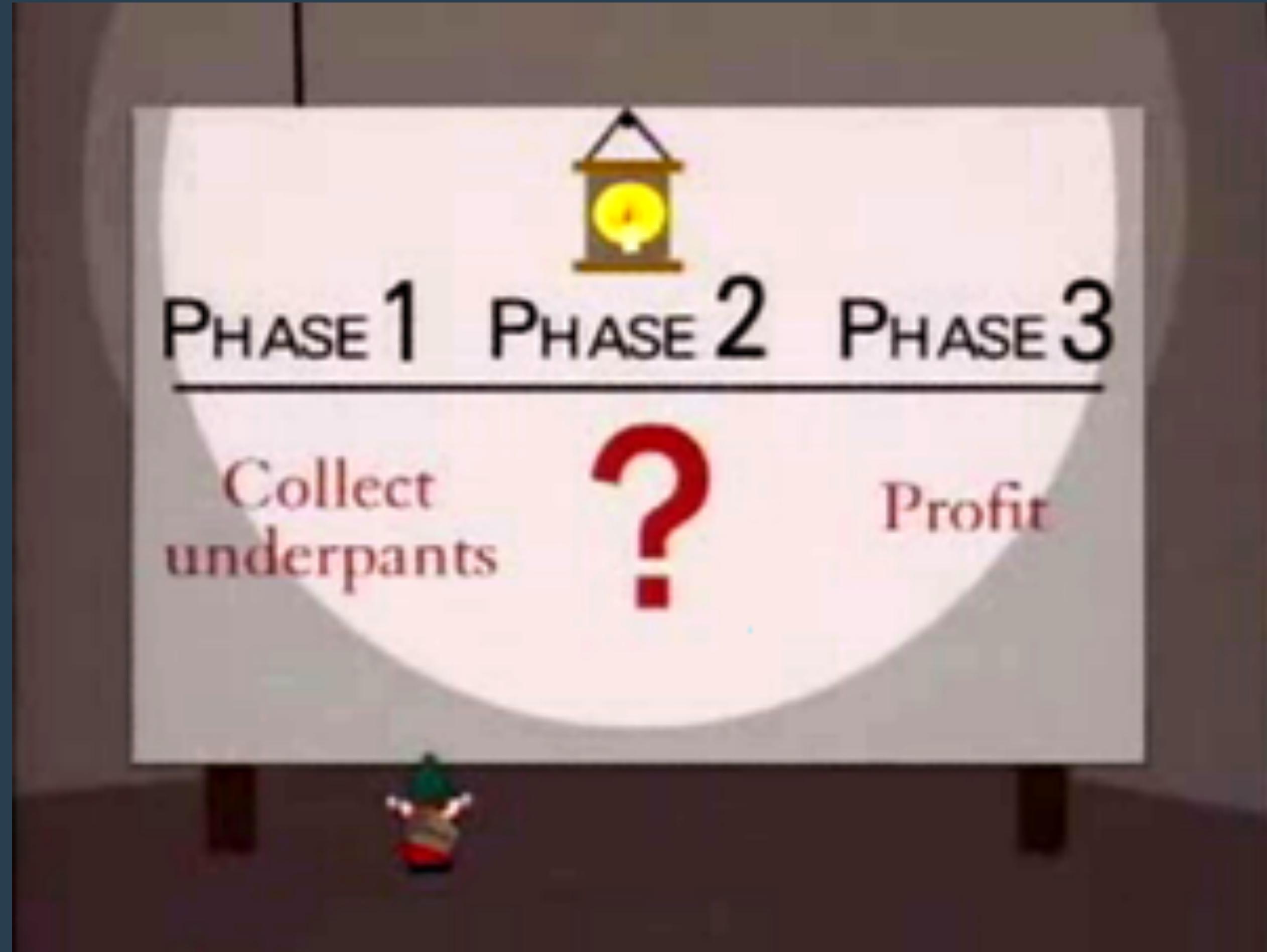


Code



Optimizations - Part #4

AI Optimizations - What everyone wants



Don't we know this from the Simpsons already?

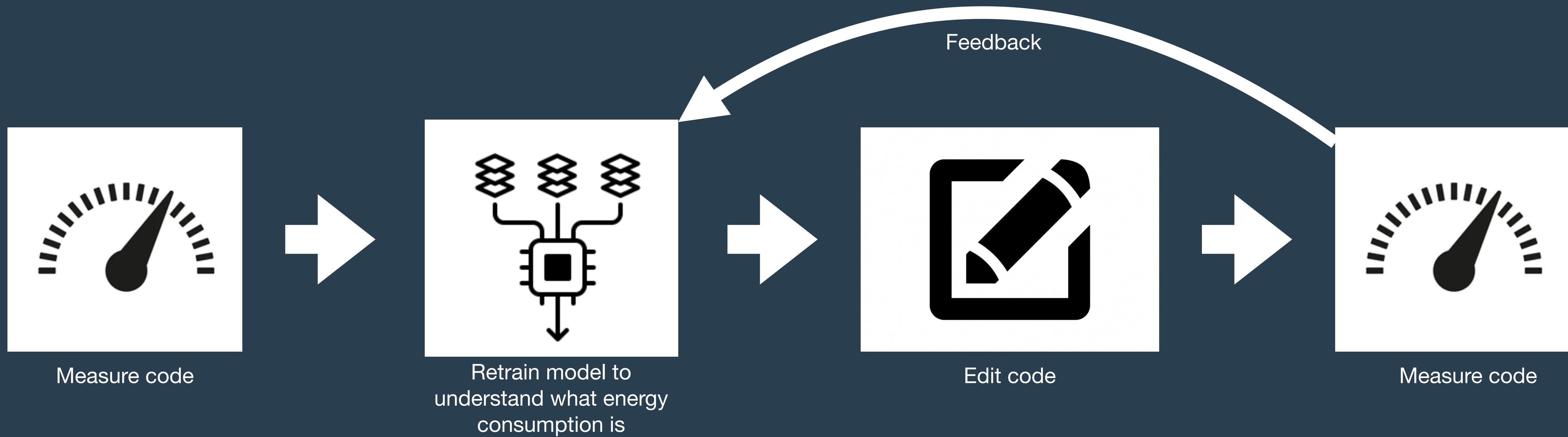
Optimizations - Part #4

AI Optimizations - This doesn't really work that well

- AI models no understanding of “green”
- AI models can not consume large complex projects
- Performance Engineering \neq Green Coding
- Modern systems use loads of libraries and functionalities that change depending on the system. Some operations great on X86 but really bad on ARM
- Normally ends up being Lint++

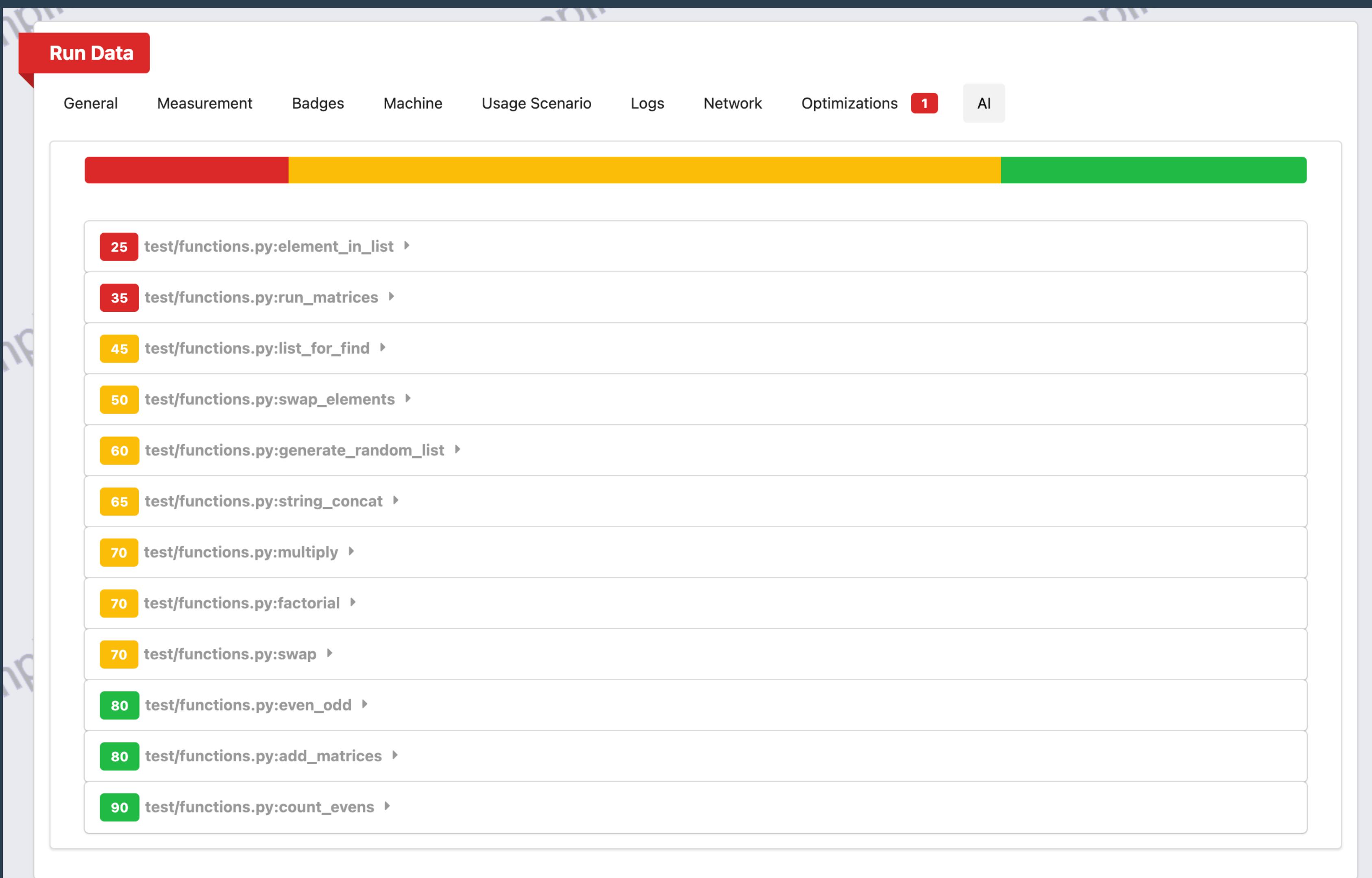
Optimizations - Part #4

AI Optimizations - What we are working on - WIP



Problem: You need a lot of compute

Optimizations - Part #4 - Our current state



Optimizations - Part #4 - Our current state

45 test/functions.py:list_for_find ▾

You code

```
def list_for_find(length):
    randome_elements = random.sample(range(MIN, MAX), 100)
    list_seq = list(generate_random_list(length))
    counter = 0
    for ele in randome_elements:
        if ele in list_seq:
            counter += 1
    return counter
```

ChatGPT

To properly assess the performance and optimization potential of the provided piece of code, we would need to consider several factors, such as code efficiency, use of data structures, algorithmic complexity, code readability, and maintainability. As the code snippet is incomplete and lacks context (e.g., definitions of `MIN`, `MAX`, and `generate_random_list`), I can only provide a general assessment based on the visible portion.

Here's a brief analysis:

1. **Algorithm Efficiency**: The `if ele in list_seq` operation is $O(n)$ for a list, where n is the length of `list_seq`. This means if `list_seq` is long, the operation will be slow since it checks every element until a match is found.
2. **Use of Data Structures**: Using a list for `list_seq` is not optimal for membership tests. A set would be more efficient, as membership tests are $O(1)$ on average for a set.
3. **Randomness**: The `random.sample` function is used correctly to obtain unique elements, assuming `MIN` and `MAX` are properly defined.
4. **Variable Naming**: The names `randome_elements` and `list_seq` could be more descriptive. There is also a typo in `randome_elements`; it should be `random_elements`.
5. **Magic Numbers**: The '100' in `random.sample(range(MIN, MAX), 100)` is a magic number and its significance is not clear without additional context.
6. **Error Handling**: There is no error handling for potential issues such as the `generate_random_list` function returning a non-list type or the range for `random.sample` being smaller than the sample size desired.
7. **Code Readability**: The code is relatively straightforward and readable, but could benefit from comments explaining the purpose and logic.

Considering these points and assuming the unseen parts of the code (like the `generate_random_list` function) are implemented with similar considerations, I would give the visible code a rating of around 45 out of 100. With optimization, such as using a set for `list_seq` and cleaning up the code, the rating could be much higher.

Rating: 45

Improve this with AI

Our showcase for testing inferencing cost of Open Source AI models

The screenshot shows the homepage of the Green Coding AI project. At the top, there's a header bar with a logo, the text "Green Coding AI", a help icon, and a button labeled "Show in Joule". Below the header, a large heading says "Welcome to the Green Coding AI.". A subtext explains the project's goal: "AI is spreading like a wildfire but at what cost? This project aims to show how much asking an GPT uses in resources. Please checkout the [FAQs](#)".

Two conversational entries are displayed:

- You**: what is green coding in 100 words?
- llama3**: Green coding refers to the practice of writing software code that is environmentally friendly and sustainable. It involves designing, developing, and deploying software applications that minimize their ecological footprint, reduce waste, and promote energy efficiency. Green coding aims to make software development more eco-friendly by using techniques such as: reducing CPU usage, minimizing data transfer, optimizing memory usage, using open-source technologies, and promoting digital literacy to reduce e-waste. The goal is to create software that not only benefits humans but also helps protect the environment.

Below the conversation, several performance metrics are listed:

29.39	112.65	184.09	0.0953
CPU WATT	GPU WATT	MACHINE WATT	GCO ₂ EQ EMITTED
0.1833	4.83	51°C	
SCI PER REPLY	TIME SECONDS	CPU TEMP	

At the bottom, there's a chat input field with a dropdown menu set to "llama3" and a placeholder "How can I help you?".

Different models have vastly different costs. Try the playground to get a glimpse!



You

what is green coding in 100 words?



tinyllama

Green Coding, also known as Lightweight Coding, is a method for software development that simplifies the process of writing and maintaining code without compromising performance or functionality. It involves breaking down complex algorithms into smaller, more manageable components called "steps" to simplify the programming process while still ensuring high-quality code. In essence, green coding reduces the number of lines of code required for a given function, thereby reducing overall code size and improving software performance. The technique is highly effective for developing web applications or large data processing systems as it helps in identifying the most critical parts of code while also ensuring that they are efficient enough to be executed efficiently. In this way, green coding facilitates a more efficient development process and ultimately provides developers with better quality code that meets performance requirements.

28.75
CPU WATT

98.61
GPU WATT

170.31
MACHINE WATT

0.0273
GCO₂EQ EMITTED

0.0545
SCI PER REPLY

1.49
TIME SECONDS

51°C
CPU TEMP

Now we want your opinions and ideas

Let's talk about how to implement and automate optimizations

- We have prepared a Miro board at: <https://t.ly/hTuwW>
- Questions are:
 - What tools are you using?
 - What green optimizations have you implemented already?
 - What problems have you faced?
 - What magical tool would you wish for?
- Then in one hour: We deep dive into the questions and discuss them in the groups.
Please also pick out Post-its you want to talk about and start a discussion :)



Now we want your opinions and ideas

Miro Board Preview

CI / CD

- Which products / tools do you use?**
 - Github Actions (Shared Services)
 - Github Pipelines
 - CircleCI

Jenkins

What do you want to add?
Grab a note!
- What do your pipelines do?**
 - Comparing buildtimes with Git for Linux
 - Unit Tests in pipeline
 - Build from pull request and cache broken builds

Visual regression testing

What do you want to add?
Grab a note!
- What metrics do you track?**
 - CPU energy via Intel Rapi
 - Container CPU utilization
 - Pipeline duration

What do you want to add?
Grab a note!
- What is your green KPI?**
 - Pipeline runtime
 - CO₂ per build
 - CO₂

What do you want to add?
Grab a note!
- What optimizations have you implemented?**
 - Build cache for shared packages
 - Cache read the package from GitHub
 - Our CI/CD does not allow caching

What do you want to add?
Grab a note!
- Problems you have faced**
 - How did I measure energy from the CPU in Jenkins
 - I cannot read energy from the CPU in Github
 - Our CI/CD does not allow caching

My variation between two measurements is very great

Network latency is too big to get reproducible measurements

Gains are smaller than my measurement error

....

What do you want to add?
Grab a note!
- Which optimizations would you like to implement?**
 - A tool that does cold starts and reads
 - On that can change our pipeline to be more efficient

What do you want to add?
Grab a note!
- Additional stuff**
 - Please put this tool for optimized jobs

What do you want to add?
Grab a note!

Problems you have faced

- How do I measure energy in Jenkins?
- I cannot read energy from the CPU in Github
- Our CI / CD does not allow caching

My variation between two measurements is very great

Network latency is too big to get reproducible measurements

Gains are smaller than my measurement error

....

What do you want to add?
Grab a note!

Close up of one box

</> GREEN CODING;

Now we want your opinions and ideas

Preview: Questions of the Miro Board

- Which products / tools do you use?
- What does your software do? / What do your pipelines do?
- What is your green KPI? / What constitutes a green software for you?
- What optimizations have you implemented
- Problems you have faced
- Which optimizations would you like to implement? / Which magical "feature" or "tool" that you would like to exist?

Thank you

Want to know more

- Website / Blog / Newsletter: <https://www.green-coding.io>
- Green Metrics Tool Free SaaS: <https://metrics.green-coding.io>
- Github page: <https://github.com/green-coding-solutions/green-metrics-tool>
- Our projects: <https://www.green-coding.io/#projects>
- Our case studies: <https://www.green-coding.io/case-studies>
- <https://www.linkedin.com/in/arne-tarara> / arne@green-coding.io

Backup Slides

Ab hier ...

Our approach for a tool

Green Metrics Tool

- We need a measurement tool that can orchestrate the application and thus knows it's infrastructure
- We need a measurement tool that introspect the application and also observe it under different conditions (OS, Network, Hardware etc.)
- We need a measurement tool that interacts with the application in a defined way (like the end use case)
- We need statistical reproducability in order to judge optimization techniques

Our approach for a tool

Green Metrics Tool

- We need a measurement tool that can orchestrate the application and thus knows it's infrastructure
- We need a measurement tool that introspect the application and also observe it under different conditions (OS, Network, Hardware etc.)
- We need a measurement tool that interacts with the application in a defined way (like the end use case)
- We need statistical reproducability in order to judge optimization techniques

Prelude

25.04.2024

- What do we need in order to optimize an application
 - We need a measurement tool that can orchestrate the application and thus knows it's infrastructure
 - We need a measurement tool that introspect the application and also observe it under different conditions (OS, Network, Hardware etc.)
 - We need a measurement tool that interacts with the application in a defined way (like the end use case)
 - We need statistical reproducability in order to judge optimization techniques
 - => This is NOT part of the talk, but please assume that we have such a system :) More details on how to set it up is in the workshop. This talk is in part an appetizer for the workshop

What is Green Coding?

Many organisations have different definitions

Definition from academia

Junger et. al - 2023

- **Green By IT:** [...] (e. g. smart grids, building automation, environmental management information systems (EMIS) like life-cycle assessment software (LCA), etc.) [...]
- **Green IT** deals with making ICT itself more sustainable (e. g. follow-the-sun load shifting, backward compatible software, designing energy- and resource-efficient hardware, etc.)
- **Green Coding:** Considered part of 'Green IT', concerned with the sustainability of the software itself, focusing on methods and tools that can be leveraged during development to impact software sustainability in all of its life-cycle phases (development, usage, and disposal phase).

Green Coding Solutions

Green Coding definition

- Making Energy and CO2 a functional KPI
- Increase transparency
- Efficiency through performance engineering and AI
- Reduce resource consumption / Increase utilization
- Prolong hardware change cycles
- Software developers contribution to climate change and sustainability
- **Summary:** Many parts are already known. However their composition is becoming a separate domain. Similar to DevOps which also distinct responsible domain.



Moving parts

What do we need in order to quantify and optimize
software for less carbon consumption?

This is where we want to be

Making a claim in CO₂ for a software and it's use case



Nextcloud
CMS [click for details](#)

BADGES

Energy Cost	4.89 kJ via PSU (AC)
Energy Cost	1.94 kJ via RAPL
SCI	162.64 mgCO ₂ e/Talk message

[Show measurements](#)



Wagtail
CMS [click for details](#)

BADGES

Energy Cost	1.14 kJ via PSU (AC)
Energy Cost	243.24 J via RAPL
SCI	18.84 mgCO ₂ e/page request

[Show measurements](#)



WORDPRESS

Blog/CMS [click for details](#)

BADGES

Energy Cost	635.62 J via PSU (AC)
Energy Cost	112.62 J via RAPL
SCI	25.08 mgCO ₂ e/page request

[Show measurements](#)

How to get embodied emissions

Using Life-Cycle-Assessment databases

- **Boavizta**

<https://dataviz.boavizta.org/manufacturerdata>

- **Microsoft**

<https://tco.exploresurface.com/sustainability/calculator>

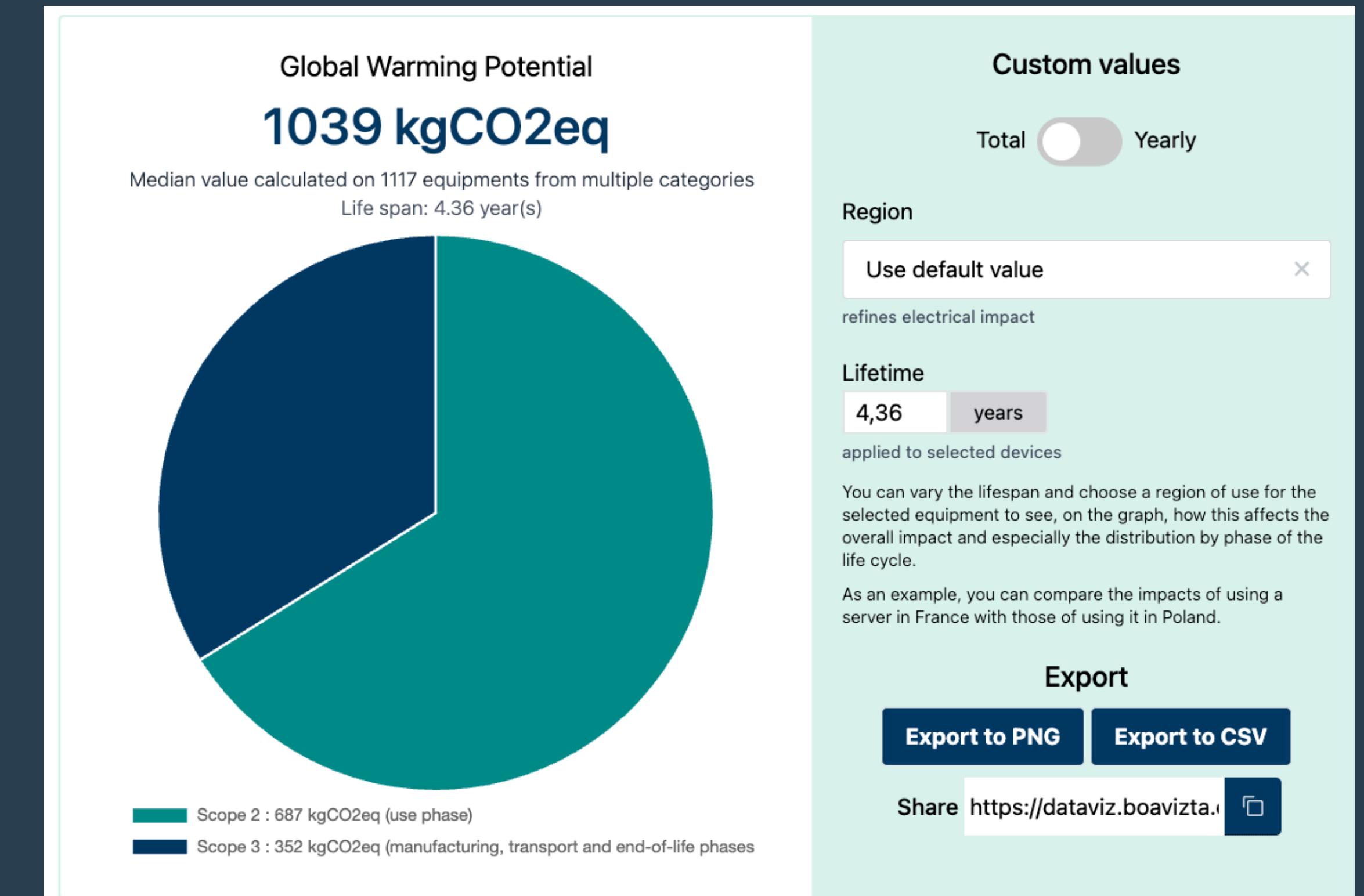
- **Dell**

Example: https://www.delltechnologies.com/asset/en-us/products/servers/technical-support/Full_LCA_Dell_R740.pdf

... many more

- **Important:**

They contain more than only CO₂, also water usage, Toxic metals etc.

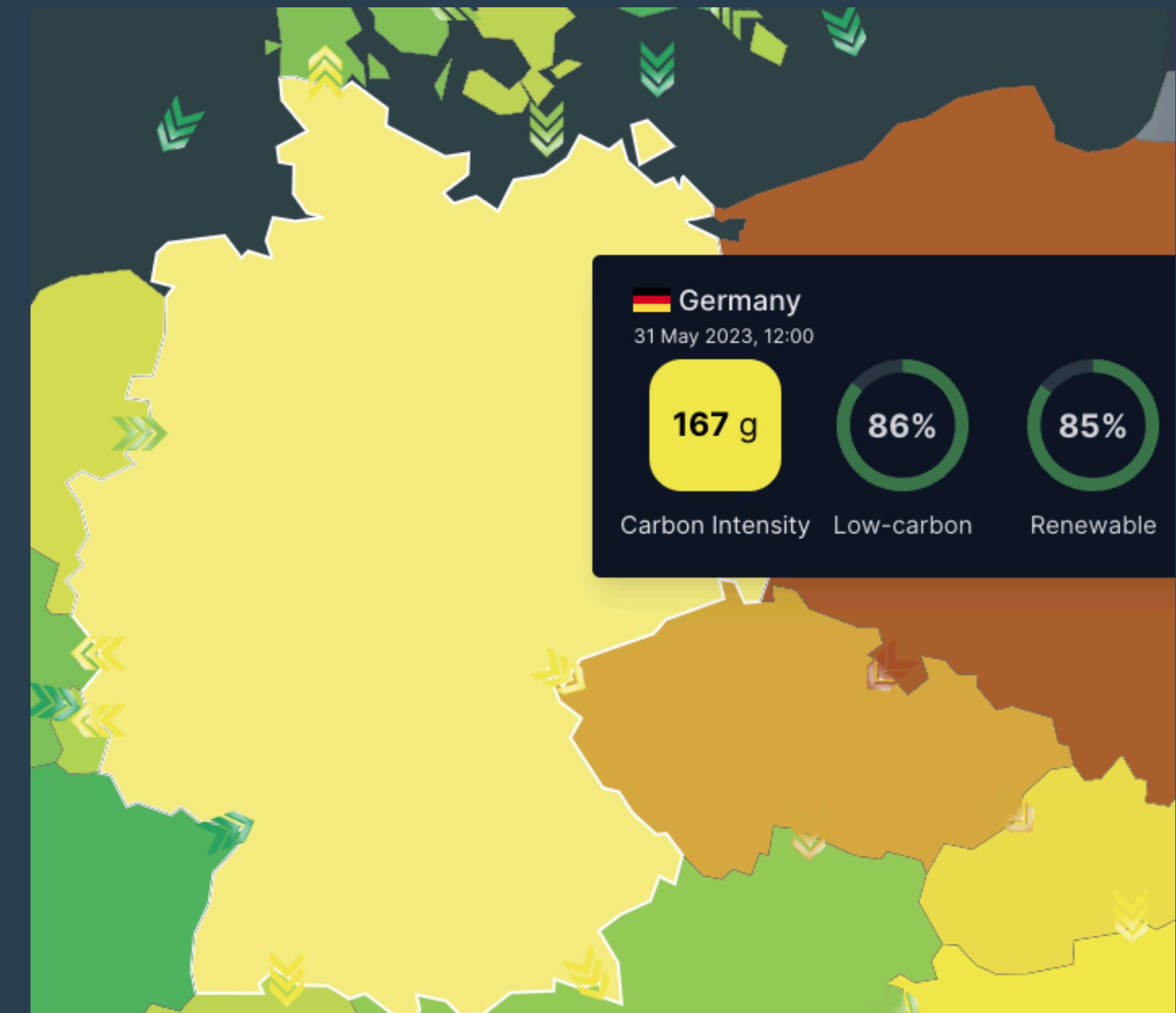


Source: <https://dataviz.boavizta.org/manufacturerdata>

How to get emission factors

Using open data sources

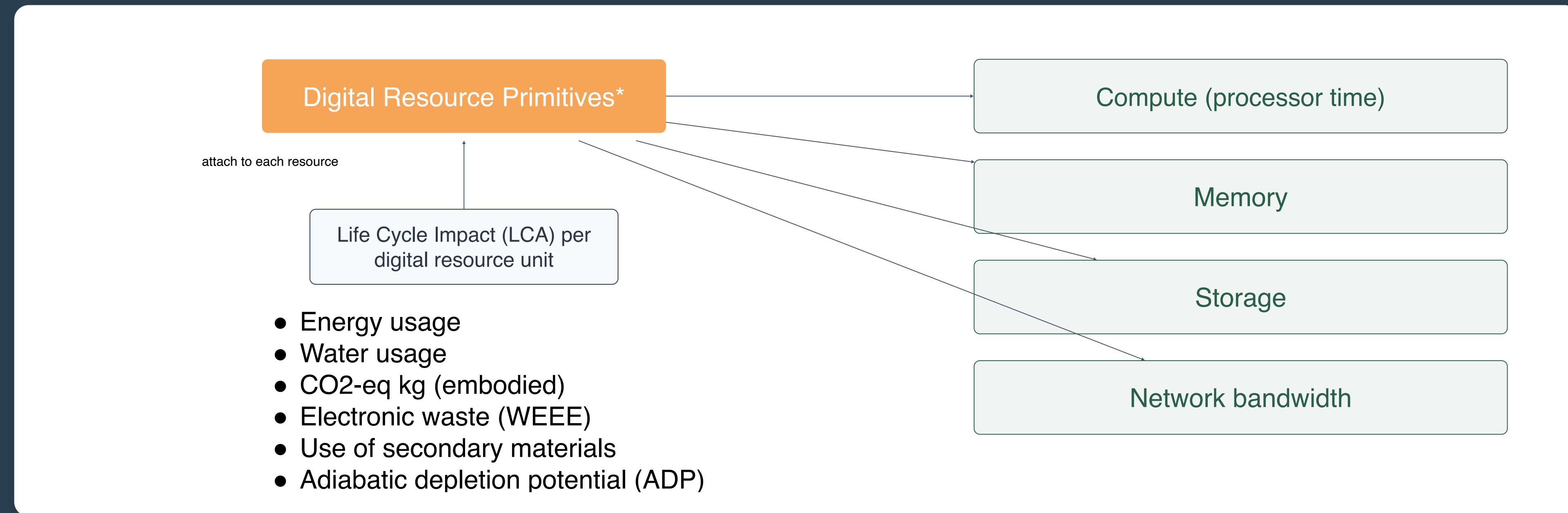
- **Electricitymaps**
<https://www.electricitymaps.com/>
- **Carbon-Aware-SDK**
<https://github.com/Green-Software-Foundation/carbon-aware-sdk>
- **Bundesnetzagentur (DE)**
<https://www.smard.de/home>
- **Watttime (US)**
<https://www.watttime.org/>
- ... many more



Source: <https://app.electricitymaps.com/zone/DE>

Sobald wir die Werte pro Einheit haben können wir diese in abstrahierte Komponenten (Cloud Ressourcen) zerlegen

SDIA Modell der "Digital Resource Primitives"



Nicht nur CO₂ zählt zu den Emissionen der Komponenten

Tools

Alrighty, let's look at some tools!

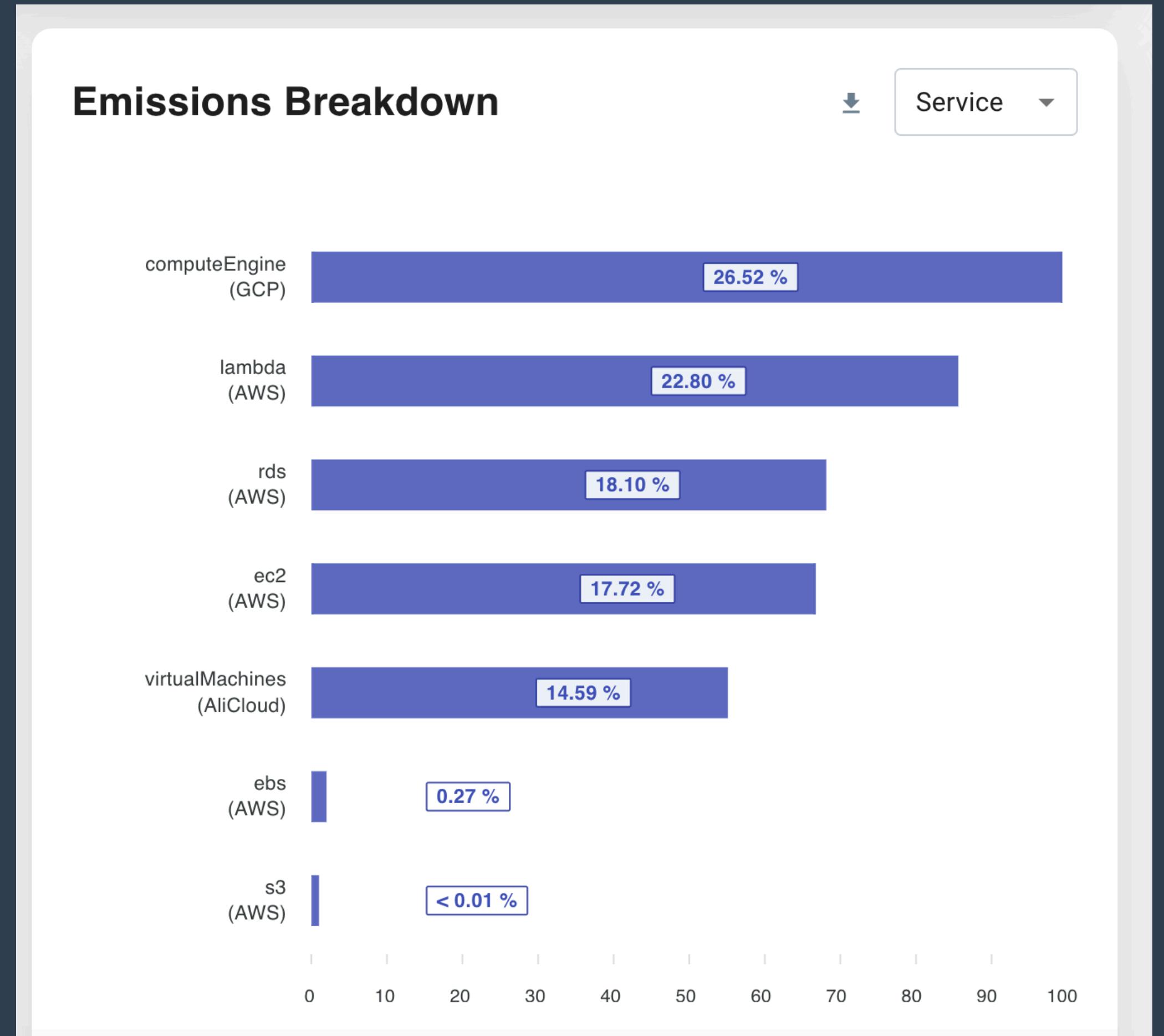
Estimation of IT CO₂ emissions

If you do not know the energy directly - Cloud Carbon Footprint

- Uses billing data
- Based on this, a static calculation model is created [1] [2]
- Most cloud native services not covered
- IAM, DynamoDB, Logstash, etc.
- Downside: Only gives you the company total operation

[1] <https://www.cloudcarbonfootprint.org/docs/methodology/>

[2] Average Watts = Min Watts +
Avg vCPU Utilization * (Max Watts - Min Watts)
DRAM, hard disk etc. via coefficients



Anwendungsfall - pro Komponente

perf_events - CPU, DRAM, GPU (onboard), SMC

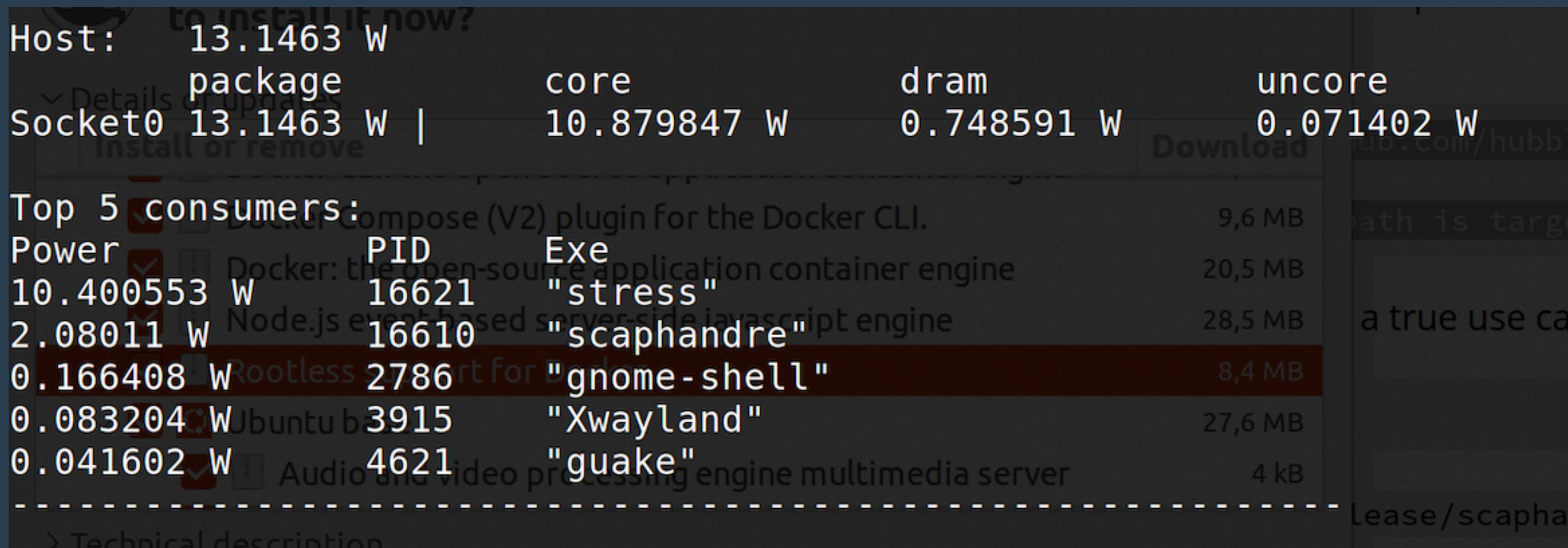
- Initial als Performance Engineering Tool / Profiler entwickelt
- Hervorragendes Tool, sofern neuer Kernel verfügbar [und man sich gut damit auskennt :)]
- Moderne Alternative für *manche* Fälle: eBPF

```
Performance counter stats for 'system wide':  
    76,49 Joules power/energy-pkg/  
    4,92 Joules power/energy-ram/  
    0,50 Joules power/energy-gpu/  
  
 6,692703695 seconds time elapsed
```

Anwendungsfall - pro Prozess

Scaphandre - open-source RAPL based CLI tool

- Nettes Feature: Schlüsselt nach Prozess auf basierend auf CPU-%



- Wie gut klappt das in der Praxis? -> Wir sehen uns im Workshop :)

Anwendungsfall - Programmcode

Beispiel: codecarbon.io



- Python
- RAPL-based
- NVIDIA GPU support

```
1 import tensorflow as tf
2
3 from codecarbon import Emission
4
5 mnist = tf.keras.datasets.mnist
6
7 (x_train, y_train), (x_test, y_test) = mnist.load_data()
8 x_train, x_test = x_train / 255.0, x_test / 255.0
9
10
11 model = tf.keras.models.Sequential([
12     [
13         tf.keras.layers.Flatten(input_shape=(28, 28)),
```

codecarbon
codecarbon

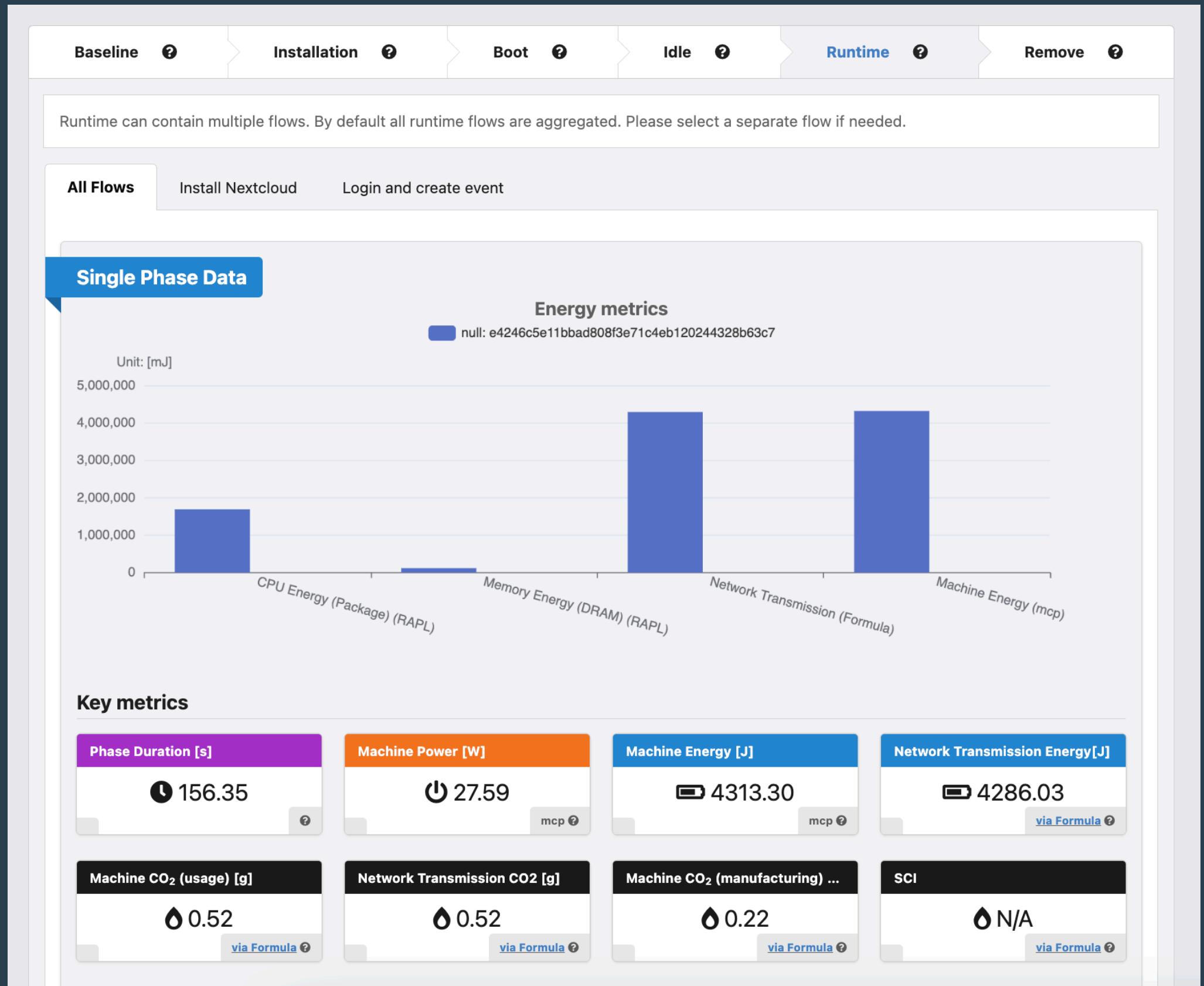
- Bessere Alternativen sind jedoch direkt die Runtime zu patchen / instrumentieren

Quantifying a product

Example: Green Metrics Tool

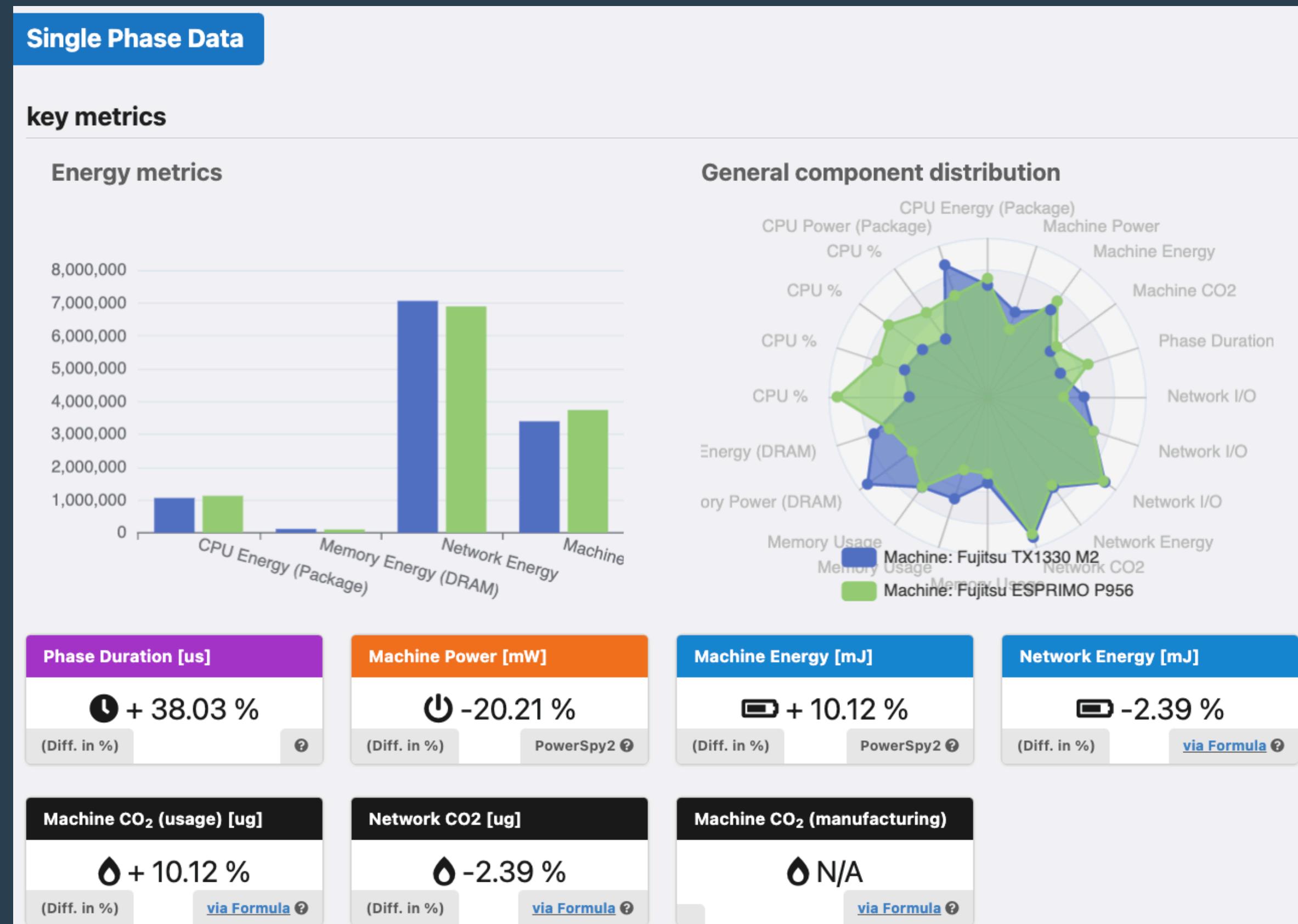
- Benchmarking via Standard-Usage-Scenarios (academia based via UCB / Öko-Insitut e.V.)
- Look at software in all phases of execution Building, Idle, Runtime etc.
- Include embodied carbon and modular metrics reporters
- SCI (ISO-Norm tbd 2024) / Blauer Engel compatible (German Eco Label)

Online Dashboard: <https://metrics.green-coding.io/index.html>

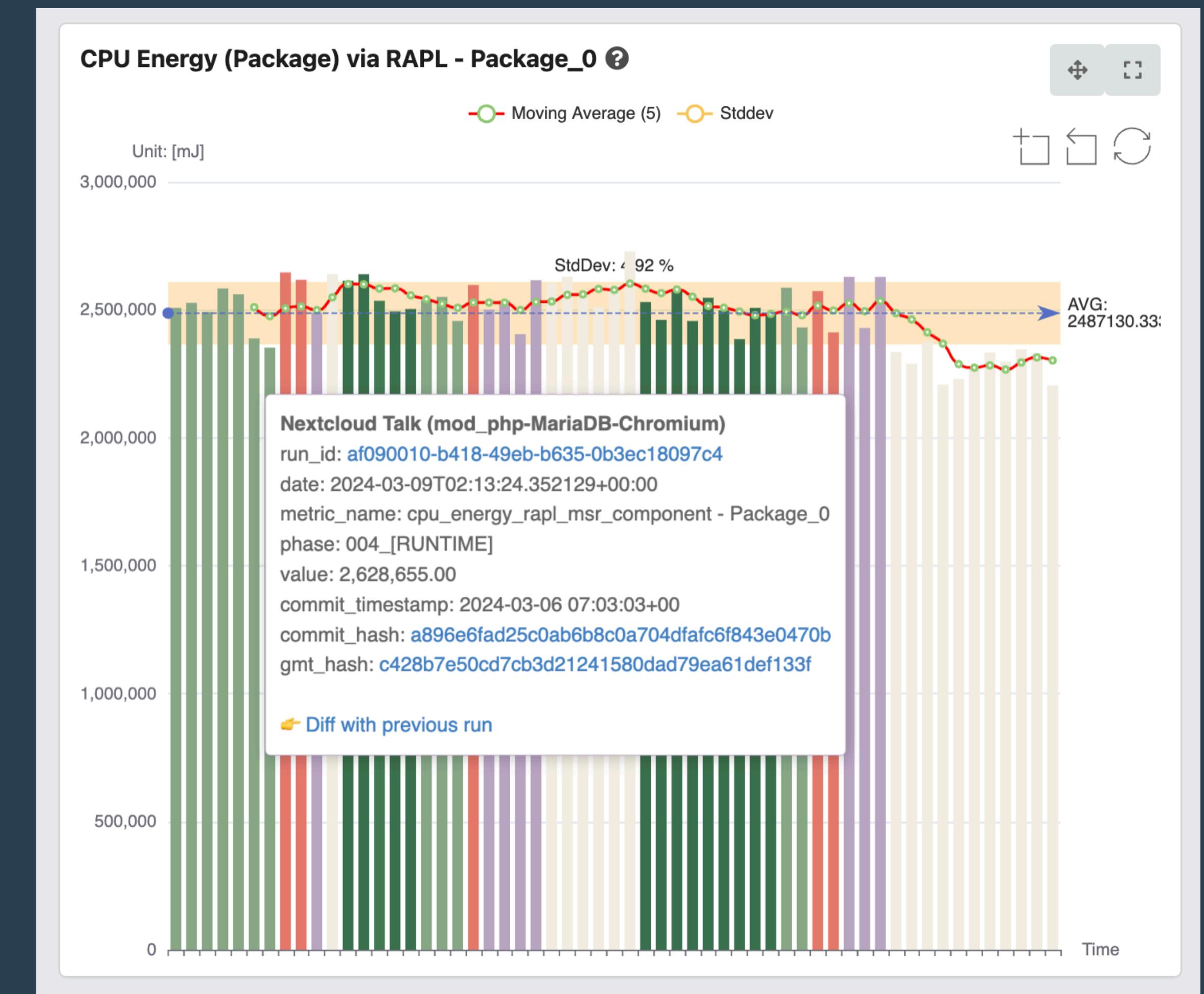


Green Metrics Tool

More Features



Comparisons between Hardware / Repository / Branch etc.



Measurement per Commit - Timeline View

Green Metrics Tool

More Features

- Optimization features
- Based on resource limits and performance metrics
- Includes runtime, build and Idle phases
- Currently in beta

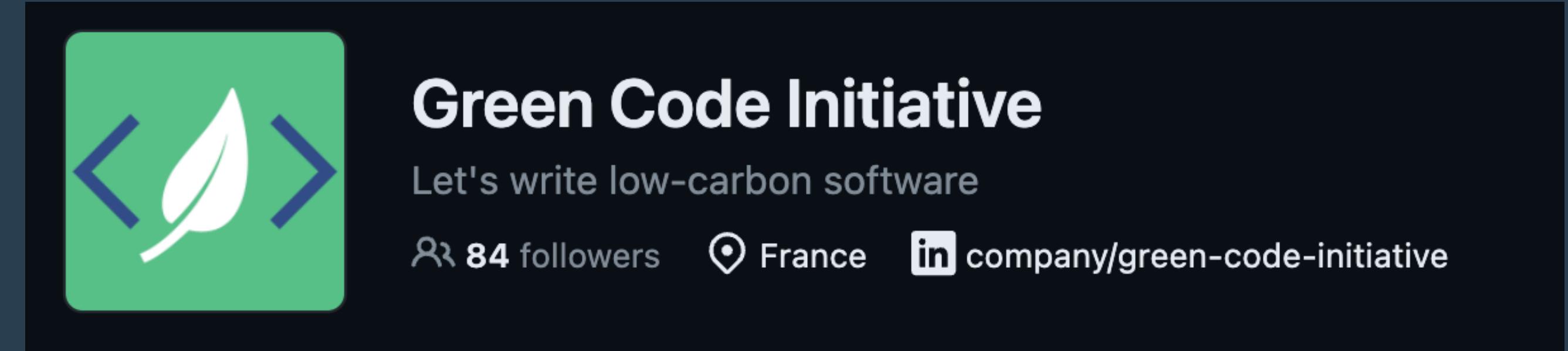
The screenshot shows the 'Run Data' section of the Green Metrics Tool. At the top, there are tabs: General, Measurement, Badges, Machine, Usage Scenario, Logs, Network, Optimizations (with a count of 17), and a red 'Run Data' button. Below the tabs, a message states: "This is a list of potential optimizations we have detected based off your run. We are constantly adding new features so please check our documentation for more details. If you think a warning does not apply to you, you can add an ignore into your usage_sceanrio.yml file." The page lists six optimization suggestions:

- Docker container boot time**: Docker docker-boot-time. The container takes very long to become usable (~ 456s). Containers are meant to be started and killed quite fast.
- Cpu container resource allocation**: Docker docker-cpu-allocation. Container 'app' is maybe overprovisioned. CPU utilization was '2%'. Max was '31%'.
- Low IPC count**: cpu low-ipc-count. The ipc count is 1. This is a sign that you are not using all features of the processor.
- High page faults**: memory high-page-faults. Your system is experiencing a lot of page faults. See documentation for a detailed description on how to maybe fix this.
- Code with high optimization potential found**: Ai ai-low-code-quality. This piece of code was rated a 18/100 by our Green Coding AI. We think this can be optimized.
object_specification.py:473: return [line.strip().split(',') for line in open(file_path, 'r')]
- Energy hotspots**: Benchmark energy-hotspots. We have detected energy hotspots in the flame graph. See documentation on how to run your code with full profiling enabled.

Anwendungsfall Web: Statische Analyse

Eco-Code / CAST etc.

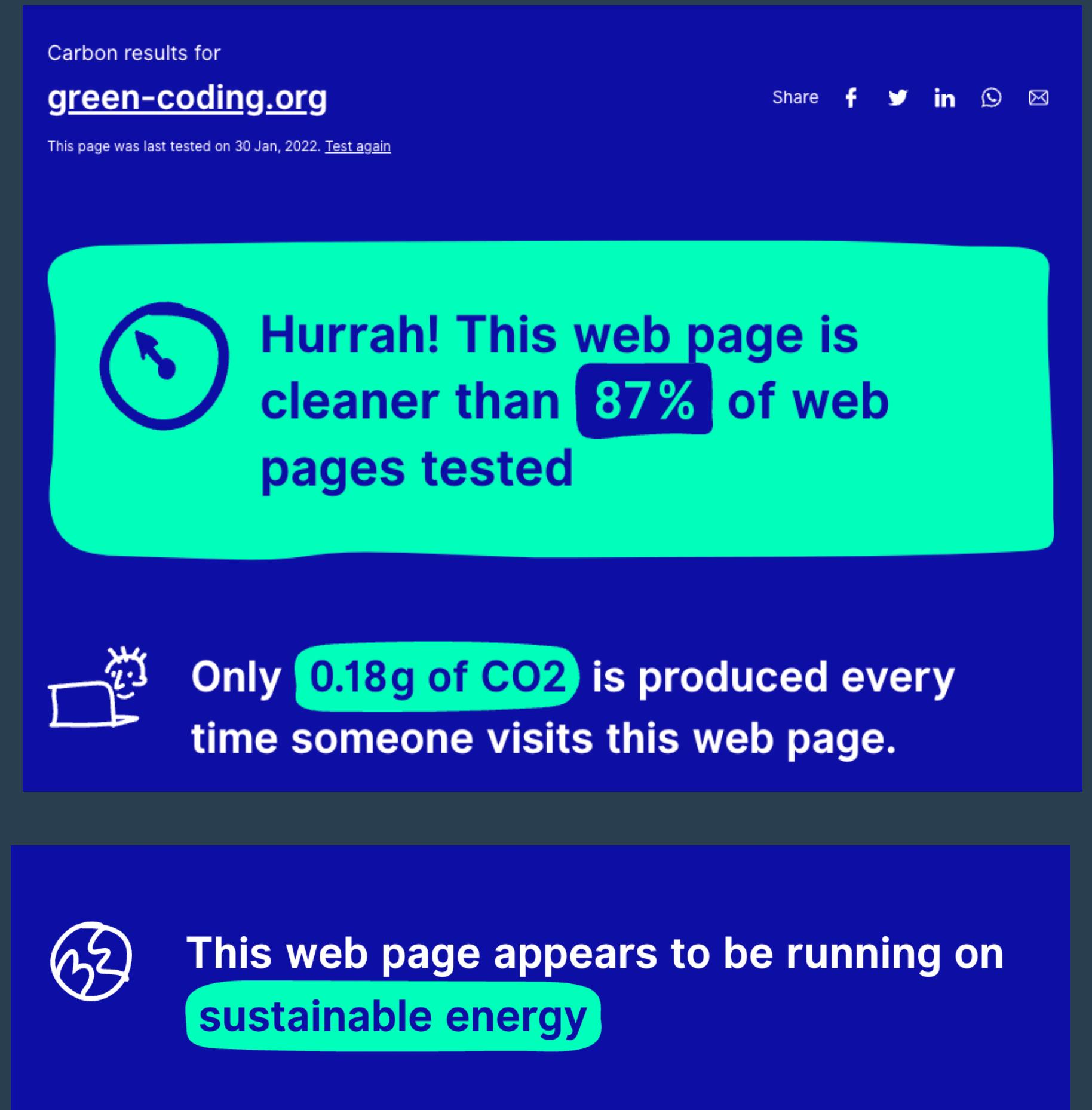
- Basierend auf Empfehlung aus Wissenschaft und französischem Positionspapier Numérique Responsable
- Kein nachgewiesener Effekt in größeren Software-Projekten
- Empfehlungen wie `i++` statt `++i` sogar im Assembly Code / Byte-Code oft identisch
- Overhead der Tools unklar



Deep dive into one label: [websitecarbon.com](https://www.websitecarbon.com)

Tool to get the carbon footprint estimate of a website

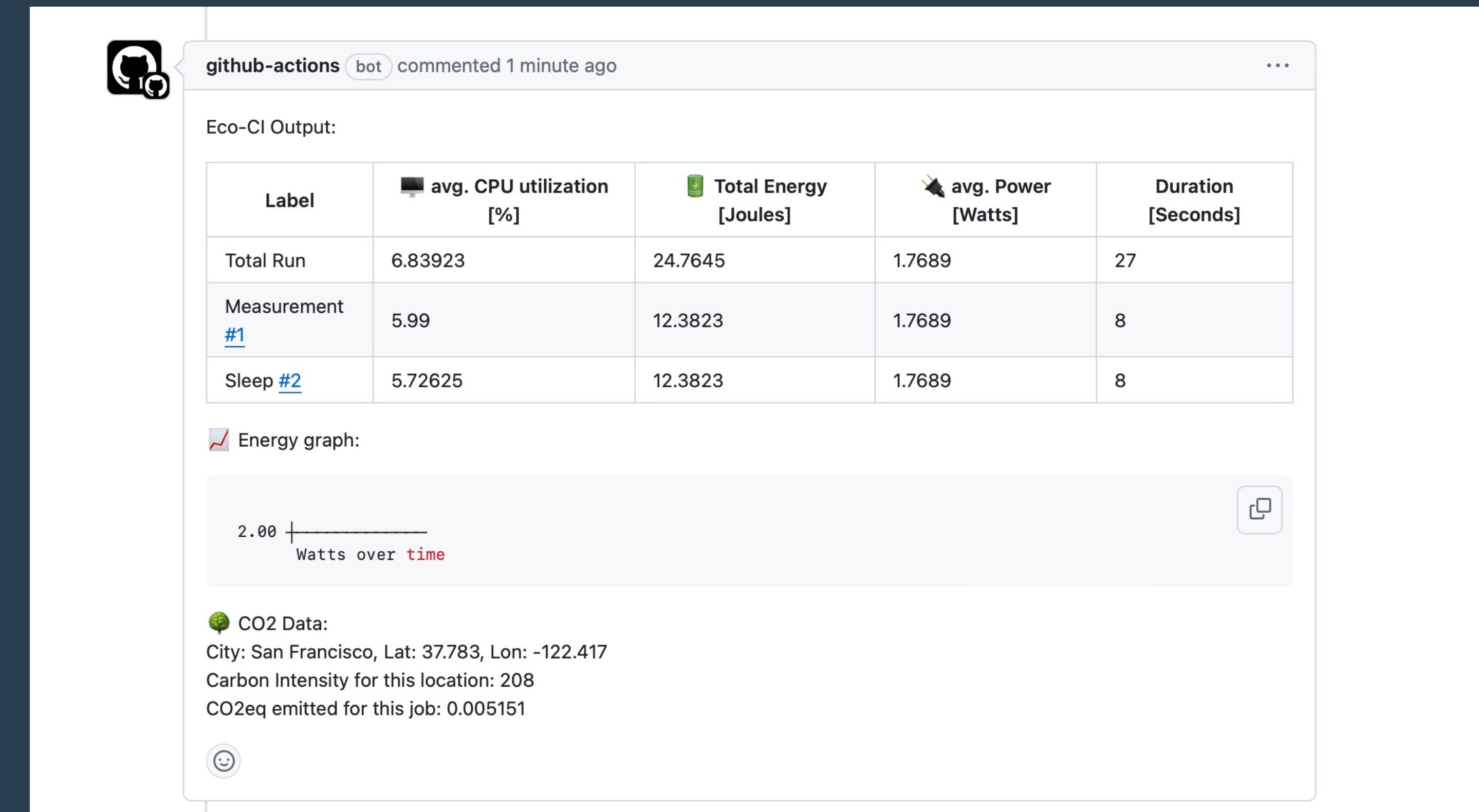
- Very visual, good usability, but also very basic
- Entering the URL is sufficient
- Unfortunately, there are many calculators and every one provides very different values
- Most only use network traffic (not packets)
- Some add constants for end device screen time or server costs
- Good for communicating that you care. Not so much for comparisons / reporting



Eco-CI

Estimation in CI / CD Pipelines

- Integration into Github / Gitlab directly
- Export to central dashboard
- Statistical comparisons over time
- PR-triggered measurements



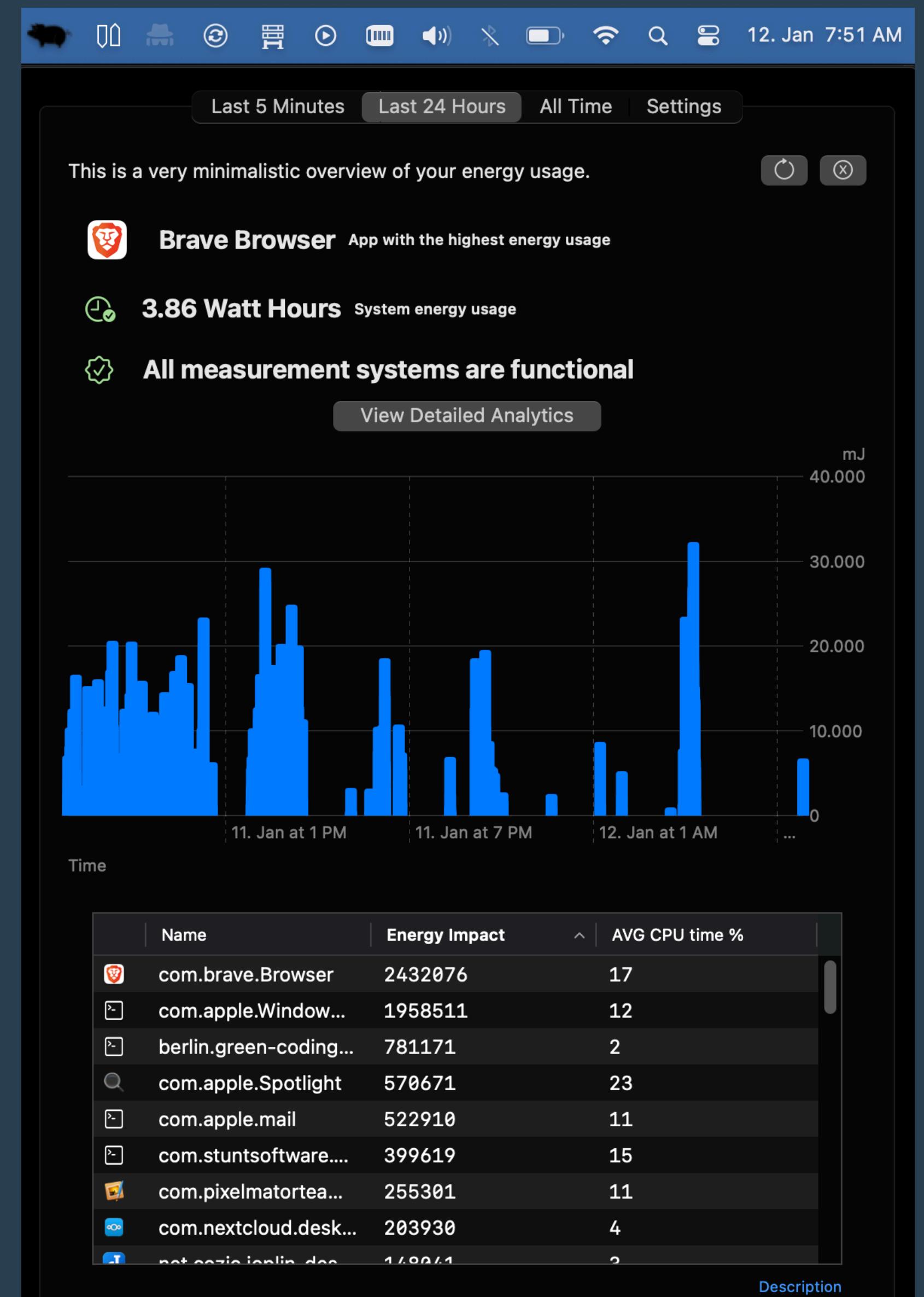
Example for Github PRs: <https://github.com/green-coding-berlin/green-metrics-tool/pull/653>

Example for Django Measurements over time: <https://metrics.green-coding.io/ci.html?repo=green-coding-berlin/django&branch=main&workflow=60545070>

Power Hog

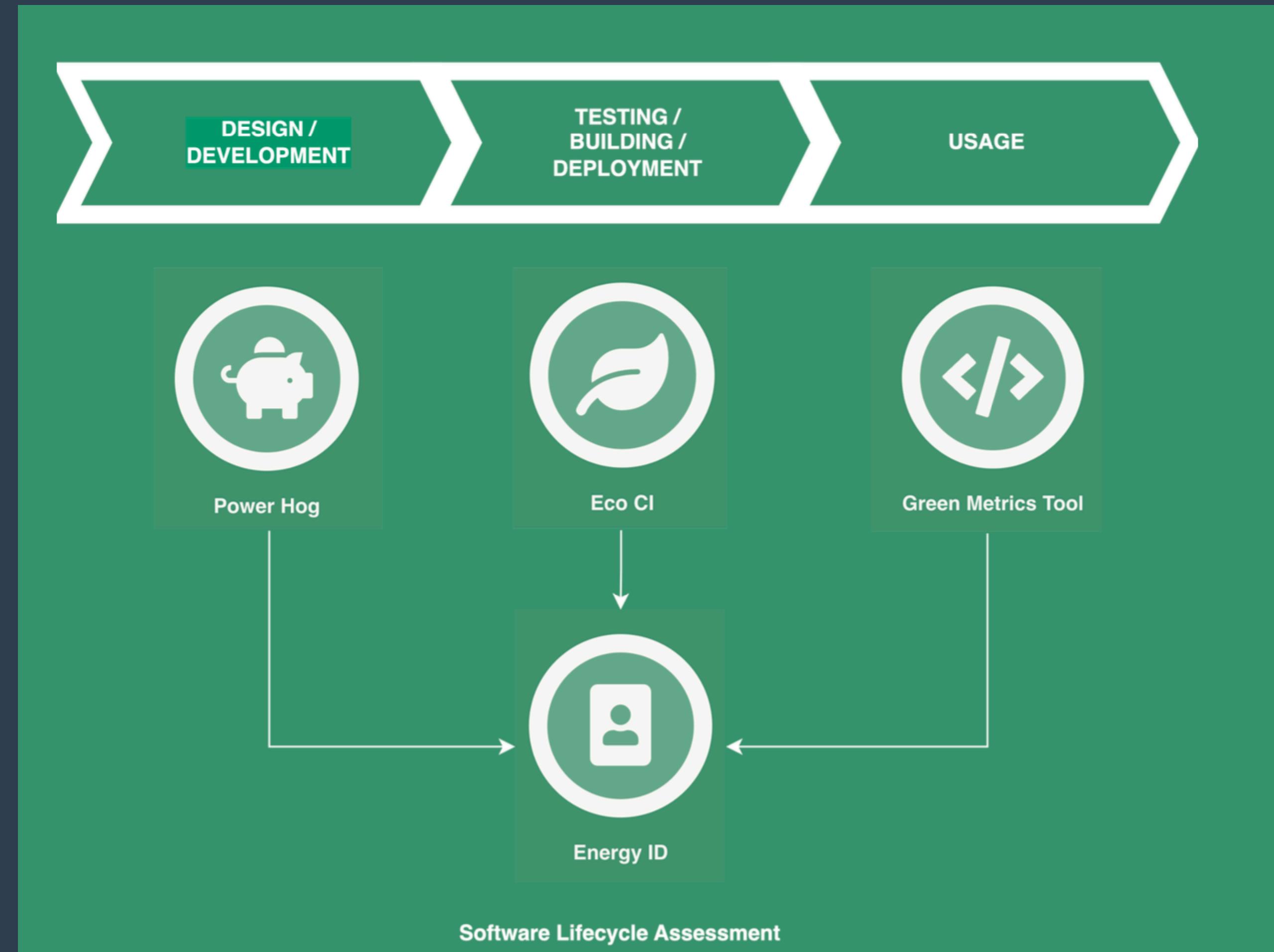
Development Cost measurement

- Direct measurement of energy on the developer machine
- Granularity per process
- Aggregation per project
- API and drill-down option in dashboard
- Local database for quick analyses



Software Lifecycle Assessment

Combines all parts to a holistic perspective



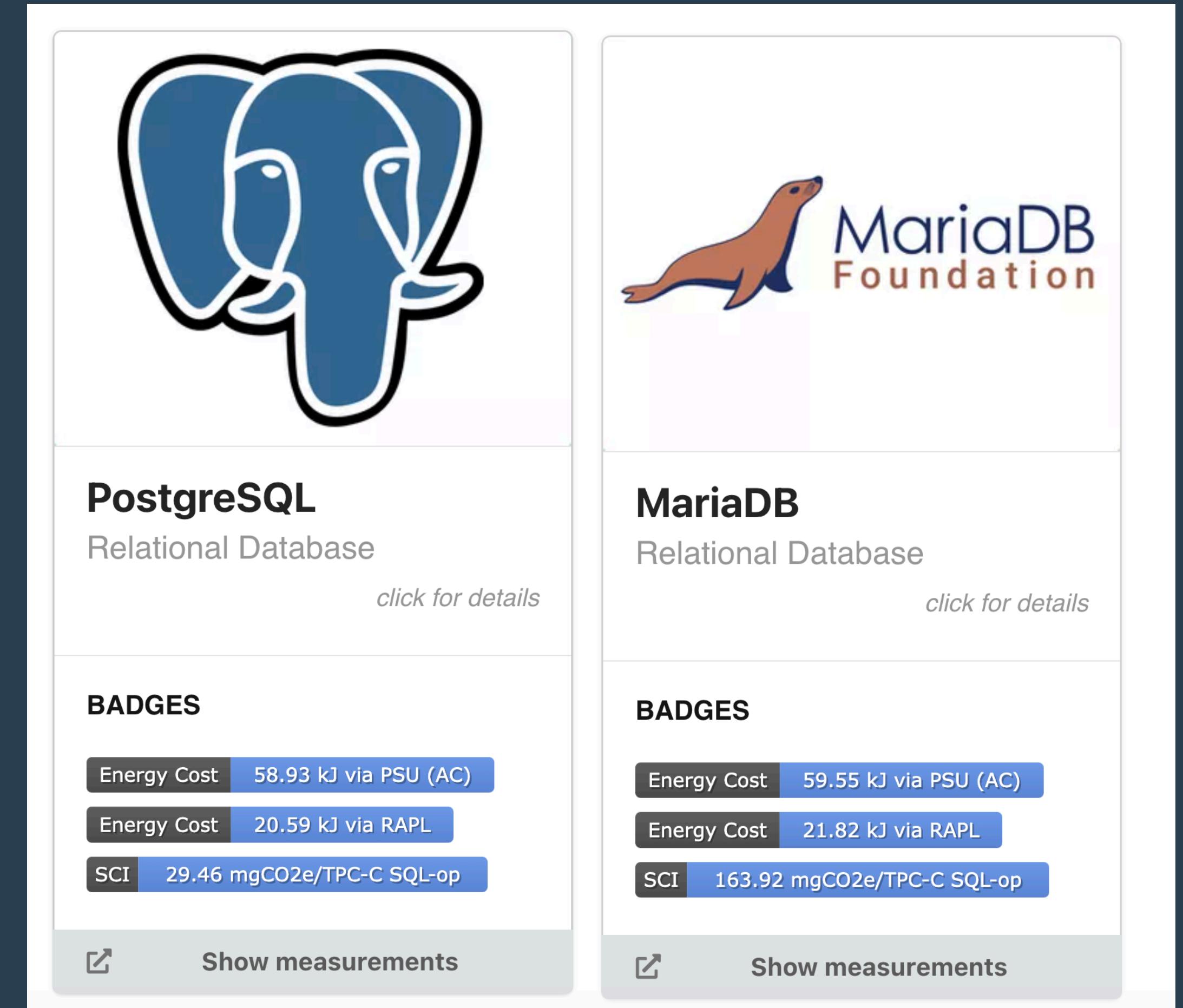
Potentials and levers

In saving and mitigation

Potentials in dependencies / components

Quantifying dependencies and toolchains in terms of carbon

- Software carbon and energy transparency
- Different software products for same use-case can have vastly different energy and carbon profiles
- Levers:
 - Library / Dependency selection
 - Default software recommendations



Potentials in Architectures

My website is already loading below 1s and very efficient

- Fast ≠ efficient
 - Although a website might be fine in Google Pagespeed saving potentials are still enormous.
 - Case Study: **Wordpress vs. HUGO**
 - Saving is 5x per Request
 - You can rebuild the page 5x for EVERY VISITOR and still would only break even.
 - On a small startup website (100.000 Page Views / month) this saving is already 2.4 kg!
 - **Take away:** Perceived effectiveness is not carbon effectiveness



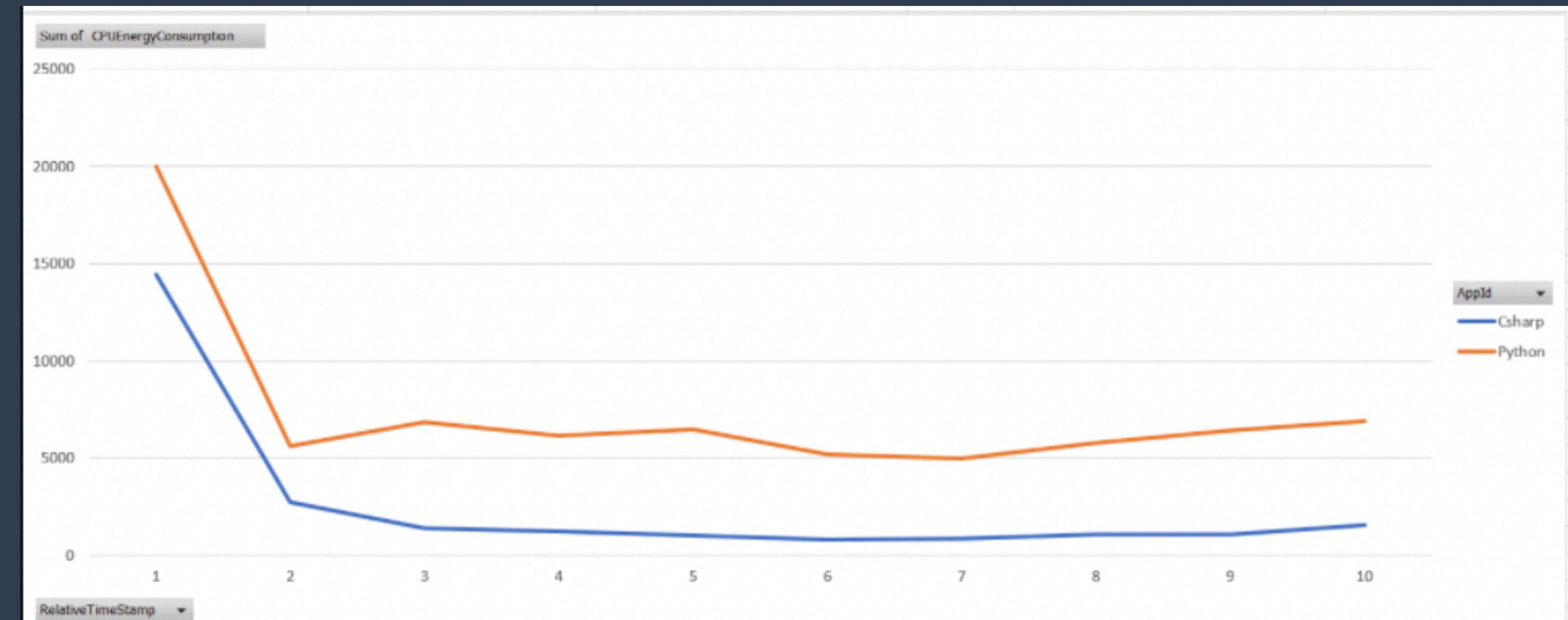
<https://www.green-coding.berlin/case-studies/wordpress-vs-hugo-cloudflare/>

Programming language potentials

- 75x in worst case!

Total				
	Energy	Time	Mb	
(c) C	1.00	1.00	(c) Pascal	1.00
(c) Rust	1.03	1.04	(c) Go	1.05
(c) C++	1.34	1.56	(c) C	1.17
(c) Ada	1.70	1.85	(c) Fortran	1.24
(v) Java	1.98	1.89	(c) C++	1.34
(c) Pascal	2.14	2.14	(c) Ada	1.47
(c) Chapel	2.18	2.83	(c) Rust	1.54
(v) Lisp	2.27	3.02	(v) Lisp	1.92
(c) Ocaml	2.40	3.09	(c) Haskell	2.45
(c) Fortran	2.52	3.14	(i) PHP	2.57
(c) Swift	2.79	3.40	(c) Swift	2.71
(c) Haskell	3.10	3.55	(i) Python	2.80
(v) C#	3.14	4.20	(c) Ocaml	2.82
(c) Go	3.23	4.20	(v) C#	2.85
(i) Dart	3.83	6.30	(i) Hack	3.34
(v) F#	4.13	6.52	(v) Racket	3.52
(i) JavaScript	4.45	6.67	(i) Ruby	3.97
(v) Racket	7.91	11.27	(c) Chapel	4.00
(i) TypeScript	21.50	26.99	(v) F#	4.25
(i) Hack	24.02	27.64	(i) JavaScript	4.59
(i) PHP	29.30	36.71	(i) TypeScript	4.69
(v) Erlang	42.23	43.44	(i) Ruby	6.01
(i) Lua	45.98	46.20	(i) Perl	6.62
(i) Jruby	46.54	59.34	(i) Lua	6.72
(i) Ruby	69.91	65.79	(v) Erlang	7.20
(i) Python	75.88	71.90	(i) Dart	8.64
(i) Perl	79.58	82.91	(i) Jruby	19.84

[17] Greenlab Study

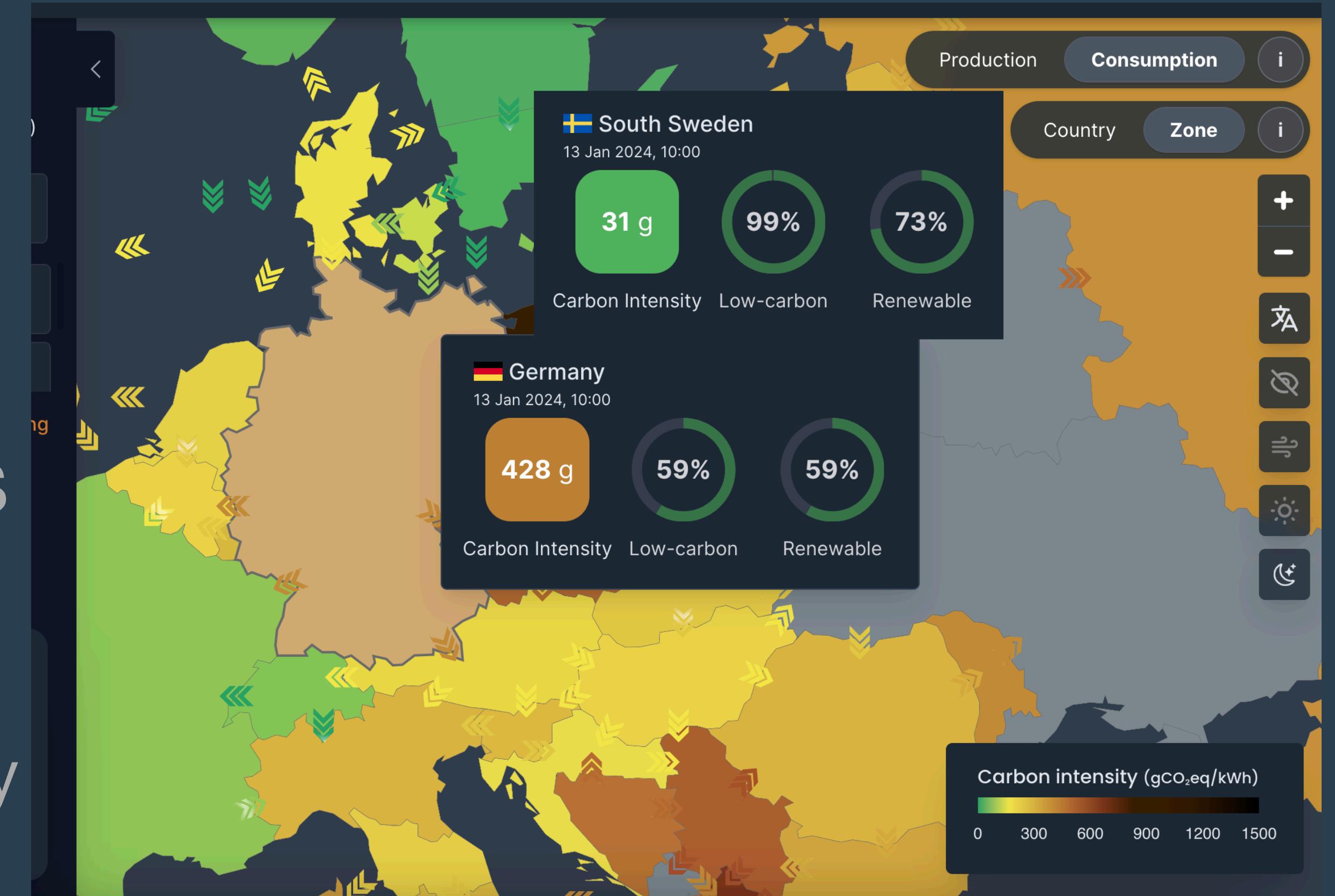


[18] Microsoft article on energy use of programming languages
1.8x - 3x difference

Potentials of Location Shifting

Example: Europe

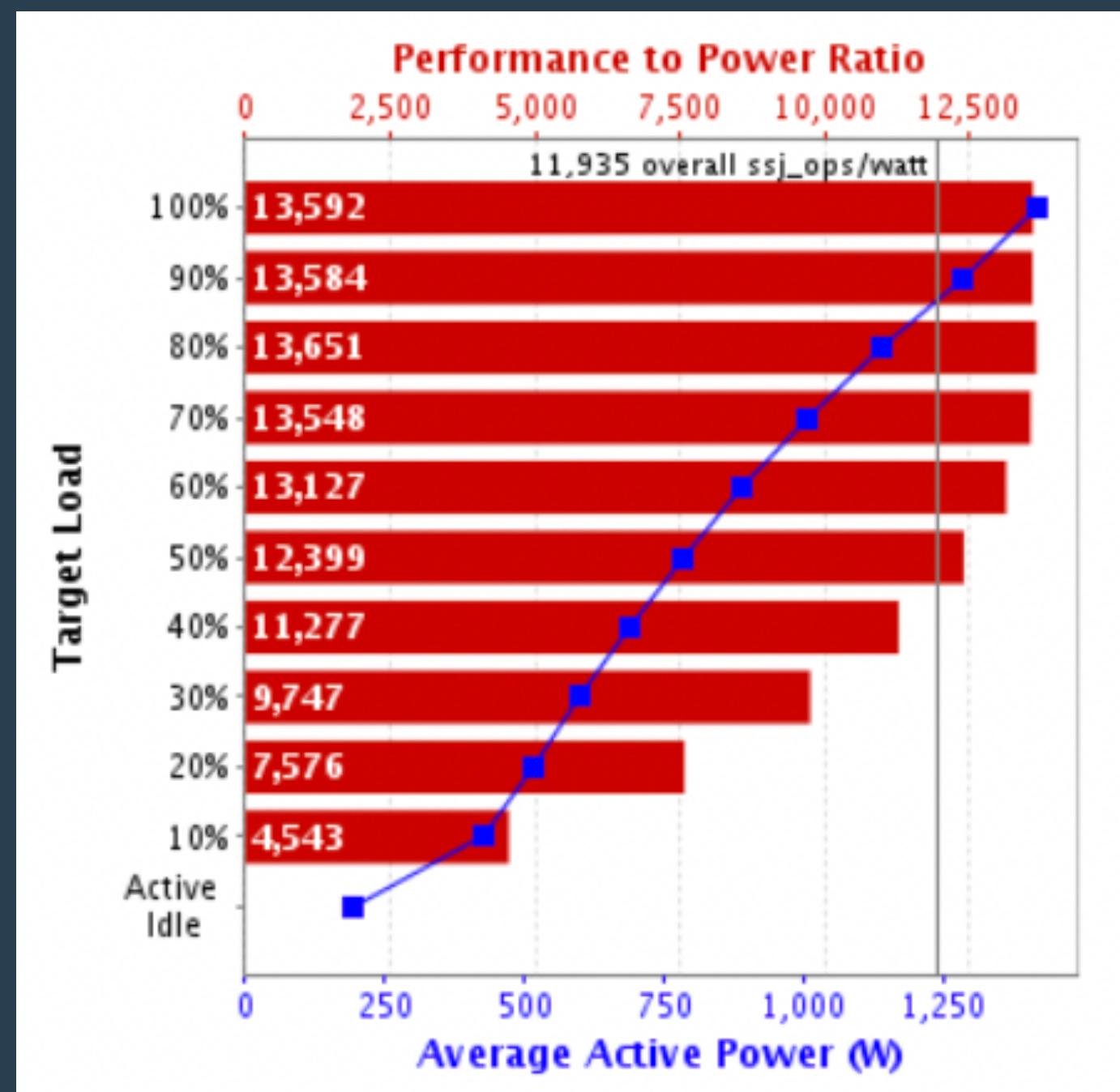
- Moving a server to a low carbon region can save CO₂
- Sweden / Germany is > 10x at times
- Moving workloads inside europe / US very easy due to low compliance risk
- Moving workload internationally tricky



Potentials of different hardware configuration

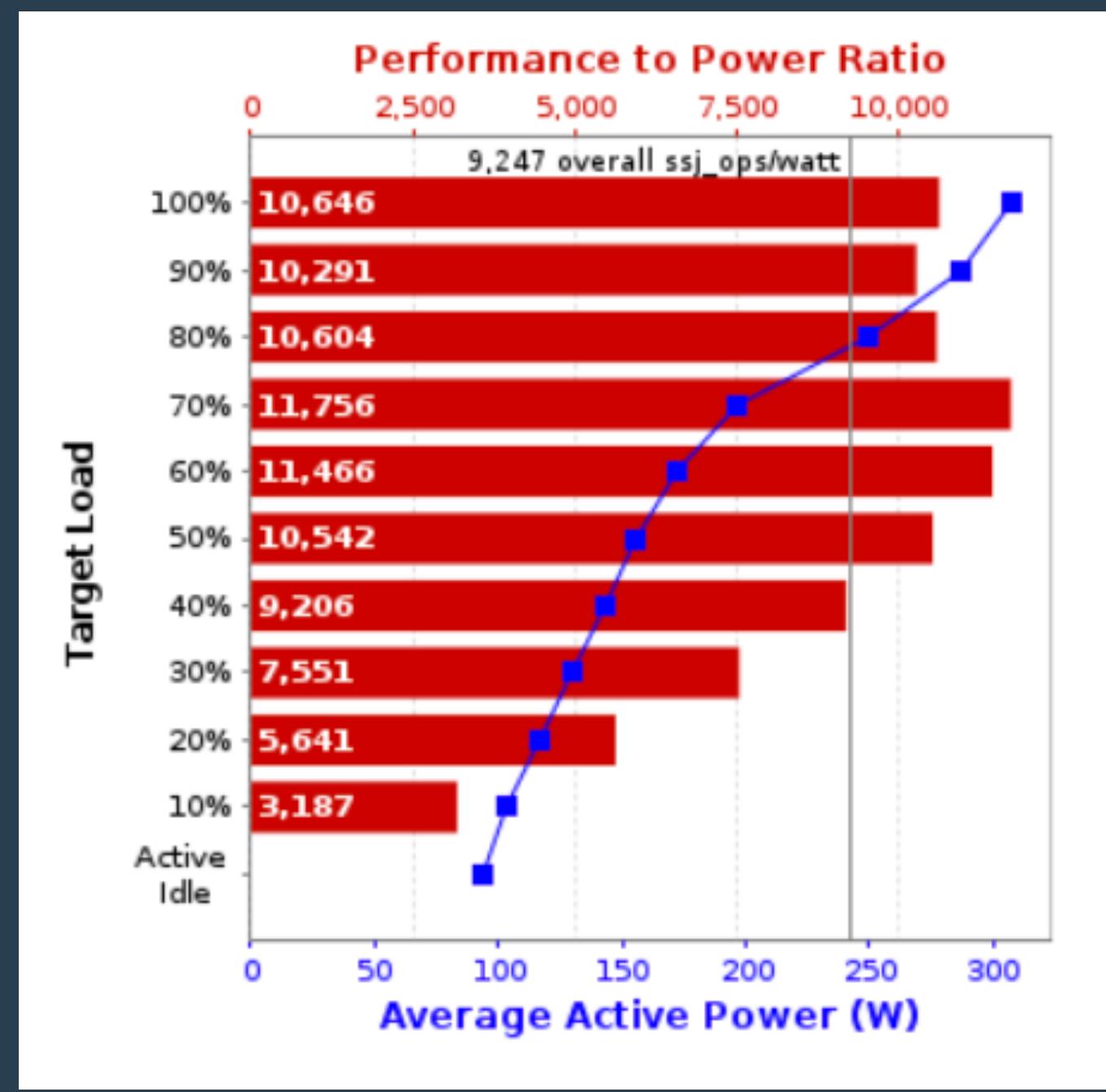
Power behaviour of hardware and how to leverage non-linearity. Cloud and non-cloud!

High Idle, but almost linear



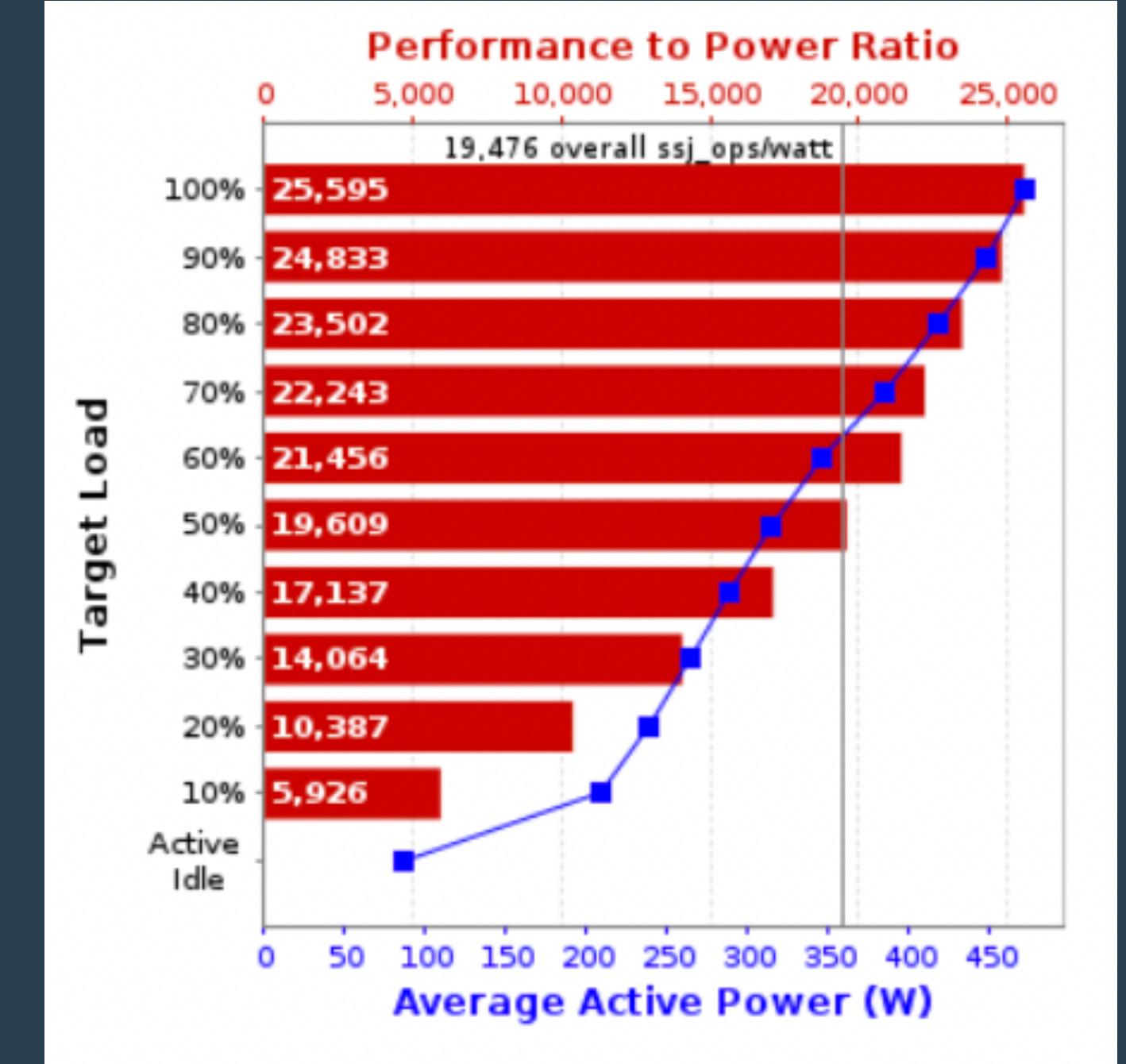
ASUSTeK Computer Inc. RS720Q-E9-RS8
(2019)

50% Power increase at 70% utilization



Hewlett Packard Enterprise ProLiant DL110
Gen10 Plus

Idle optimized



QuantaGrid D43K-1U (2022)

Summary

Green Coding is a software technique FOR software

- Efficiency gains through performance engineering
- Making hardware last longer through software optimizations
- Using special CO₂ reduction techniques through using green energy wherever possible
 - load-shifting
 - time-shifting
 - Demand-shaping
- Reduction of compute / data storage / network / complexity

Thank you

Want to know more

- Website / Blog / Newsletter: <https://www.green-coding.io>
- Demo Open Data Repository: <https://metrics.green-coding.io>
- Unsere Projekte: <https://www.green-coding.io/#projects>
- Unsere Case-Studies: <https://www.green-coding.io/case-studies>
- Meetup Gruppe (Berlin): <https://www.meetup.com/green-coding>
- <https://www.linkedin.com/in/arne-tarara> / arne@green-coding.io

Thank you

Want to know more

- Website / Blog / Newsletter: <https://www.green-coding.io>
- Give our tool a try:
 - Demo Open Data Repository: <https://metrics.green-coding.io>
 - Github page: <https://github.com/green-coding-solutions/green-metrics-tool>
- Unsere Projekte: <https://www.green-coding.io/#projects>
- Unsere Case-Studies: <https://www.green-coding.io/case-studies>
- Meetup Gruppe (Berlin): <https://www.meetup.com/green-coding>
- <https://www.linkedin.com/in/arne-tarara> / arne@green-coding.io
- Interested in more content and networking -> Join our conference!

Thank you

Want to know more



EcoCompute 2024

25-26 April - Munich

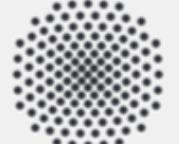
The first engineering conference on sustainability in hardware & software.

www.eco-compute.io

Thank you

Want to know more

Line-up

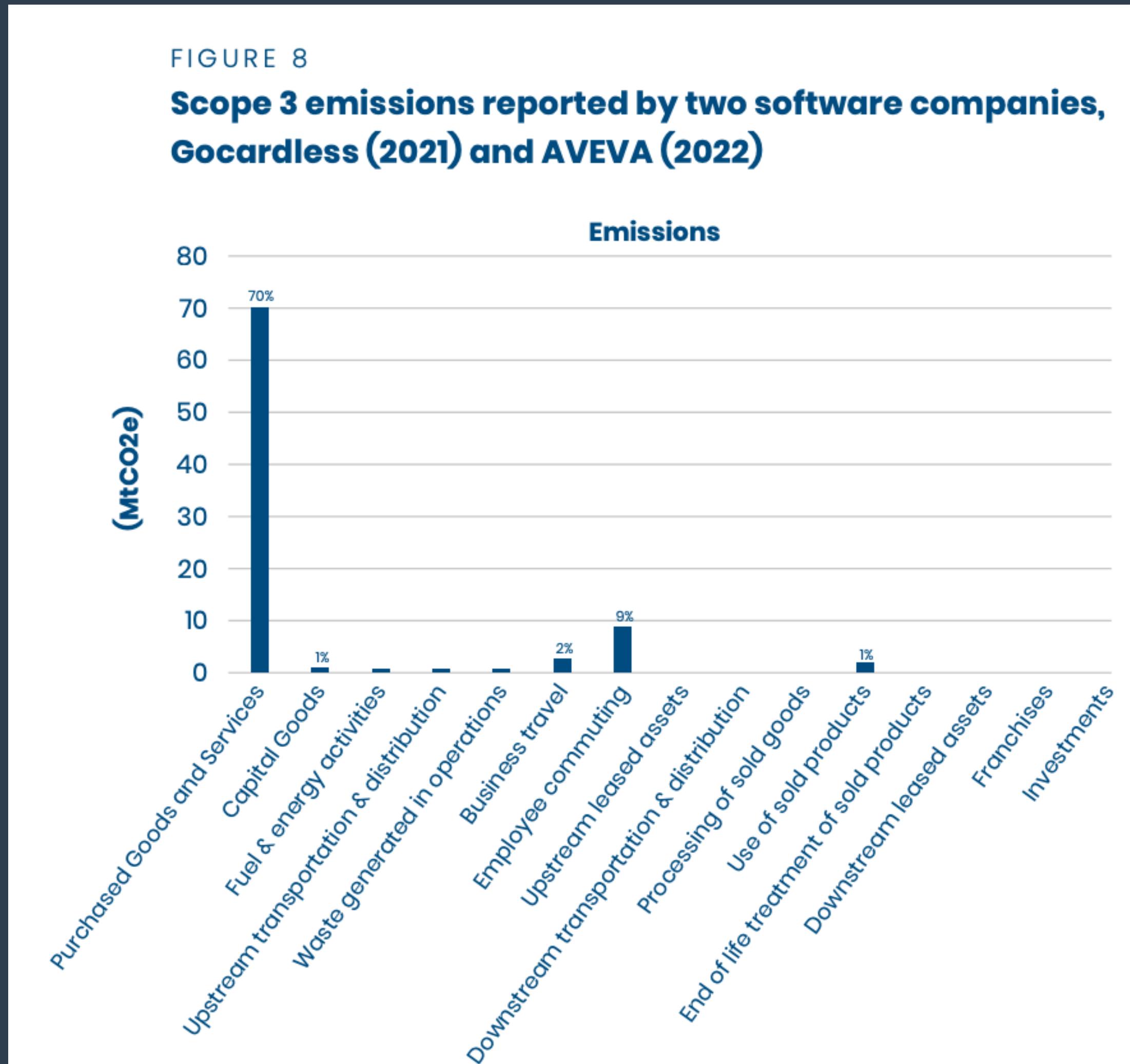
 Hagen Pfeifer 	 Arne Tarara 	 Sasche Böhme 	 Didi Hoffmann 	 Yelle Lieder 	 David Kopp 	 Max Körbächer 
 Anna Zagorski 	 Max Schulze 	 Verena Majuntke 	 Andreas Brunnert 	 Petteri Kivimäki 	 Anita Schüttler 	 Ghazal Aakel & Eric Jochum 

Backup

Potentials For industries

Distribution of scope emissions

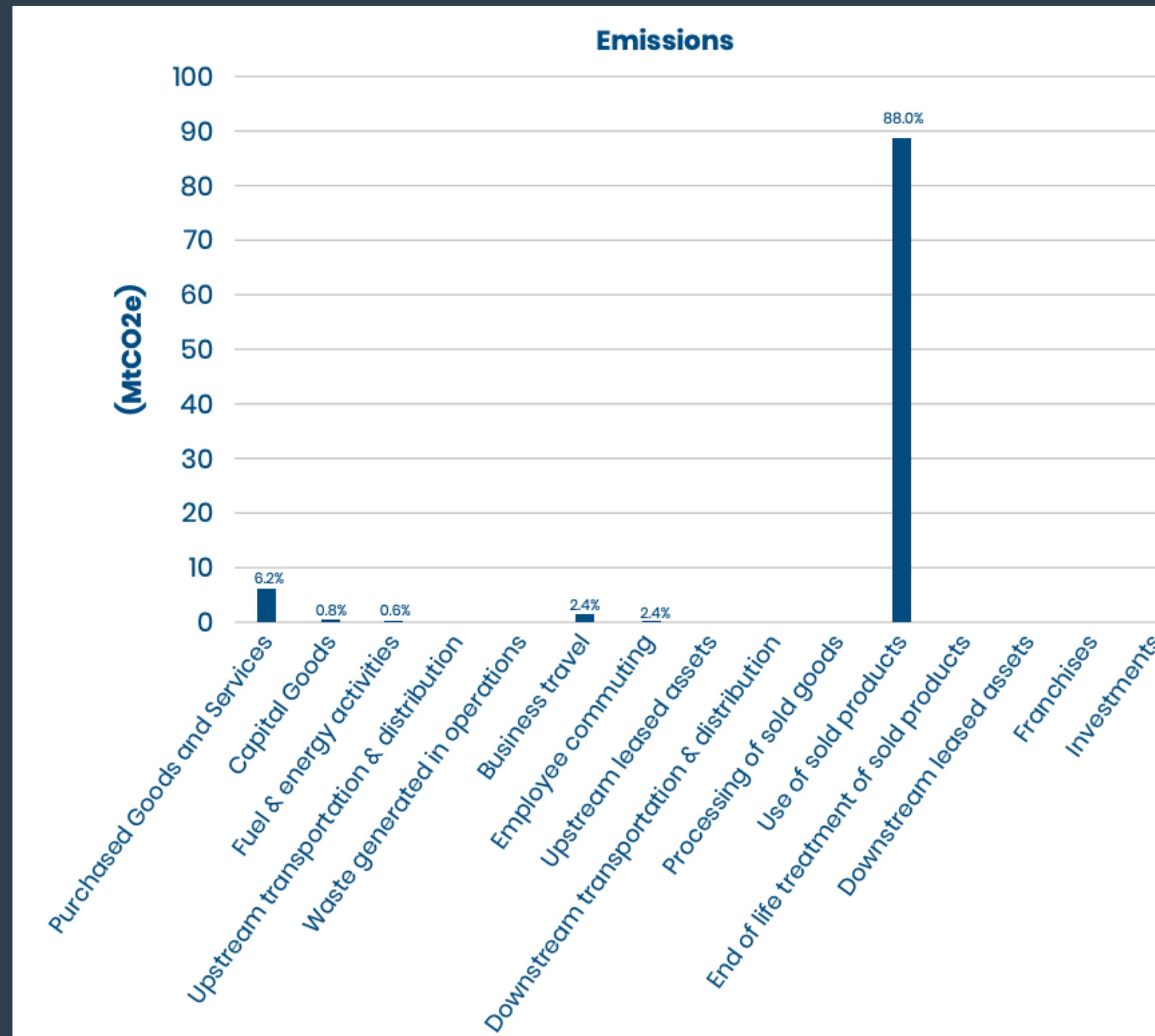
Example GoCardless



- GoCardless is an online payment processor
- If you are a SaaS company your emissions are mostly Scope 3 upstream
- They originate from buying cloud services

Distribution of scope emissions

Example AVEVA



- AVEVA is an industrial software company.
- If you are an software integrator your emissions are mostly Scope 3 downstream
- They originate through clients using your software in production

ABN Amro - Datacenter emissions

Bank from the Netherlands - 11.45 Billion EUR market cap.

- Datacenters and SaaS account for 11.89 kTon CO₂
- ~30% of Scope 3
- ~25% of total CO_{2e}

3.1 Summary of carbon emissions within ABN AMRO scope			
Reported kton GHG emissions by region	NL	RoW*	2022 Total
Scope 1 Tank-to-Wheel (TTW)			
Energy (natural gas + solar PV + other)	-	0,52	0,52
Mobility (lease cars - internal combustion engine)	2,18	-	2,18
Total scope 1	2,18	0,52	2,70
Scope 2 Tank-to-Wheel (TTW)**			
Energy (electricity, heating and cooling)	0,65	2,28	2,93
Mobility (lease - electric vehicles)	0	-	-
Total scope 2	0,65	2,28	2,93
Total scopes 1 + 2	2,83	2,80	5,63
Note: Decrease in scope 1 & 2 GHG emissions is mainly due to Covid-19 effects.			
Scope 3 Tank-to-Wheel (TTW) Own Operations			
Air travel, international business rail travel and hotel visits	NL	RoW*	2022 Total
Mobility (commuting in private vehicles and business travel)	1,93	1,37	3,30
Public transport	5,28	-	5,28
Home workplace	0,55	-	0,55
Off-premise datacenters + Software-as-a-Service	9,00	1,74	10,74
Total scope 3 TTW	11,89	-	11,89
	28,65	3,11	31,77
Scope 3 Well-to-Tank (WTT) Own Operations			
Energy (electricity, gas, heating and cooling)	NL	RoW*	2022 Total
Home office (Netherlands)	0,67	-	0,67
Mobility (lease cars - internal combustion engine)	2,64	-	2,64
Mobility (lease - electric vehicles)	0,66	-	0,66
Private vehicles	2,94	-	2,94
Public transport	1,73	-	1,73
Total scope 3 WTT	0,21	-	0,21
	8,85	-	8,85
Total Scope 3 emissions - Own operations			
	37,51	3,11	40,62

Source: <https://www.abnamro.com/en/information/sustainability-reporting-and-publications?selectedTabs=Reporting>

Allianz

Insurer - Germany - 93.73 Billion EUR market cap.

- Datacenters account for 325,000 TJ
- => 90 GWh
- => 45 kTons CO2e
- => 16% of total energy consumption
- Total CO2 not disclosed
- ~ 4 times the CO2 / 8-times the market cap

Table ENV-3
Energy consumption

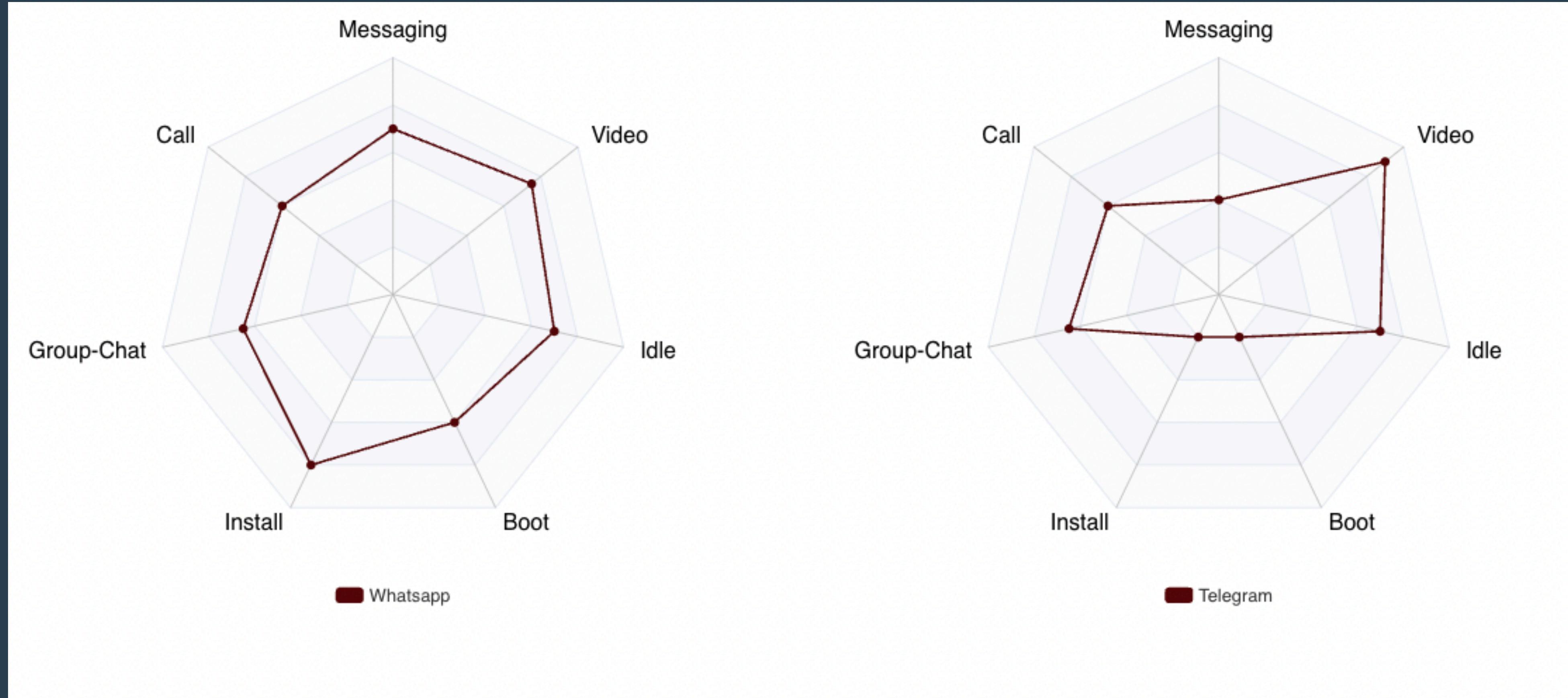
As of December 31		2022
Energy consumption from our office buildings	GJ	1,578,675
Energy consumption from our data centers	GJ	325,212
Total energy consumption¹	GJ	1,965,294

Source: https://www.allianz.com/content/dam/onemarketing/azcom/Allianz_com/sustainability/documents/Allianz_Group_Sustainability_Report_2022-web.pdf

Potentials of different application classification

By looking at software via usage scenario

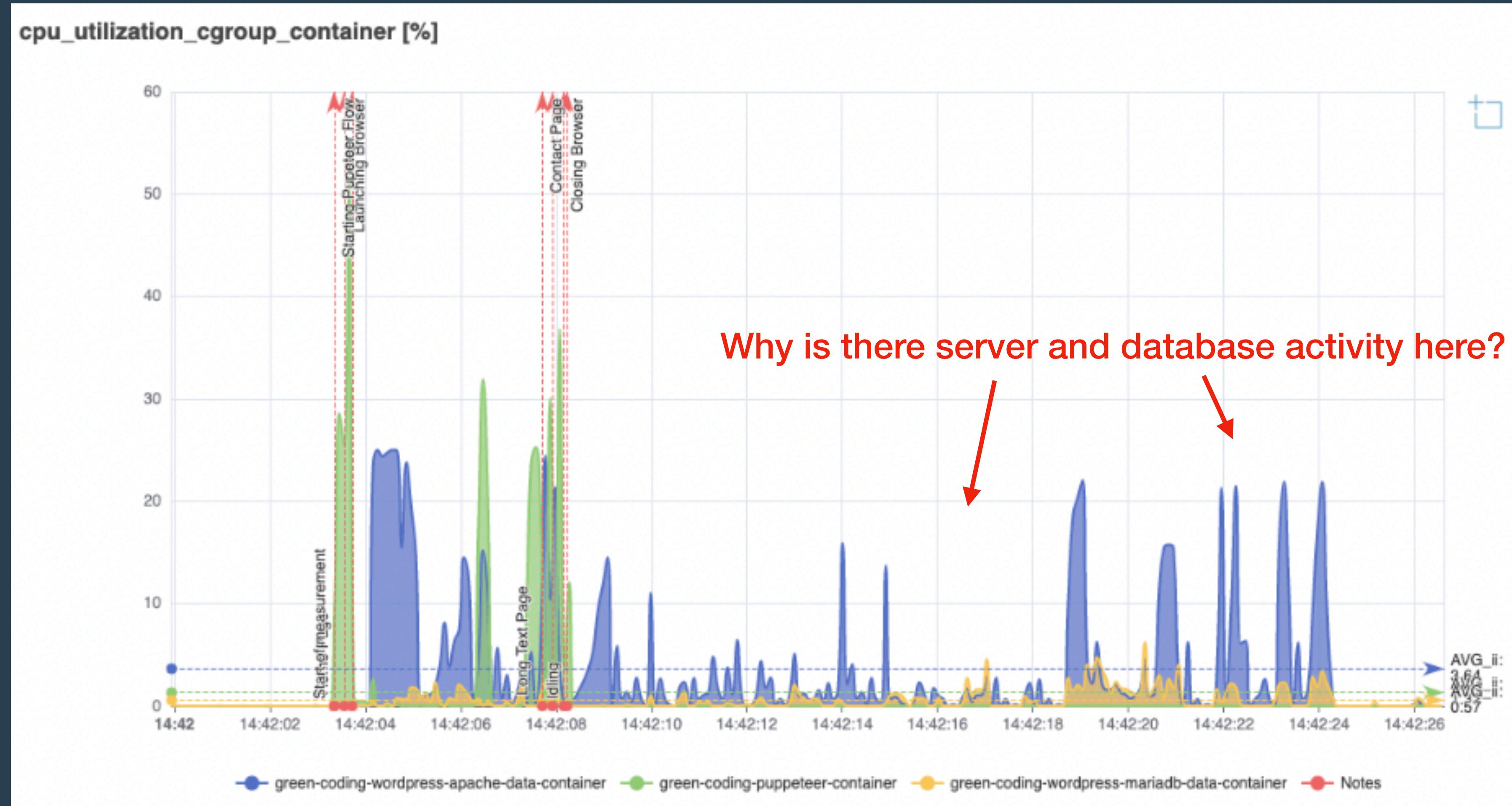
No actual data! Concept picture!



Energy consumption of Whatsapp vs. Telegram - per usage scenario

Potentials of simple energy bugs

In background application measurement data



The Green Web Foundation

Green Energy Label for Datacenters / Servers

- The Green Web Foundation is pursuing and accelerating the transition to a fossil-free Internet
- Provides datasets/API for green hosting
- Main topics:
 - climate justice
 - green energy



Green Software Foundation

NGO for the sustainability of the digital sector



- UK founded NGO
- The goal is to reduce the CO2 emissions of software
- Main work is standardization and reference implementations
 - Software Carbon Intensity (ISO Standard)
 - Carbon Aware SDK
 - SCER - Software Carbon Efficiency Rating

SDIA

NGO for sustainability in the digital sector

- Association for stakeholders in the digital industry
 - datacenters / software vendors /
 - hardware manufacturers etc.
- Main topics:
 - Digital carbon footprint
 - Consume renewable energy
 - Increase utilization of hardware
 - Leaner Architecture of software
 - Extend hardware lifecycle



Open Source potentials

Due to scaling effects can be massive even with minimal effort

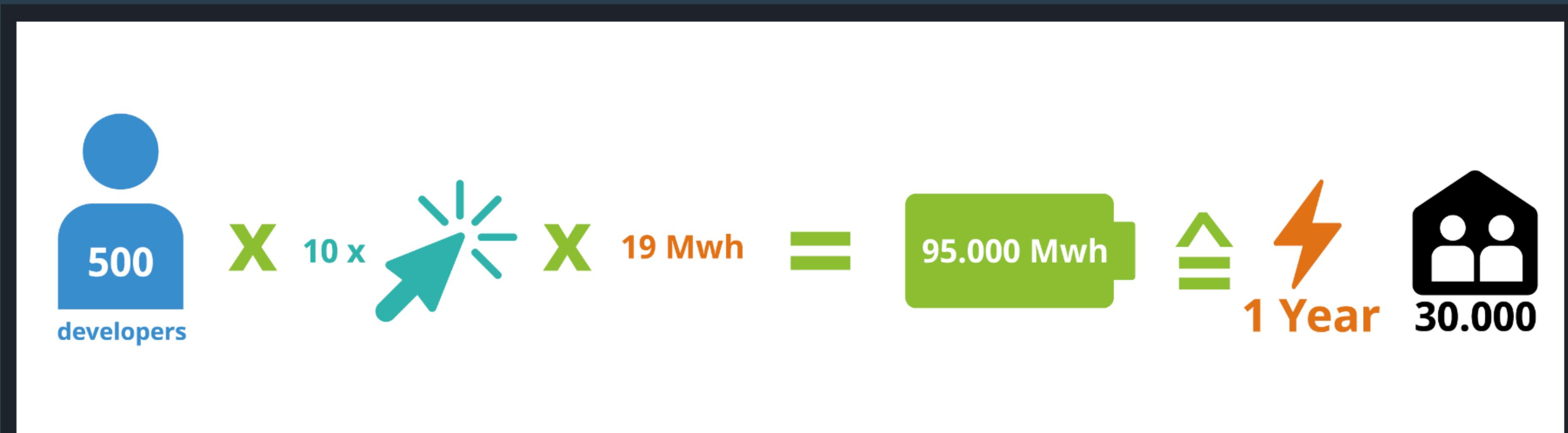
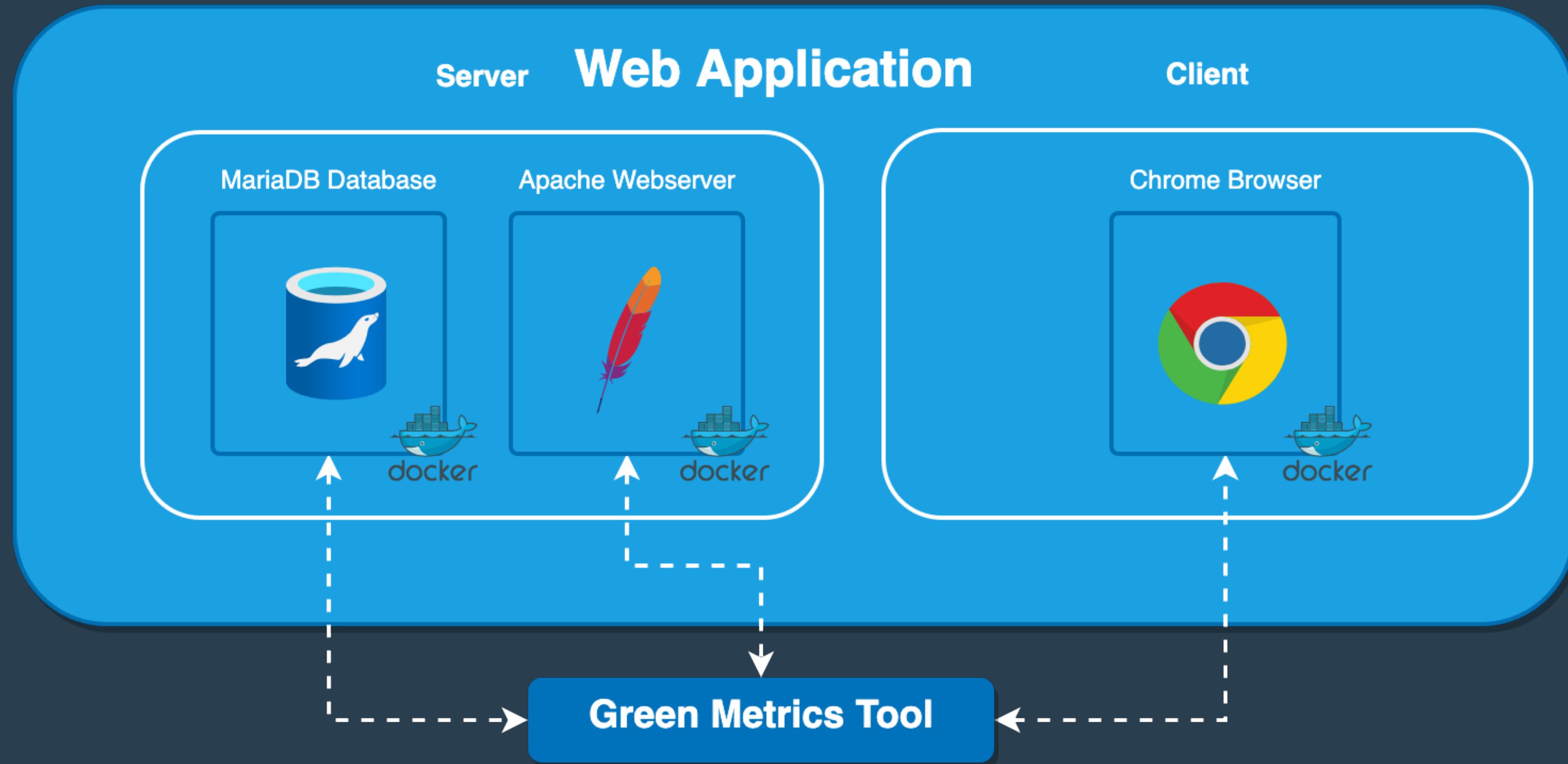


Figure : If 500 developers make 10 one CPU-second reductions, this is equal to 95 thousand megawatt-hours of savings, or the energy consumption of 30-thousand two-person households over one year. (Image from KDE published under a CC-BY-SA-4.0 license. Cursor icon by Alice-vector licensed under a CC-BY license. Example from Detlef Thoms. Design by Lana Lutz.)

Green Metrics Tool

Aufbau einer Applikation für ein Usage Szenario - Client / Server



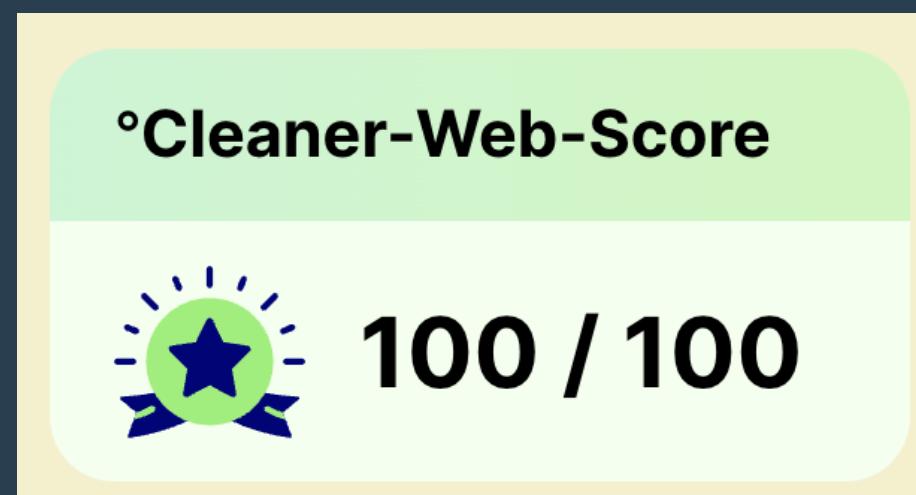
Labels and certifications

An excerpt

Labels for Green Software

Sparse ... only some really high profile / applicable

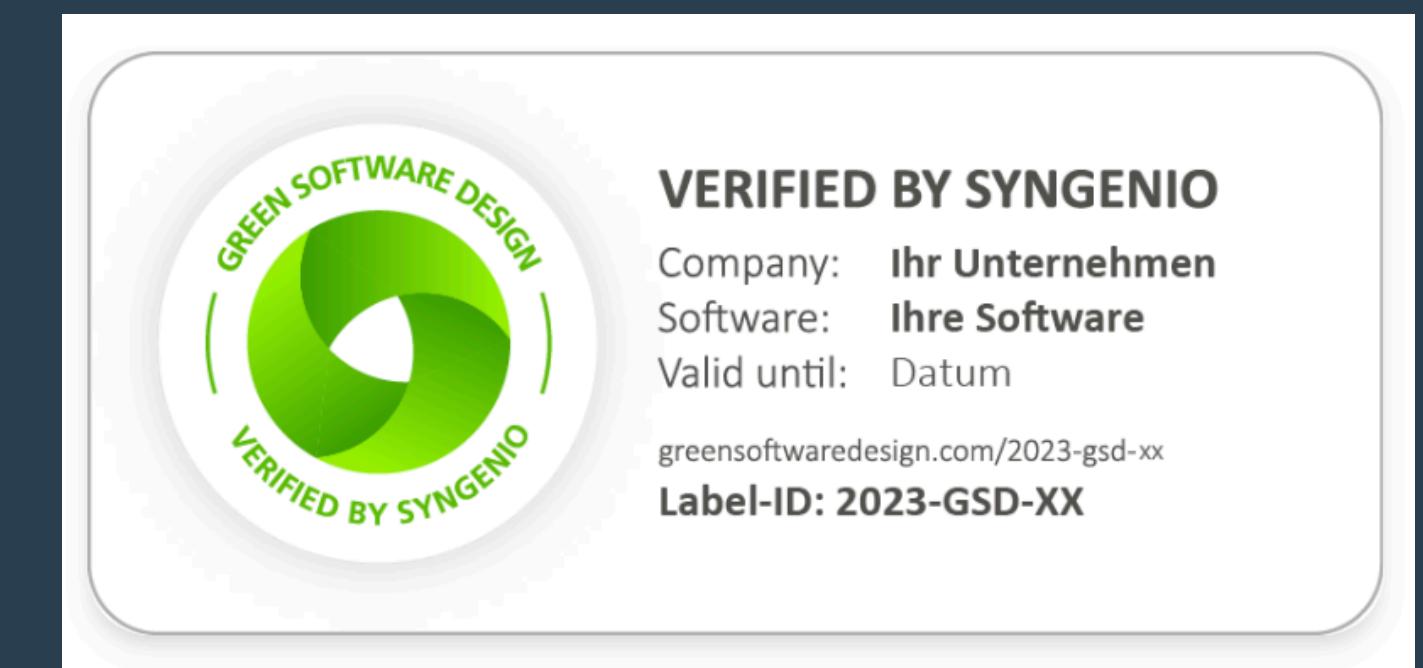
Website



Hosting



Application



**Gut für mich.
Gut für die Umwelt.**

Blue Angel - German Eco Label

Ressource- and energy-efficient software products (DE-UZ 215)

- Reduce hardware obsolescence
- Increase transparency about software energy consumption
- Increase user autonomy
- Increase modularity and software re-use
- No CO2 concerns



**Good for me.
Good for the environment.**

Measurement

How do we measure?

Wie messen wir Energie?

An der Steckdose natürlich, ist doch klar!



Die Amazon Version

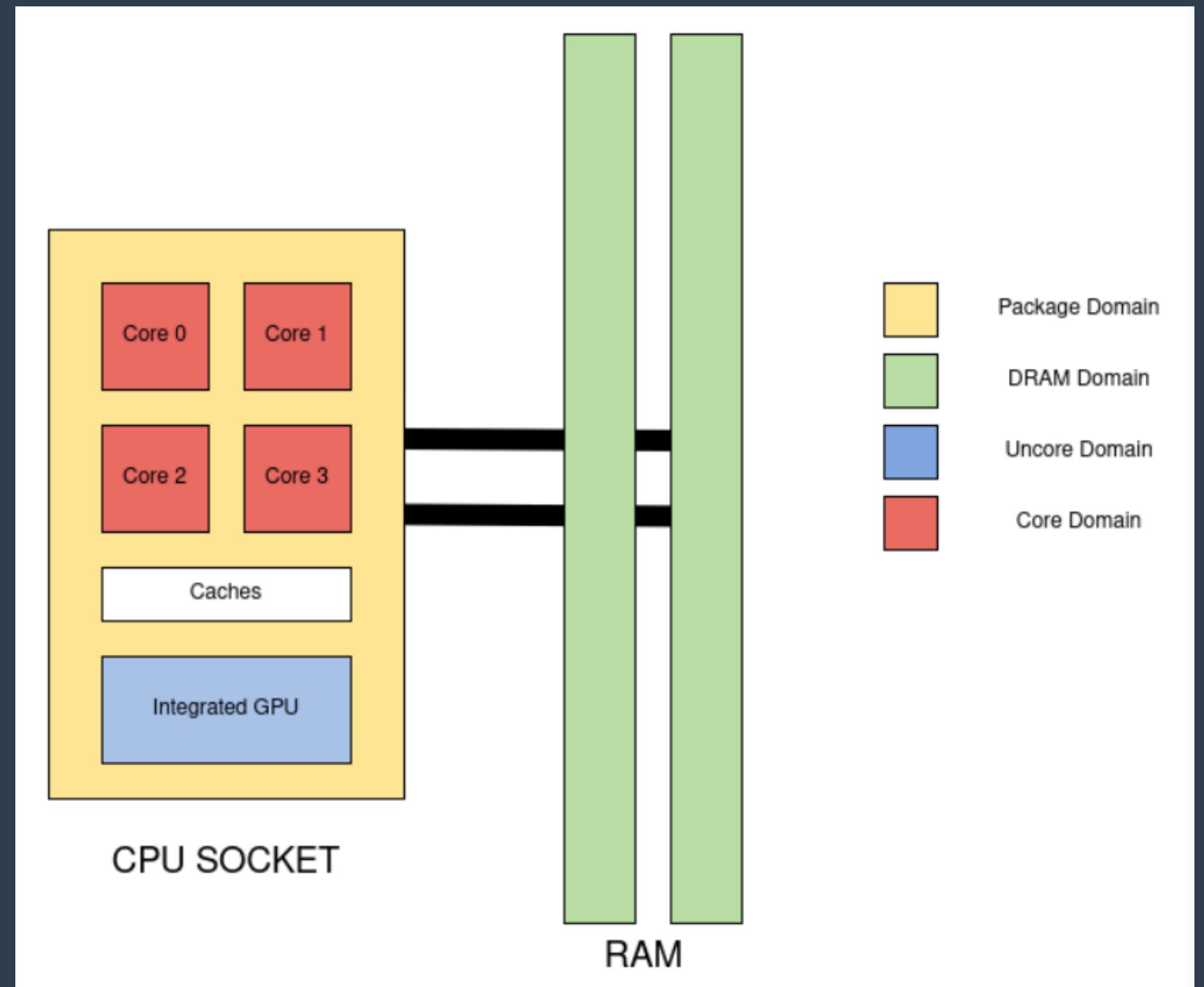


Die Laborversion

Wie messen wir Energie?

On-Board Hardware Interfaces

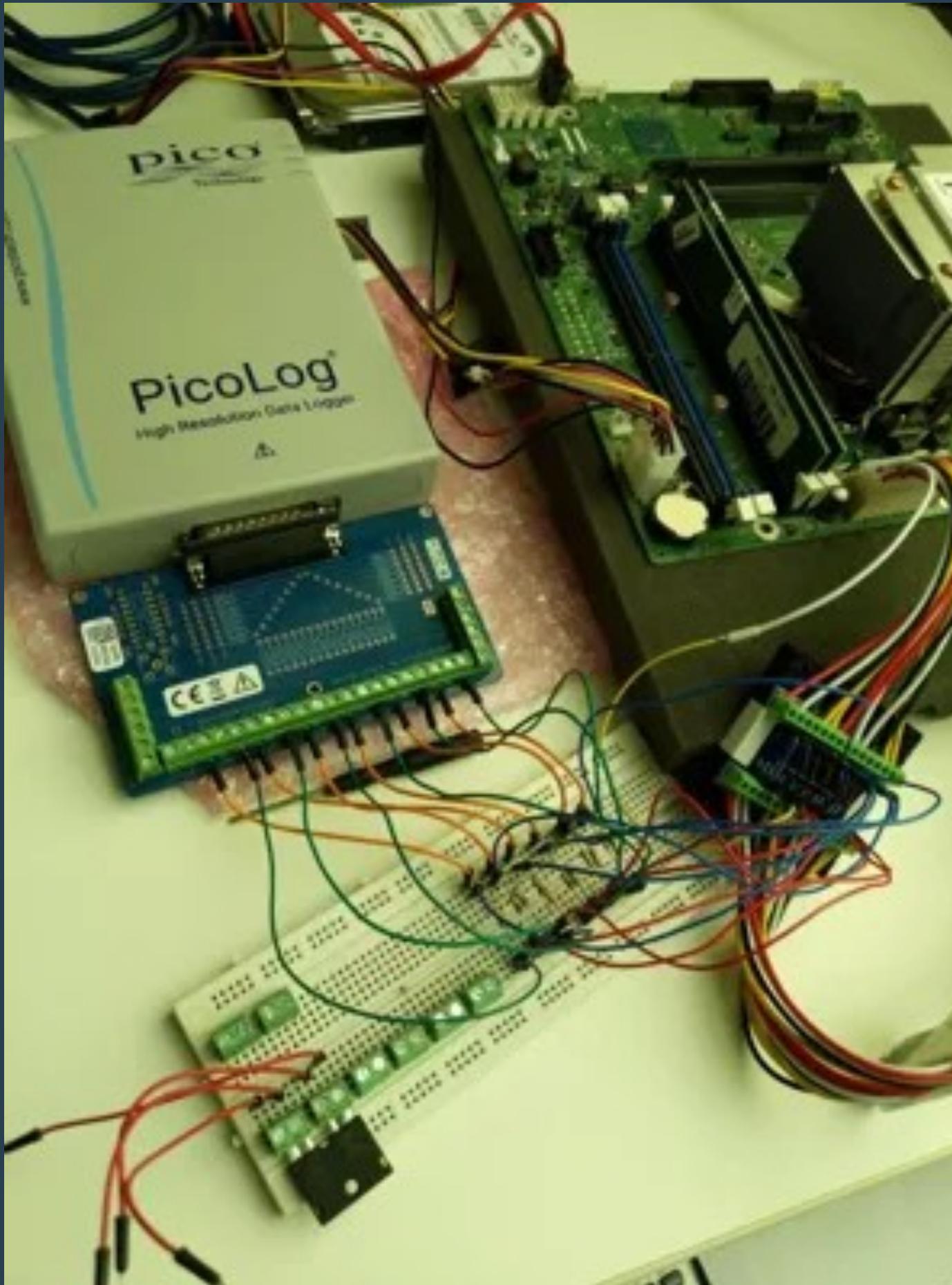
- Energiemessfunktionen für die meisten modernen Intel / AMD Prozessoren
- Messen:
 - CPU-Energie pro Kern/Paket
 - RAM
 - Integrierte GPU
- Software-Modell (proprietär - Spannungsregler etc.), aber mehrfach akademisch validiert
- Auflösung 1ms / 15,3 microJoules
- Bereitgestellt als Linux Kernel-Modul o. MSR



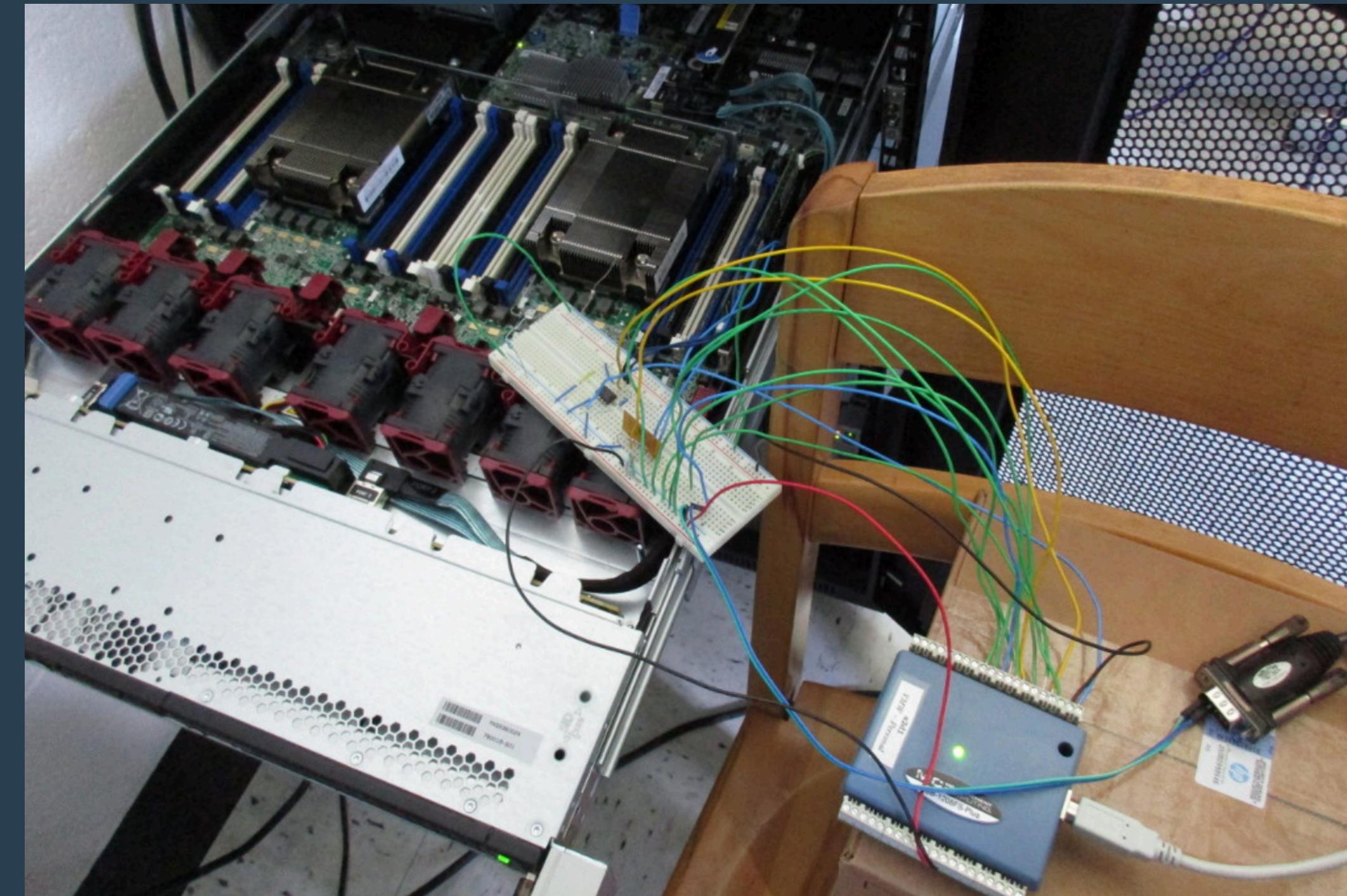
Source: https://pyjoules.readthedocs.io/en/stable/devices/intel_cpu.html

Wie messen wir Energie

Wie weit möchte man gehen?



Auftrennen der ATX-Channels für DC Eingang
ins Mainboard [Green Coding Berlin]



Direktanschluss an Versorgungsleitung im Mainboard
[https://web.eece.maine.edu/~vweaver/projects/rapl/2016_memsys_rapl.pdf]

DC Energie-Messungen

Was wäre der bestmögliche Ansatz?

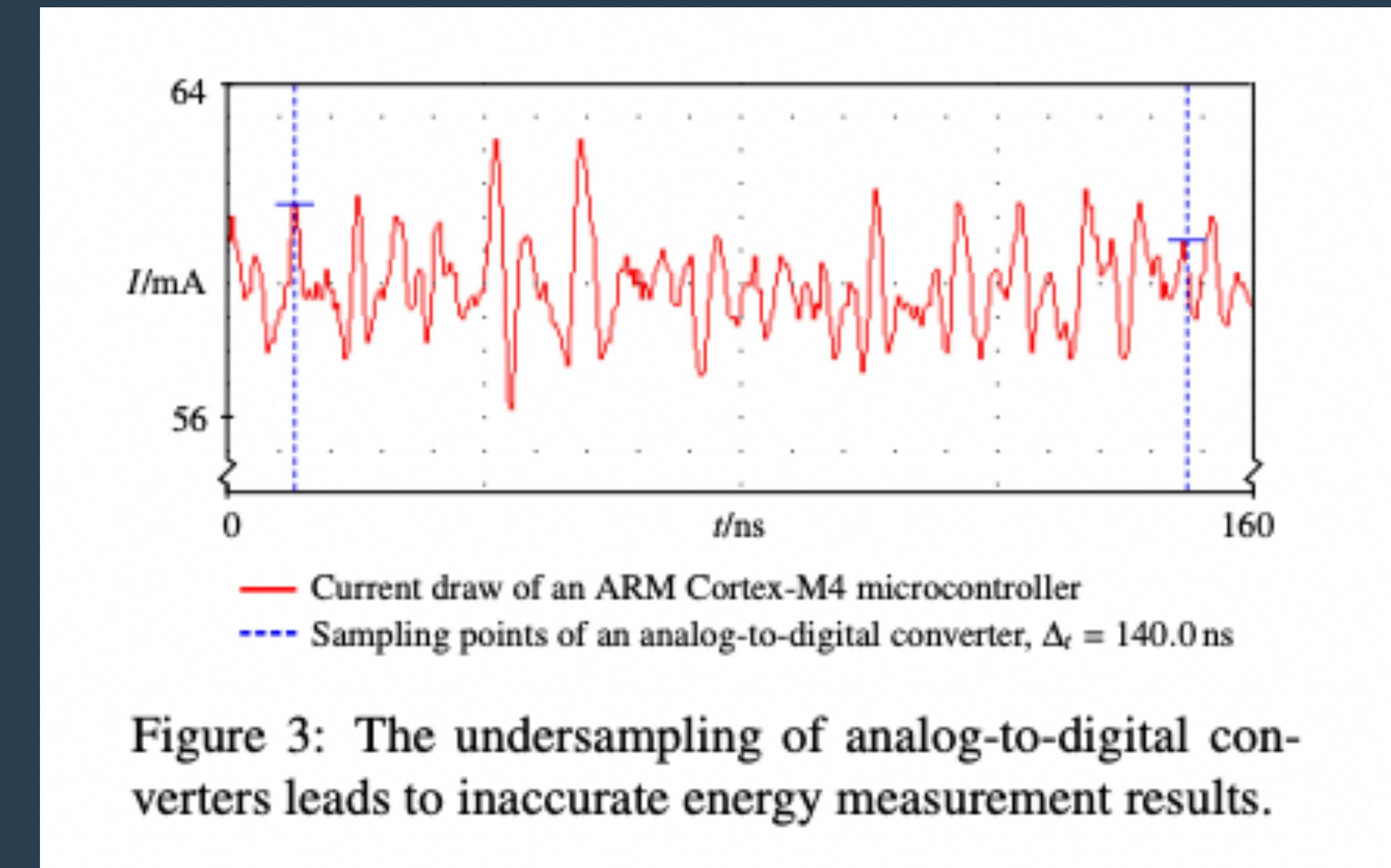
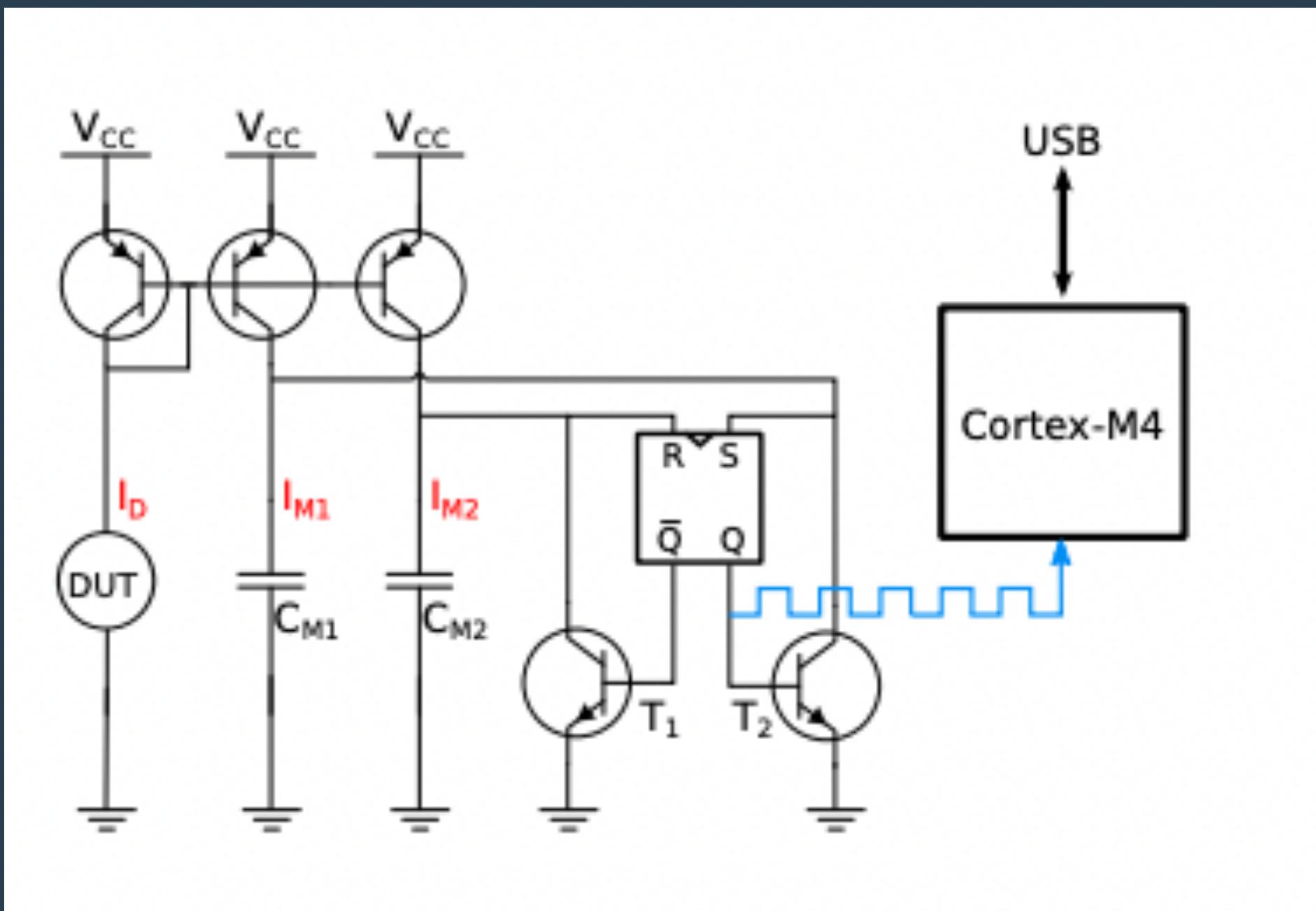


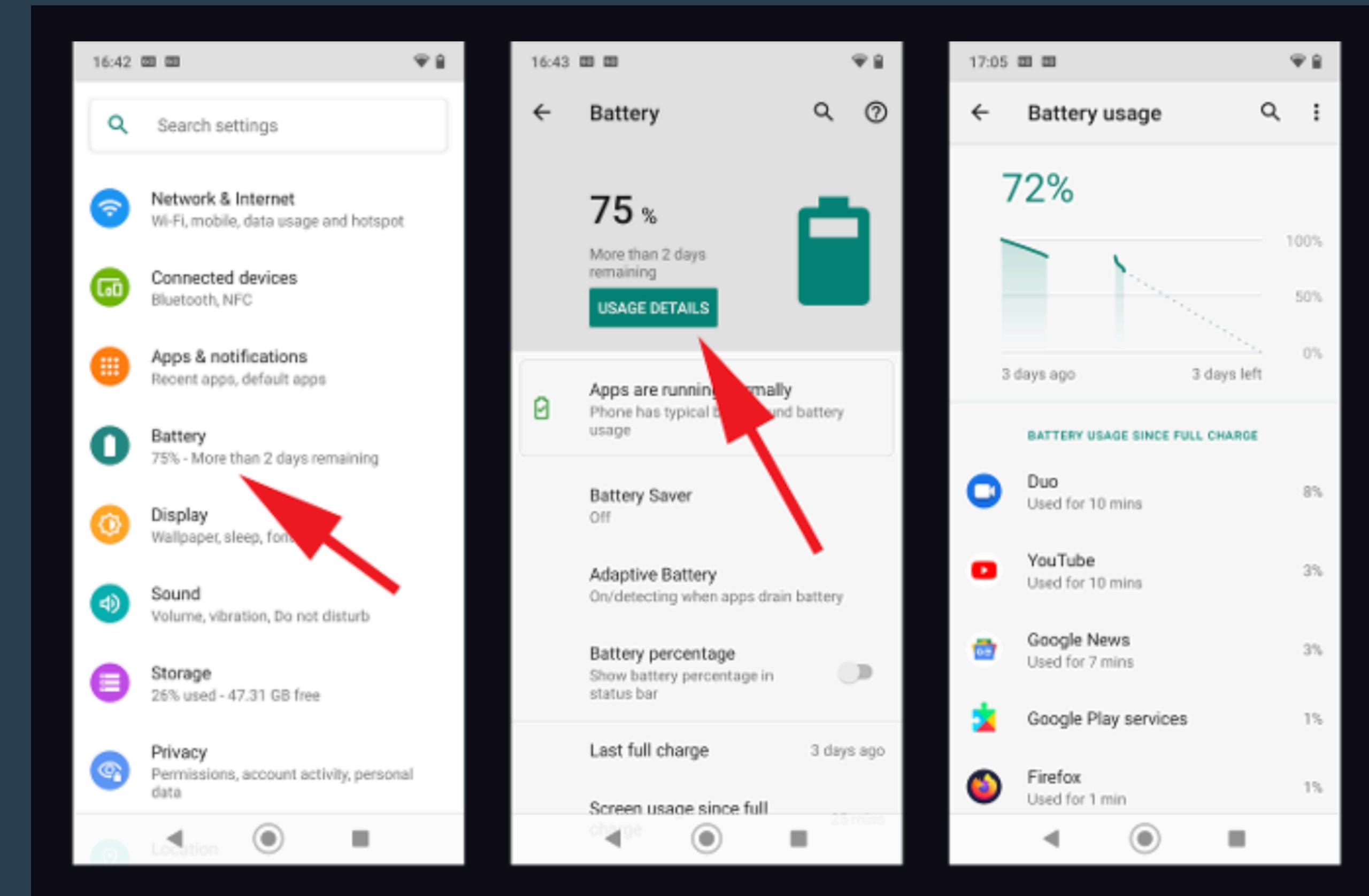
Figure 3: The undersampling of analog-to-digital converters leads to inaccurate energy measurement results.

Wie messen wir Energie

Eine mögliche Alternative: Durch Batterieentladung bei mobilen Geräten



Example: Coconut Battery for
macOS / iOS

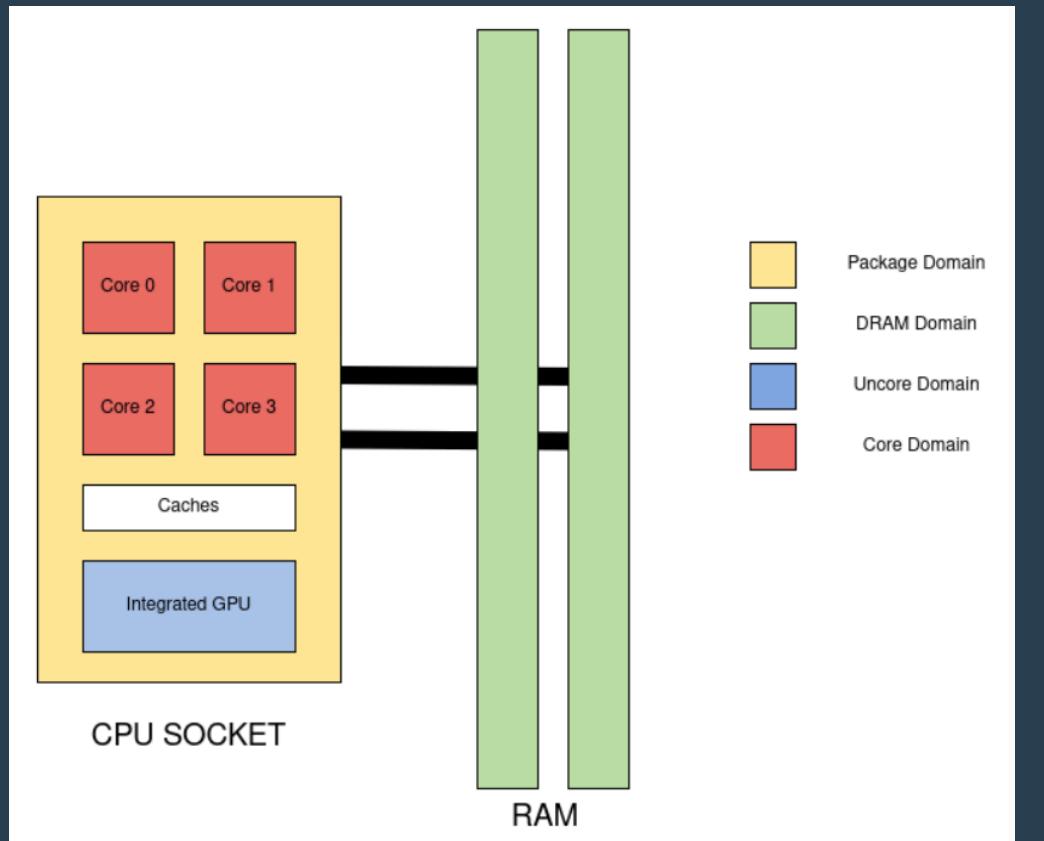


Android Battery usage (model)

Green Metrics Tool

Mess-Hardware / Cluster

- Desktop: Blue Angel Compatible Machine
- Server:
 - Fujitsu TX1330 M3 / M2 (2x)
 - Quanta Leopard - Multi-Tenant Server
SoftAWERE compatible
 - Developer: MacBook Pro 2015 13"



RAPL: https://pyjoules.readthedocs.io/en/stable/devices/intel_cpu.html



MCP39F511N / AMD00706



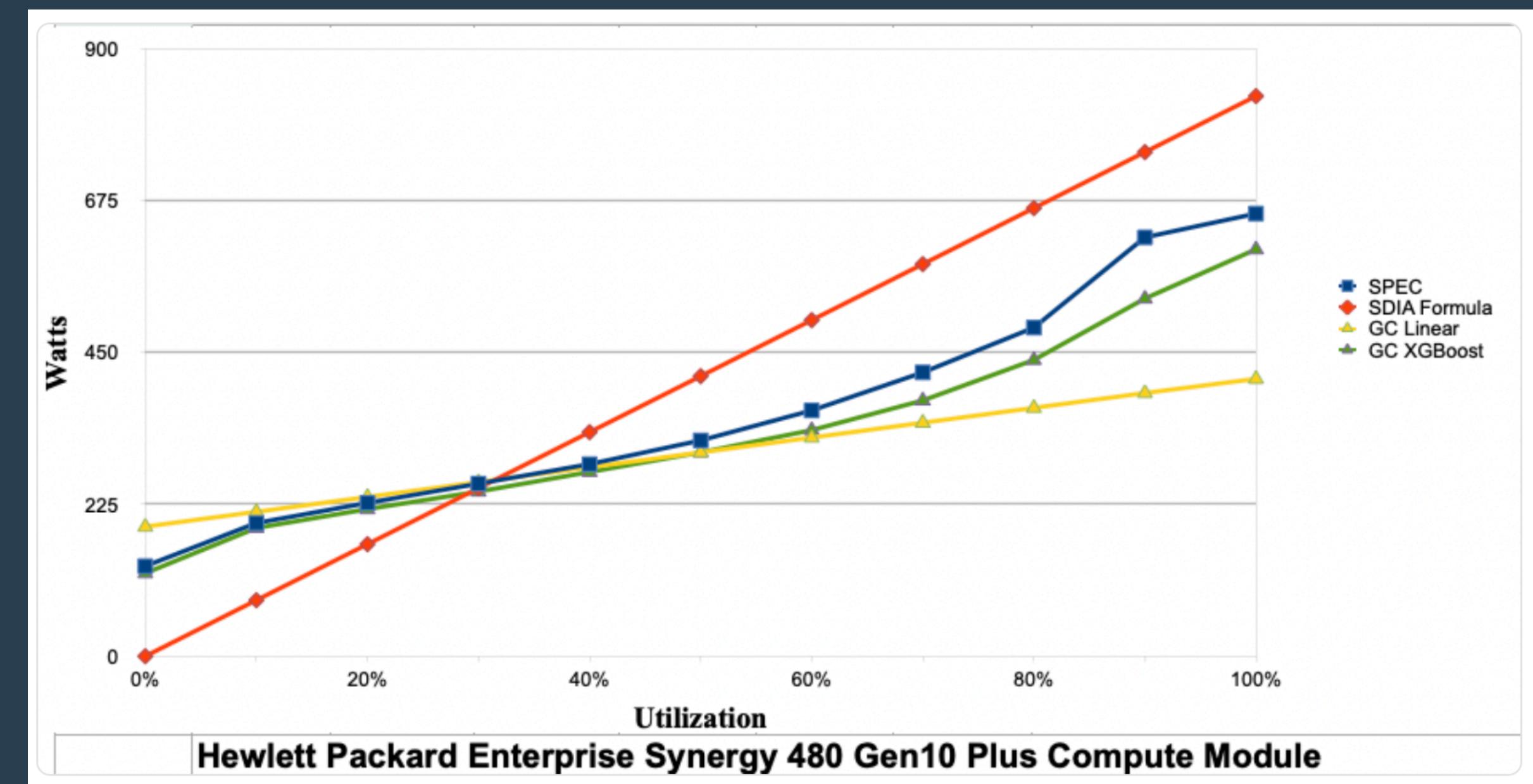
NVIDIA SMI

Cloud

How can we estimate in the cloud?

Cloud Energy Estimation for Cloud and VMs

- The setup of the model is based on a research paper from Interact DC and the University of East London
- 90%+ Accuracy in/out-of sample
- Near 0% overhead
 - XGBoost + POSIX stream implementation



Estimation of IT CO₂ emissions

Static approach: Back of the envelope - to get first snapshot

- Assumptions:
 - \$2.5 - 5\$ per hour for full machine (bare metal) [1]
 - Typical machine draws ~ 600 W [2]
 - Worldwide grid intensity is 442 gCO₂e/kWh [3]
- ~6,500 - 13,000 EUR in cloud spends equals 1 ton of CO₂ emissions
- Downside: Only gives you the company total operation

[1] <https://aws.amazon.com/ec2/pricing/on-demand/>

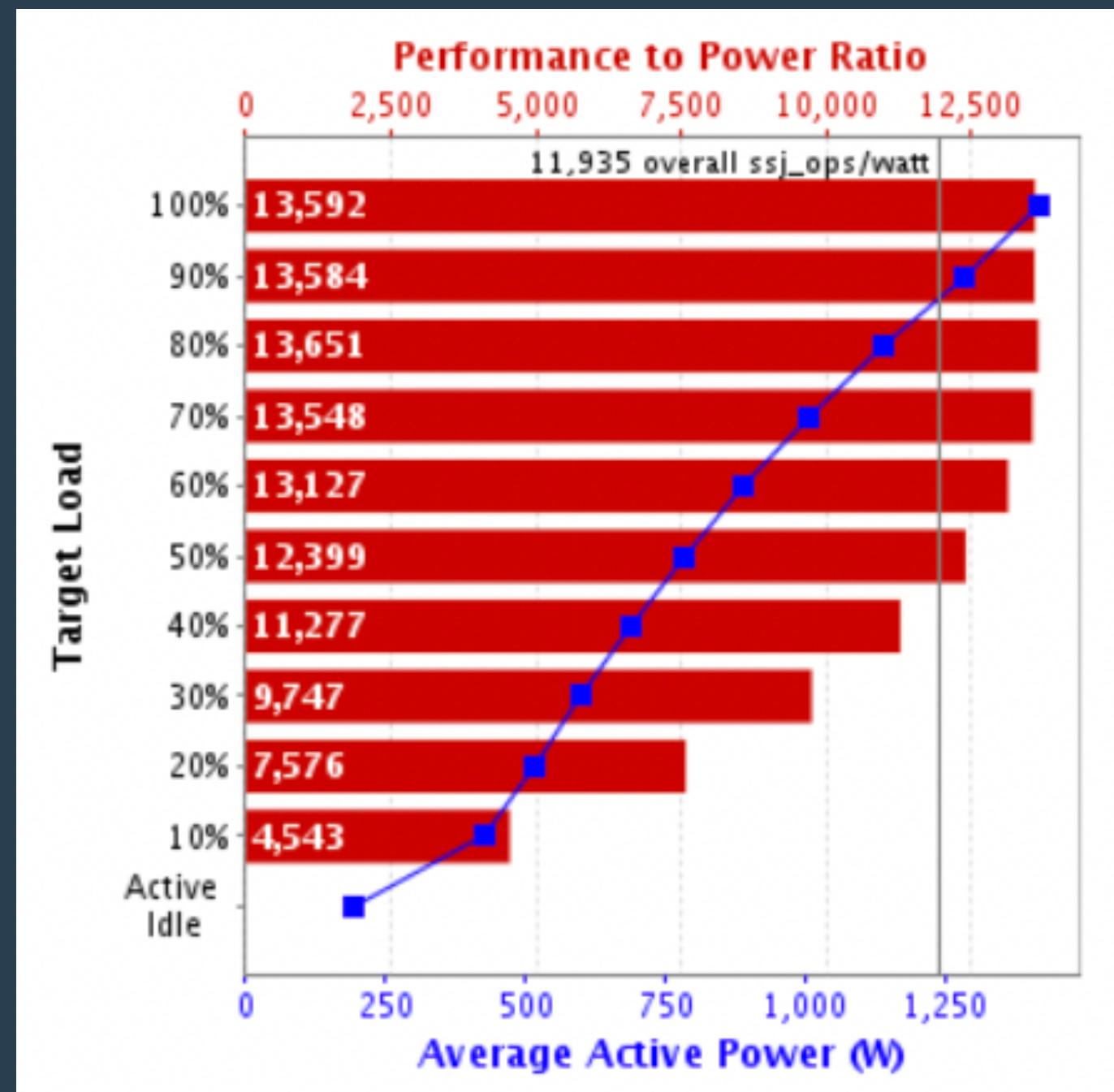
[2] <https://datavizta.boavizta.org/cloudimpact>

[3] <https://www.green-coding.io/co2-formulas/>

Potentials of different hardware configuration

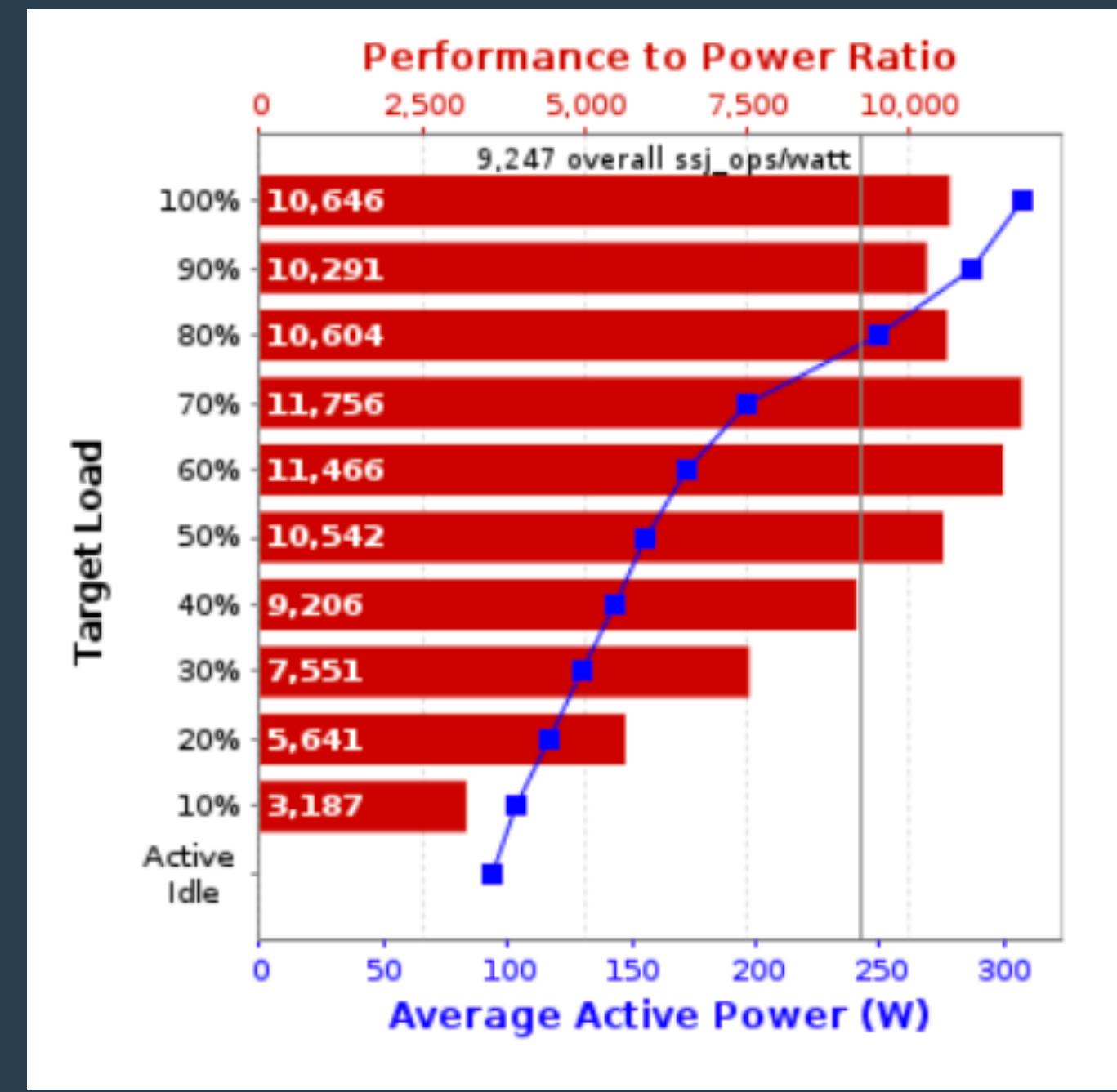
Power behaviour of hardware and how to leverage non-linearity

High Idle, but almost linear



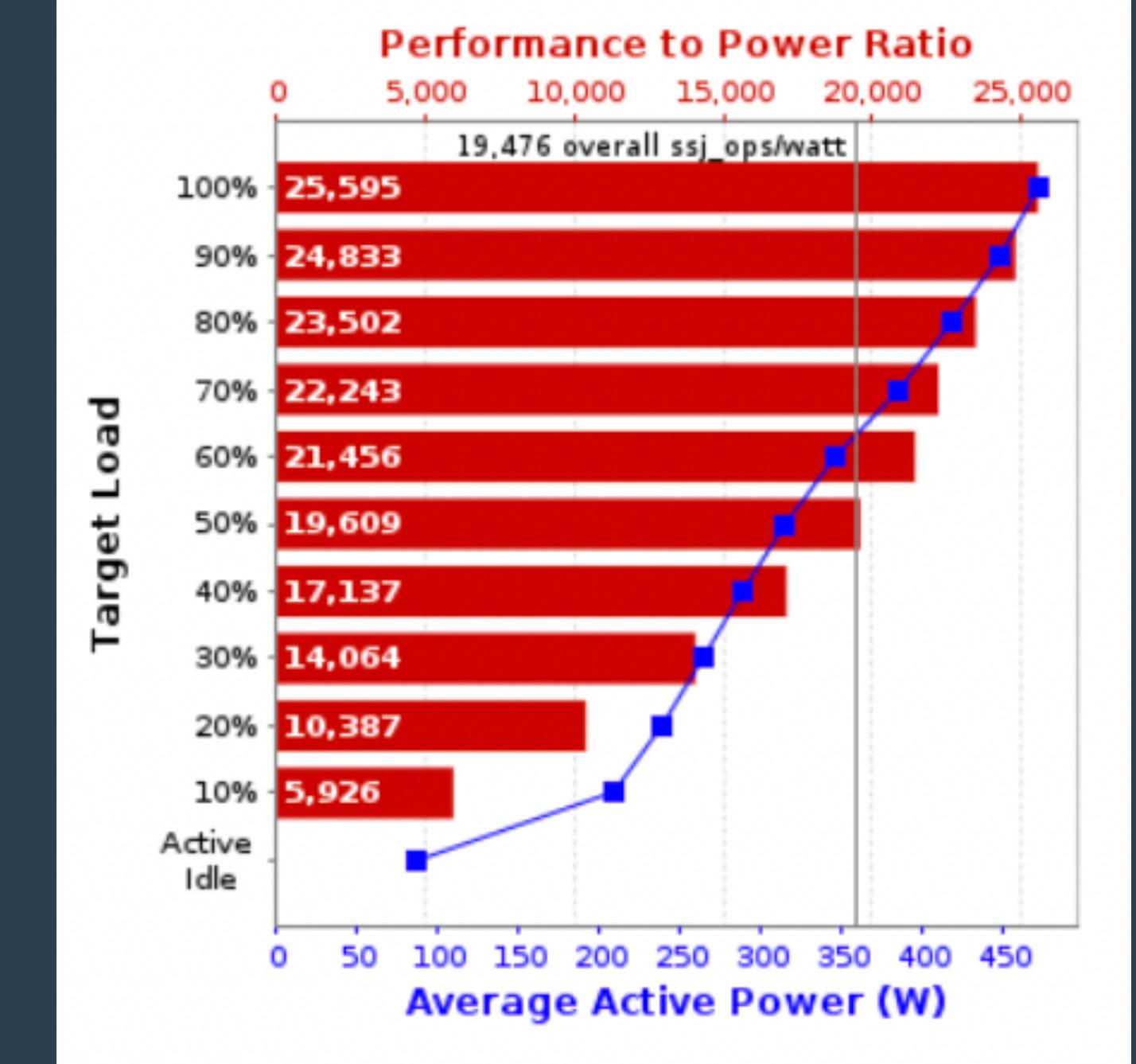
ASUSTeK Computer Inc. RS720Q-E9-RS8
(2019)

50% Power increase at 70% utilization



Hewlett Packard Enterprise ProLiant DL110
Gen10 Plus

Idle optimized



QuantaGrid D43K-1U (2022)