Energy saving techniques

For modern cloud services and on-prem software

</> GREEN CODING;

Who am I

Arne Tarara - Green Coding Solutions

- CEO & Founder for Green Coding Solutions
- Software Developer 16+ years
- We specialize in making software sustainble through benchmarking and optimization
- All our tools are open source

Lufthansa Industry Solutions

Green Software Foundation

SDIA

wagtail

</> GREEN CODING;

Before we talk about savings

Let's look at the techniques we can leverage in order to achieve a saving

The anatomy of a saving

Three kinds

- **Saving through efficiency**
  - Doing stuff different / using different approach
  - Not doing stupid stuff / Extra roundtrips etc.
- **Saving through doing less**
  - Just do not log that much etc.
  - Demand shaping
- **Turning off when you do not need machine**

What we will focus on in this talk

This talk is about energy savings. Not CO2

- An energy saving is almost always a CO2 saving
- Unless when it is not ...
  - Migrating to a better processor to save energy. But the old one goes to the trash. Lifetime of new hardware is too short to trade-off the energy savings
- So for every energy saving we look at we still must take rebound effects / backlashes into consideration ❗
- Also: We will focus on efficiency and doing less in this talk. Not idle time savings.

How do you approach a saving?

When you have a software in front of you?

Approaching a saving

Three scenarios

- You know the software, because you have written it
  - => **<u>Every</u>** programmer can tell you X parts of your software where a saving could be possible but was not implemented because of XYZ
- You have benchmarked the software and found a bottleneck
  - => **<u>Every</u>** programmer can fix a bottleneck as soon as it is identified
- You have an unknown software in front of you, that seems to be running fine
  - => How do you know if you can save? -> This is the talks topic

Approaching a saving

Methodologies out there

- **Static code analysis according to "best practices"**
  - EcoCode / CAST etc. [1] [2]
- **General Tips**
  - Green Software Foundation "Patterns" [2] / Dark Patterns list by Digital Sustainability Center [3]
  - Software Design Patterns /Performance Engineering Tips
- **Theoretical computer science approaches**
  - Big O Notations - O(n) > O(log(n))

[1] https://github.com/green-code-initiative/ecoCode/releases
[2] https://learn.castsoftware.com/green-software
[3] https://patterns.greensoftware.foundation/catalog/cloud/match-utilization-requirements-of-vm/
[4] https://s2group.cs.vu.nl/AwesomeAndDarkTactics/

Approaching a saving

Problems with these approaches

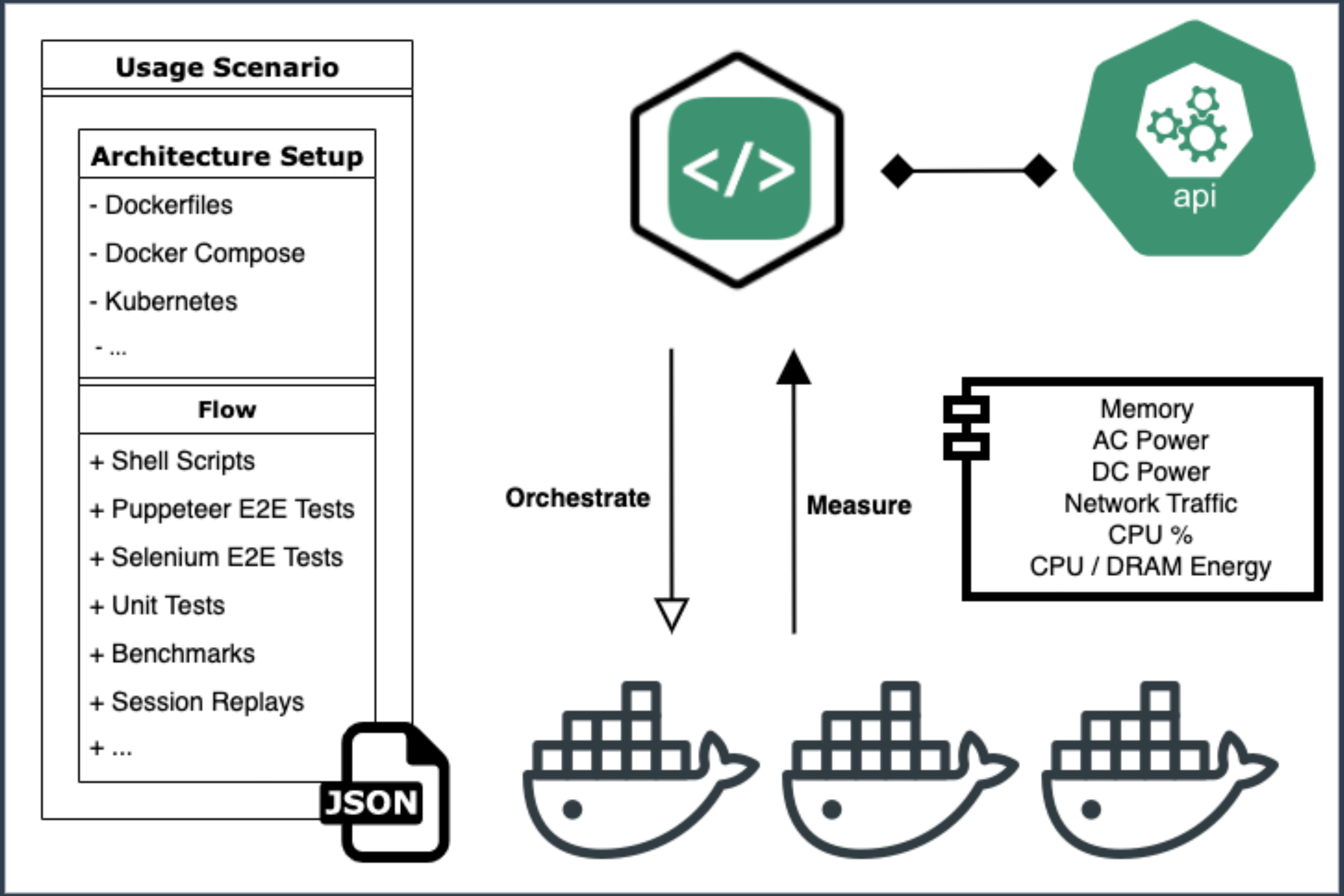
- All of these approaches have not seen the software running
- A recommendation like use O(n) instead of O(n^2) algorithm might even be less helpful if the set you are iterating over is very small because the preparation time of the algorithm might be higher
- Software is very often so complex that you receive thousands of small recommendations, but where is the saving really?
- So we believe in order to really optimize a software you **must** see it running and look at a top down picture.

So

Let's run some software

We used the Green Metrics Tool

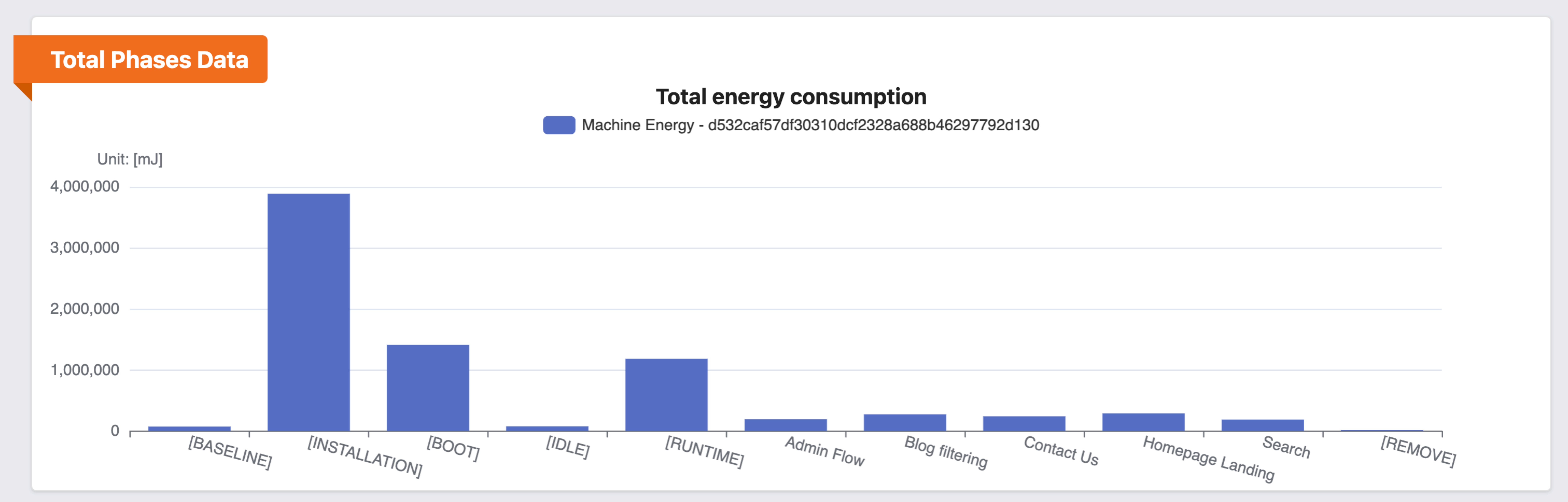
To put some open source software into benchmarks, unit-tests etc.



</> GREEN CODING;

Showcase #1: High level life cycle overview

Django Unit tests: Build vs. Runtime view to get high level info



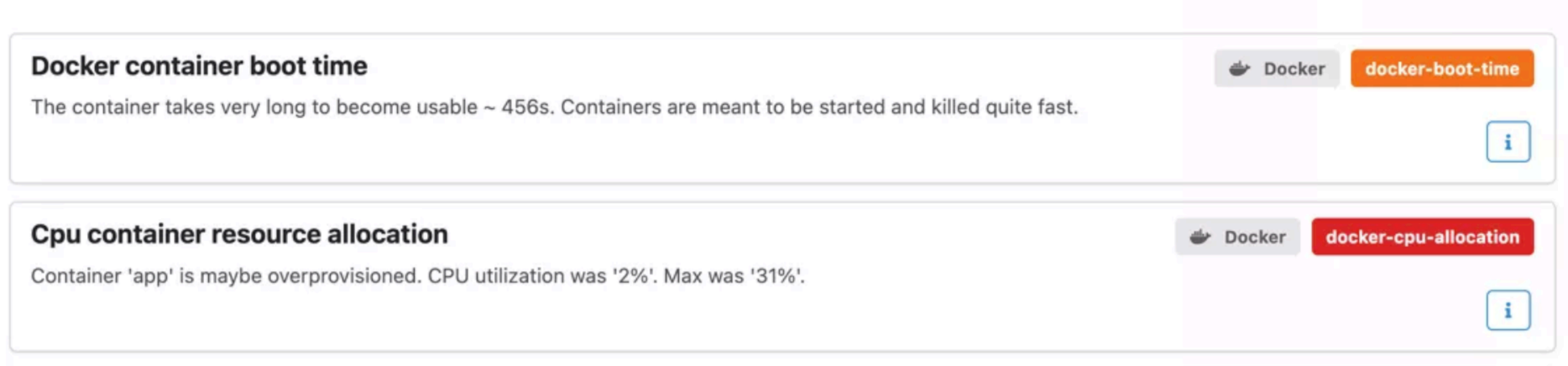
Boot compared to runtime very bad. Problematic in FAAS and CI/CD cases where caching is possible

Take away: The technique is of course very easy (caching). But we need tools that give you a one shot view of where to look deeper

</> GREEN CODING;

Showcase #1: High level life cycle overview

Django Unit tests: Build vs. Runtime view to get high level info



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%'.

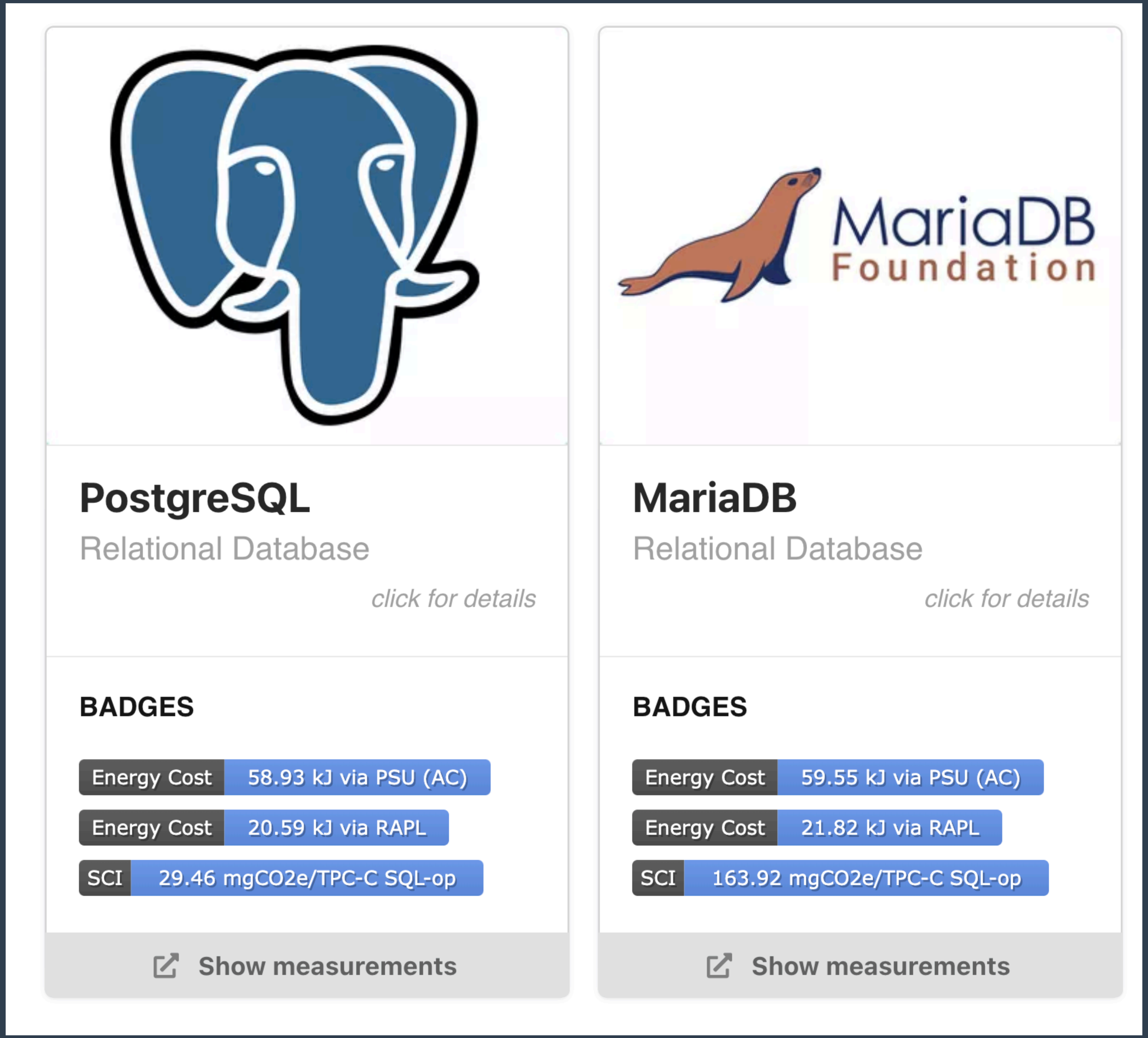
If you know the application, then you can also automate tips (See our workshop later!)

GREEN CODING;

Showcase #2: Software Scorecards

We need data libraries where we can compare use cases

- We did a case with PostgreSQL and MariaDB
- Both were given same hardware, same benchmark (TPC-C)
- Both have SCI score written down
- Postgres 5x better than MariaDB for absolute standard use case!
- => Of course this is not always the case. Configuration plays a huge role. But the standard behaviour of a software counts!



PostgreSQL

Relational Database

click for details

BADGES

Energy Cost 58.93 kJ via PSU (AC)

Energy Cost 20.59 kJ via RAPL

SCI 29.46 mgCO2e/TPC-C SQL-op

Show measurements

MariaDB

Relational Database

click for details

BADGES

Energy Cost 59.55 kJ via PSU (AC)

Energy Cost 21.82 kJ via RAPL

SCI 163.92 mgCO2e/TPC-C SQL-op

Show measurements

Showcase #3: On-Prem software

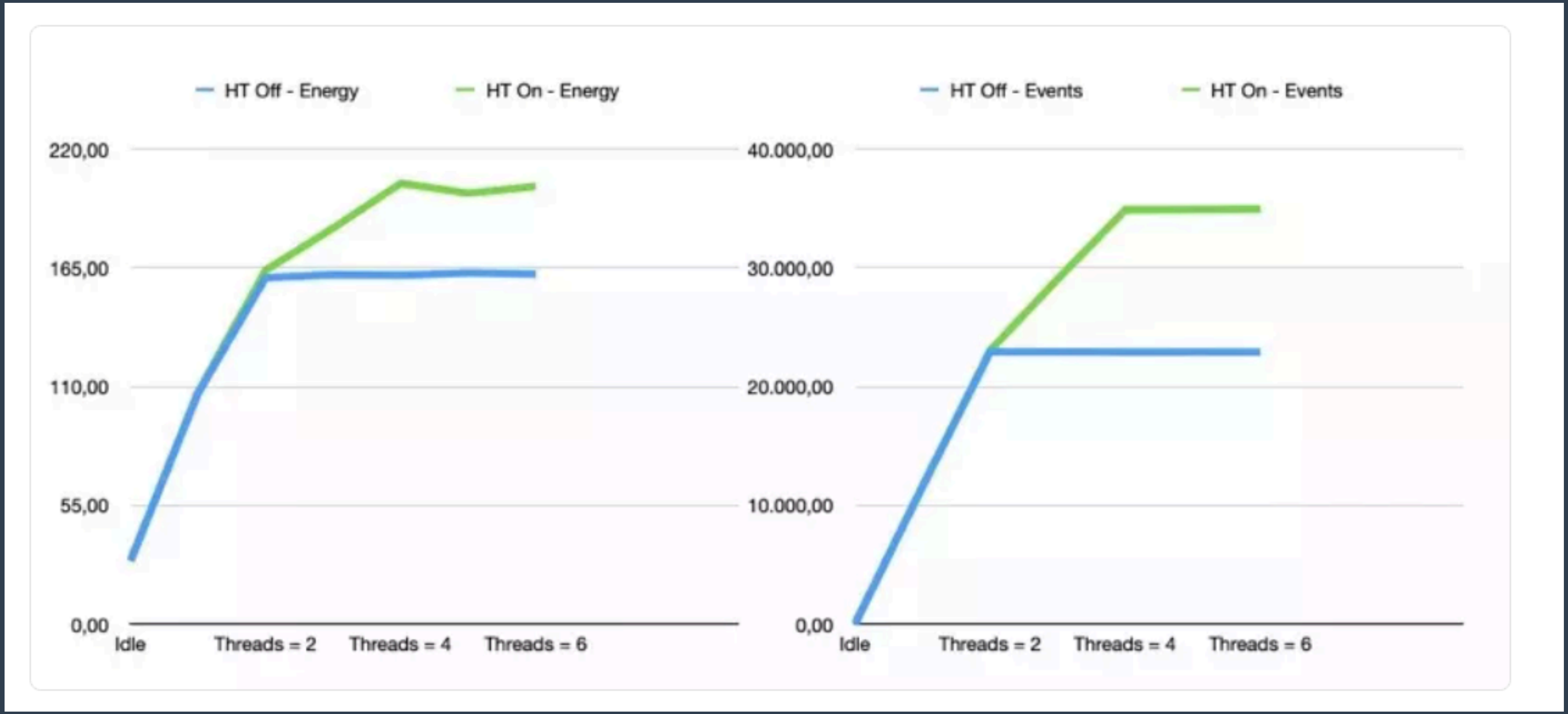
When you can actually influence the hardware

- Using different means in the operating system
  - PowerCapping (GPU / CPU) On / Off [1]
    - Sets maximum energy limit
  - TurboBoost On / Off [2]
    - Puts CPU into "boosted" frequency for short while
  - HyperThreading On / Off [3]
    - Creates extra virtual threads to have better multi-threading

[1] https://www.green-coding.io/case-studies/cpu-power-capping/
[2] https://www.green-coding.io/case-studies/turbo-boost-and-energy/
[3] https://www.green-coding.io/case-studies/hyper-threading-and-energy/

Showcase #3: On-Prem software

When you can actually influence the hardware



HyperThreading performance surplus in many applications far exceeds the additional needed energy

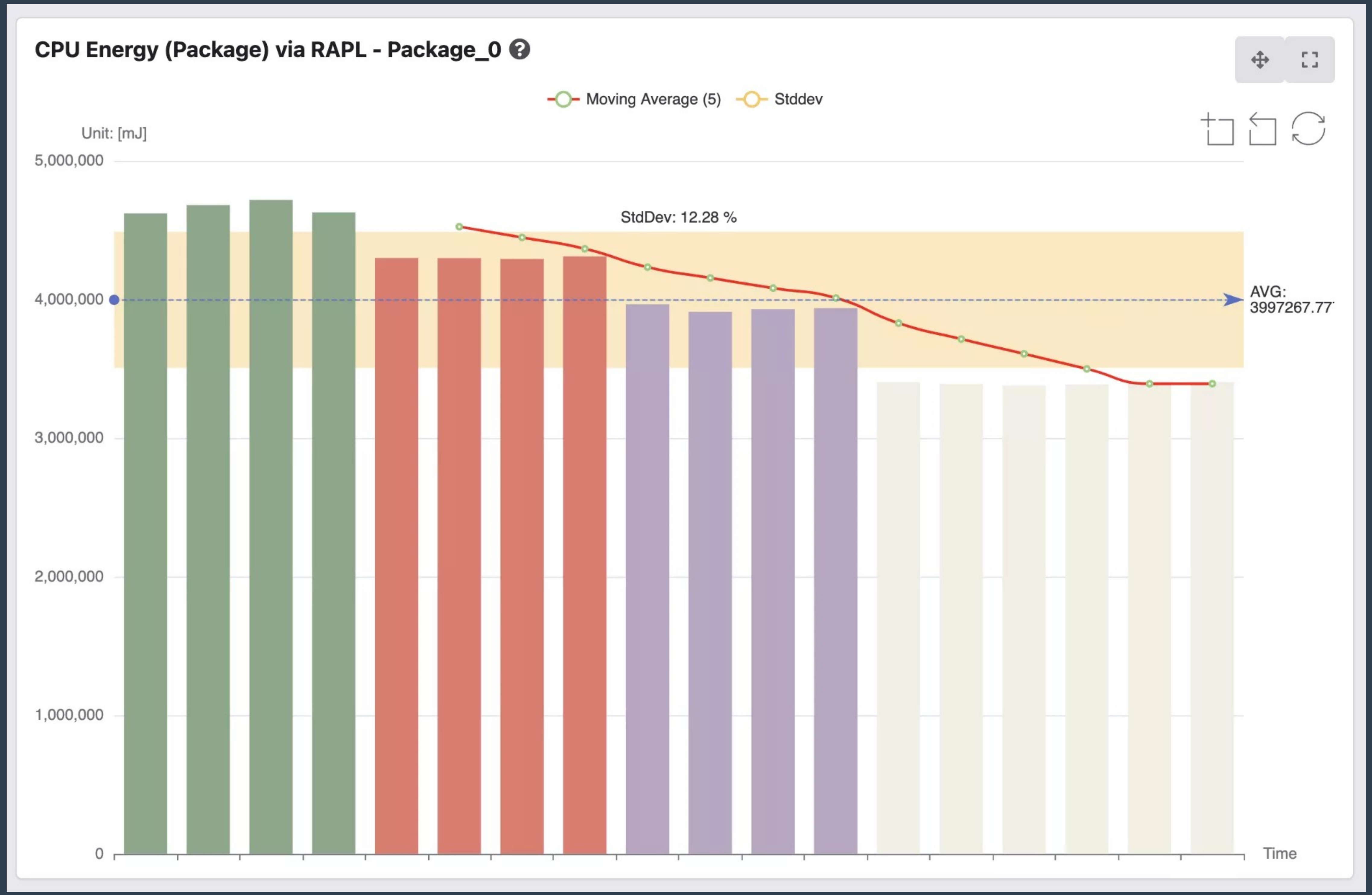
-

Be aware of bad virtualization and / or HPC

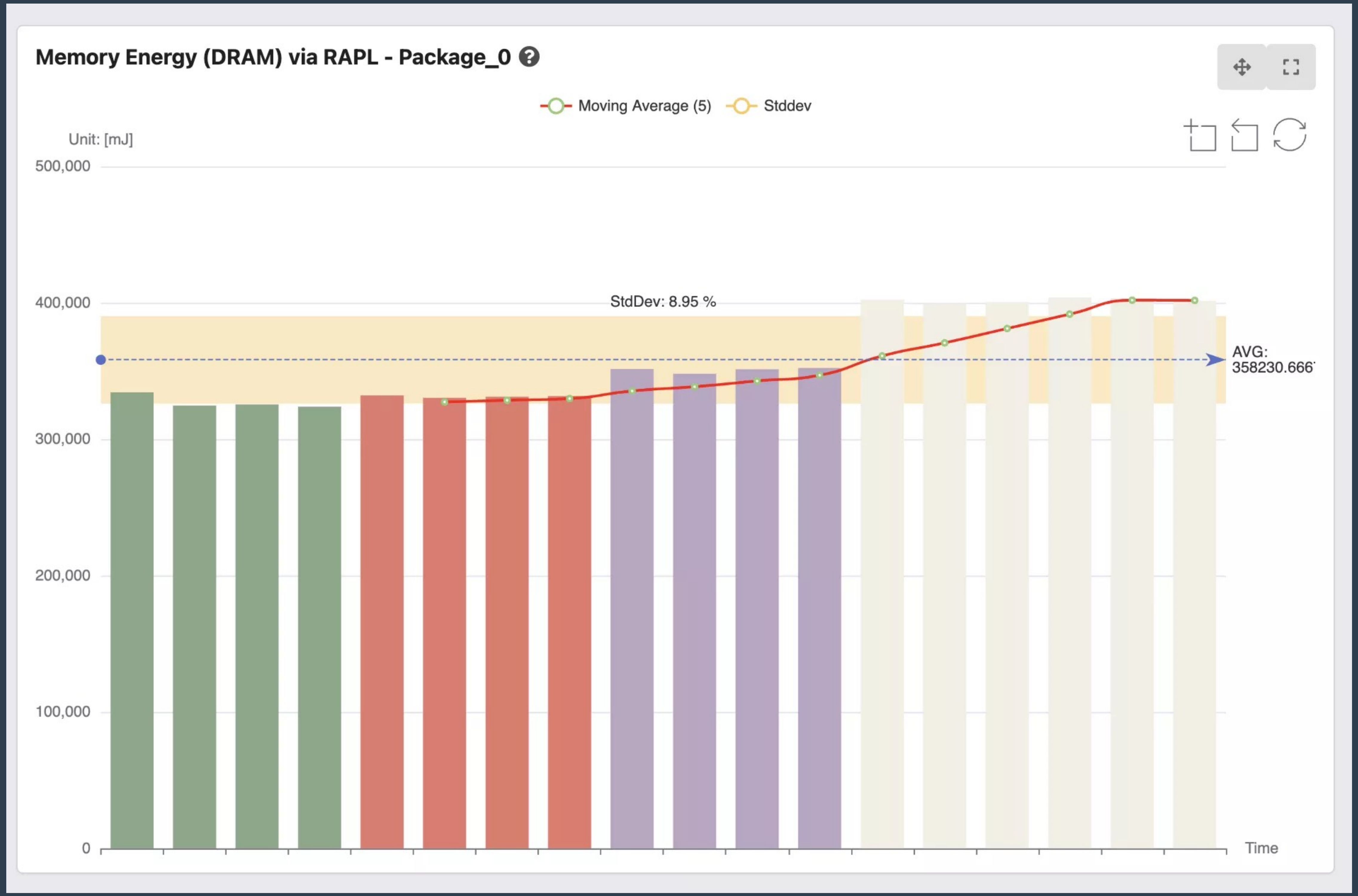
</> GREEN CODING;

Showcase #3: On-Prem software

When you can actually influence the hardware - Power Capping



The more power we cap, the lower the CPU energy. Whoohoo!

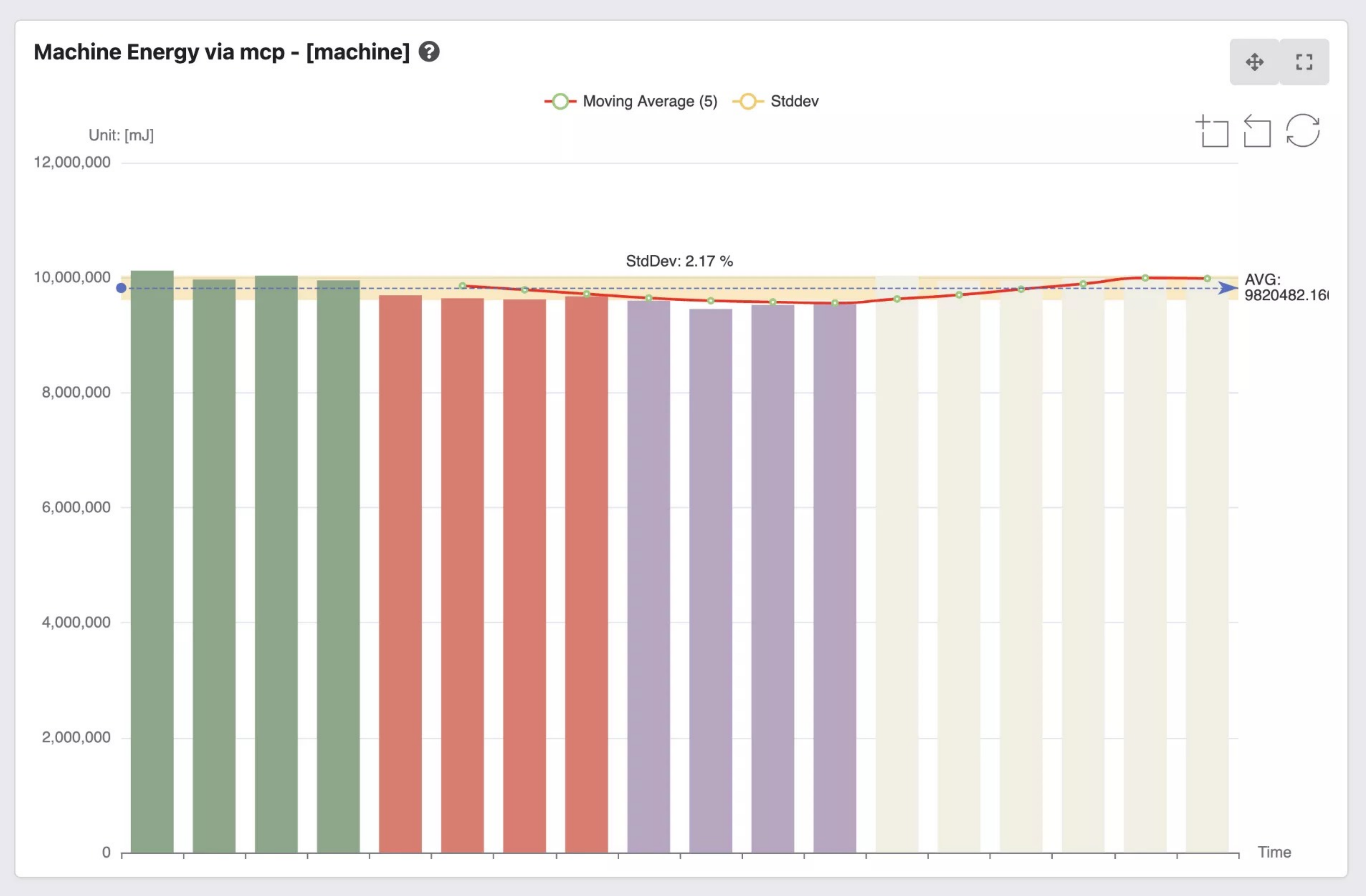


But DRAM energy is going up? Why? ... but it is stil better in total!

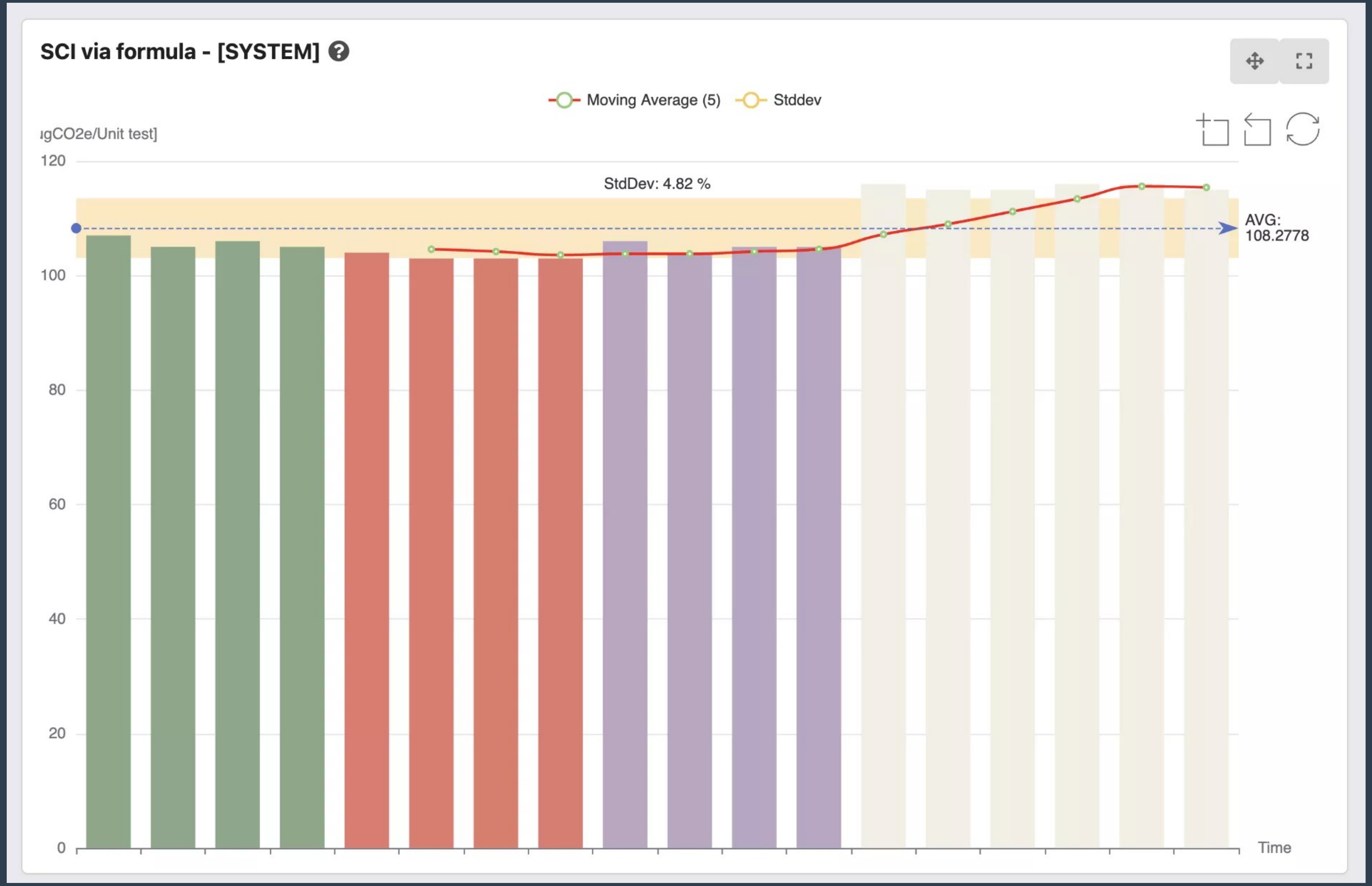
GREEN CODING;

Showcase #3: On-Prem software

When you can actually influence the hardware - Power Capping



But wait, the machine energy is actually going up at some point?
(Energy = Power * Time. Increase in time is now hitting)



Argh, and SCI is even worse ....

</> GREEN CODING;

Showcase #3: On-Prem software

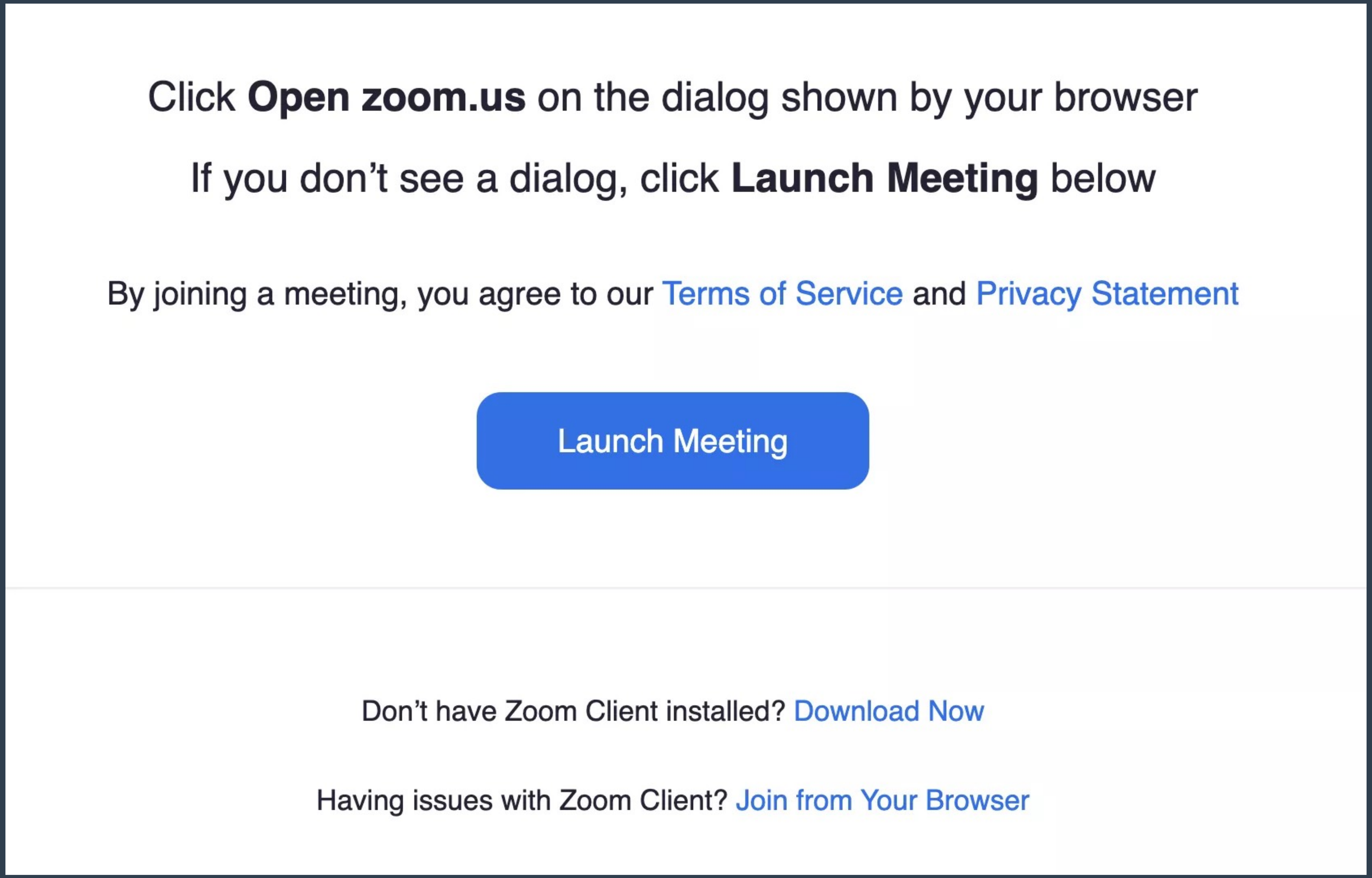
When you can actually influence the hardware

- If you know think of second order effects
  - Network storage
  - Display attached to device
  - Cooling of the system
  - ...
- Then being fast actually becomes more relevant. However, this is only true for high load
- **Take away**: Energy savings do not exist in a vacuum. But always in a use-case! You MUST see the software in action. Even with "general tips" like power capping.

Showcase #4: Doing useful work

The Zoom auto download case

Zoom .exe downloads EVERY time on link visit (cookies deleted)



Click **Open zoom.us** on the dialog shown by your browser

If you don't see a dialog, click **Launch Meeting** below

By joining a meeting, you agree to our Terms of Service and Privacy Statement

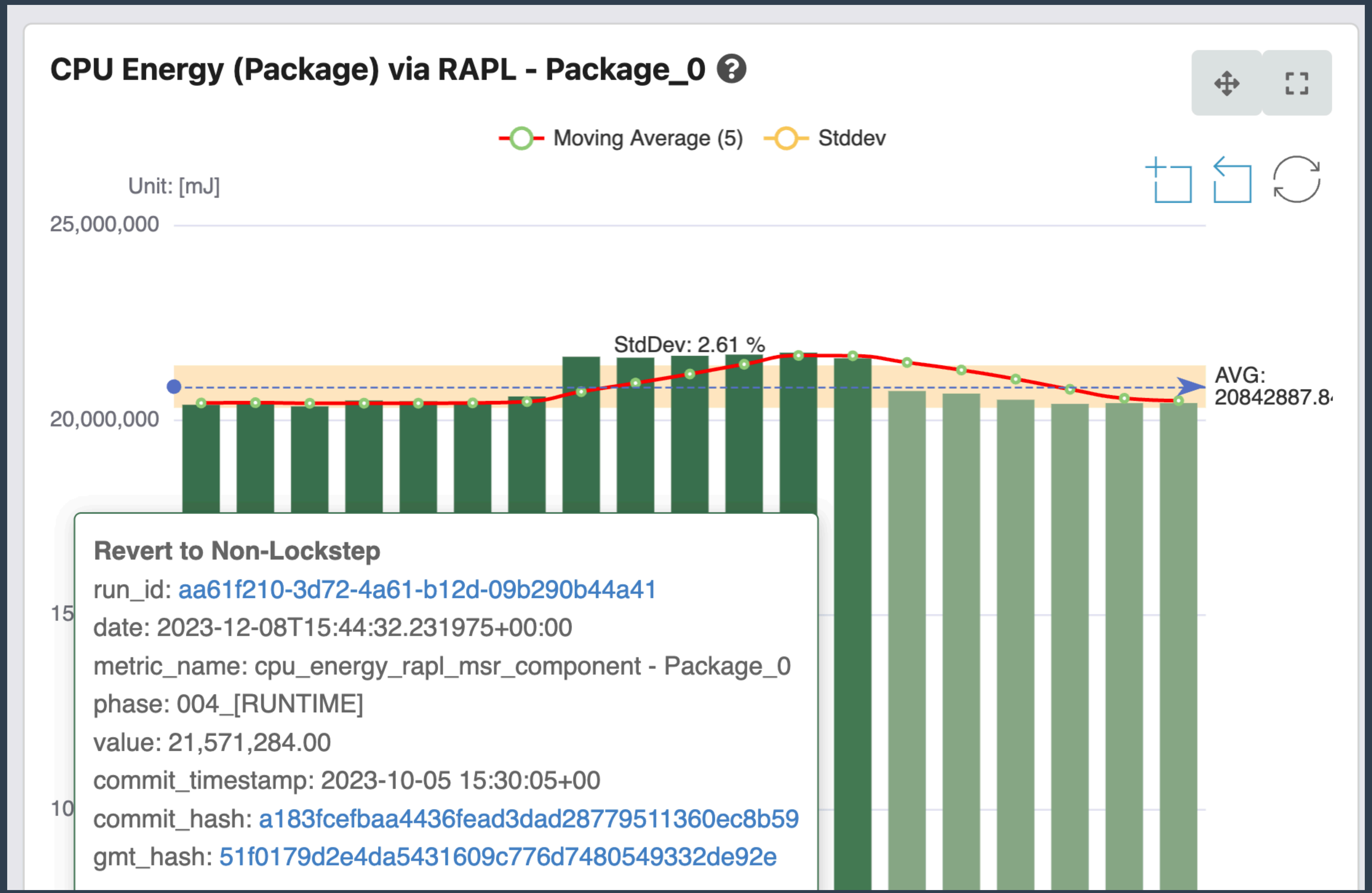
Don't have Zoom Client installed? Download Now

Having issues with Zoom Client? Join from Your Browser

Full case study: https://www.green-coding.io/case-studies/co2-savings-at-scale-zoom-auto-download/

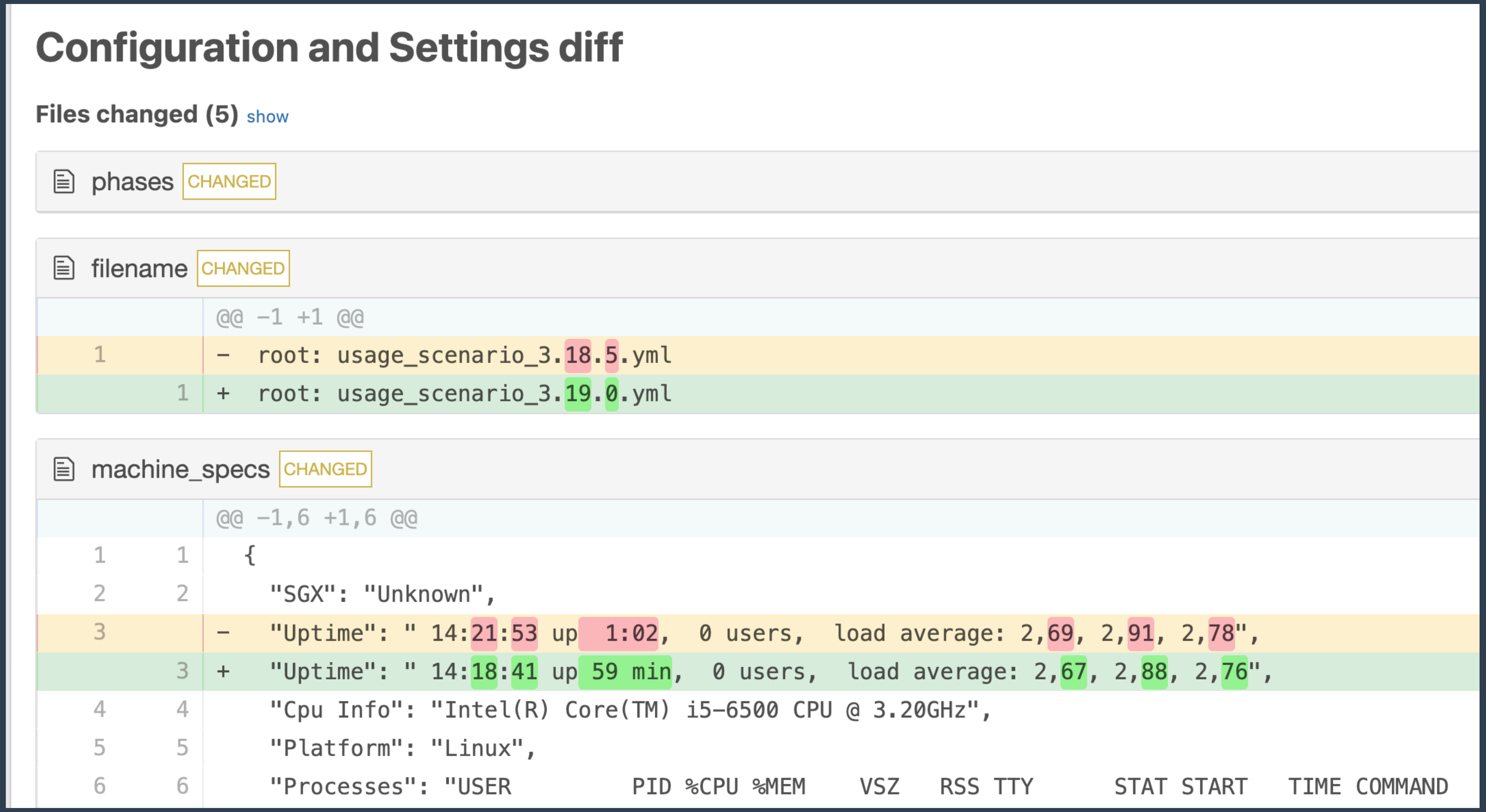
Showcase #5: Investigating libraries

By integrating energy awareness over time



Revert to Non-Lockstep
run_id: aa61f210-3d72-4a61-b12d-09b290b44a41
date: 2023-12-08T15:44:32.231975+00:00
metric_name: cpu_energy_rapl_msr_component - Package_0
phase: 004_[RUNTIME]
value: 21,571,284.00
commit_timestamp: 2023-10-05 15:30:05+00
commit_hash: a183fcefbaa4436fead3dad28779511360ec8b59
gmt_hash: 51f0179d2e4da5431609c776d7480549332de92e

Energy-Timeline Feature of Green Metrics is integrated in every git commit and hinted where the regression happened



Configuration and Settings diff

Files changed (5) show

phases CHANGED

filename CHANGED

```
@@ -1 +1 @@
1     - root: usage_scenario_3.18.5.yml
   1  + root: usage_scenario_3.19.0.yml
```

machine_specs CHANGED

```
@@ -1,6 +1,6 @@
1  1    {
2  2      "SGX": "Unknown",
3     -   "Uptime": " 14:21:53 up  1:02,  0 users,  load average: 2,69, 2,91, 2,78",
   3  +   "Uptime": " 14:18:41 up 59 min,  0 users,  load average: 2,67, 2,88, 2,76",
4  4      "Cpu Info": "Intel(R) Core(TM) i5-6500 CPU @ 3.20GHz",
5  5      "Platform": "Linux",
6  6      "Processes": "USER         PID %CPU %MEM    VSZ   RSS TTY      STAT START   TIME COMMAND
```

Green Metrics Tool includes extensive diffing. Energy regression happened in dependency

Issue: https://github.com/alpinelinux/docker-alpine/issues/385
Detailed Analysis: https://github.com/green-coding-solutions/alpine-energy-regression/blob/main/README.md

</> GREEN CODING;

Thank you for this appetizer tour!

We could only show some, but I hope the message was clear

We advocate for actually measuring software according to use cases in order to advance the green coding field with actionable insights and optimizations

- Look at our blog and case studies for the details from this talk [1][2]
- Look at our Energy-ID project for the open source projects we investigate for optimizations [3]
- Look at the measurements and try our platform. It's FOSS! [4]

[1] https://www.green-coding.io/blog
[2] https://www.green-coding.io/case-studies
[3] https://www.green-coding.io/projects/energy-id
[4] https://www.green-coding.io/projects/green-metrics-tool

GREEN CODING;