Andreas Brunnert

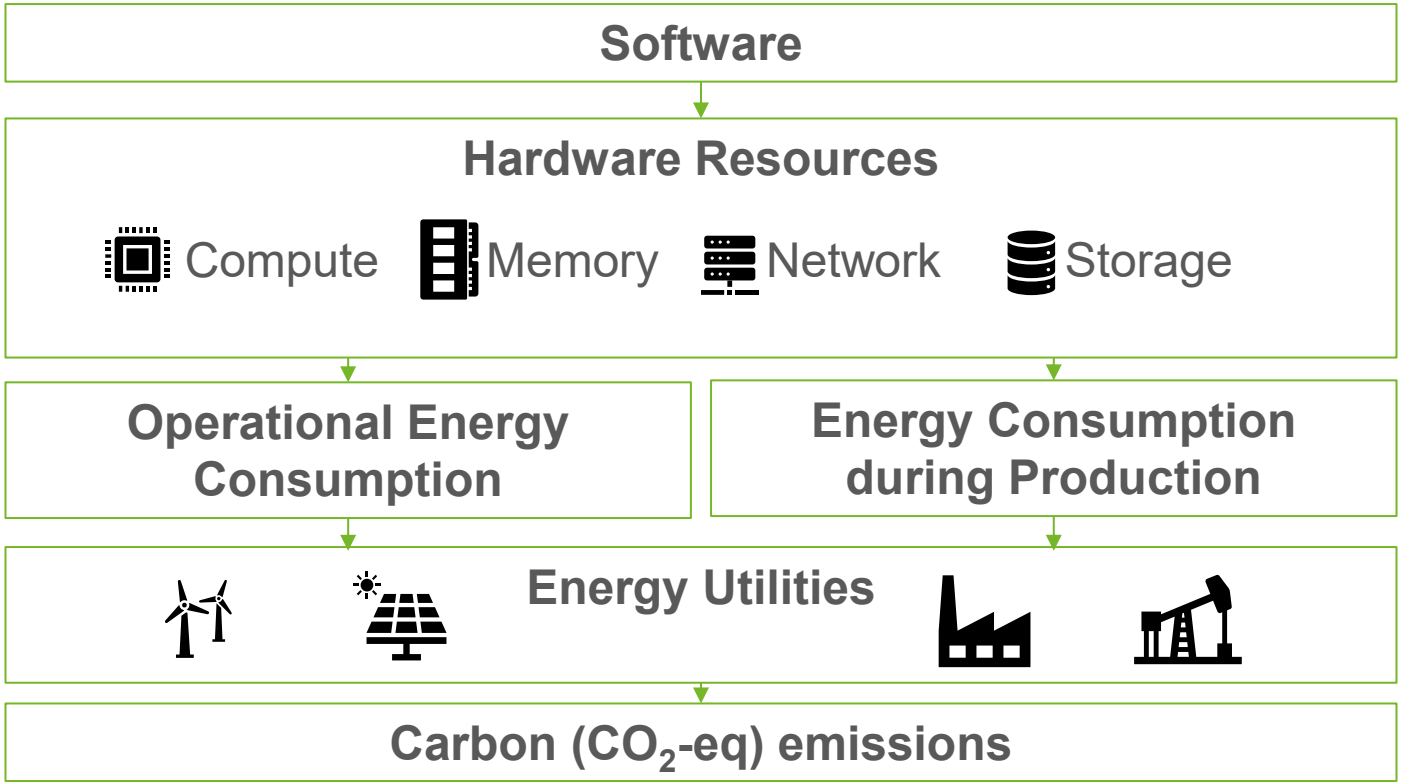
Wie misst man die CO2-Emissionen von Microservices?

How to measure carbon emissions for every API call of your microservices?

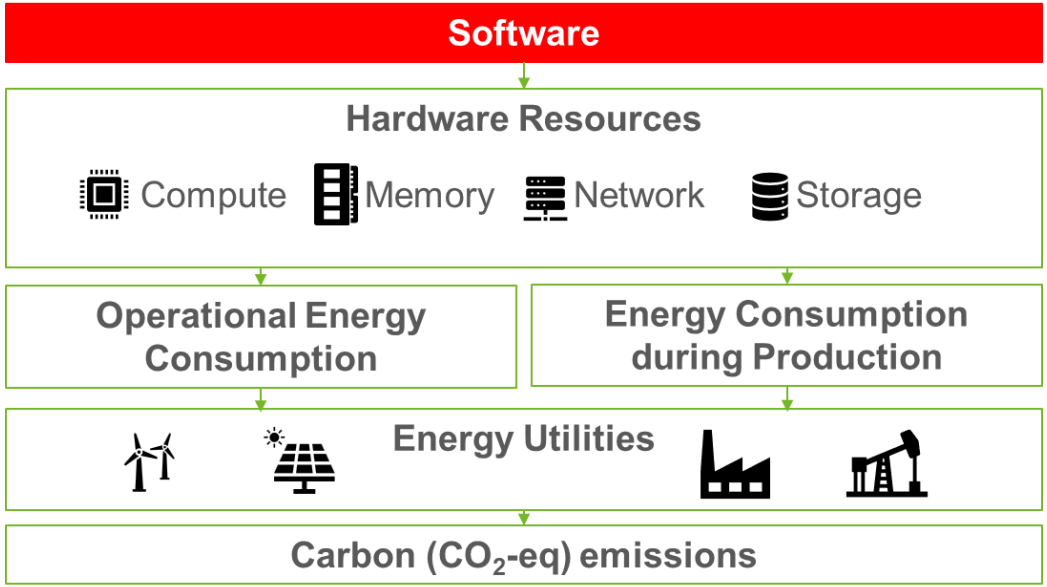
Andreas Brunnert

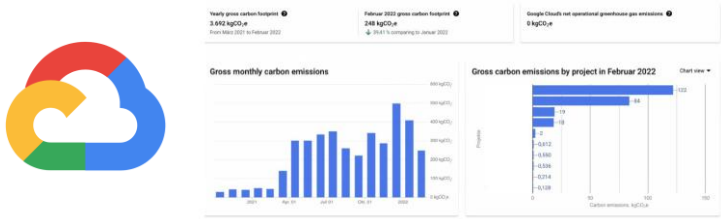
Professor @ Munich University of Applied Sciences HM
Founder @ RETIT GmbH

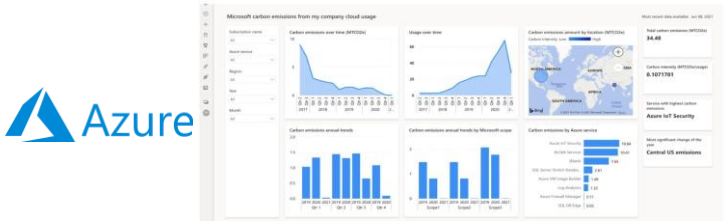
Software Carbon Emissions



Software Carbon Emissions





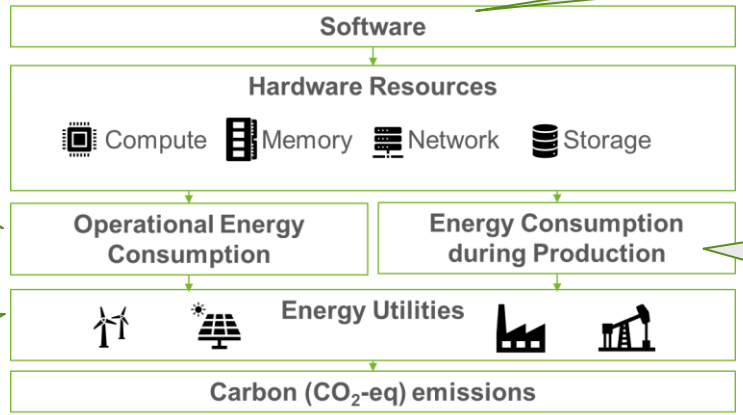


Software Carbon Intensity (SCI) Specification

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

R = Functional unit; this is how the software scales, for example per user or per API call

E = This is the energy that a software system consumes at runtime.



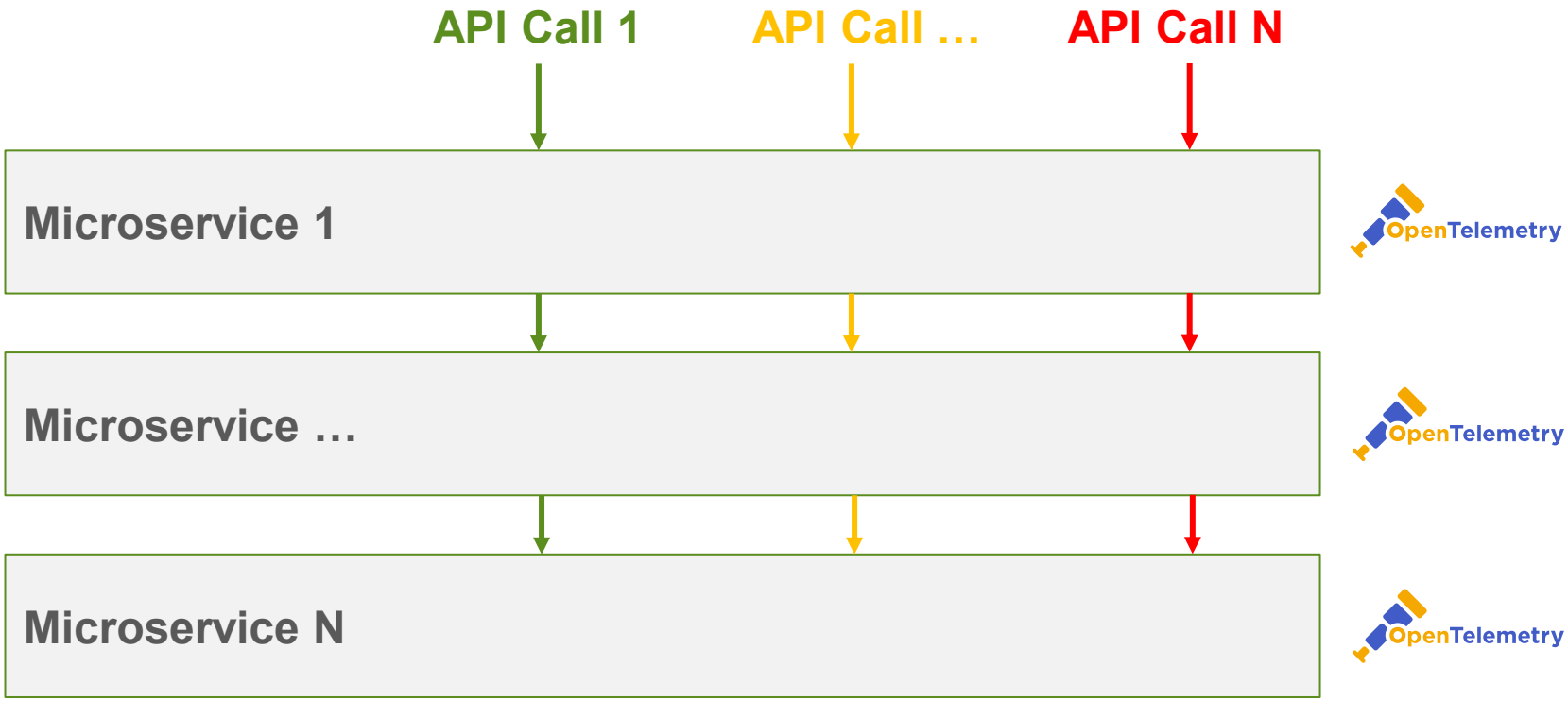
I = Carbon emissions per kWh of energy, gCO_2eq/kWh

M = CO_2-eq emissions for the production of hardware / software

- See https://www.iso.org/standard/86612.html and https://sci.greensoftware.foundation/ for more details

*SCI = ((*) +) per R*

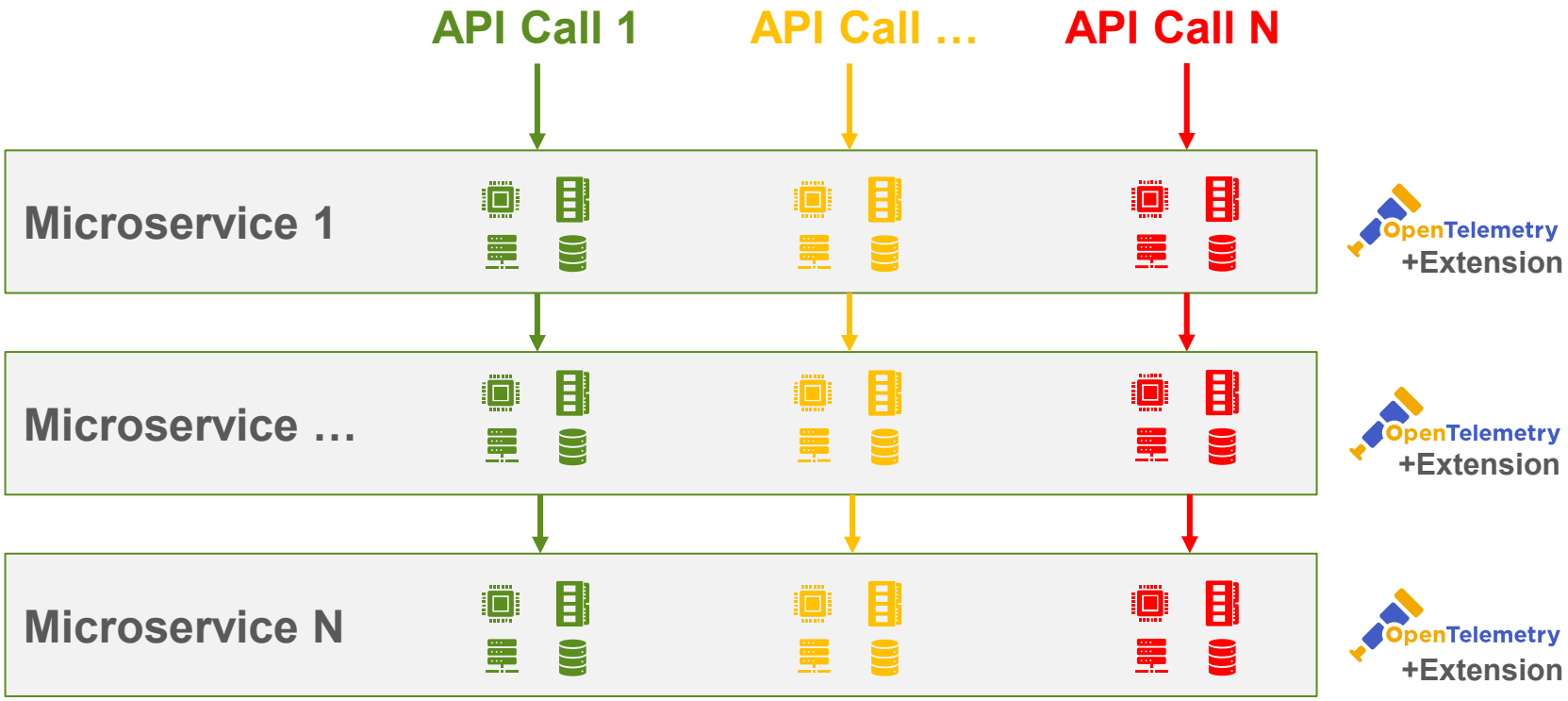
Microservices – *R* is typically an “API Call”



https://opentelemetry.io/

*SCI = ((E *) +) per R*

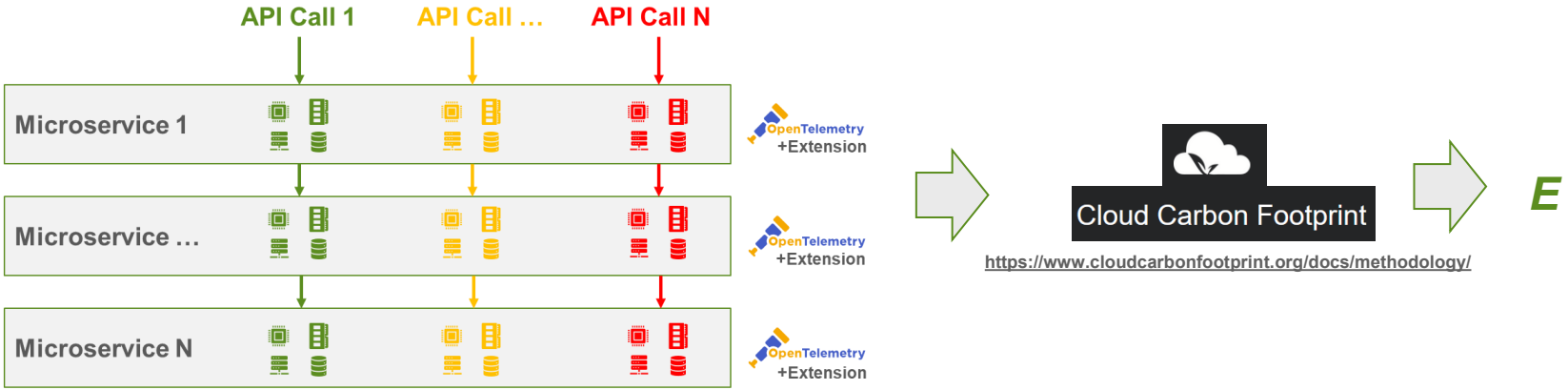
Microservices – *E* cannot be measured directly (in many cases)



https://github.com/RETIT/opentelemetry-javaagent-extension

SCI = ((E *) +) per R

Microservices – *E* is derived using models from the collected data



https://github.com/RETIT/opentelemetry-javaagent-extension

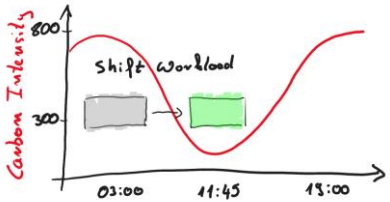
SCI = ((E * I) +) per R

Microservices – I can come from several sources

Cloud Carbon Footprint

https://github.com/cloud-carbon-footprint/ccf-coefficients

https://app.electricitymaps.com/



https://www.carbon-aware-computing.com/

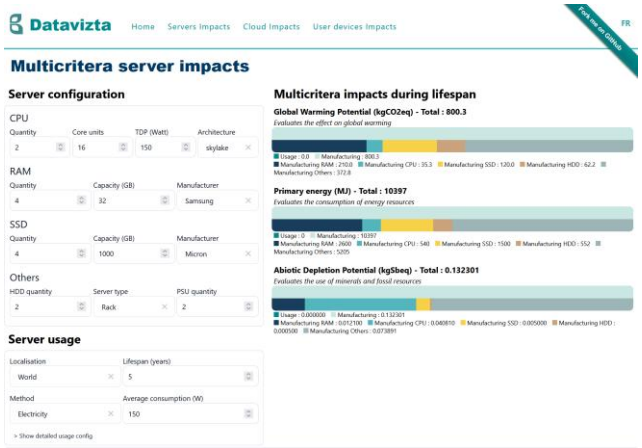
I

SCI = ((E * I) + M) per R

Microservices – M is provided by various sources

Cloud Carbon Footprint

https://github.com/cloud-carbon-footprint/ccf-coefficients

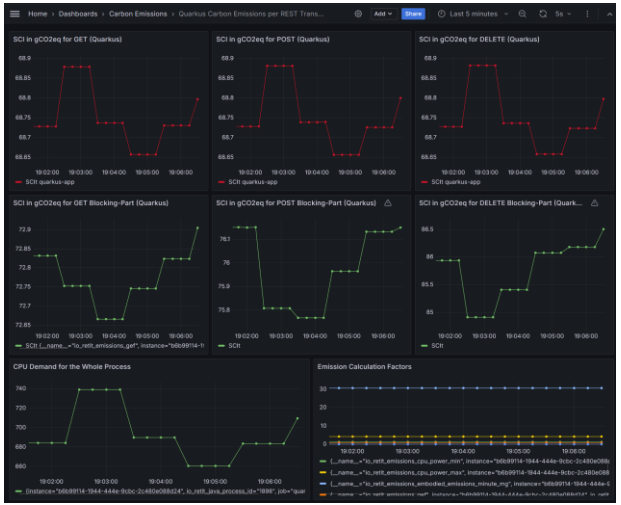


https://dataviz.boavizta.org/serversimpact

M

Demo for Microservices in Java (Quarkus + Spring)

https://github.com/RETIT/opentelemetry-javaagent-extension/tree/main/examples/spring-rest-service

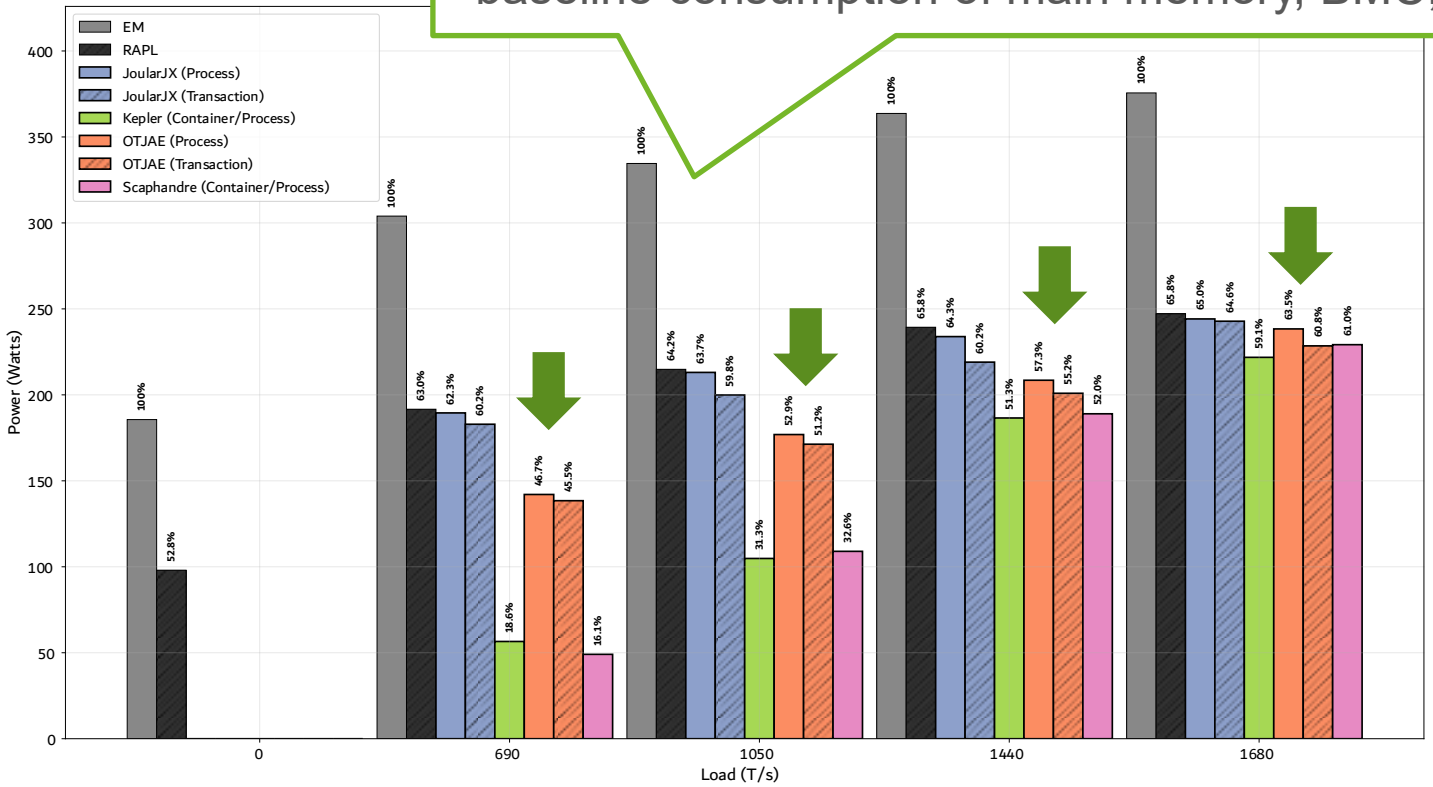


https://github.com/RETIT/opentelemetry-javaagent-extension/tree/main/examples/quarkus-rest-service

java -javaagent:opentelemetry-javaagent-all.jar -Dotel.javaagent.extensions=io.retit.opentelemetry.javaagent.extension.jar -Dio.retit.emissions.cloud.provider=***<your_cloud_provider>*** -Dio.retit.emissions.cloud.provider.region=***<your_cloud_region>*** -jar ***<your_app>***.jar

Accuracy

Inefficiency of power supplies (~4-12%), cooling, baseline consumption of main memory, BMC, …

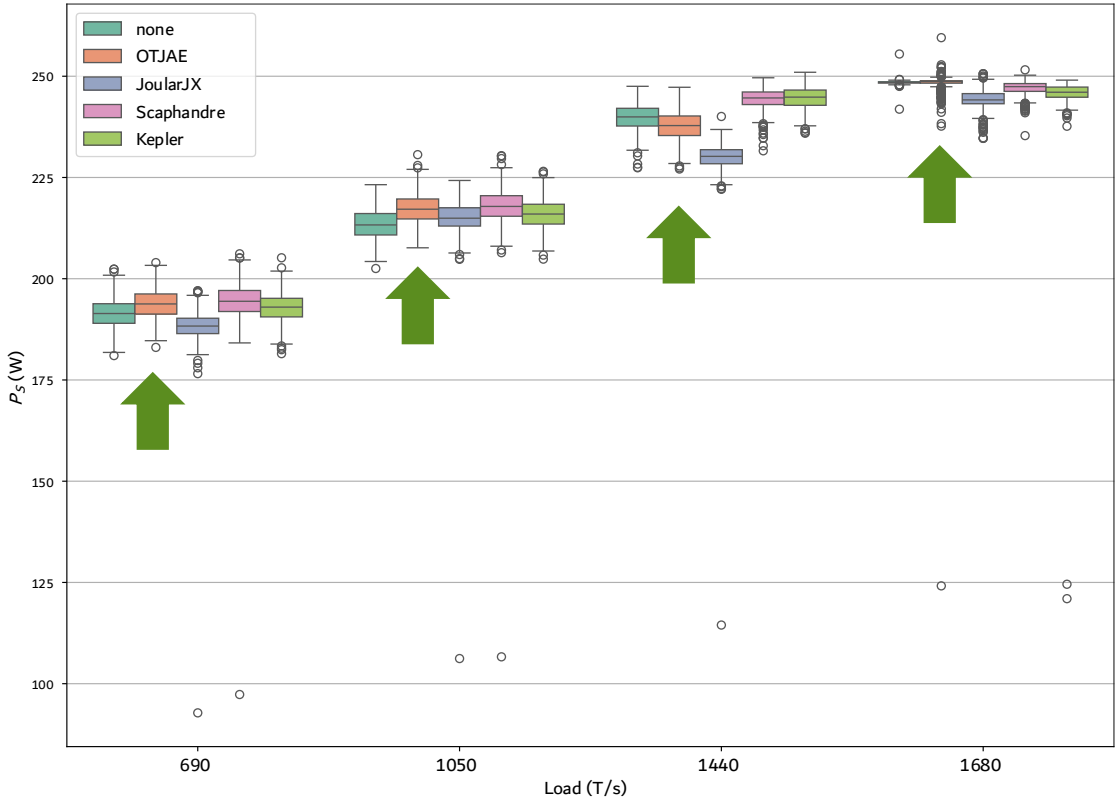


CPU Overhead

Joular⚡X

Load	CPU_{none}	CPU_{Kepler}	$CPU_{Scaphandre}$	CPU_{OTJAE}	$CPU_{JoularJX}$
0T/s	0.01%	0.08%	0.09%	0.02%	0.10%
690T/s	28.52%	29.35%	29.66%	29.23%	30.57%
1050T/s	48.58%	48.74%	51.61%	52.64%	54.31%
1440T/s	74.67%	77.10%	78.84%	73.76%	85.05%
1680T/s	94.30%	89.72%	95.89%	93.78%	93.44%

Power Overhead



References

- OpenTelemetry Java Agent:
  - https://github.com/open-telemetry/opentelemetry-java-instrumentation
- OpenTelemetry Java-Agent Extension:
  - https://github.com/RETIT/opentelemetry-javaagent-extension
- Papers about the OpenTelemetry Java-Agent Extension:
  - https://dl.acm.org/doi/10.1145/3629527.3652883
  - https://fb-swt.gi.de/fileadmin/FB/SWT/Softwaretechnik-Trends/Verzeichnis/Band_44_Heft_4/SSP24_16_camera-ready_5255.pdf
  - https://dl.acm.org/doi/10.1145/3696630.3728709
- Cloud Carbon Footprint – Methodology:
  - https://www.cloudcarbonfootprint.org/docs/methodology
- Cloud Carbon Footprint – Coefficients:
  - https://github.com/cloud-carbon-footprint/ccf-coefficients

Thank you for your attention!

Questions?

Andreas Brunnert
brunnert@retit.de

RETIT
Resource Efficient Technologies & IT Systems