How to measure CO2 emissions for every API call of your microservices

Andreas Brunnert

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

EcoCompute 2024

Agenda

1. Software CO2 Emissions

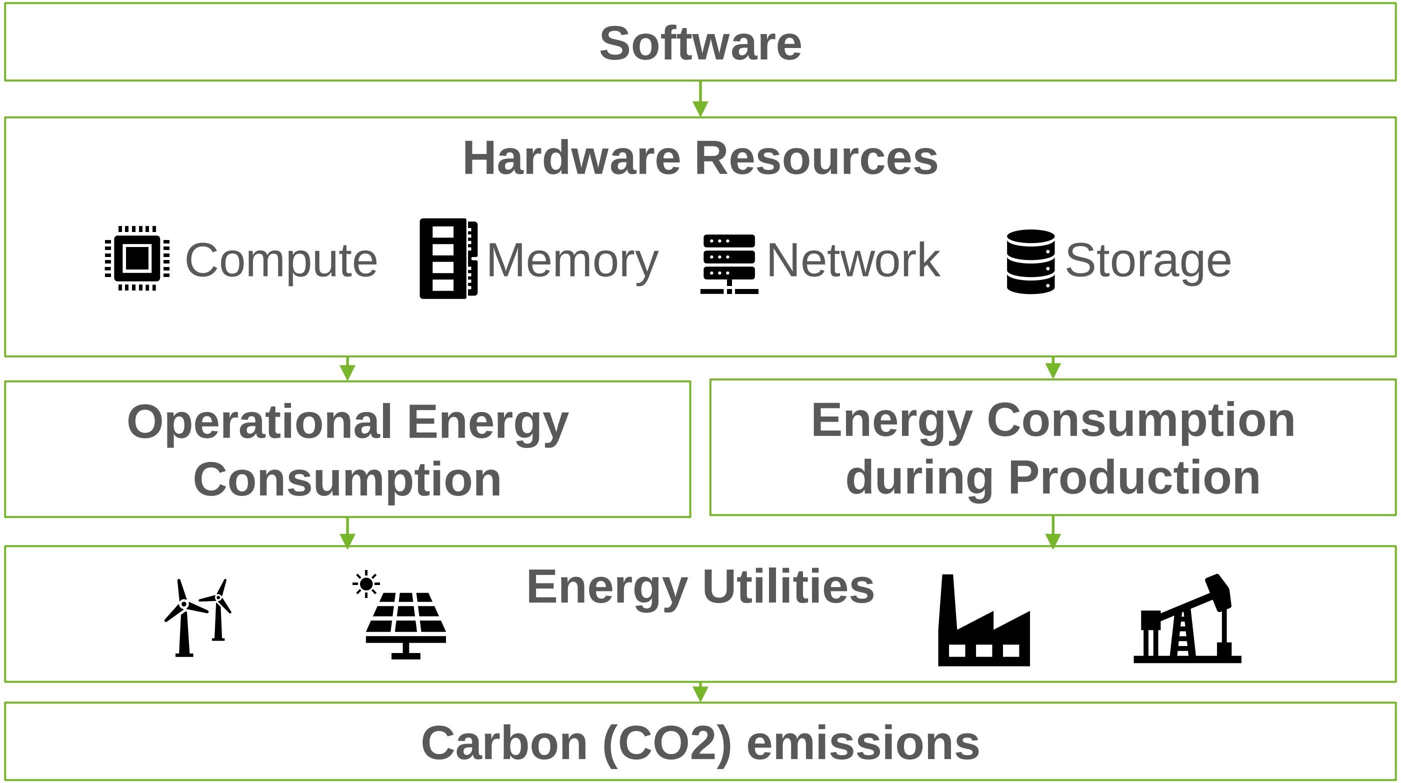
2. Software Carbon Intensity (SCI) Specification

3. Limitations of the SCI Specification

4. How to get to CO2 values for single API calls?

5. Demo!

Software CO2 Emissions



Software

Hardware Resources

Compute Memory Network Storage

Operational Energy Consumption

Energy Consumption during Production

Energy Utilities

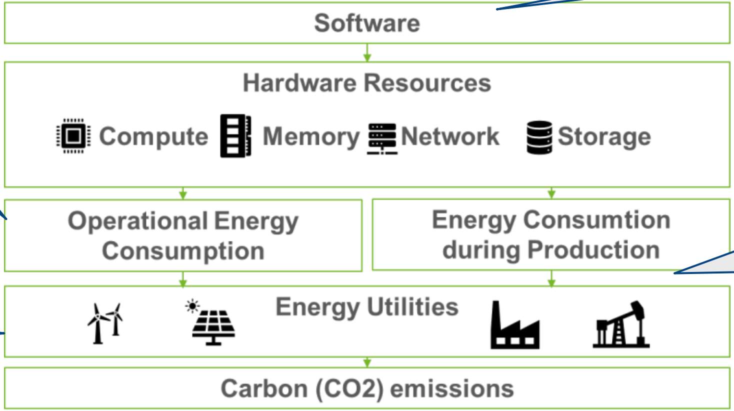
Carbon (CO2) emissions

Software Carbon Intensity (SCI) Specification

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

E = This is the energy consumed by a software system for a functional unit of work.

R = Functional Unit; this is how software scales, for example per user or per device



Software

Hardware Resources

Compute Memory Network Storage

Operational Energy Consumption

Energy Consumtion during Production

Energy Utilities

Carbon (CO2) emissions

I = Carbon emitted per kWh of energy, gCO2/kWh

M = Embodied carbon of the hardware that the software is running on

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

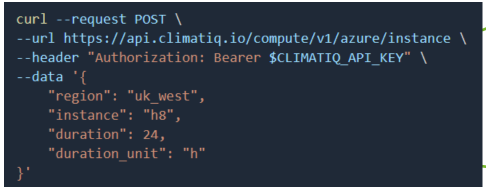
Software Carbon Intensity (SCI) Specification

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

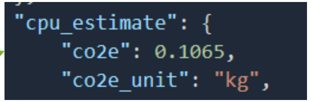
- Example: Microservice running on a Azure VM of type h8 (8CPU, 56 GB RAM) in region uk_west and scales according to one API and has around 200.000 calls per day

$$SCI = ((106{,}5gCO2e + 138{,}2gCO2e) + 498{,}9gCO2e) / 200.000$$

$$= 0{,}003718 \text{ gCO2e per API call}$$

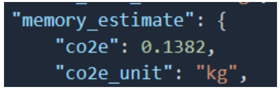


```
curl --request POST \
--url https://api.climatiq.io/compute/v1/azure/instance \
--header "Authorization: Bearer $CLIMATIQ_API_KEY" \
--data '{
    "region": "uk_west",
    "instance": "h8",
    "duration": 24,
    "duration_unit": "h"
}'
```



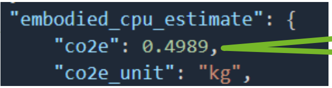
```
"cpu_estimate": {
    "co2e": 0.1065,
    "co2e_unit": "kg",
```

(E * I) for CPU (8 Cores)



```
"memory_estimate": {
    "co2e": 0.1382,
    "co2e_unit": "kg",
```

(E * I) for Memory (56GB)



```
"embodied_cpu_estimate": {
    "co2e": 0.4989,
    "co2e_unit": "kg",
```

(M) for CPU (8 Cores)

Data sources: https://www.climatiq.io/docs/api-reference/computing and https://azure.microsoft.com/en-us/pricing/details/virtual-machines/linux/#pricing

Software Carbon Intensity (SCI) Specification

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

- Alternative open source for ***E * I*** and ***M*** if you don‘t want to use climatiq: https://www.cloudcarbonfootprint.org/
- Alternative sources for ***E*** if you are not on one of the major clouds:
  - Ask your provider
  - Own measurements
    - Using energy meters
    - Software tools like RAPL / IPMI
- Alternative sources for ***I*** if you are not on one of the major clouds:
  - https://app.electricitymaps.com/
- Alternative sources for ***M*** if you are not on one of the major clouds or not interested in buying Climatiq API keys:
  - https://sci-guide.greensoftware.foundation/M/Datasets/

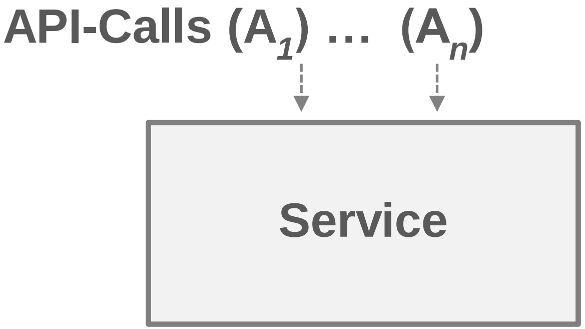
Limitations of the SCI Specification

- So, SCI is great, right?

- SCI takes only one functional unit (R) into account
  - *How do you handle multiple functional units (e.g., API calls)?*

- Current tooling to get the E, I and M values is mainly applicable for dedicated runtimes like virtual machines or bare metal servers
  - *How do you handle serverless systems or functions only allocating resources during request processing?*

How to get to CO2 values for single API calls?



API-Calls (A_1) … (A_n)

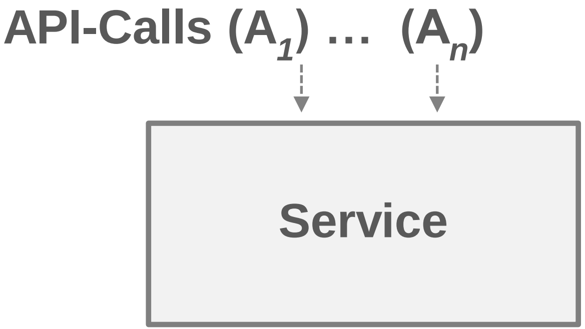
Service

$SCI\ for\ A_1 = ((E * I) + M)\ per\ A_1$

…

$SCI\ for\ A_n = ((E * I) + M)\ per\ A_n$

How to get to CO2 values for single API calls?



$$SCI\ for\ A_1 = ((E * I) + M)\ per\ A_1$$

…

$$SCI\ for\ A_n = ((E * I) + M)\ per\ A_n$$

$$\begin{pmatrix} d_{CPU} \\ d_{STOr} \\ d_{STOw} \\ d_{MEM} \\ d_{NETi} \\ d_{NETo} \end{pmatrix} * \begin{pmatrix} c_{CPU} \\ c_{STO} \\ c_{STO} \\ c_{MEM} \\ c_{NET} \\ c_{NET} \end{pmatrix} = \begin{pmatrix} c_{CPU/A} \\ c_{STOr/A} \\ c_{STOw/A} \\ c_{MEM/A} \\ c_{NETi/A} \\ c_{NETo/A} \end{pmatrix}$$

d = Resource Demand
c = carbon emissions
STO = Storage
r = read
w = write
MEM = Memory
NET = Network
i = Input
o = Output

$$SCI\ for\ A = c_A = c_{CPU/A} + c_{STOr/A} + c_{STOw/A} + c_{MEM/A} + c_{NETi/A} + c_{NETo/A}$$

How to get to CO2 values for single API calls?

In order to get the resource demands (d) you need to measure them within the service:

```
YourService {
    yourAPI {
        $d_{before}$ = measureResourceDemandBefore()

        doBusinessWork(…)

        $d_{after}$ = measureResourceDemandAfter()

        $d = d_{after} - d_{before}$
    }
}
```

$$\begin{pmatrix} d_{CPU} \\ d_{STO_r} \\ d_{STO_w} \\ d_{MEM} \\ d_{NET_i} \\ d_{NET_o} \end{pmatrix} * \begin{pmatrix} c_{CPU} \\ c_{STO} \\ c_{STO} \\ c_{MEM} \\ c_{NET} \\ c_{NET} \end{pmatrix} = \begin{pmatrix} c_{CPU/A} \\ c_{STO_r/A} \\ c_{STO_w/A} \\ c_{MEM/A} \\ c_{NET_i/A} \\ c_{NET_o/A} \end{pmatrix}$$

How to get to CO2 values for single API calls?

$$\begin{pmatrix} d_{CPU} \\ d_{STO_r} \\ d_{STO_w} \\ d_{MEM} \\ d_{NET_i} \\ d_{NET_o} \end{pmatrix} * \begin{pmatrix} c_{CPU} \\ c_{STO} \\ c_{STO} \\ c_{MEM} \\ c_{NET} \\ c_{NET} \end{pmatrix} = \begin{pmatrix} c_{CPU/A} \\ c_{STO_r/A} \\ c_{STO_w/A} \\ c_{MEM/A} \\ c_{NET_i/A} \\ c_{NET_o/A} \end{pmatrix}$$

In Java this looks like this for d_{CPU}

```
YourService {
    yourAPI {

        ThreadMXBean mxBean = ManagementFactory.getThreadMXBean();
        $d_{CPU-before}$ = mxBean. getCurrentThreadCpuTime()

        doBusinessWork(…)

        $d_{CPU-after}$ = mxBean. getCurrentThreadCpuTime()

        $d_{CPU} = d_{CPU-after} – d_{CPU-before}$
    }
}
```

- For d_{MEM} in Java you can do the same with com.sun.management. ThreadMXBean. getCurrentThreadAllocatedBytes()

How to get to CO2 values for single API calls?

$$\begin{pmatrix} d_{CPU} \\ d_{STO_r} \\ d_{STO_w} \\ d_{MEM} \\ d_{NET_i} \\ d_{NET_o} \end{pmatrix} * \begin{pmatrix} c_{CPU} \\ c_{STO} \\ c_{STO} \\ c_{MEM} \\ c_{NET} \\ c_{NET} \end{pmatrix} = \begin{pmatrix} c_{CPU/A} \\ c_{STO_r/A} \\ c_{STO_w/A} \\ c_{MEM/A} \\ c_{NET_i/A} \\ c_{NET_o/A} \end{pmatrix}$$

- Similar approaches are available for many programming languages

- If your programming language does not provide direct APIs for reading resource demands you can read the proc file system on Linux [1] (or in Containers [2]):

  - /proc/<process-id>/task/<thread-id>/stat file for CPU demands
  - /proc/<process-id>/task/<thread-id>/io file for STO demands
  - /proc/<process-id>/task/<thread-id>/mem file for MEM demands
  - /proc/<process-id>/task/<thread-id>/net/dev file for NET demands

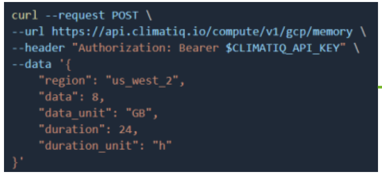
- If you are not interested in doing this on your own, you are welcome to use our service at https://www.retit.io/
- [1] See https://www.man7.org/linux/man-pages/man5/proc.5.html for details
- [2] if you are running on Windows, please note that you cannot measure CPU demands below 15ms

How to get to CO2 values for single API calls?

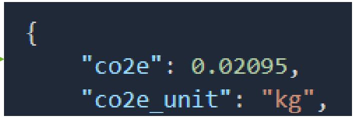
- In order to get to carbon emissions of single resources you can again do own measurements or just use Climatiq

- Climatiq is able to return the carbon emissions of single resources (CPU/MEM/STO) in a given region for a given time interval (s,m,h,…)
  (Source: https://www.climatiq.io/docs/api-reference/computing)

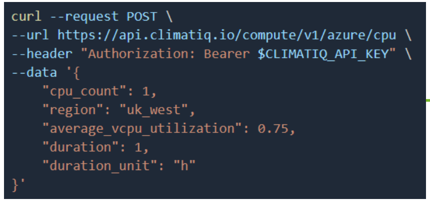
$$\begin{pmatrix} d_{CPU} \\ d_{STO_r} \\ d_{STO_W} \\ d_{MEM} \\ d_{NET_i} \\ d_{NET_O} \end{pmatrix} * \begin{pmatrix} c_{CPU} \\ c_{STO} \\ c_{STO} \\ c_{MEM} \\ c_{NET} \\ c_{NET} \end{pmatrix} = \begin{pmatrix} c_{CPU/A} \\ c_{STO_r/A} \\ c_{STO_W/A} \\ c_{MEM/A} \\ c_{NET_i/A} \\ c_{NET_O/A} \end{pmatrix}$$



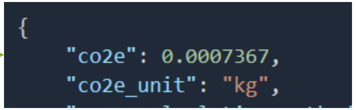
```
curl --request POST \
--url https://api.climatiq.io/compute/v1/gcp/memory \
--header "Authorization: Bearer $CLIMATIQ_API_KEY" \
--data '{
    "region": "us_west_2",
    "data": 8,
    "data_unit": "GB",
    "duration": 24,
    "duration_unit": "h"
}'
```



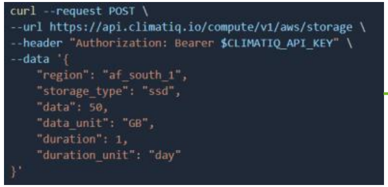
```
{
    "co2e": 0.02095,
    "co2e_unit": "kg",
```



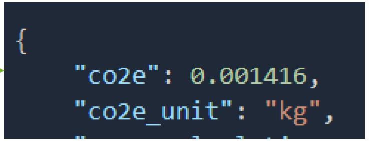
```
curl --request POST \
--url https://api.climatiq.io/compute/v1/azure/cpu \
--header "Authorization: Bearer $CLIMATIQ_API_KEY" \
--data '{
    "cpu_count": 1,
    "region": "uk_west",
    "average_vcpu_utilization": 0.75,
    "duration": 1,
    "duration_unit": "h"
}'
```



```
{
    "co2e": 0.0007367,
    "co2e_unit": "kg",
```



```
curl --request POST \
--url https://api.climatiq.io/compute/v1/aws/storage \
--header "Authorization: Bearer $CLIMATIQ_API_KEY" \
--data '{
    "region": "af_south_1",
    "storage_type": "ssd",
    "data": 50,
    "data_unit": "GB",
    "duration": 1,
    "duration_unit": "day"
}'
```



```
{
    "co2e": 0.001416,
    "co2e_unit": "kg",
```

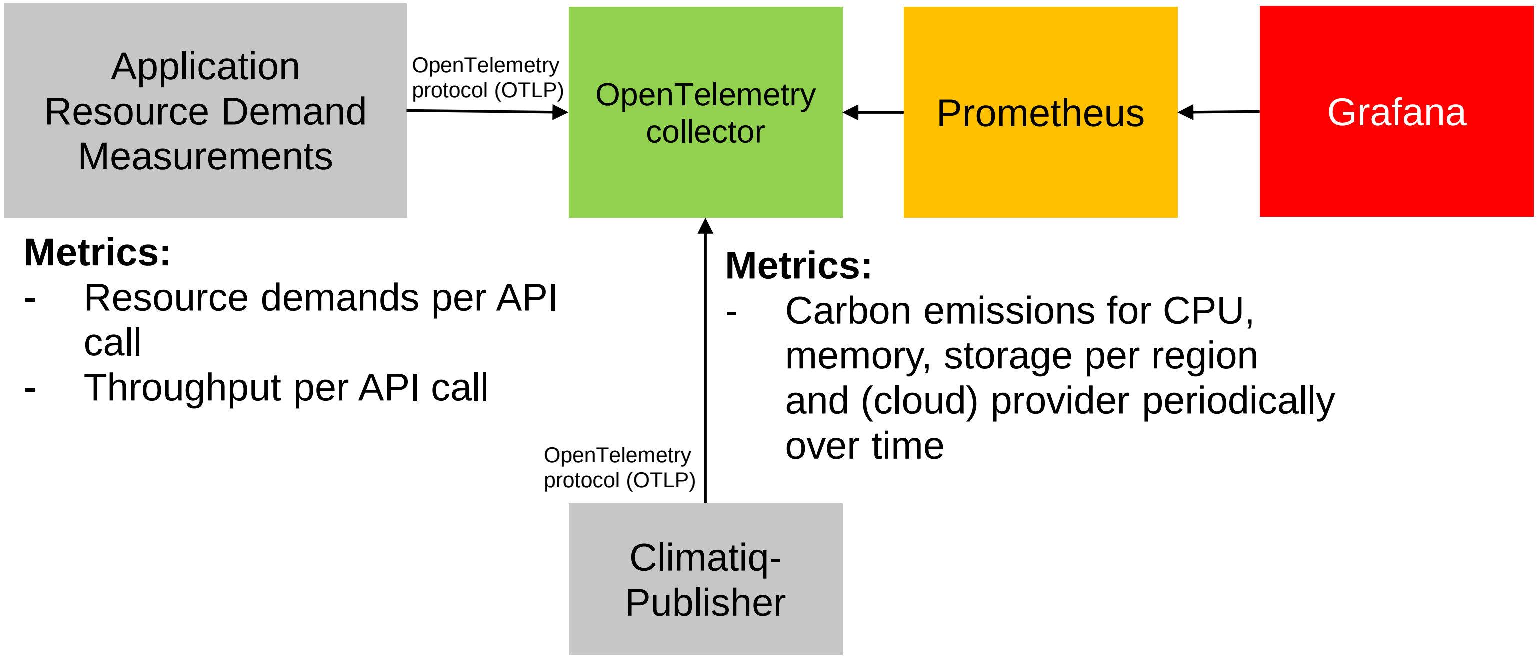
How to get to CO2 values for single API calls?

- Example for an example API Call (A) TestService.getData() (HTTP GET REST Endpoint)
  - Resource Demand and Carbon Emission Data collected for one minute (60s)
  - The API call was executed 100 times in a minute (60s)
  - Application did not read or write from Storage

$$\begin{pmatrix} d_{CPU} = 75s \\ d_{STO_r} = 0GB \\ d_{STO_w} = 0GB \\ d_{MEM} = 2GB \\ d_{NET_i} = 0{,}3GB \\ d_{NET_o} = 0{,}1GB \end{pmatrix} * \begin{pmatrix} c_{CPU} = 0{,}2mgCO2e/s \\ c_{STO} = 0{,}0001mgCO2e/GB \\ c_{STO} = 0{,}0001mgCO2e/GB \\ c_{MEM} = 0{,}04mgCO2e/GB \\ c_{NET} = 0{,}1mgCO2e/GB \\ c_{NET} = 0{,}1mgCO2e/GB \end{pmatrix} = \begin{pmatrix} c_{CPU} = 15mgCO2e \\ c_{STO_r} = 0 \\ c_{STO_w} = 0 \\ c_{MEM} = 0{,}08mgCO2e \\ c_{NET_i} = 0{,}03mgCO2e \\ c_{NET_o} = 0{,}01mgCO2e \end{pmatrix} / \begin{pmatrix} 100 \\ 100 \\ 100 \\ 100 \\ 100 \\ 100 \end{pmatrix} = \begin{pmatrix} c_{CPU/A} = 0{,}15mgCO2e \\ c_{STO_r/A} = 0 \\ c_{STO_w/A} = 0 \\ c_{MEM/A} = 0{,}0008mgCO2e \\ c_{NET_i/A} = 0{,}0003mgCO2e \\ c_{NET_o/A} = 0{,}0001mgCO2e \end{pmatrix}$$

$$SCI\ for\ A = c_A = c_{CPU/A} + c_{STO_r/A} + c_{STO_w/A} + c_{MEM/A} + c_{NET_i/A} + c_{NET_O/A} = c_A = 0{,}1512mgCO2e$$

How to get to CO2 values for single API calls?



Application Resource Demand Measurements

OpenTelemetry protocol (OTLP)

OpenTelemetry collector

Prometheus

Grafana

Metrics:

- Resource demands per API call
- Throughput per API call

OpenTelemetry protocol (OTLP)

Metrics:

- Carbon emissions for CPU, memory, storage per region and (cloud) provider periodically over time

Climatiq-Publisher

Demo!

Example Quarkus-based Microservice that emits resource demand and emission data including a Grafana Dashboard for single API Calls

https://github.com/RETIT/quarkus-carbon-emissions

Example Spring-based Microservice that emits resource demand and emission data including a Grafana Dashboard for single API Calls

https://github.com/RETIT/spring-carbon-emissions

Thanks a lot for your attention.

Questions?

Andreas Brunnert

brunnert@retit.de

Resource Efficient Technologies & IT Systems