


Software Carbon Intensity

Ivano Malavolta

	International Standard
Information technology — Software Carbon Intensity (SCI) specification	ISO/IEC 21031:2024
Reference number ISO/IEC 21031:2024	Edition 1 2024-03
© ISO 2024	

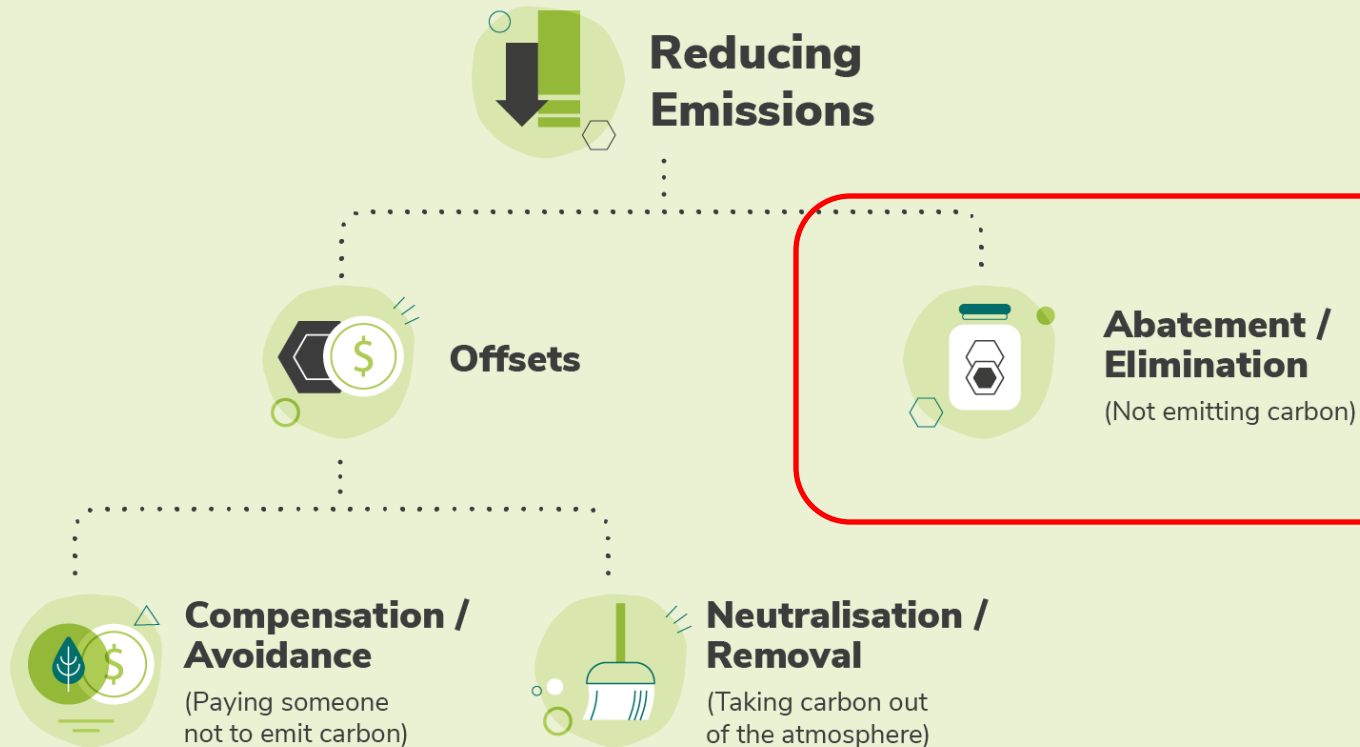


LOOKING FURTHER

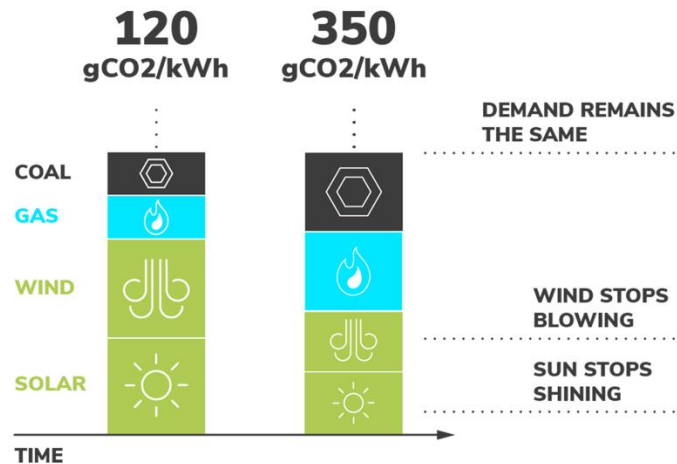
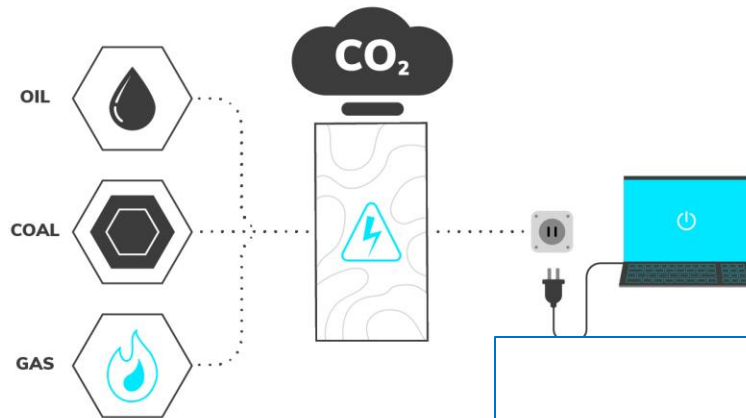
Roadmap

- Basic concepts
- The Software Carbon Intensity formula
- Running example
- Procedure (step by step)
 - Bound
 - Scale
 - Define
 - Quantify
 - Report

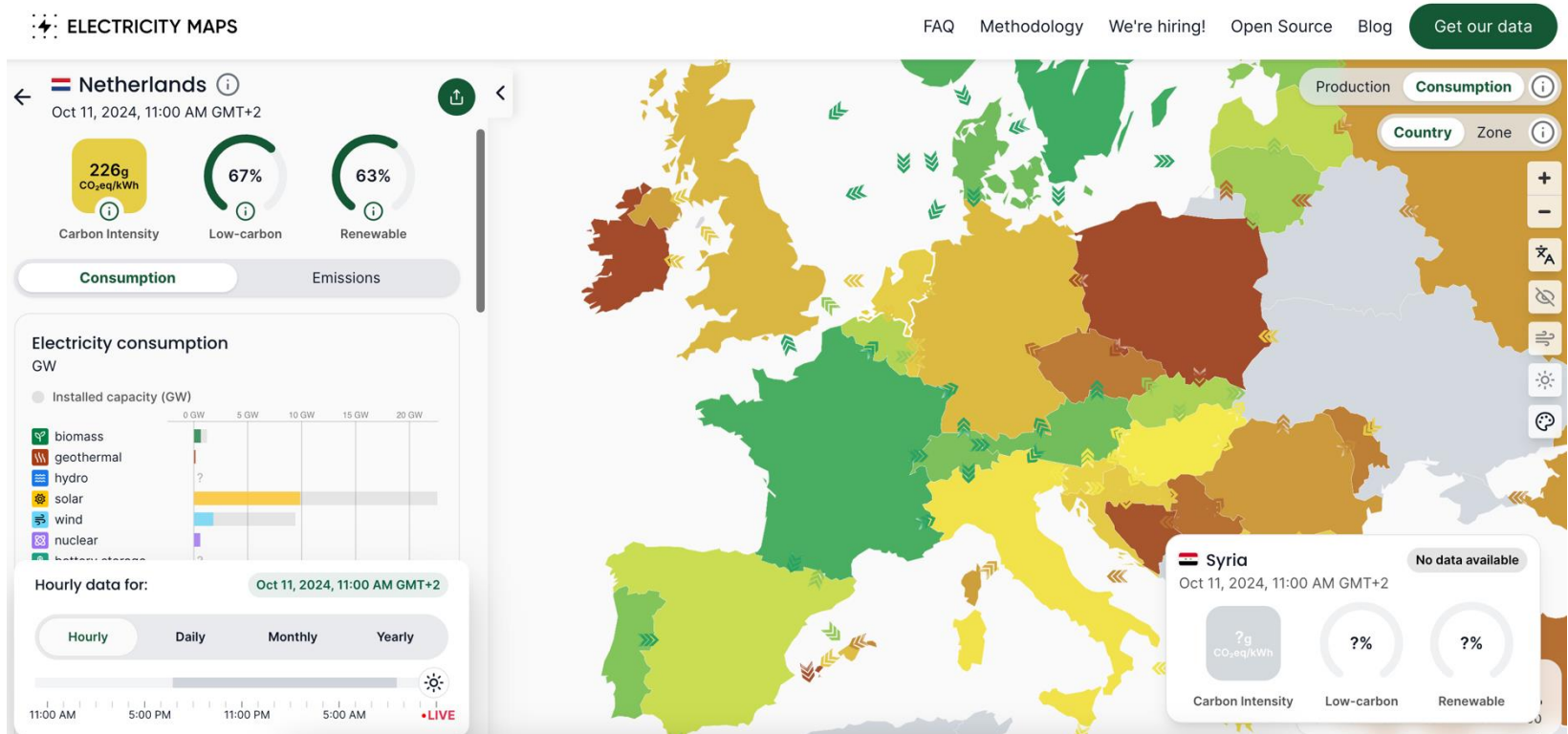
Basic concepts: Reducing carbon emissions



Basic concepts: Carbon intensity



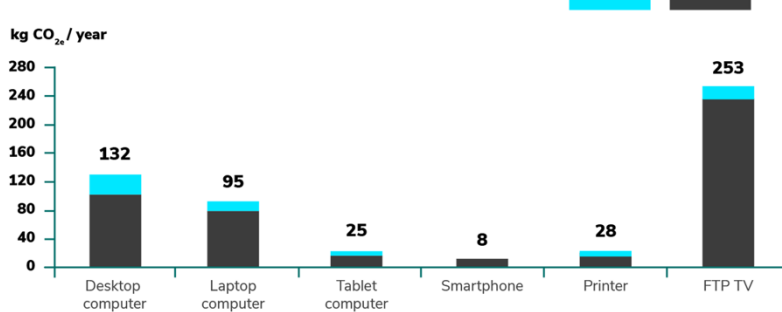
Basic concepts: Carbon intensity



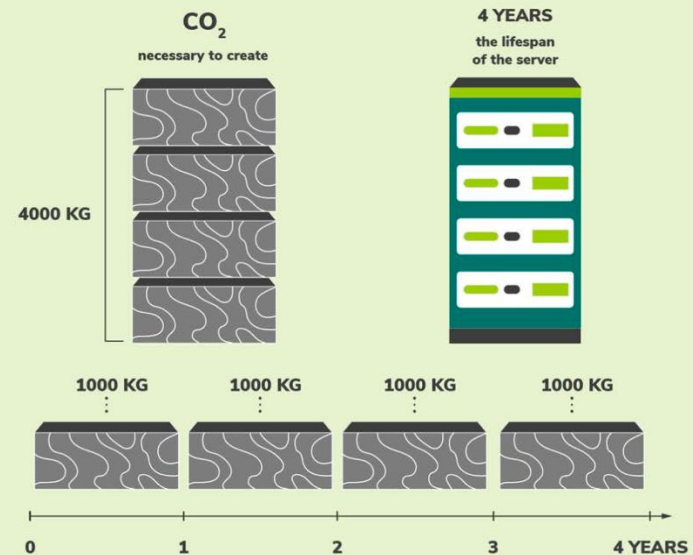
<https://app.electricitymaps.com/zone/NL>
<https://www.electricitymaps.com/methodology>

Basic concepts: embodied emissions

CO_{2e} emission per ICT end user device



Carbon amortization + utilization



Basic concepts: Strategies for greener software



Green Software Principles



Energy Efficiency

Consume the least amount of electricity possible



Hardware Efficiency

Use the least amount of embodied carbon possible



Carbon Awareness

Do more when the electricity is clean and less when it's dirty

The Software Carbon Intensity formula

Carbon emitted per kWh
of energy, gCO₂/kWh

Carbon emitted through
the hardware that the
software is running on

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

Energy consumed by
software in kWh

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

Running example

Goal: to calculate the SCI of a software application called “MovieRecommender” running on a Google Cloud VM

Scaling factor: #API requests

Average monthly requests: 20k

Region: US-East*

VM config: [e2-standard-4](#)

Machine types	vCPUs	Memory (GB)	Max number of Persistent Disk (PDs) [†]	Max total PD size (TiB)	Local SSD	Maximum egress bandwidth (Gbps) [‡]
e2-standard-2	2	8	128	257	No	4
e2-standard-4	4	16	128	257	No	8
e2-standard-8	8	32	128	257	No	16
e2-standard-16	16	64	128	257	No	16
e2-standard-32	32	128	128	257	No	16

[†] Persistent Disk and Hyperdisk usage is charged separately from [machine pricing](#).
[‡] Maximum egress bandwidth cannot exceed the number given. Actual See [Network bandwidth](#).

Procedure

1. **BOUND:** Decide on the software boundary, i.e., the components of a software system to include.
2. **SCALE:** As the SCI is a rate (carbon emissions per one functional unit), pick the functional unit which best describes how the application scales.
3. **DEFINE:** For each software component of your software boundary, decide on the quantification method.

Examples: real-world measurements, based on telemetry, or lab-based measurements, based on models.

4. **QUANTIFY:** Calculate a rate for every software component.

The SCI value of the whole application is the sum of the SCI values for every software component in the system

5. **REPORT:** Disclose the SCI score, software boundary, and the calculation methodology

Software boundary - Step 1

Software boundary = all supporting infrastructure and systems that significantly contribute to the software's operation

Examples:

- compute resources
- storage
- networking equipment
- memory
- monitoring
- idle machines
- logging
- scanning
- build and deploy pipelines
- testing
- ...

MovieRecommender

If evaluating a complex system, compute the SCI of one component at a time, then aggregate

NOTE: If the boundary includes on-premise and/or cloud data center operations, **E** should take into account the efficiency of the data center, including cooling and other energy consumption necessary to operate a data center. The data center's energy efficiency is usually available as a **PUE (Power Usage Effectiveness)** value.

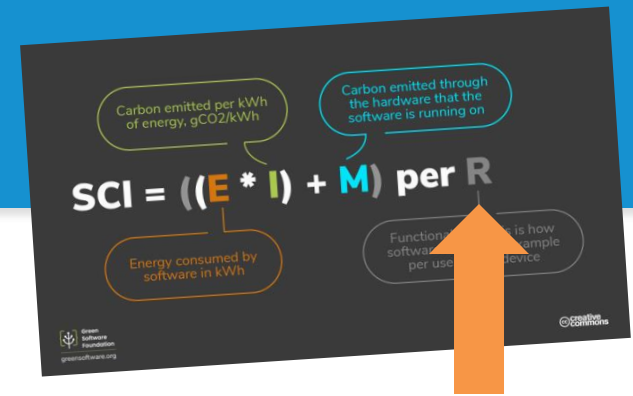
Boundary:

- all services of the application
- the infrastructure for running them (vCPUs and server only)

We are leaving out: memory, disk, racks, cooling water resources, CI/CD pipelines, etc.

Functional unit (R) - Step 2

Functional unit defines how your software scales



The goal of the SCI is to quantify how much carbon is emitted per one unit of R.

Examples:

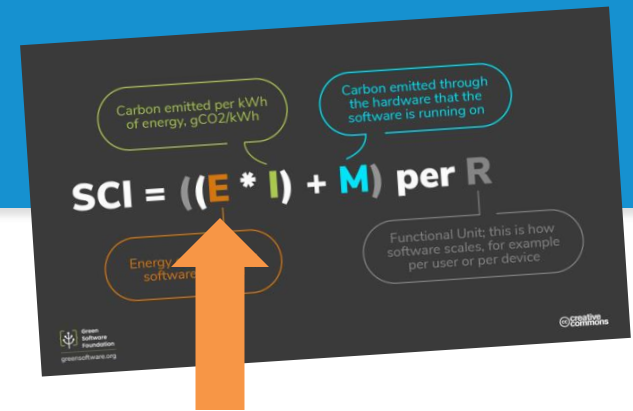
MovieRecommender Functional unit: one API call

- #API call/request
- Benchmarks executed
- #users
- Minute/time unit
- Devices
- Physical site
- Data volume
- Batch/Scheduled Jobs
- Database transactions

What would be R for your Green Lab project?

It must be **consistent** across all the components in the software boundary

Energy (E) - Step 3



E is the energy consumed by a software system for a functional unit of work R

Unit of measure: kWh

Four main strategies to compute E:

1. Tool-based [**MEASURED**]
2. API-based techniques [**CALCULATED**]
3. Performance engineering techniques [**CALCULATED**]
4. Public datasets [**CALCULATED**]

E -- Tool based

In tool-based approaches you integrate your software with tools that are directly measuring the energy consumed for each functional unit R.

Examples of useful tools:

- PyJoular
- energibridge
- Scaphandre
- any RAPL wrapper
but check your SCI boundary!
- etc.

This is what you are doing
in the Green Lab

Extensive list [here](#)

E -- API-based techniques

You integrate your software with APIs that provide at runtime energy values for VM instances, based on CPU usage, instance type, location and duration of usage, etc.

Examples of available APIs:

- [Climatiq](#)
- [CloudCarbonFootprint](#)

Available APIs typically report Energy Carbon Intensity ($E * I$), not only E !
The SCI specification calls it Operational emissions O .

Example of Climatiq request and response

```
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"  
'
```

<https://www.climatiq.io/docs/api-reference/computing#vm-instance>

((E * I)) for memory

((E * I)) for CPU

```
{  
  "total_co2e": 0.7436,  
  "total_co2e_unit": "kg",  
  "memory_estimate": {  
    "co2e": 0.1382,  
    "co2e_unit": "kg",  
    "co2e_calculation_method": "ar5",  
    "co2e_calculation_origin": "source",  
    "emission_factor": {  
      "name": "Electricity supplied from grid",  
      "activity_id": "electricity-supply_grid-source_supplier_  
id": "efac3e9c-89c4-4ac0-9af6-d2bb428302d8",  
      "access_type": "public",  
      "source": "BEIS",  
      "source_dataset": "Greenhouse gas reporting: conversion  
year": 2024,  
      "region": "GB",  
      "category": "Electricity",  
      "source_lca_activity": "electricity_generation",  
      "data_quality_flags": []  
    },  
    "constituent_gases": {  
      "co2e_total": 0.1382,  
      "co2e_other": null,  
      "co2": 0.1368,  
      "ch4": 0.00002143,  
      "n2o": 0.000003071  
    },  
    "activity_data": {  
      "activity_value": 0.6675,  
      "activity_unit": "kWh"  
    }  
  }  
}
```

```
"cpu_estimate": {  
  "co2e": 0.1065,  
  "co2e_unit": "kg",  
  "co2e_calculation_method": "ar5",  
  "co2e_calculation_origin": "source",  
  "emission_factor": {  
    "name": "Electricity supplied from grid",  
    "activity_id": "electricity-supply_grid-source_supplier_  
id": "efac3e9c-89c4-4ac0-9af6-d2bb428302d8",  
    "access_type": "public",  
    "source": "BEIS",  
    "source_dataset": "Greenhouse gas reporting: conversion  
year": 2024,  
    "region": "GB",  
    "category": "Electricity",  
    "source_lca_activity": "electricity_generation",  
    "data_quality_flags": []  
  },  
  "constituent_gases": {  
    "co2e_total": 0.1065,  
    "co2e_other": null,  
    "co2": 0.1054,  
    "ch4": 0.00001651,  
    "n2o": 0.000002366  
  },  
  "activity_data": {  
    "activity_value": 0.5143,  
    "activity_unit": "kWh"  
  }  
}
```


E -- Performance engineering techniques

These techniques are generally about:

- 1) identifying which hardware components to consider in each machine (at least, CPU, GPU, memory)
- 2) collecting data from tech specs/datasets about the average power consumption P_i of each component i and summing up all of them into P
- 3) the application of the usual $E = P * t$ formula, where t is the total time window considered in the SCI calculation

Points of attention:

- Which hardware components to consider
It depends on your SCI boundaries
- The sources you use for collecting the P_i values
Example: [Intel Xeon Platinum 8270 datasheet](#)
- How to include utilization in your P (e.g., average CPU utilization)
Example: see [DEF formula](#)

E -- Public datasets

You use public sources and references for energy estimates for computing resources

Examples of available datasets:

- [Boavizta Cloud dataset](#)
- [Boavizta servers dataset](#)
- [Climatiq open data explorer](#)
- [Spec.org average Watts for CPUs](#)

As an alternative, Cloud Jewels coefficients (created by the Sustainable Task Force at Etsy):

- 2.10 Wh per vCPUh [Server]
- 0.89 Wh/TBh for HDD storage [Storage]
- 1.52 Wh/TBh for SSD storage [Storage]

Cloud Jewels coefficients are (very rough) fixed estimates, which can be used as final resort. See [here](#) for the details about the used method.

Example of calculation via a public dataset

MovieRecommender

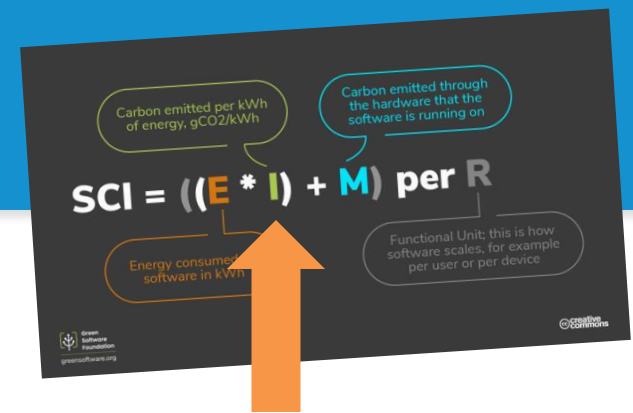
$$\text{Energy Carbon intensity (E * I)} = (0.0013 * 24 * 30) * 1000 = 936 \text{ gCO}_2\text{e}$$

✓	GCP (us-east-4) CPU ⓘ	CCF	2021	Virginia, US	NumberOver Time
	LCA Activity: use_phase				
	0.0013 kgCO ₂ e/CPU-hour				

Emission intensity for usage of a single CPU in kg CO₂e per hour for the Google Cloud Platform data centers in the given location using the assumptions and grid emissions factors available in the source as of date accessed. It is assumed that vCPU utilization is 50%. The source does not clarify if the kgCO₂e value is calculated using either IPCC Fourth Assessment Report (AR4) or IPCC Fifth Assessment Report (AR5) methodologies.

ACTIVITY ID	cpu-provider_gcp-region_us_east_4 ⓘ
ID	523aa3b8-81d2-468b-bcf2-846f48d86266 ⓘ
SOURCE	CCF
YEAR	2021
YEAR RELEASED	2021
REGION	Virginia, US (US-VA)
SECTOR	Information and Communication
CATEGORY	Cloud Computing - CPU
UNIT TYPE(S)	Number Over Time ⓘ
EMISSION FACTORS	CO ₂ e 0.001331 kg/CPU-hour ⓘ Data Quality
CO₂e CALCULATION METHOD	Method applied: AR4 Methods supported: AR4 Origin: Source

Energy carbon intensity (I) Step 4



Carbon emitted per kWh of energy, gCO2/kWh

Carbon emitted through the hardware that the software is running on

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

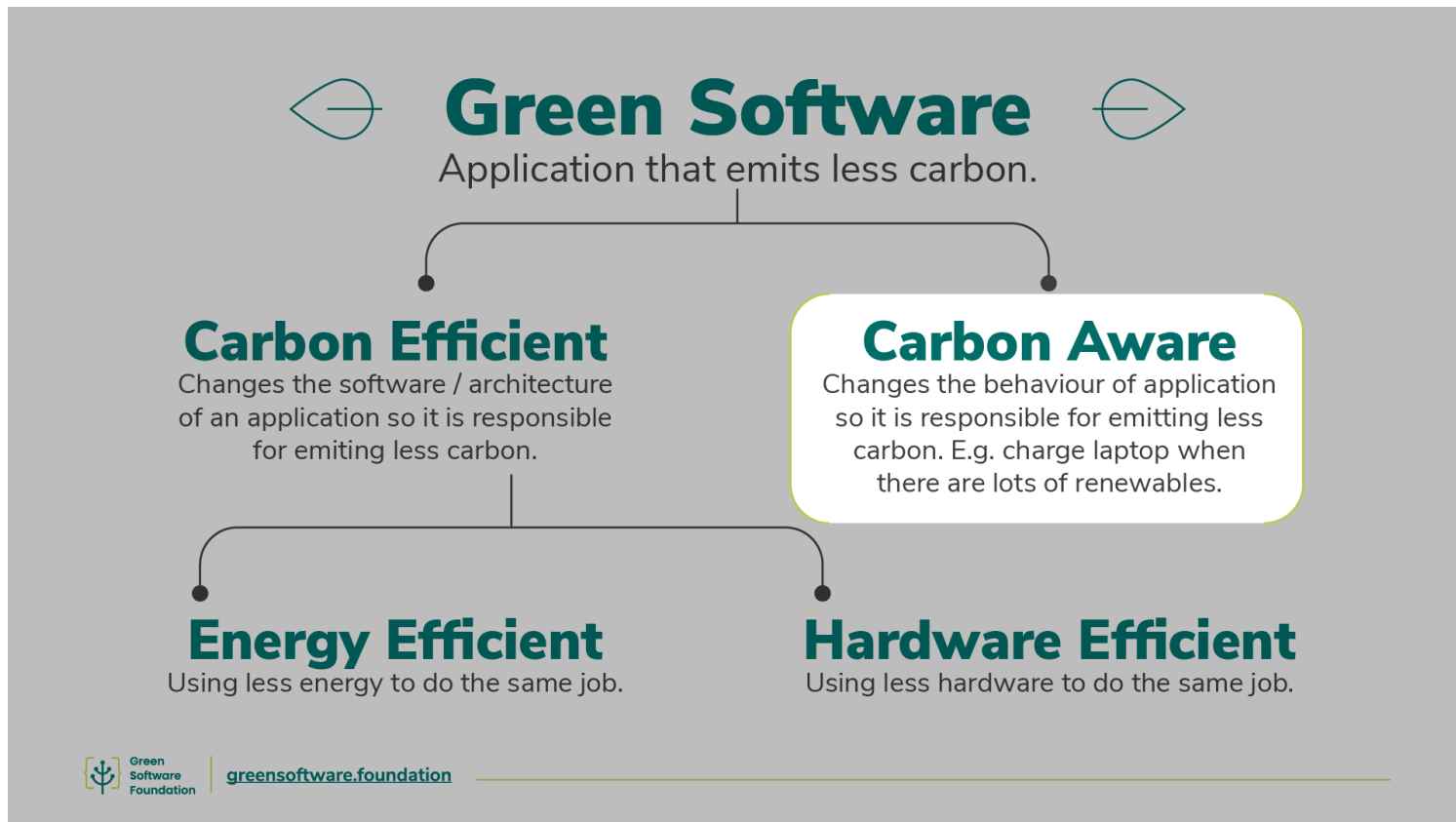
Energy consumed software in kWh

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

Green Software Foundation
greensoftware.org

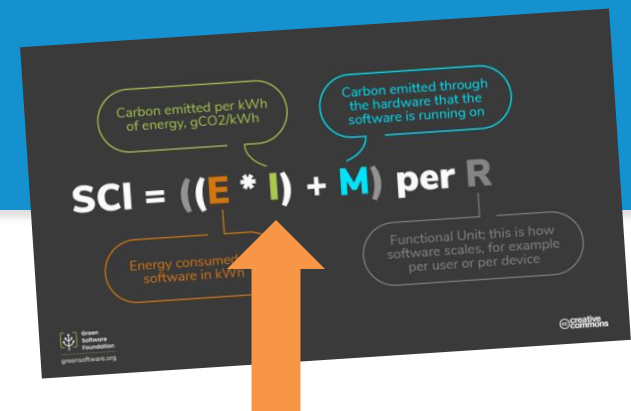
© 2018 Green Software Foundation

An orange arrow points from the 'Energy consumed software in kWh' callout to the 'E' in the formula.



Energy carbon intensity (I) Step 4

or better:
Location-Based Marginal Carbon Intensity



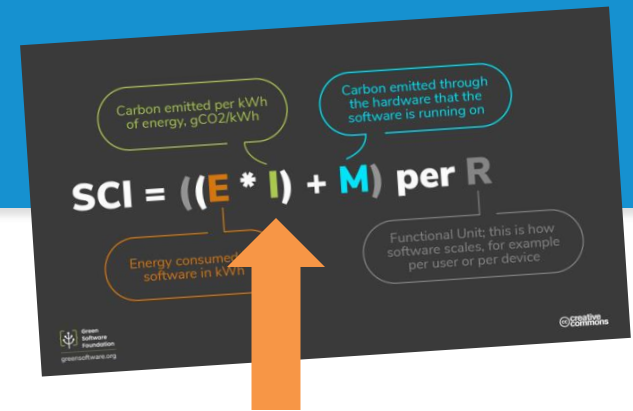
Carbon intensity = the measure of the greenhouse gas emissions associated with producing electricity

It is expressed in **gCO₂eq/kWh** - grams of carbon dioxide equivalents emitted per kilowatt hour of consumed electricity

No single method to calculate this, it depends primarily on the region where energy is consumed

Energy carbon intensity (I)

Step 4

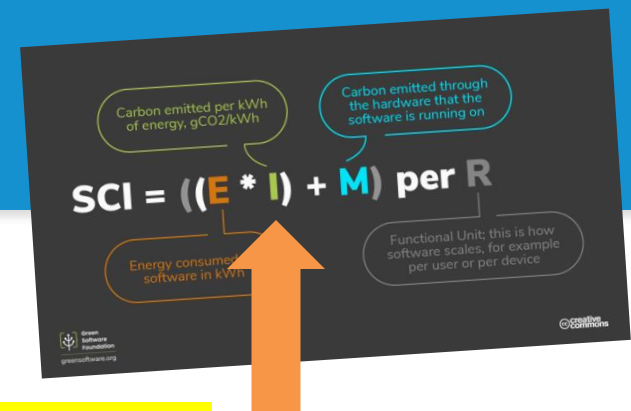


Two main strategies to follow to compute it:

- **API-based techniques:** your software uses real-time APIs and act based on the levels of carbon intensity
 - This is the preferred method
 - Examples: <https://codecarbon.io>
<https://app.electricitymaps.com>
<https://github.com/Green-Software-Foundation/carbon-aware-sdk>
- **Lookup open datasets:** you use historical databases
 - Example: <https://ourworldindata.org/grapher/carbon-intensity-electricity>

Energy carbon intensity (I)

Step 4



From a developer perspective, only the **location-based info** is important in terms of the impact on eliminating carbon emissions. This excludes market-based measures

Market-based measures are financial instruments designed to neutralize or offset carbon emissions (e.g., carbon credits, removal units)

IMPORTANT: the SCI specification also contains the definition of **operational emissions O**, defined as:

$$O = (E * I)$$

Note: Climatiq returns already O (not only E or I)

MovieRecommender

$$O = \text{Energy Carbon intensity } (E * I) = (0.0013 * 24 * 30) * 1000 = 936 \text{ gCO2e}$$

Another example

Use Emissions

Energy of software/VM (kWh)

×

Power usage effectiveness (PUE)

×

Carbon intensity of the grid
(gCO₂eq / kWh)

=

Greenhouse Gas Emissions
(gCO₂eq/h)


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

AWS m4.large : 2 CPU / 8 Gb RAM
50% use
• **10 Wh** = 0.01 kWh

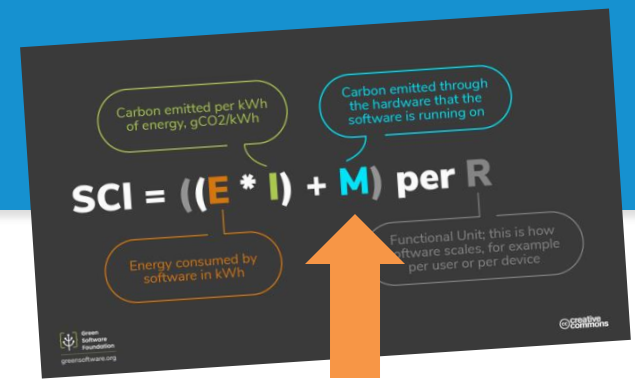
AWS Ireland
• PUE = **1.2**

316 gCO₂/kWh

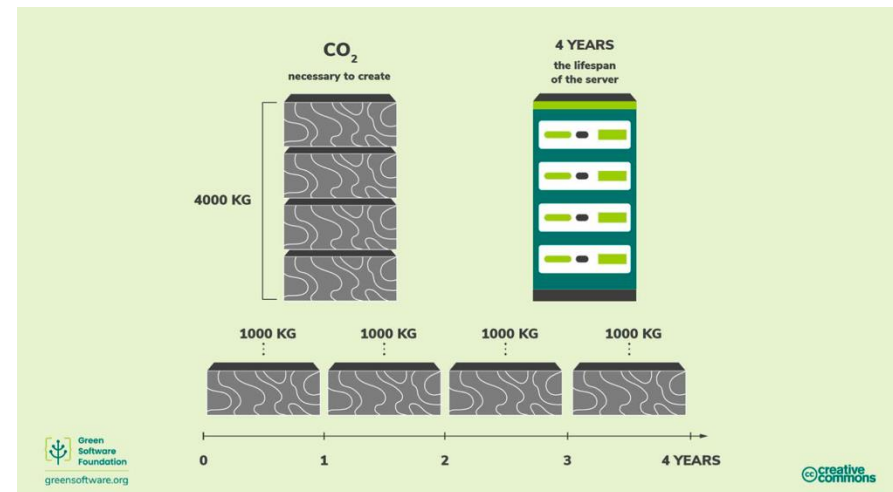
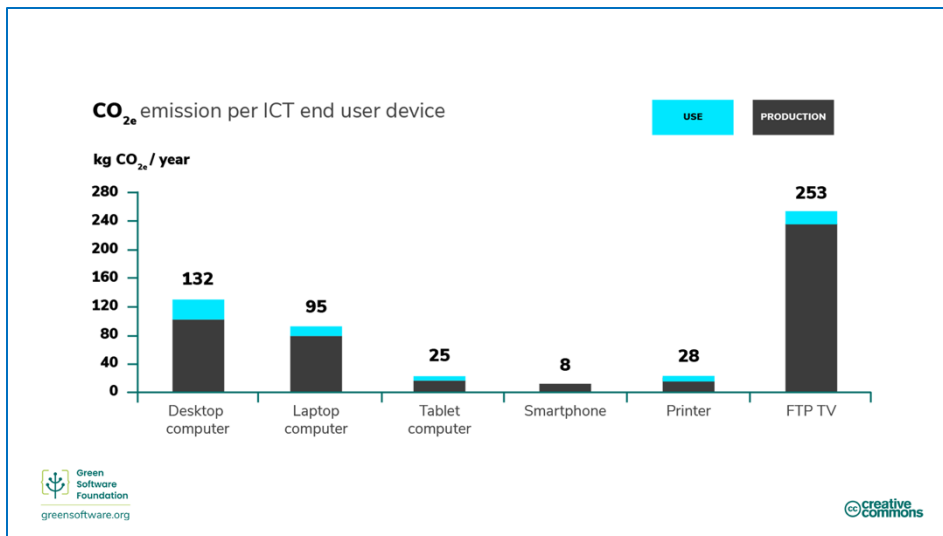
0.01 x 1.2 x 316 =
3.8 gCO₂eq/h

Embodied emissions (M)

Step 4



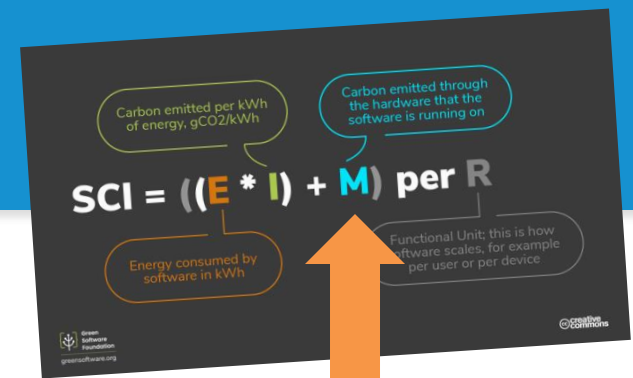
Embodied emissions = the amount of carbon emitted during the creation and disposal of a hardware device



Carbon amortization

Embodied emissions (M)

Step 4



M = the fraction of the total embodied emissions of the device is allocated to the software in grams of carbon (gCO₂eq)

$$M = TE * TS * RS$$

where:

TE = Total Embodied Emissions = the sum of Life Cycle Assessment (LCA) emissions for all hardware

components

TS = Time-share = the share of the total life span of the hardware reserved for use by the software

RS = Resource-share = the share of the total available resources of the hardware reserved for use by the software

Embodied emissions (M)

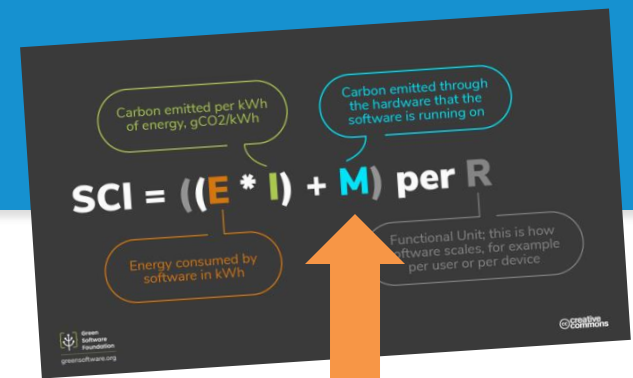
Step 4

	Time	Resources
$M = TE$	TS	RS
$M = TE$	(TiR/EL)	(RR/ToR)

where:

- TiR** = Time Reserved = the length of time the hardware is reserved for use by the software in hours/days/months/years
- EL** = Expected Lifespan = the anticipated time that the equipment will be installed in hours/days/months/years
- RR** = Resources Reserved = the amount of resources reserved for use by the software
- ToR** = Total Resources = the total amount of resources available

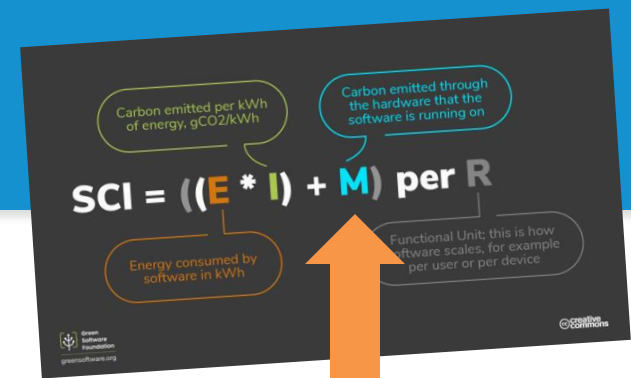
Similarly to *I*, market-based measures must not be included when computing *TE*.
The goal here is to **REDUCE** emissions!



Embodied emissions (M)

Step 4

$$M = TE * (TiR/EL) * (RR/ToR)$$



How to compute TE?

- Datasets for VMs in the Cloud

- [Cloud Carbon Footprint - Embodied Emissions constants](#) (refer to the “Total Platform Scope 3 Emissions (kgCO₂eq)” column)
- [Clamtiq API](#) (expressed per vCPU hour)

- Datasets for consumer devices

- [Boavizta user devices database](#)
- [Dell devices](#)
- etc.

MovieRecommender

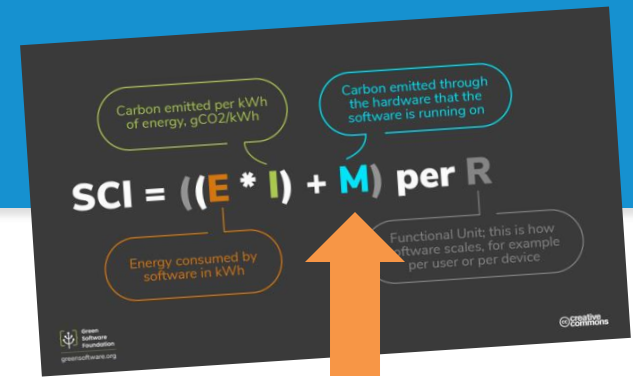
TE = 1230.3 kgCO₂eq
(Google Cloud e2-standard-4 instance – see [here](#))

- **Manual approach:** check the website of the vendor of your machine for LCA data (if not available, look up an identical bare metal machine on [spec.org](#) and repeat)

Embodied emissions (M)

Step 4

$$M = TE * (TiR/EL) * (RR/ToR)$$



How to compute **TiR**?

This depends on the length of time the SCI score is being reported for (which is part of R)

For example, if the SCI score is x kgCo₂eq for a software per 500 users per 1 day, then the time reserved should be 1 day or 24 hours

If using ClimaTiq data, it is the number of hours used in the ClimaTiq API

MovieRecommender

TiR = 1 month (this comes from our own time scale)

How to compute **EL**?

For most cloud servers or consumer devices, the expected lifetime is 4 years

→ ~35,040 hours

MovieRecommender

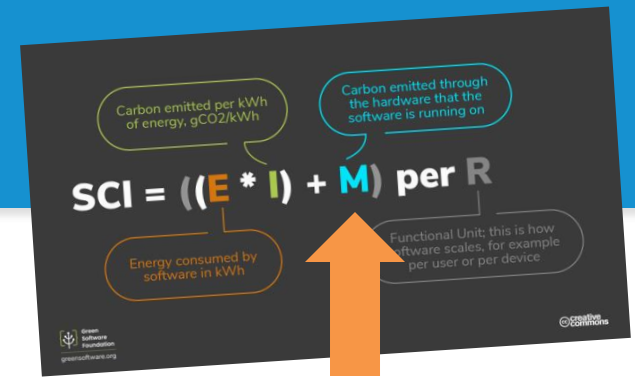
EL = 4 years (average lifespan for bare metal server)

You can also consider the number of years of warranty provided by the manufacturer

Embodied emissions (M)

Step 4

$$M = TE * (TiR/EL) * (RR/ToR)$$



How to compute RR?

- **Bare metal:** all hardware resources
- **Cloud:** the number of virtual resources reserved to the application

Examples: 8 vCPUs, 32Gb memory

MovieRecommender

RR = 4 (the number of vCPUs of e2-standard-4 instances in [Google Cloud](#))

How to compute ToR?

- **Bare metal:** the entire hardware resources are at the disposal of the application

Examples: 64Gb of memory, 8 CPUs, etc.

- **Cloud:** the virtual resources of the largest instance for the given family of VMs

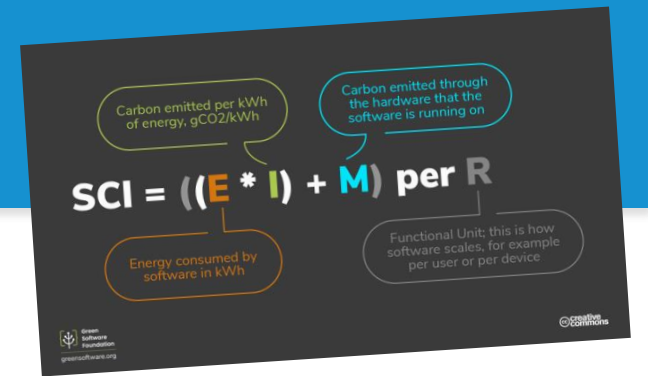
Example: 96 vCPUs

MovieRecommender

ToR = 32 (this is the max #vCPUs supported in e2-standard instances in [Google Cloud](#))

Final SCI – Step 4

Here we compute the final value of SCI for the system under analysis



Time horizon = 1 month

R = 20,000 API requests

O = (E * I) = 936 gCO₂e

$$M = TE * (TiR/EL) * (RR/ToR)$$

$$TE = 1,230.3 \text{ kgCO}_2\text{e} = 1,230,300 \text{ gCO}_2\text{e}$$

$$TiR = 1 \text{ month}$$

$$EL = 4 \text{ years} = 48 \text{ months}$$

$$RR = 4 \text{ vCPUs}$$

$$ToR = 32 \text{ vCPUs}$$

$$M = 1,230,300 * (1/48) * (4/32) = 3,203.90625 \text{ gCO}_2\text{e}$$

$$SCI = (936 + 3,203.9) \text{ gCO}_2\text{e per month}$$

$$= (936 + 3,203.9) / 20000 = 0.206995 \text{ gCO}_2\text{e per API call}$$

Reporting – Step 5

In this step you disclose the SCI score, software boundary, and the calculation methodology

- When dealing with a large system, you can compute the SCI scores for all its main subsystems and then sum them up
 - R and time horizon must remain the same
- If your goal is to take actions to reduce carbon footprint, you need a **baseline** when taking actions to reduce carbon footprint
 - The baseline has to be computed using the same methodology

What this module means to you?

- The SCI is an ISO/IEC standard designed specifically to calculate **software emissions**
- SCI is a **rate**
 - The functional unit R is not prescribed in the SCI
 - Key decision for your specific case/application
- The SCI is designed to be **easy to implement** and to do not imply additional costs
- SCI encourages calculation using **granular real-world data**
 - This might be challenging to obtain, in those cases SCI supports the usage of estimates coming from open datasets

Readings

- Basics about green software
<https://learn.greensoftware.foundation>
- Software Carbon Intensity Specification
<https://sci.greensoftware.foundation>
- SCl official guide
<https://sci-guide.greensoftware.foundation>
- SCl case studies (for inspiration)
<https://sci-guide.greensoftware.foundation/CaseStudies>
- Example of usage of SCl in a scientific study:
 - YouTube video: <https://www.youtube.com/watch?v=BOHKSK8GFYg>
 - Study: <https://hotcarbon.org/assets/2024/pdf/hotcarbon24-final109.pdf>