The state of Green Washing - or how to build sustainable systems with Kubernetes

Max Körbächer | Liquid Reply

Max Körbächer - Co-Founder @ REPLY LIQUID

My work is all about
Kubernetes Consultancy & Cloud Native Advisory

CNCF TAG Environmental Sustainability Co-Chair,
CNCF Ambassador, LF Europe Advisory Board,
Contributed 3y to the Kubernetes release team

AMBASSADOR CLOUD NATIVE COMPUTING FOUNDATION SPRING 2023

THE LINUX FOUNDATION

TAG ENVIRONMENTAL SUSTAINABILITY

maxkoerbaecher

mkoerbi

The Challenges

Global Data Centers

Consuming around **2%** of the global energy.

Expected to grow within the next couple of years by **additional 2%**.

Some forecast assume a peak of **12%** of the consumed energy by 2024

*treat this numbers with care, studies to this are old

Data, Distribution and Digitalization

The explosion of **data generation, connecting everything and the digital opening** of not yet well connected countries will lead to an **exponential growth**.

Old systems and hardware as well as data center are not very efficient.

Carbon emissions are everywhere

Carbon emissions are caused in **any step of the production** of products and services.

This also counts for IT. The **major part** of carbon emissions are caused by the production of chips, server and other hardware components.

The transparency of power usage, Carbon emissions and other waste is very poor within the ICT.

A missing link:

Data about energy consumption

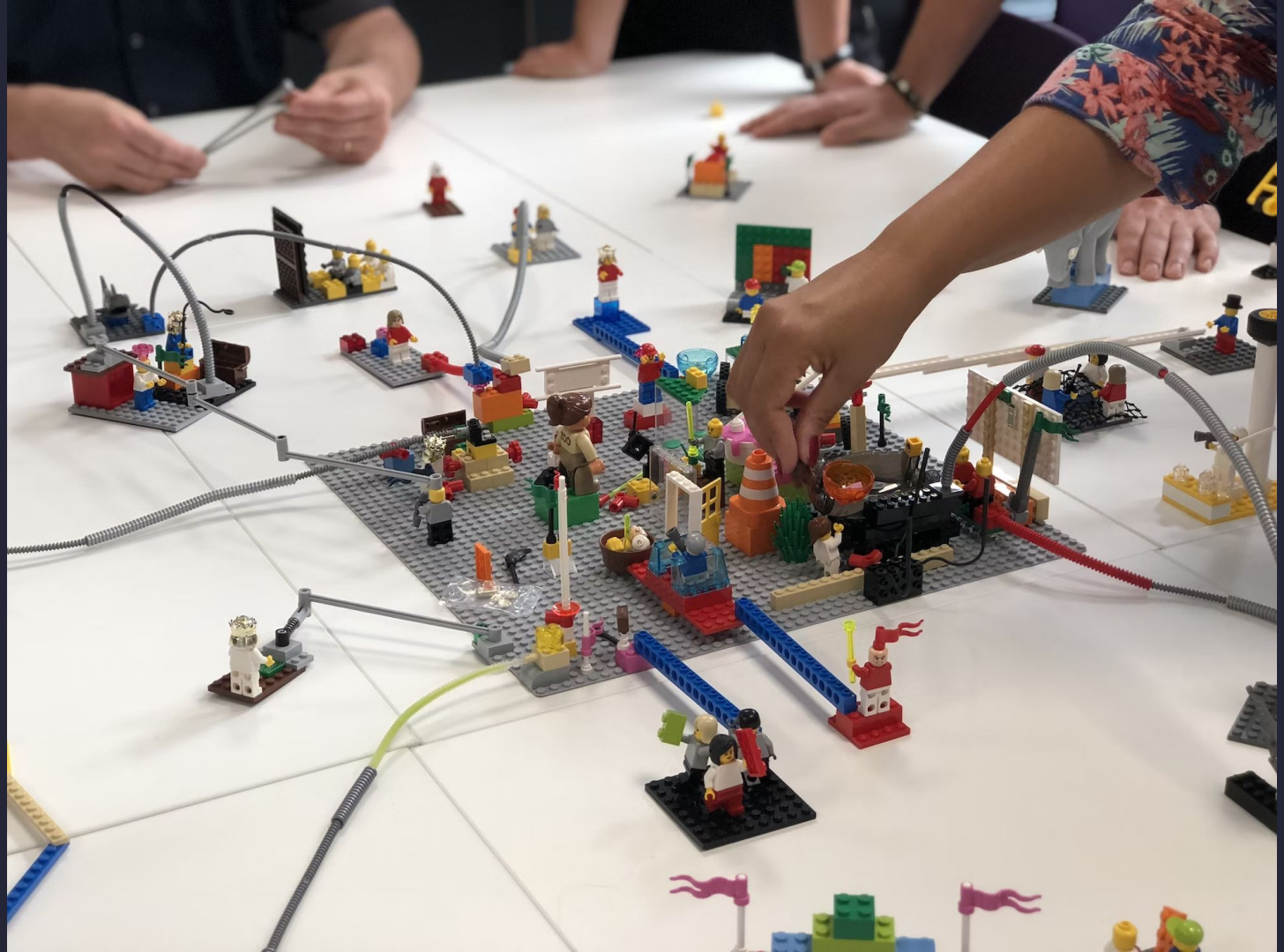
TO

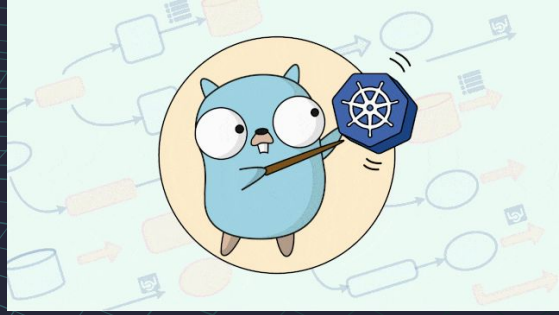
Data on caused CO2 emissions per kWh

*but we are getting close


MAERSK


THIS IS NOT
KUBERNETES



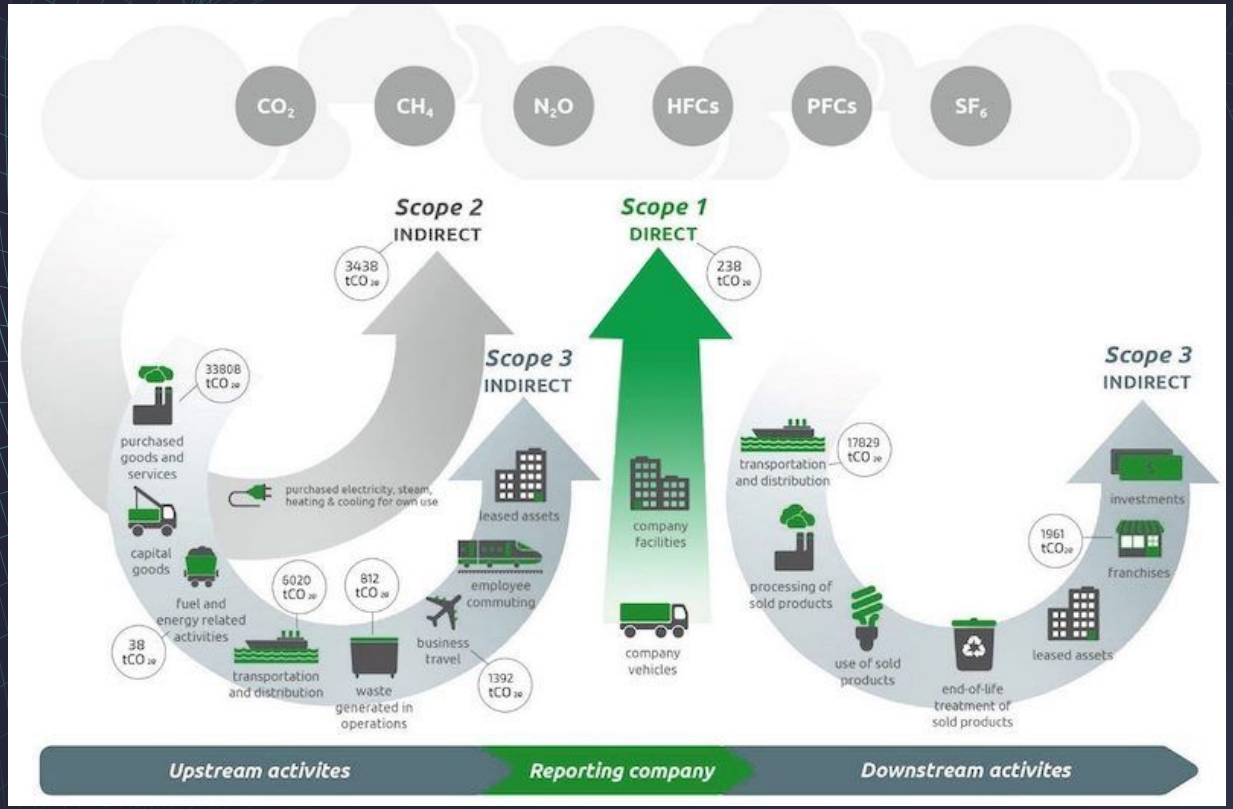


Kubernetes provides a unified approach to integrate various solutions and to make them act on each other.*

*yes, we still need better data at the node level, beyond this, only the creativity is a limit

One more thing…

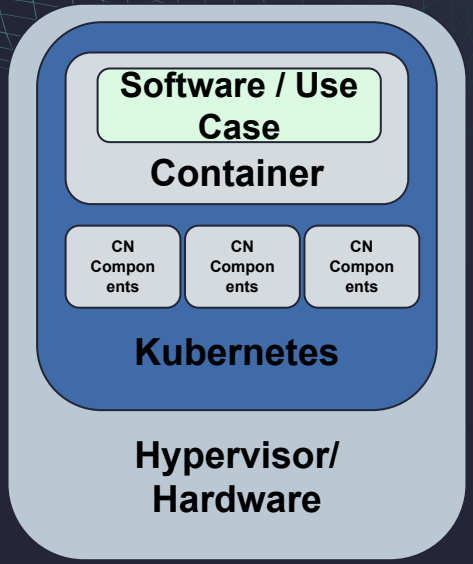
Scopes



Source: https://sustainlab.co/blog/what-are-scope-1-2-3-emissions

What can we do?

The Cloud Native “Can and have to”



The Cloud Native “Can and have to”

Can do*

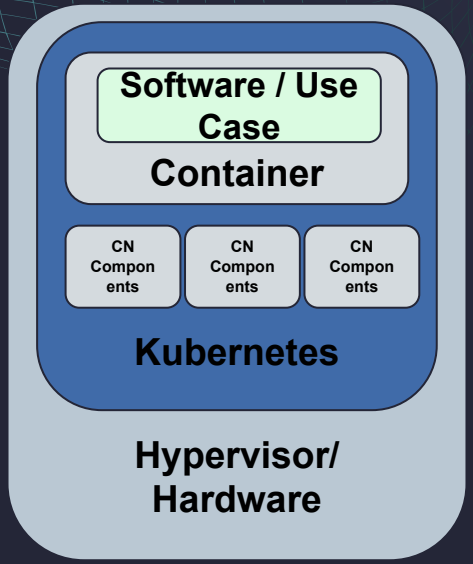
Optimize Container Images

Schedule containers for high density

Scale containers to zero

Scale clusters to zero

Optimize nodes, HW (e.g. ARM based) and OS



*selection of topics that are obvious

The Cloud Native “Can and have to”

Can do*

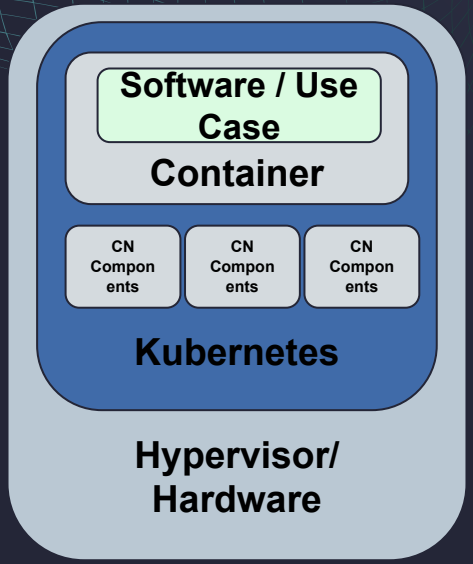
Optimize Container Images

Schedule containers for high density

Scale containers to zero

Scale clusters to zero

Optimize nodes, HW (e.g. ARM based) and OS



Have to*

A future without container?

Schedule based on carbon data

Scale based on carbon data

Design architectures for sustainability

Improve power management

*selection of topics that are obvious

Scale, reduce & rightsize

Change hardware or compute architecture

Adjust systems architecture

Optimize Software & build process

Scale, reduce & rightsize

Approach

Measure resource consumption.
Identify what you eliminate entirely.
Implement event, time or metrics based scaling.

Solutions

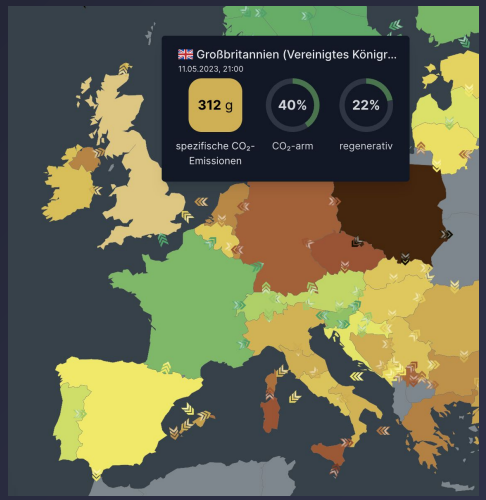
- Autoscaling Groups
- Karpenter
- kube-green
- KEDA
- kepler
- scaphandre

Impact

Drastic consumption reduction.
Easy to achieve, potential for further improvement over time.

Kepler & Scaphandre

Combine the power consumption with the location and your local energy mix to get a realistic estimation



Change hardware or compute architecture

Approach

Switch to more efficient CPU, Memory & Storage. Utilize event driven or serverless solutions.

Solutions

- ARM/AWS Graviton
- (just the latest instance type)
- Fermyon Spin
- OpenFaas
- “Green” Regions

Impact

Reduce required runtime & energy. Depending on effort invested, can have similar good impact as scaling & reduction.

Could be wasm a relevant game changer?

- Only single digit MB size
- Incredible fast startup time ->
  - more scalability
  - scale to 0

Adjust systems architecture

Approach

Change system configuration, HA, used middleware solutions, data formats, storage options and so on.

Solutions

- Stateless
- Reduce transmitted data
- Implement more lightweight tools

Impact

Complicated approach that requires a well thought through plan. Minimal to large reduction of energy demand.

Green Software Patterns

Summary

An online open-source database of software patterns reviewed and curated by the Green Software Foundation across a wide range of categories. You can be confident that applying any of our published and live patterns will reduce your software emissions.

Any software practitioner can find the patterns related to their field, technology, or domain. Anyone can submit a pattern that triggers a detailed review process by the Foundation.

Be smart with data!

And find a more holistic approach. To copy tons around isn’t helpful.

Optimize Software & Build Process

Approach

Rewrite software with more efficient algorithms, libraries or “better” programming languages.
Don’t build every commit!

Solutions

← ???

And change the whole solution design, as mentioned before.

Impact

High effort to implement those changes, except software is already highly modular and can be adjusted.

Total

	Energy
(c) C	1.00
(c) Rust	1.03
(c) C++	1.34
(c) Ada	1.70
(v) Java	1.98
(c) Pascal	2.14
(c) Chapel	2.18
(v) Lisp	2.27
(c) Ocaml	2.40
(c) Fortran	2.52
(c) Swift	2.79
(c) Haskell	3.10
(v) C#	3.14
(c) Go	3.23
(i) Dart	3.83
(v) F#	4.13
(i) JavaScript	4.45
(v) Racket	7.91
(i) TypeScript	21.50
(i) Hack	24.02
(i) PHP	29.30
(v) Erlang	42.23
(i) Lua	45.98
(i) Jruby	46.54
(i) Ruby	69.91
(i) Python	75.88
(i) Perl	79.58

	Time
(c) C	1.00
(c) Rust	1.04
(c) C++	1.56
(c) Ada	1.85
(v) Java	1.89
(c) Chapel	2.14
(c) Go	2.83
(c) Pascal	3.02
(c) Ocaml	3.09
(v) C#	3.14
(v) Lisp	3.40
(c) Haskell	3.55
(c) Swift	4.20
(c) Fortran	4.20
(v) F#	6.30
(i) JavaScript	6.52
(i) Dart	6.67
(v) Racket	11.27
(i) Hack	26.99
(i) PHP	27.64
(v) Erlang	36.71
(i) Jruby	43.44
(i) TypeScript	46.20
(i) Ruby	59.34
(i) Perl	65.79
(i) Python	71.90
(i) Lua	82.91

	Mb
(c) Pascal	1.00
(c) Go	1.05
(c) C	1.17
(c) Fortran	1.24
(c) C++	1.34
(c) Ada	1.47
(c) Rust	1.54
(v) Lisp	1.92
(c) Haskell	2.45
(i) PHP	2.57
(c) Swift	2.71
(i) Python	2.80
(c) Ocaml	2.82
(v) C#	2.85
(i) Hack	3.34
(v) Racket	3.52
(i) Ruby	3.97
(c) Chapel	4.00
(v) F#	4.25
(i) JavaScript	4.59
(i) TypeScript	4.69
(v) Java	6.01
(i) Perl	6.62
(i) Lua	6.72
(v) Erlang	7.20
(i) Dart	8.64
(i) Jruby	19.84

Time & Memory	Energy & Time	Energy & Memory	Energy & Time & Memory
C • Pascal • Go	C	C • Pascal	C • Pascal • Go
Rust • C++ • Fortran	Rust	Rust • C++ • Fortran • Go	Rust • C++ • Fortran
Ada	C++	Ada	Ada
Java • Chapel • Lisp • Ocaml	Ada	Java • Chapel • Lisp	Java • Chapel • Lisp • Ocaml
Haskell • C#	Java	OCaml • Swift • Haskell	Swift • Haskell • C#
Swift • PHP	Pascal • Chapel	C# • PHP	Dart • F# • Racket • Hack • PHP
F# • Racket • Hack • Python	Lisp • Ocaml • Go	Dart • F# • Racket • Hack • Python	JavaScript • Ruby • Python
JavaScript • Ruby	Fortran • Haskell • C#	JavaScript • Ruby	TypeScript • Erlang
Dart • TypeScript • Erlang	Swift	TypeScript	Lua • JRuby • Perl
JRuby • Perl	Dart • F#	Erlang • Lua • Perl	
Lua	JavaScript	JRuby	
	Racket		
	TypeScript • Hack		
	PHP		
	Erlang		
	Lua • JRuby		
	Ruby		

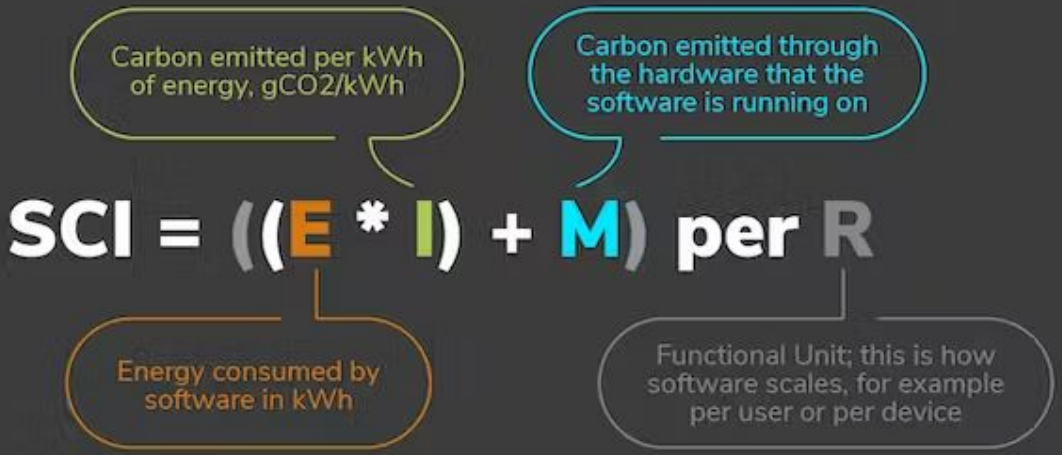
Please forget this, this is not not true, but outdated

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

Time & Memory	Energy & Time	Energy & Memory	Energy & Time & Memory
C • Pascal • Go	C	C • Pascal	C • Pascal • Go
Rust • C++ • Fortran	Rust	Rust • C++ • Fortran • Go	Rust • C++ • Fortran
Ada	C++	Ada	Ada
Java • Chapel • Lisp • Ocaml	Ada	Java • Chapel • Lisp	Java • Chapel • Lisp • Ocaml
Haskell • C#	Java	OCaml • Swift • Haskell	Swift • Haskell • C#
Swift • PHP	Pascal • Chapel	C# • PHP	Dart • F# • Racket • Hack • PHP
F# • Racket • Hack • Python	Lisp • Ocaml • Go	Dart • F# • Racket • Hack • Python	JavaScript • Ruby • Python
JavaScript • Ruby	Fortran • Haskell • C#	JavaScript • Ruby	TypeScript • Erlang
Dart • TypeScript • Erlang	Swift	TypeScript	Lua • JRuby • Perl
JRuby • Perl	Dart • F#	Erlang • Lua • Perl	
Lua	JavaScript	JRuby	
	Racket		
	TypeScript • Hack		
	PHP		
	Erlang		
	Lua • JRuby		
	Ruby		

The Software Carbon Intensity (SCI)

The SCI score is a rate of carbon emissions, not a total.
The equation is a simple and elegant solution to the extremely complex problem behind it:



Green Software Foundation
greensoftware.foundation

The “per R” is what makes the SCI into a tool that works for every software domain, every use case, and every person.

A REAL LIFE EXAMPLE

Prometheus as example in its SCI specification

E = 0.34Wh

M = (4283kg CO2e/35.000h) = 122gCO2e/Wh x 0,1

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

I = 0,323gCO2e/Wh*

* in Germany, in Sweden that would be 0,200g

R = number of nodes?

A REAL LIFE EXAMPLE

Prometheus as example in its SCI specification

E = 0.34Wh

M = (4283kg CO2e/35.000h) = 122gCO2e/Wh x 0,1

SCI = ((**0,34**Wh * **0,323**gCO2e/Wh) + **12.2**gCO2e/Wh) / **3** = **4.01gCO2e/Wh**

I = 0,323gCO2e/Wh*

* in Germany, in Sweden that would be 0,200g

R = number of nodes?

WHAT TO DO WITH THE SCI?

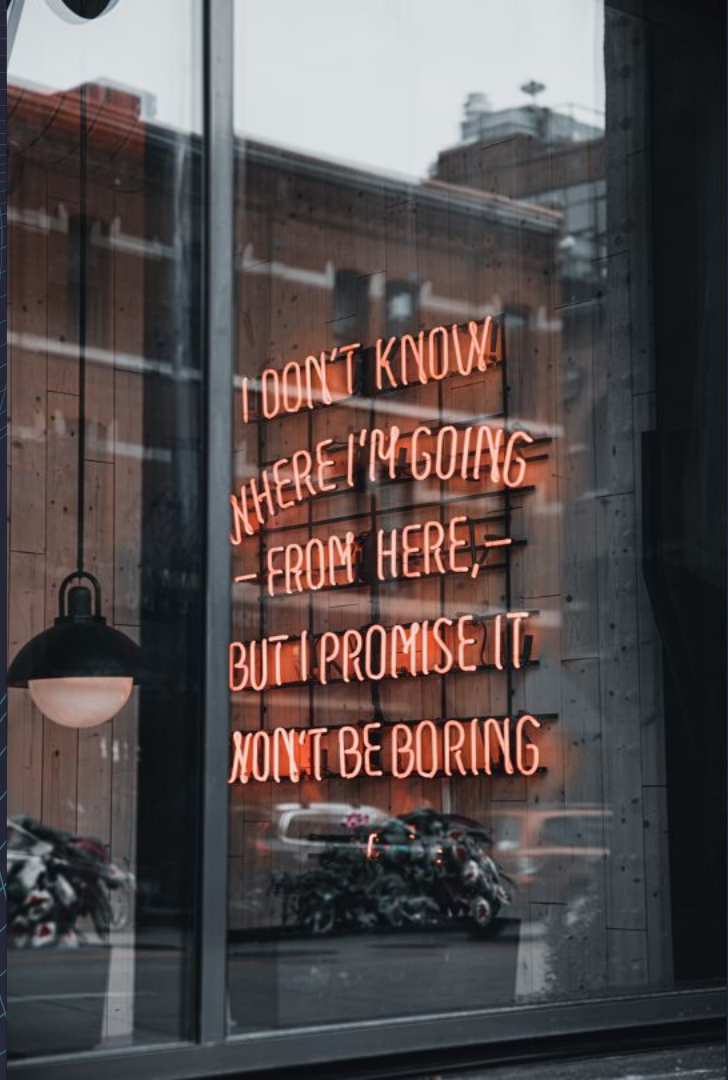
A current open source tool we support in its SCI specification

SCI = 4,01gCO2e/Wh

Calculate the SCI with every change:

- Move to another region
- Optimize the software
- Combine multiple SCIs (App, Middleware, DBs)

The SCI is there to help understanding your actions on optimize your systems.



Processes matter, make your SDLC better!

- ➔ don’t need to run checks on every commit
- ➔ optimize your container builds - order the layers correctly
- ➔ use the right tools, if you want to optimize for low carbon footprint


MYTHBUSTERS

Good ideas but not calculated till the end

I. **Time-shifted jobs** - 0 benefits for you, most likely even the CSP doesn’t recognize the impact

Good ideas but not calculated till the end

II.

Relocation or follow the sun - might save you some coins, but is it green to shift data?

Good ideas but not calculated till the end

III.

Optimize for costs will optimize for CO2e reduction - possible, but optimize for CO2e reduction can be even most costly

Good ideas but not calculated till the end

I. **Time-shifted jobs** - 0 benefits for you, most likely even the CSP doesn't recognize the impact

II. **Relocation or follow the sun** - might save you some coins, but is it green to shift data?

III. **Optimize for costs will optimize for CO2e reduction** - possible, but optimize for CO2e reduction can be even most costly

The Sustainability Paradoxon

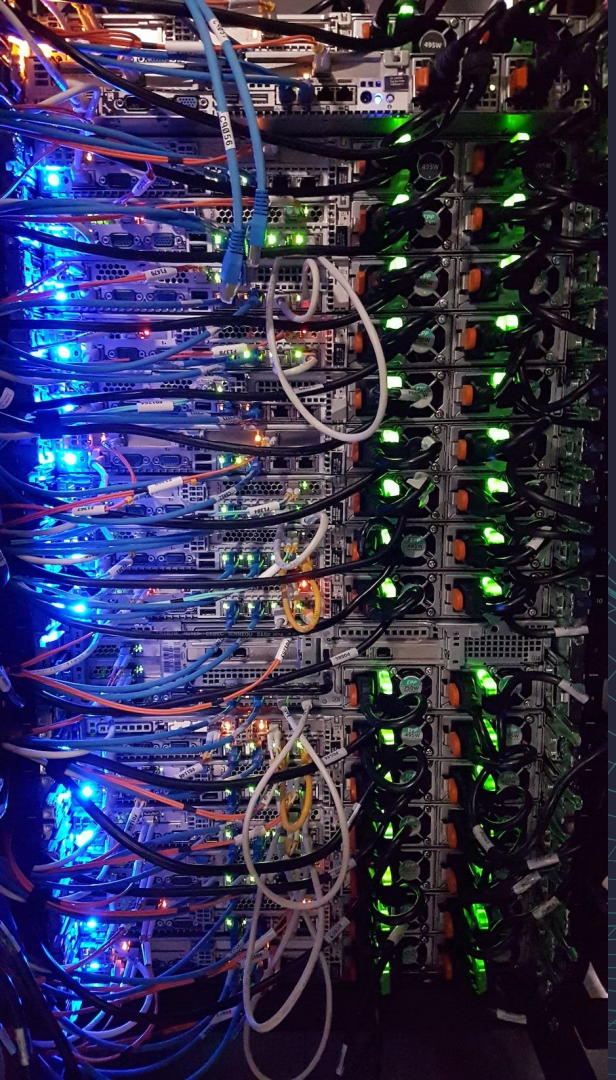
ARM based server

- needs up to **60% less energy**
- provides around **25% more performance**
- **40% better price** performance

We just have to go ARM to do better right? EASY!

Generally it is better to:

- Use hardware **as long as possible**
- The **embodied carbon** make up to 85% of the whole CO2e during the lifecycle of a server



So, we all better go back to private data centers and run old HW as long as possible?

No! Building a DC is super harmful due to the insane carbon footprint of cement!

Whatever we want to optimize, it currently always has a down side.

Therefore, you need to identify your right way.


ACT
NOW

Whatever Scope you reduce, it’s better than nothing

Reducing Scope 2 also means to reduce passively Scope 3 by lowering the demand!

To drive this change in the cloud native universe we have founded the CNCF TAG Environmental Sustainability

TAG ENVIRONMENTAL SUSTAINABILITY

… and there are other fantastic organizations

Talk with us on the CNCF Slack, find a team to work with or show us your ideas!
#tag-environmental-sustainability

Find us on the CNCF GitHub, discuss current working artifacts and review our deliverables.
https://github.com/cncf/tag-env-sustainability

Join our mailing list and most importantly virtual meetings!
https://tag-env-sustainability.cncf.io

THANKS!

Does anyone have any questions?

Max Körbächer

maxkoerbaecher

mkoerbi