Karlsruher Institut für Technologie

14th HPC Café

02.04.2026

Agenda HPC Café – 02.04.2026

Topics for today:

- **Sustainability in HPC - a brief overview (Dr. Andreas Baer, SCC)**
- **Updates for HPC@KIT**
  - **HoreKa 2 - Transition from HoreKa to HoreKa 2**
  - **Future Technologies Partition**
- Discussion and open floor

As usual this format is meant to be interactive.
Don’t hesitate to ask questions!

Karlsruher Institut für Technologie

Sustainability in HPC – a brief overview

Andreas Baer, SCC

Sustainability in HPC
A brief overview

Andreas Baer,
Scientific Computing Center, 02.04.26

bw | RSE 4 HPC

KIT | NHR
National High-Performance Computing Center

What is sustainability?

"**Sustainability** [...] is the ability to continue over a long period of time. In modern usage it generally refers to a state in which environment, economy, and society will continue to exist over a long period of time." [1]

=> Careful use of resources

=> Favoring persistence over re-creation

[1] Wikipedia: Sustainibility, 09.02.2025, https://en.wikipedia.org/wiki/Sustainability

Sustainability in High Performance Computing

Energy and carbon efficiency of the code

Minimal energy consumption during development

Adaptability and reusability of the code

Outline

Green HPC and energy efficiency

- Introduction
- Factors influencing energy efficiency
- Practical advice

Sustainability in software development

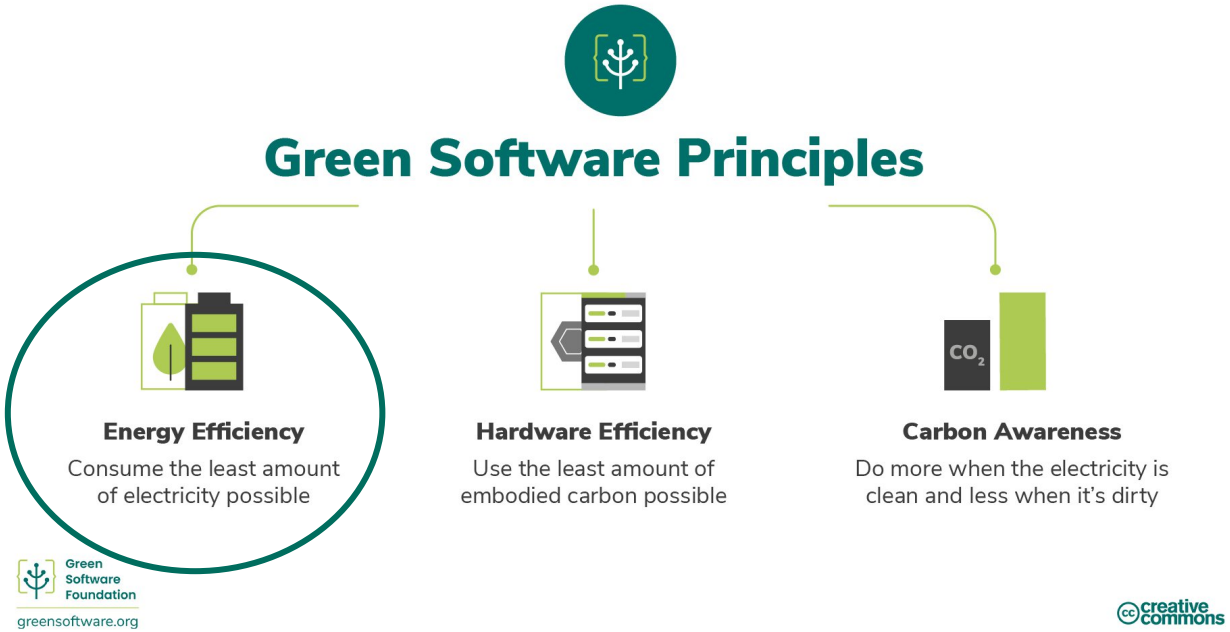
- Principles of energy efficient software
- Why testing matters
- Reusability and adaptability

Green HPC and energy efficiency

Based on the course “Green software use on HPC” by A. Turner
https://carpentries-incubator.github.io/green-software-hpc/key-points.html

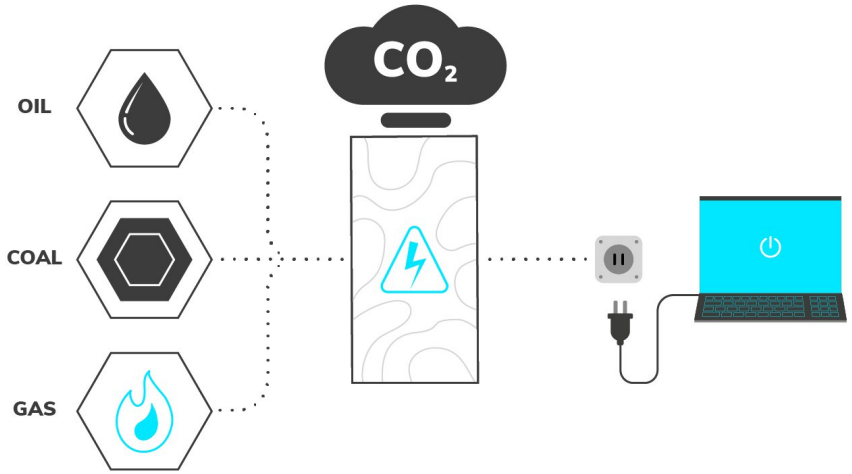
What is green HPC?

- Carbon-efficient HPC system use: Emit the **least carbon possible**
- Here, focus on **Energy efficiency**
- Hardware efficiency and carbon awareness:
  - Not much to do from the user side
  - Task for HPC system providers



From carbon to electricity

- Electricity sounds green and clean – but is it the case?
- Depends strongly on the region and time of day/year
- In Germany, 2025, one kWh generated on average 332g CO_2-equivalent [1]
  => in Germany, electricity is (unfortunately) a good proxy for carbon



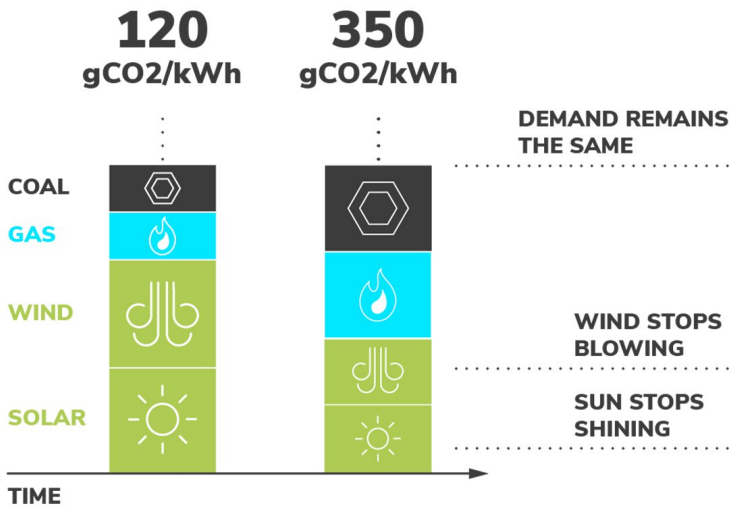
Green Software Foundation
greensoftware.org

creative commons

[1] Our World in Data, 09.02.2025, https://ourworldindata.org/grapher/carbon-intensity-electricity?tab=table

Low-carbon sources of energy

- **Clean energy**: does not produce emissions, e.g. nuclear
- **Green energy**: natural sources
- **Renewable energy**: sources will not expire, e.g. solar, wind



Green Software Foundation
greensoftware.org

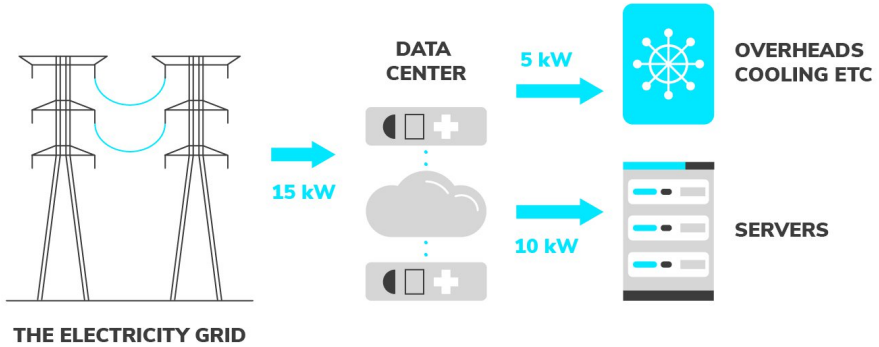
Green HPC and energy efficiency

Factors impacting energy efficiency

Factors impacting energy efficiency

Power usage effectiveness (PUE):

- Metric developed by Green Grid in 2006
- Measures **data center energy efficiency**
- Relates energy used for cooling and other overheads to energy for computations



Green Software Foundation
greensoftware.org

creative commons

Factors impacting energy efficiency

Power usage effectiveness (PUE)

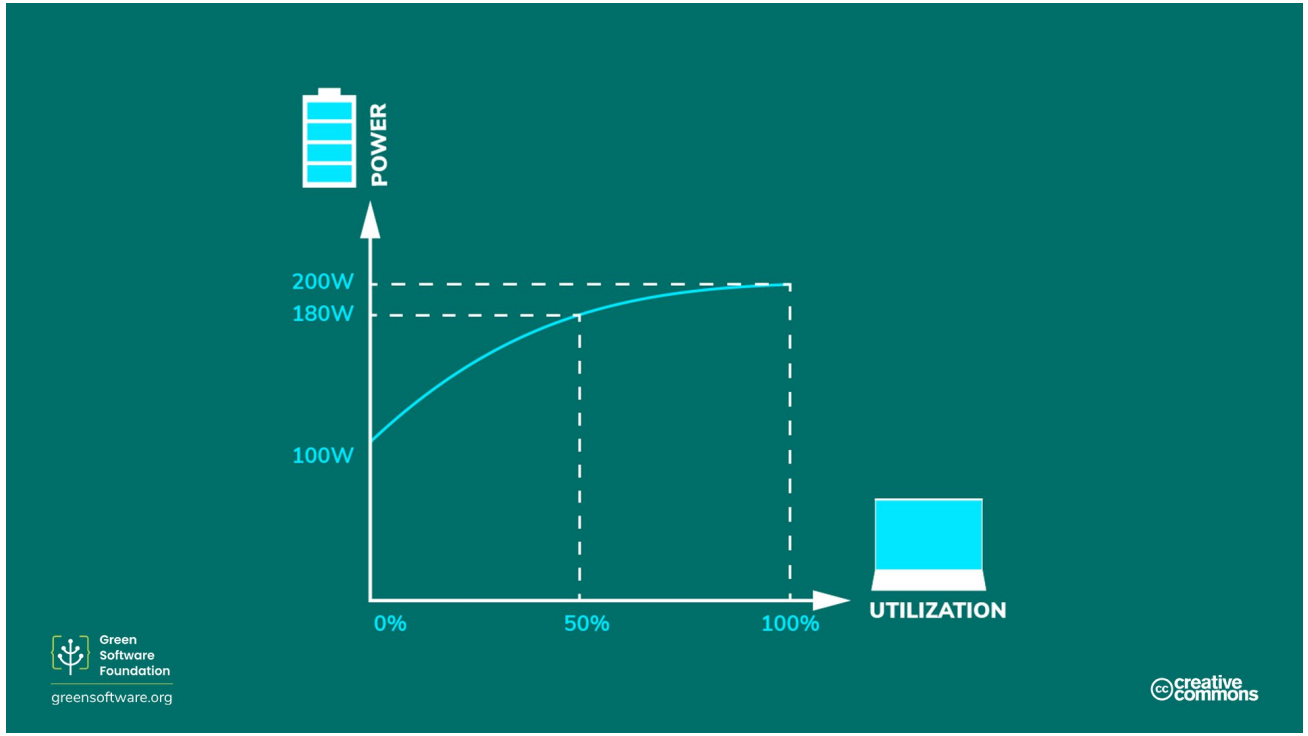
- Main contribution for cooling
- Additional factors (e.g.):
  - Power supply and distribution in the building
  - Monitoring and administrative infrastructure
- Typically depends on environmental conditions, e.g. outside temperature
- Example HoreKa: PUE (2025) of ~1.14, which is **very low**

Factors impacting energy efficiency

Energy proportionality

- First proposed in 2007 by engineers at Google
- Measures relationship between utilization (rate at which useful work is done) and power consumption
- Typically best ratio for 100% utilization

=> Aim for maximum utilization



Usage efficiency in HPC systems

- Energy proportionality argument very valid for *compute-bound* applications
- Many HPC applications are actually *memory-bound*, i.e. performance is not limited by CPU but by data moving from the memory to the processor
  - Here: once memory bandwidth is saturated, additional power draw will **not** lead to increased utilization (i.e. useful performance)
  - Additional power draw is just wasted energy
  - With parallel applications, ratio between *compute* and *memory* bound parts might shift

=> For HPC applications, energy efficiency is not straightforward and might differ for same problem and software

GROMACS on ARCHER2 (molecular dynamics simulation)

Node count	Ns simulated	Runtime (s) 2.0 GHz	kWh	Runtime (s) 2.25 GHz + boost	kWh
1	0.020	369	0.0464	288	0.0464
2	0.020	198	0.0450	156	0.0465
3	0.020	155	0.0438	109	0.0465
4	0.020	117	0.0471	93	0.0513

Which setup is the **most energy efficient**?

Which setup is the **most performant**?

Which setup is the **cheapest** (least node-hours)?

GROMACS on ARCHER2 (molecular dynamics simulation)

Node count	Ns simulated	Runtime (s) 2.0 GHz	kWh	Runtime (s) 2.25 GHz + boost	kWh
1	0.020	369	0.0464	**288**	0.0464
2	0.020	198	0.0450	156	0.0465
3	0.020	155	**0.0438**	109	0.0465
4	0.020	117	0.0471	**93**	0.0513

Which setup is the **most energy efficient**?

Which setup is the **most performant**?

Which setup is the **cheapest** (least node-hours)?

Practical advice

Assess the energy consumption

- What you don’t measure, you cannot improve!
- Job monitoring (e.g. JobMon on HoreKa)
- Job reporting (e.g. Slurm reports)
- Hardware counters (however, often not accessible for the user)

=> Check out the options of your HPC provider

Summary

- Carbon intensity of HPC heavily depends on the source of electricity
- Goal of Green HPC: reduce carbon emissions
- Straightforward way for the user: reduce energy consumption
- Many other people involved, e.g. service providers have additional options
- Energy consumption is very complex: depending on how the job is bound, its parallelization, the language and many other factors
  - Benchmarking and interactive tuning are often the only way to runtime optimizations
  - Highest throughput or best performance per resources typically do not align with most efficient configuration

Sustainability in software development

Principles of energy efficient software

Principles of energy efficient software

- Goal: make the code run as energy efficient as possible
- No objective quality standards yet
- But: many basic principles already known
- Fraunhofer institute for experimental software engineering published a blog with a set of principles
- Mostly best practices, easy to implement

Let’s take a look, what this means for HPC applications!

Fraunhofer IESE, Blog, accessed 02.02.25, https://www.iese.fraunhofer.de/blog/sustainable-software-design/

Principles of energy efficient software

- Select an efficient algorithm after analyzing the complexity of the problem at hand
- Avoid regular and high-frequency polling by applying asynchronous communication patterns
- Pick an energy-efficient programming language for high-intensity modules
- Reduce data scope by transferring, querying or processing only the relevant data
- Use caches where possible and appropriate
- Perform dynamic code analysis to identify code passages with high energy consumption
- Use Artificial Intelligence (AI) wisely, i.e. only where necessary and only trained to the level required by the application
- ... Some more, less relevant for HPC

Summary

- Efficient software often is performant software and vice versa
- Many principles are just best practices, easily to be implemented
- Many people stop caring for their analysis – but this is where most can be achieved
- In-depth optimization is best done with help of experts (e.g. SSPE team, bwRSE4HPC team)

Sustainability in software development

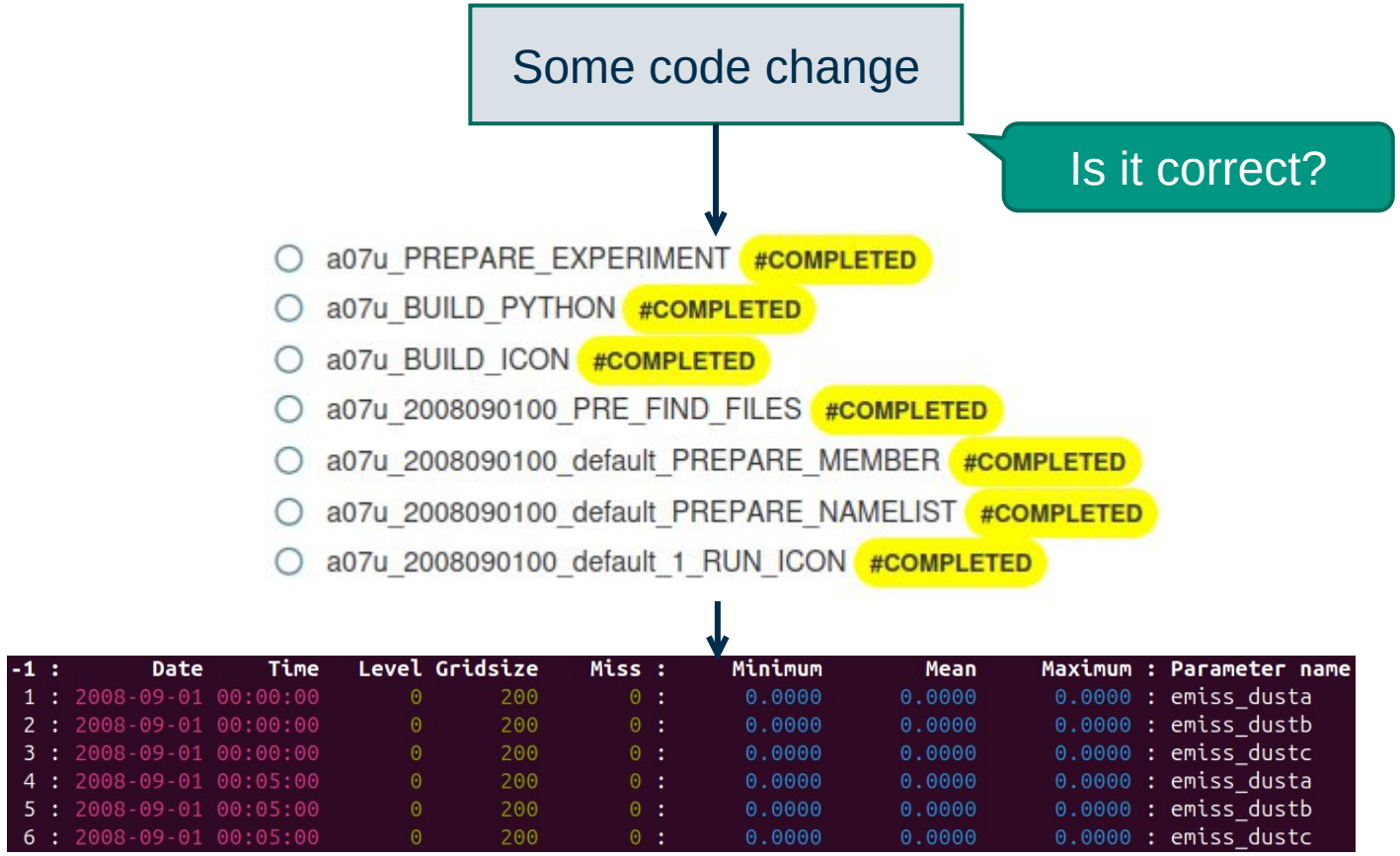
Why testing matters

Energy efficient testing

- Correctness is crucial for (scientific) applications
- How to ensure correctness?
  => Have a test case and check its output.

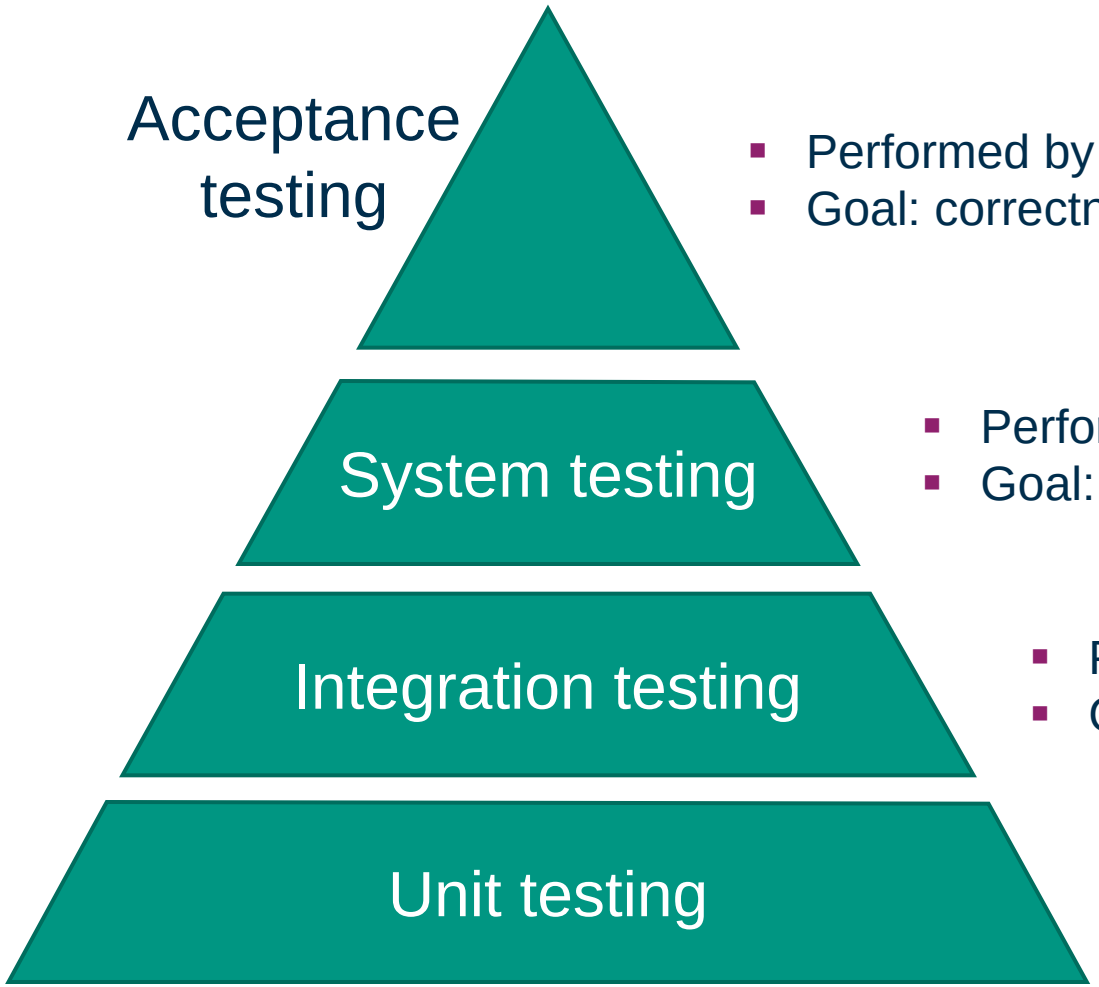
Problem with HPC software:

- Even simple case costs much resources
- Most time spent in routines not developed



Test only the routine of interest with minimal data

The testing hierarchy



- Performed by end user
- Goal: correctness of **end product**

In Research Software, often user = tester = developer

- Performed by testers/maintainers
- Goal: correctness of **end-to-end system**

- Performed by developers/CI
- Goal: correctness of **interfaces / module interactions**

- Performed by developers/CI
- Goal: correctness of **single functional units**

Summary

- End-to-end (or system) testing requires lots of resources for HPC software
- Unit test can be (if efficiently used) save lots of energy (and manpower)
- Integration test can and should be used for more complex software to bridge the gap between system and unit test

Use unit tests even in small codes, e.g. for analysis

Choose a trade of between effort and result

Sustainability in software development

Reusability and adaptability

Why is reusability an issue?

- You will repeat to use same or similar code
- Your colleagues will also do so
- Other people in your field and probably other fields will also do so
- Each code requires testing => resources required
- Reusable code can help:
  - You have to get adapted to how to use the code
  - Adaption to own needs might be required
  - Collaborative development of the code can help to profit from features, others contribute

How will it look like for my colleagues?

The FAIR principles

- Basic principles for sharing scientific data
- Established to “optimize data sharing and reuse by humans and machines”

But what about software?

==> FAIR4RS

(FAIR principles for research software, Chue Hong *et al.* 2021)

... but software is evolving!

Box 2 | The FAIR Guiding Principles

To be Findable:
F1. (meta)data are assigned a globally unique and persistent identifier
F2. data are described with rich metadata (defined by R1 below)
F3. metadata clearly and explicitly include the identifier of the data it describes
F4. (meta)data are registered or indexed in a searchable resource

To be Accessible:
A1. (meta)data are retrievable by their identifier using a standardized communications protocol
A1.1 the protocol is open, free, and universally implementable
A1.2 the protocol allows for an authentication and authorization procedure, where necessary
A2. metadata are accessible, even when the data are no longer available

To be Interoperable:
I1. (meta)data use a formal, accessible, shared, and broadly applicable language for knowledge representation.
I2. (meta)data use vocabularies that follow FAIR principles
I3. (meta)data include qualified references to other (meta)data

To be Reusable:
R1. meta(data) are richly described with a plurality of accurate and relevant attributes
R1.1. (meta)data are released with a clear and accessible data usage license
R1.2. (meta)data are associated with detailed provenance
R1.3. (meta)data meet domain-relevant community standards

Chue Hong *et al.* 2021

CODE beyond FAIR

Research software

- Is diverse:
  - Data processing
  - Analysis
  - Driver for apparatus
  - Model complex phenomena, etc.
- Has a “living nature”:
  - Constantly evolving
  - Many different versions possibly in used parallelly
- Has a porous frontier:
  - Trend to go open source

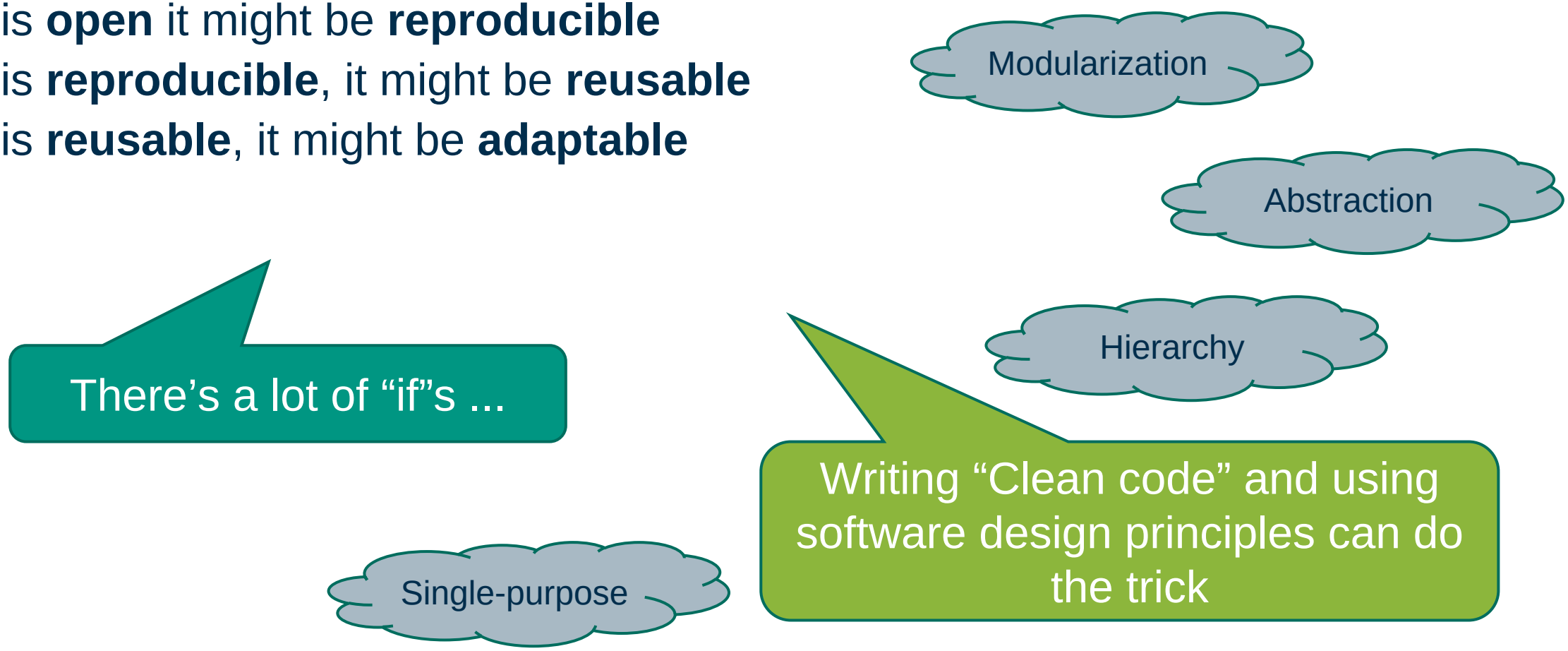
=> CODE principles by Di Cosmo *et al.* 2025

OPEN	
• **Publish your source code on a public forge**	**mandatory**
• **Save your repository on dedicated archive**	**mandatory**
• **License your code with an open license**	**strongly recommended**
• Declare authorship & rightholders	recommended
DOCUMENT	
• Choose meaningful names	recommended
• Comment code	recommended
• Provide examples, notebooks & tutorials	recommended
• Describe API	optional
EXECUTE	
• List software & hardware dependencies	recommended
• Provide a computational environment	optional
• Implement a test suite	optional
• Show real-life usage example with expected results	optional
COLLABORATE	
• Respond to issues & feature requests	optional
• Describe maintenance, features & support limits	optional
• Explain how to contribute	optional
• Build and animate a community	optional

Di Cosmo *et al.* 2025

Open – reusable – adaptable

- If it is **open** it might be **reproducible**
- If it is **reproducible**, it might be **reusable**
- If it is **reusable**, it might be **adaptable**



Summary

- Code needs to be readable and structured by adhering to design principles
- Your colleagues should be able to tell you quickly what your code does!
- When you have a well-written code, you can apply the CODE principles
  - OPEN: publish and license the code
  - DOCUMENT: add code documentation and tutorials
  - EXECUTE: environment and test suite
  - COLLABORATE: get your community engaged

Scientific necessity for reproducibility

From here on, you can gain from the effort

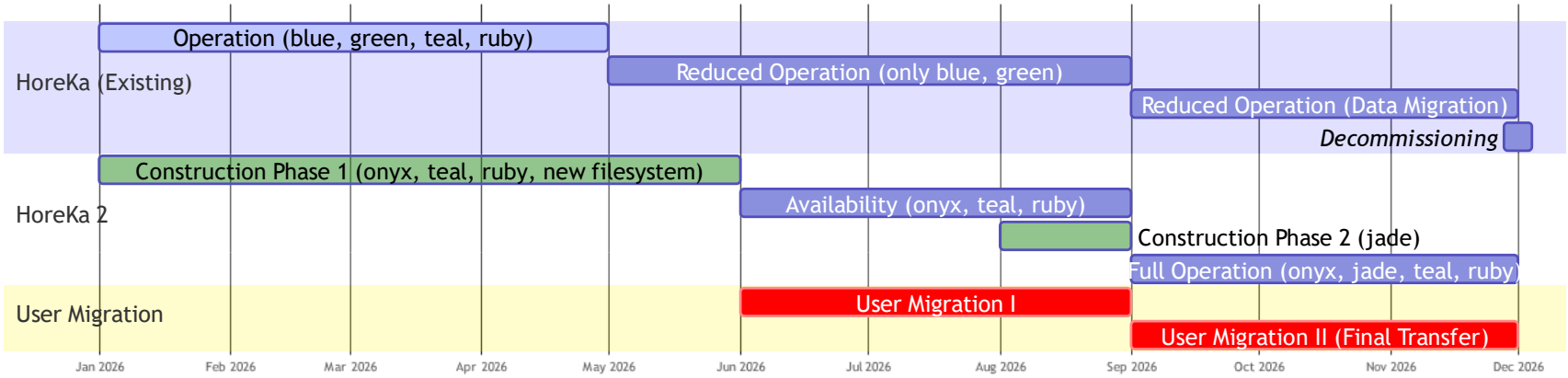
Thank you for your attention!

Karlsruher Institut für Technologie

Updates for HPC@KIT

Transition to HoreKa 2

- Setup for the HoreKa 2 CPU partition ongoing
  - Reduced availability of HoreKa Blue
  - Migration of HoreKa Teal and Ruby (accelerated partitions with Nvidia H100 and H200) to HoreKa 2 together with new CPU partition
- We will let you know when actions are needed

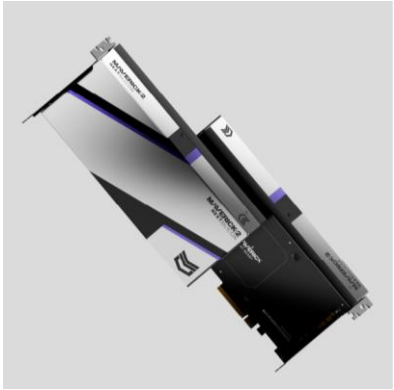


Taking into account scientific conferences for HoreKa

- The current situation
  - Higher load on the compute clusters can lead to waiting times for job to start
  - Especially ahead of deadlines for conferences this can be daunting
  - Expressed wish for some level of plannability
- Plans to account for this
  - First step: Gathering information for relevant conferences and deadlines
    → Please fill out the survey until end of April, we can only act on what we know
  - Afterwards:
    - Evaluation of Survey: conferences with larger user base, demand, and similar
    - Evaluate weeks with possible prioritization for users targeting a conference
      Think of it like a SC/NeurIPS/… week
- Currently: pilot phase and evaluation

Future Technologies Partition

- New addition of NextSilicon Maverick 2 accelerators
  - Designed as a dataflow architecture (in contrast to CPUs and GPUs)
  - Suited for HPC applications
  - Servers with standard x86 host processors
  - Some more information here: https://www.nextsilicon.com/maverick
- Availability for users soon
  - Currently in the last phases of the setup
  - Upcoming update to the FTP Documentation
- Interested in trying it out? Let us know!
  - Options for support by us and experts from NextSilicon



© NextSilicon

Karlsruher Institut für Technologie

Time for Discussion

Next HPC Café

- Next HPC Café planned for **7 May 2026, 10 am**
- Reminder: regular timeslot for the HPC Café
  - Every **first Thursday of the month 10:00 am**

- As always, open to hear ideas for topics and talks, from …
  - Yourself
  - Colleagues and collaborators
  - Also just expressions for “topics of interest”