

MANCHESTER
1824

The University of Manchester

Sustainable software: Some perspectives for HEP

CATERINA DOGLIONI - UNIVERSITY OF MANCHESTER
SHE/HER



Outline

- Definitions: what is **sustainability**?
- Sustainable software:
 - **FAIR** software principles
 - **Environmental** sustainability (green software)
- *So, how do I make my software sustainable?*
- *This is all great but what incentives do I have?*
- **Ongoing initiatives and collaborative efforts** on making software more sustainable

What is sustainable software, to *you*?

Slido link
slido.com #4492991



What sustainable software (engineering) means to Microsoft

<https://learn.microsoft.com/en-us/training/modules/sustainable-software-engineering-overview/>



The Principles of Sustainable Software Engineering

• Module • 10 Units

 [Feedback](#)

Beginner Developer Administrator Solution Architect Student DevOps Engineer Data Scientist Data Engineer
Database Administrator AI Edge Engineer AI Engineer Technology Manager Azure .NET Microsoft Power Platform

Sustainable Software Engineering is an emerging discipline at the intersection of climate science, software, hardware, electricity markets, and data center design. The Principles of Sustainable Software Engineering are a core set of competencies needed to define, build, and run sustainable software applications.

Carbon
efficiency

Energy
efficiency

Carbon
awareness

Hardware
efficiency

Measurement

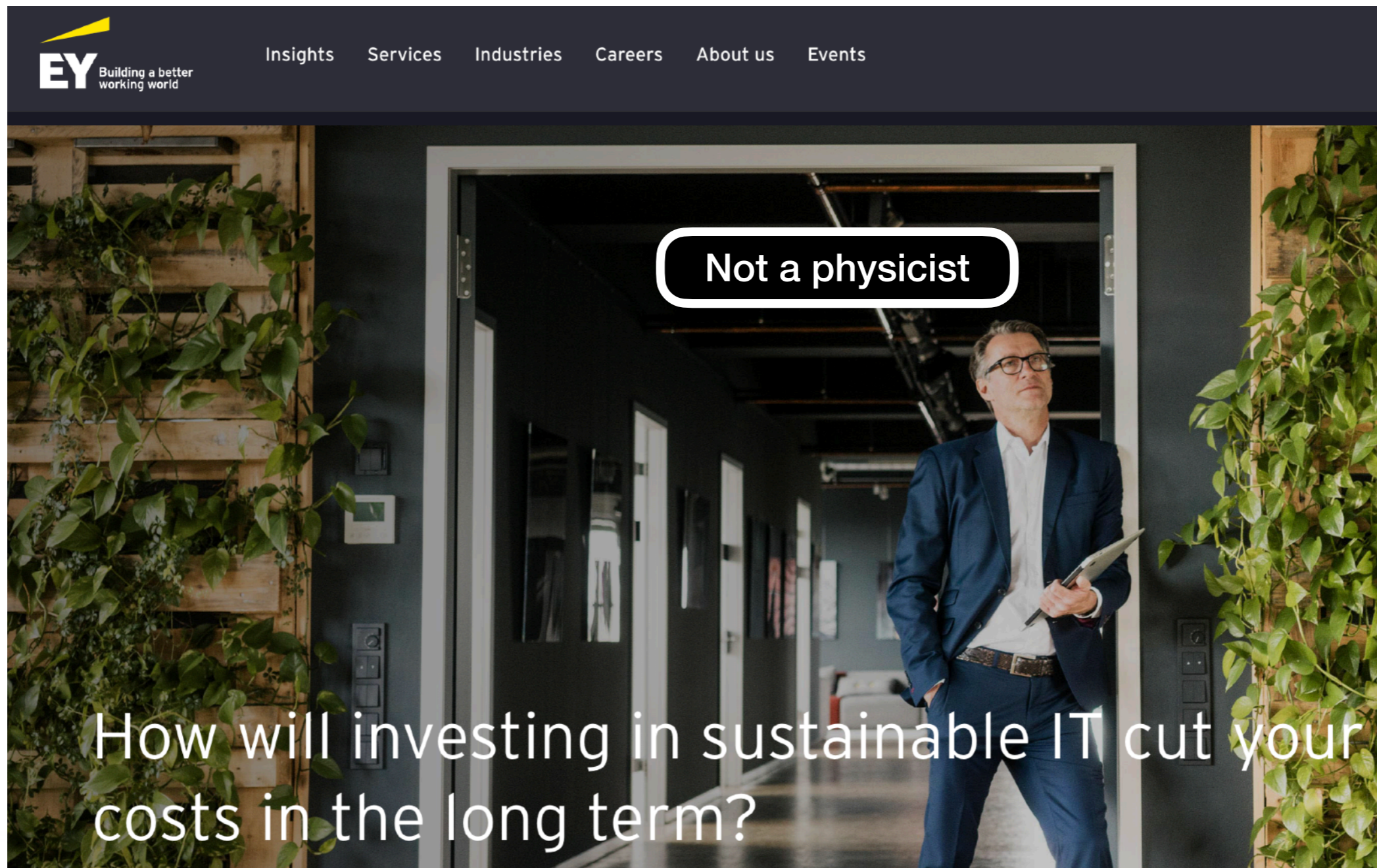
Climate
commitments



Focus: **environmental / climate** impacts

What sustainable IT & software means to Ernst & Young

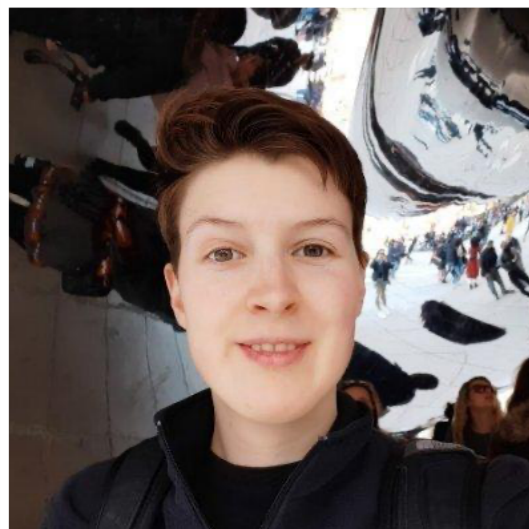
https://www.ey.com/en_ch/sustainability/how-will-investing-in-sustainable-it-cut-your-costs-in-the-long-term



Focus: **costs** and **reputation**

What sustainable software means to an expert RSE

Eli Chadwick is a Research Software Engineer (RSE)
from the Software Sustainability Institute



<https://github.com/elichad>

Talk by E. Chadwick at the HEP Software Foundation + Swift-HEP C++ course in Manchester

Definition of Sustainable Software

Sustainability means that the software you use today will be available - and continue to be improved and supported - in the future. ¹

Sustainable software is software which: ²

- Is easy to evolve and maintain
- Fulfils its intent over time
- Survives uncertainty
- Supports relevant concerns (political, economic, social, technical, legal, environmental)

1. [About the Software Sustainability Institute](#)
2. [Defining Software Sustainability](#) by Daniel S. Katz



THE
CARPENTRIES

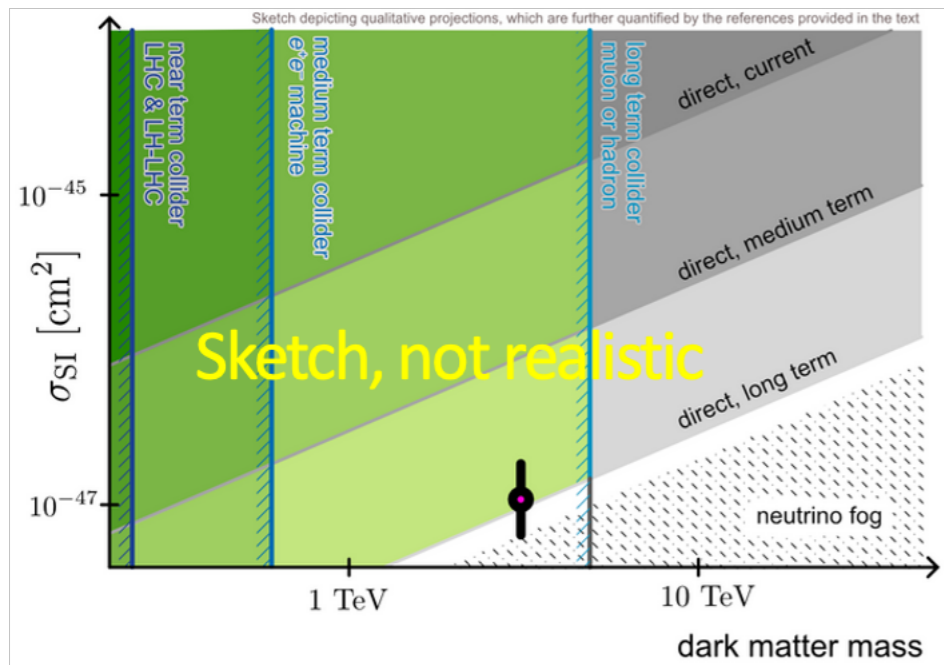


MANCHESTER
1824

The University of Manchester

Focus: **evolution, maintenance, intent and future**

What sustainable software means to us in HEP (examples)



Physicists who are going to discover dark matter at direct, indirect detection and FCC experiments



<https://www.iybssd2022.org/en/100-years-of-physics/>

Physicists who are developing / using complex tools used by the whole HEP community



This year's MCNet school picture will go here!

<https://pythia.org/authors/>



Focus:

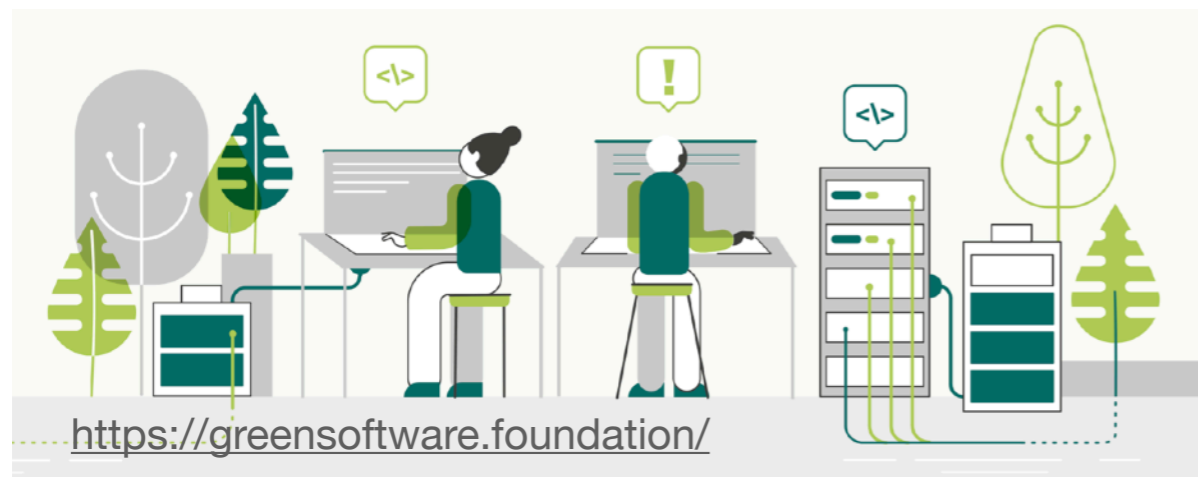
reproducibility, usability, maintainability...



Today, we'll explore two of these perspectives

Focus on **reproducibility, sharing and reuse:**
FAIR software

<https://www.go-fair.org/go-fair-initiative/>



Focus on **environmental (and climate) impacts:**
Green software

Big disclaimer: I've been writing software for a while, but only started thinking about sustainable software in the last 1-2 years → personal perspective with a lot of pointers to others!

Take home point 1:

Software communities in different fields (including industry) are thinking/talking about sustainability

(and definitions/recommendations are still both very much in flow, see later)

From FAIR principles for *data*...

<https://www.nature.com/articles/sdata201618>

<https://www.go-fair.org/go-fair-initiative/>

scientific data



Explore content ▾ About the journal ▾ Publish with us ▾

[nature](#) > [scientific data](#) > [comment](#) > [article](#)

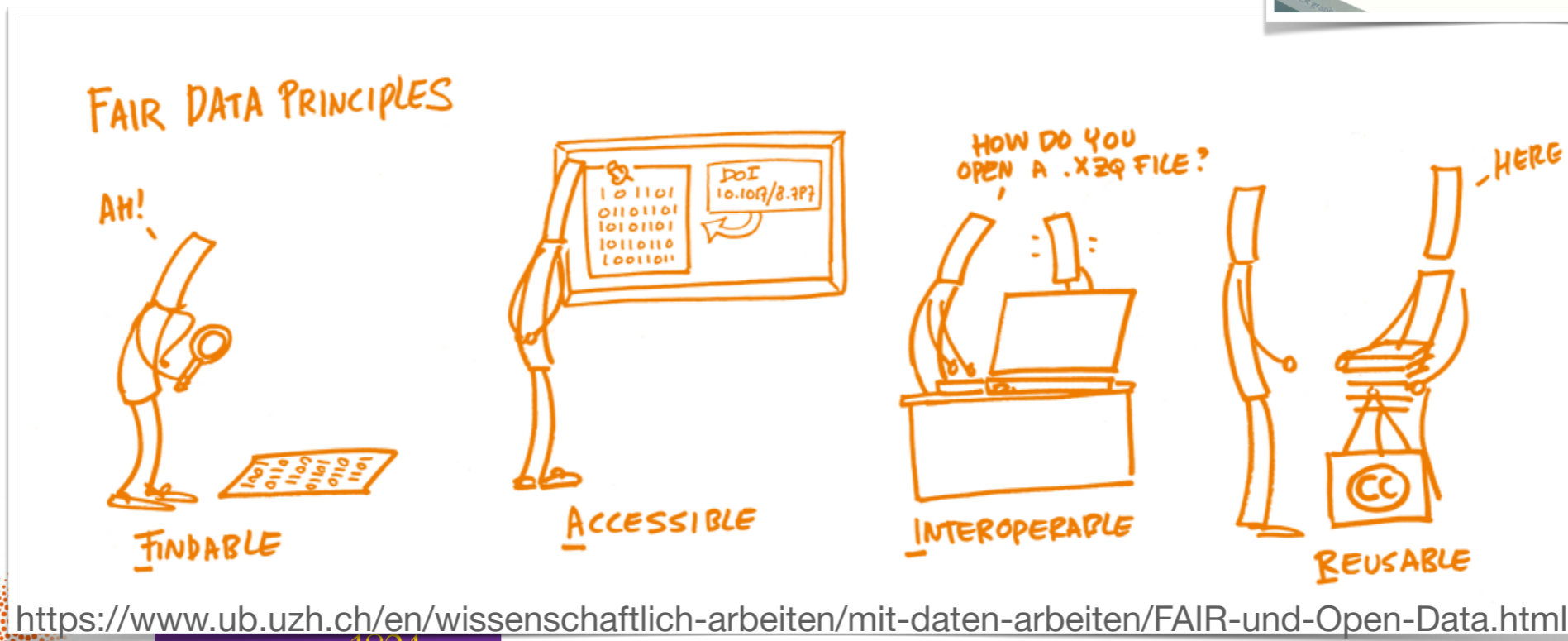
Comment | [Open access](#) | Published: 15 March 2016

The FAIR Guiding Principles for scientific data management and stewardship

FAIR Principles

GO FAIR is committed to making data and services **findable, accessible, interoperable and reusable (FAIR)**.

- Findable:** Metadata and data should be easy to find for both humans and computers.
- Accessible:** The exact conditions under which the data is accessible should be provided in such a way that humans and machines can understand them.
- Interoperable:** The (meta)data should be based on standardized vocabularies, ontologies, thesauri etc. so that it integrates with existing applications or workflows.
- Reusable:** Metadata and data should be well-described so that they can be replicated and/or combined in different research settings.



TLDR: FAIR basics for software

<https://fair-software.eu/>

FIVE RECOMMENDATIONS FOR FAIR SOFTWARE

ENDORSE

LET'S GO! →

Public repository
with version control

Provide a
license

Code appears in a
community
catalogue

Enable
software
citations

Use a software
quality checklist



<https://zenodo.org/>

More in depth: FAIR for research software

scientific data

<https://www.nature.com/articles/s41597-022-01710-x>

Explore content ▾


About the journal ▾

Publish with us ▾

[nature](#) > [scientific data](#) > [articles](#) > [article](#)

Article | [Open access](#) | Published: 14 October 2022

Introducing the FAIR Principles for research software

[Michelle Barker](#) , [Neil P. Chue Hong](#), [Daniel S. Katz](#), [Anna-Lena Lamprecht](#), [Carlos Martinez-Ortiz](#),
[Fotis Psomopoulos](#), [Jennifer Harrow](#), [Leyla Jael Castro](#), [Morane Gruenpeter](#), [Paula Andrea Martinez](#) &
[Tom Honeyman](#)

[Scientific Data](#) **9**, Article number: 622 (2022) | [Cite this article](#)

23k Accesses | **74** Citations | **229** Altmetric | [Metrics](#)



FAIR for research software: recommendations

The FAIR4RS Principles are: Final recommendations: <https://zenodo.org/records/6623556>

<p>F: Software, and its associated metadata, is easy for both humans and machines to find.</p>
<p>F1. Software is assigned a globally unique and persistent identifier.</p> <ul style="list-style-type: none"> • F1.1. Components of the software representing levels of granularity are assigned distinct identifiers. • F1.2. Different versions of the software are assigned distinct identifiers. <p>F2. Software is described with rich metadata.</p> <p>F3. Metadata clearly and explicitly include the identifier of the software they describe.</p> <p>F4. Metadata are FAIR, searchable and indexable.</p>
<p>A: Software, and its metadata, is retrievable via standardized protocols.</p>
<p>A1. Software is retrievable by its identifier using a standardized communications protocol.</p> <ul style="list-style-type: none"> • A1.1. The protocol is open, free, and universally implementable. • A1.2. The protocol allows for an authentication and authorization procedure, where necessary. <p>A2. Metadata are accessible, even when the software is no longer available.</p>
<p>I: Software interoperates with other software by exchanging data and/or metadata, and/or through interaction via application programming interfaces (APIs), described through standards.</p>
<p>I1. Software reads, writes and exchanges data in a way that meets domain-relevant community standards.</p> <p>I2. Software includes qualified references to other objects.</p>
<p>R: Software is both usable (can be executed) and reusable (can be understood, modified, built upon, or incorporated into other software).</p>
<p>R1. Software is described with a plurality of accurate and relevant attributes.</p> <ul style="list-style-type: none"> • R1.1. Software is given a clear and accessible license. • R1.2. Software is associated with detailed provenance. <p>R2. Software includes qualified references to other software.</p> <p>R3. Software meets domain-relevant community standards.</p>

Table 1: The FAIR Principles for Research Software



FAIR for research software: recommendations

The FAIR4RS Principles are: Final recommendations: <https://zenodo.org/records/6623556>

<p>F: Software, and its associated metadata, is easy for both humans and machines to find.</p>
<p>F1. Software is assigned a globally unique and persistent identifier.</p> <ul style="list-style-type: none"> F1.1. Components of the software representing levels of granularity are assigned distinct identifiers. F1.2. Different versions of the software are assigned distinct identifiers. <p>F2. Software is described with rich metadata.</p> <p>F3. Metadata clearly and explicitly include the identifier of the software they describe.</p> <p>F4. Metadata are FAIR, searchable and indexable.</p>
<p>A: Software, and its metadata, is retrievable via standardized protocols.</p>
<p>A1. Software is retrievable by its identifier using a standardized communications protocol.</p> <ul style="list-style-type: none"> A1.1. The protocol is open, free, and universally implementable. A1.2. The protocol allows for an authentication and authorization procedure, where necessary. <p>A2. Metadata are accessible, even when the software is no longer available.</p>
<p>I: Software interoperates with other software by exchanging data and/or metadata, and/or through interaction via application programming interfaces (APIs), described through standards.</p>
<p>I1. Software reads, writes and exchanges data in a way that meets domain-relevant community standards.</p> <p>I2. Software includes qualified references to other objects.</p>
<p>R: Software is both usable (can be executed) and reusable (can be understood, modified, built upon, or incorporated into other software).</p>
<p>R1. Software is described with a plurality of accurate and relevant attributes.</p> <ul style="list-style-type: none"> R1.1. Software is given a clear and accessible license. R1.2. Software is associated with detailed provenance. <p>R2. Software includes qualified references to other software.</p> <p>R3. Software meets domain-relevant community standards.</p>

This opens a new question: what are HEP-specific, *feasible* standards? For later...

Table 1: The FAIR Principles for Research Software



Take home point 2:

Recommendations regarding software sustainability exist, but inevitably have a domain-specific component

Context: examples of software impacting emissions

<https://hbr.org/2020/09/how-green-is-your-software>

Harvard
Business
Review

Sustainable Business Practices | How Green Is Your Software?

Sustainable Business Practices

How Green Is Your Software?

by Sanjay Podder, Adam Burden, Shalabh Kumar Singh, and Regina Maruca

September 18, 2020



Since then, a lot more AI
(and large language models)

- Bitcoin 🦴🦴🦴🦴🦴
- **IT + communication** sector: estimated to account for **14% of carbon footprint by 2040**
- **AI training** of complex/large networks consumes resources with **limited returns** for the last % of performance...
- Also: **hardware** carbon cost >> **software** running costs

Should we set limits to software energy consumption?

Should all humans arrange to get eaten by bears for a greener planet? 🐻

No, but we optimise what we can (given the trade-offs)

and use what we need ←



The fact that CERN (via our funding agencies) pays our energy bills makes us think computing resources are infinite, but they are not!

Sustainable (green) software: Microsoft training

Taken from: <https://learn.microsoft.com/en-us/training/modules/sustainable-software-engineering-overview/>

Updates at: <https://learn.greensoftware.foundation/>

**Carbon
efficiency**

Build applications that are carbon efficient.

Minimise the amount of carbon emitted per unit of work

**Energy
efficiency**

Build applications that are energy efficient.

Write software that maximises/matches your hardware's energy efficiency

**Carbon
awareness**

Consume electricity with the lowest carbon intensity.

Choose your energy sources wisely, if you can

**Hardware
efficiency**

Build applications that are hardware efficient.

Software should be well matched to hardware, and optimise hardware lifecycles

Measurement

Improve sustainability through measurement.

More information is better!

**Climate
commitments**

Defining the exact mechanism of carbon reduction.

Strive to improve, in a quantitative and well-defined way



Context: sustainability in HEP (software)

[Clicking on the poster leads to the indico agenda](#)



[V. Boisvert's talk at SUSTHEP24](#)

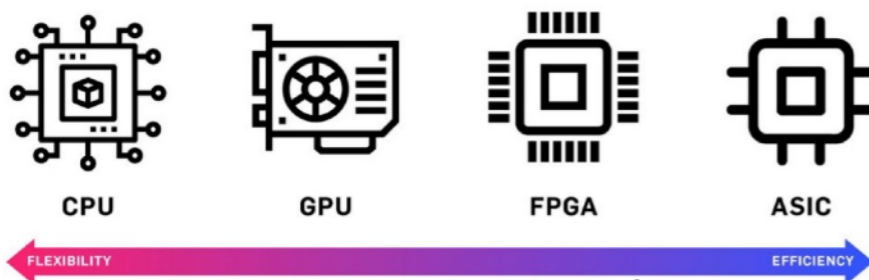


Image probably from here?



[Link to sustainable HECAP recommendations](#)

Recommendations – Computing

Individual actions:

- Make sustainable personal computing choices by considering the necessity of hardware upgrades, the repurposing of hardware, and the environmental credentials of suppliers and their products.
- Assess and improve the efficiency and portability of codes by considering, e.g., the required resolutions and accuracy.
- Assess and optimise data transmission and storage needs.
- Follow best practice in open-access data publishing, prioritising reproducibility and limiting repeat processing.
- Read the section on E-waste (Section 7).



Further group actions:

- Right-size IT requirements and optimise hardware lifecycles.
- Schedule queueing systems with environmental sustainability in mind, so as to maximise the use of renewables, accounting for the geographical location of servers/data centres.



Example of a success story close to MCNet

[Link to sustainable HECAP recommendations](#)

Best Practice : Optimization of software

A targeted effort enabled by the UK-based SoftWare and InFrastructure Technology for High Energy Physics (SWIFT-HEP) [49] project recently brought together experimentalists and Monte Carlo (MC) developers to greatly improve the computational efficiency of multi-leg next to leading order calculations by focussing on two major components of general purpose MC event generators: The evaluation of parton-distribution functions along with the generation of perturbative matrix elements. A dedicated CPU profiling illustrated that for the cost-driving event samples employed by the ATLAS experiment at CERN to model irreducible Standard Model backgrounds, these components dominate the overall run time by up to 80%. Improved interpolation and caching strategies in LHAPDF [50], the main evaluation tool for parton-distribution functions used by the experiments, along with the introduction of a simplified pilot run in the MC generator Sherpa [51] for the unweighting achieves a reduction of the computing footprint by factors of around 50 for multi-leg next to leading order event generation, while maintaining the formal accuracy of the event sample [52]. The speed-up translates into a direct CPU (and hence energy) saving, paving the way towards affordable and sustainable state-of-the-art event simulation in the hl-lhc era.

Energy
efficiency

Carbon
efficiency



The Green Software Foundation Software Carbon Intensity

At a glance...more details in this course

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 by
software in kWh

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

The Green Software Foundation Software Carbon Intensity

Carbon of ene

SCI =

Energy softw

s is how
example
device



Google Summer of Code / IRIS-HEP Fellows 2023

Manas Pratin Biswas

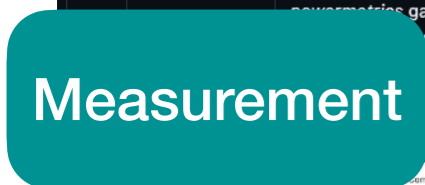
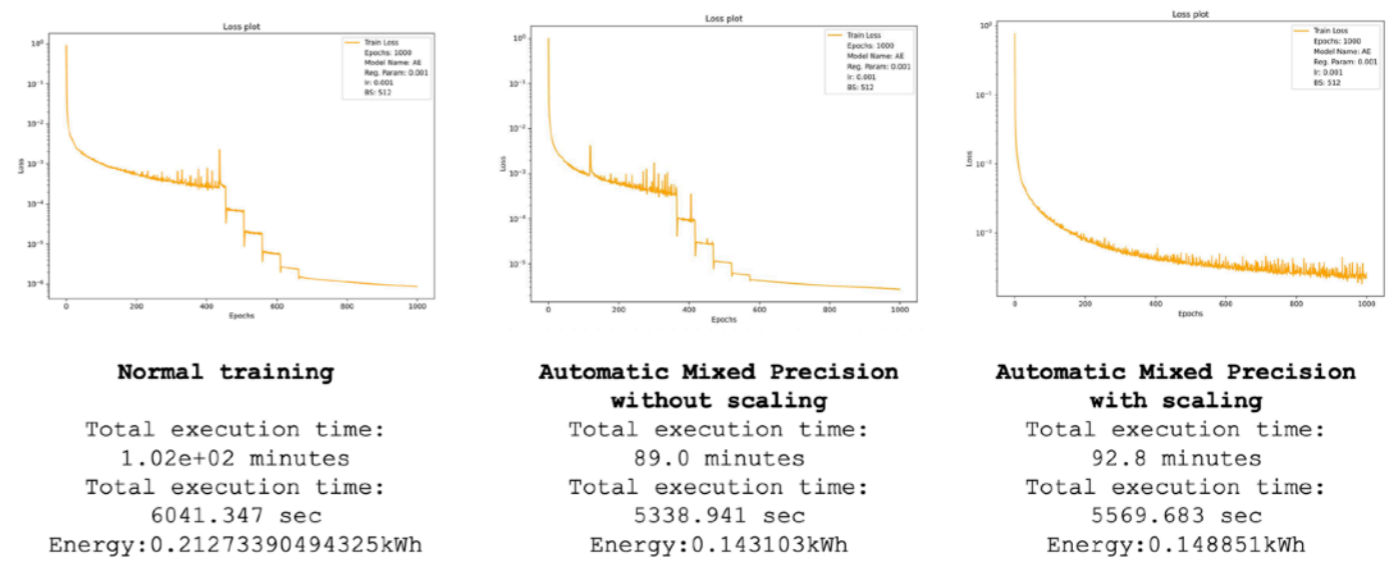
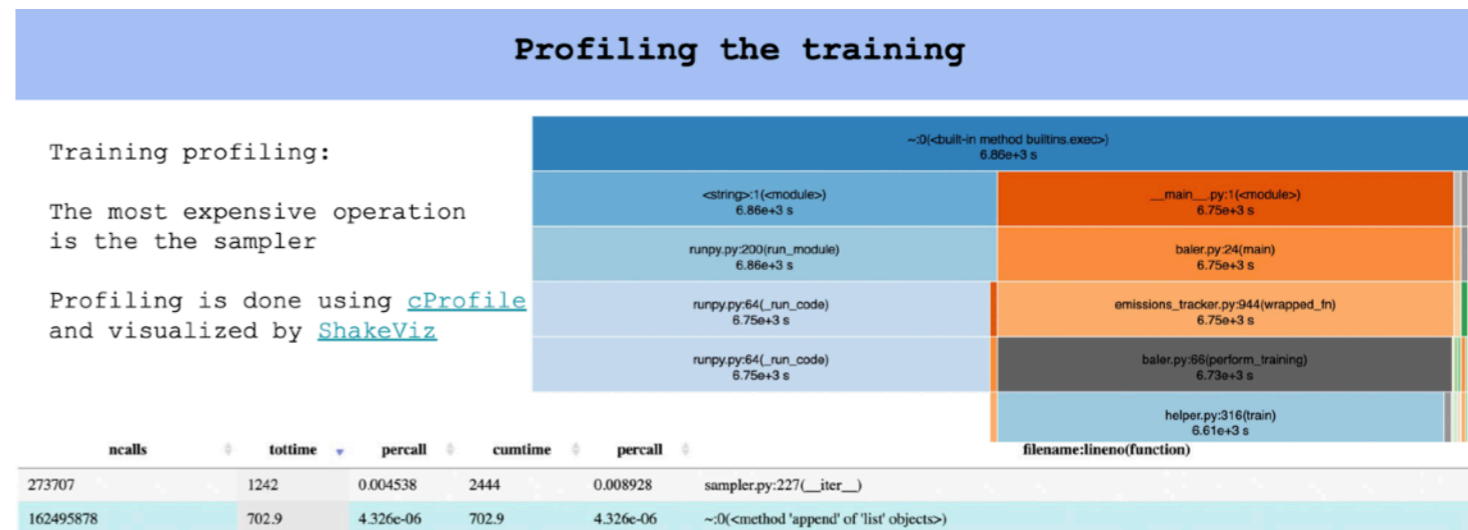


Started to explore the connection between software profiling and energy consumption in a ML compression software (Baler)
(More projects will be available next summer!)

Leonid Didukh



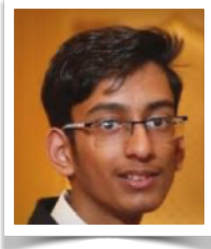
No.	Profiler/Tool	Description
1	cProfile	<i>cProfile</i> provides Deterministic Profiling of Python programs. A profile is a set of statistics that describes how often and for how long various parts of the program executed. It measures the <i>CPU</i> time
2	pyinstruments	<i>pyinstruments</i> provides Statistical Profiling of Python programs. It doesn't track every function call that the program makes. Instead, it records the call stack every 1ms and measures the <i>Wall Clock</i> time
3	experiment-impact-tracker	<i>experiment-impact-tracker</i> tracks energy usage, carbon emissions, and compute utilization of the system. Currently, on Linux systems with Intel chips (that support the <i>RAPL</i> or <i>powergadget</i> interfaces) and <i>NVIDIA GPUs</i> . It records power draw from CPU and GPU, hardware information, python package versions and estimated carbon emissions information
4	scalene	<i>Scalene</i> is a high-performance CPU, GPU and memory profiler for Python that incorporates AI-powered proposed optimizations
5	memory-profiler	<i>memory-profiler</i> is a python module for monitoring memory consumption of a process as well as line-by-line analysis of memory consumption for python programs
6	memray	<i>Memray</i> is a memory profiler for Python. It can track memory allocations in Python code, in native extension modules, and in the Python interpreter itself. It can generate several different types of reports to analyze the captured memory usage data.
7	codecarbon	<i>codecarbon</i> is a Python package that estimates the hardware electricity power consumption (GPU + CPU + RAM) and apply to it the carbon intensity of the region where the computing is done. The methodology behind the package involves the use a scheduler that, by default, call for the measure every 15 seconds and measures the CO ₂ as per the formula Carbon dioxide emissions ($CO_{2eq} = C * E$) Here, C = Carbon intensity of the electricity consumed for computation: quantified as g of CO ₂ emitted per kilowatt-hour of electricity and E = Energy Consumed by the computational infrastructure: quantified as kilowatt-hours.
8	Eco2AI	<i>Eco2AI</i> is a Python library for CO ₂ emission tracking. It monitors energy consumption of CPU & GPU devices and estimates equivalent carbon emissions taking into account the regional emission coefficient.



The University of Manchester

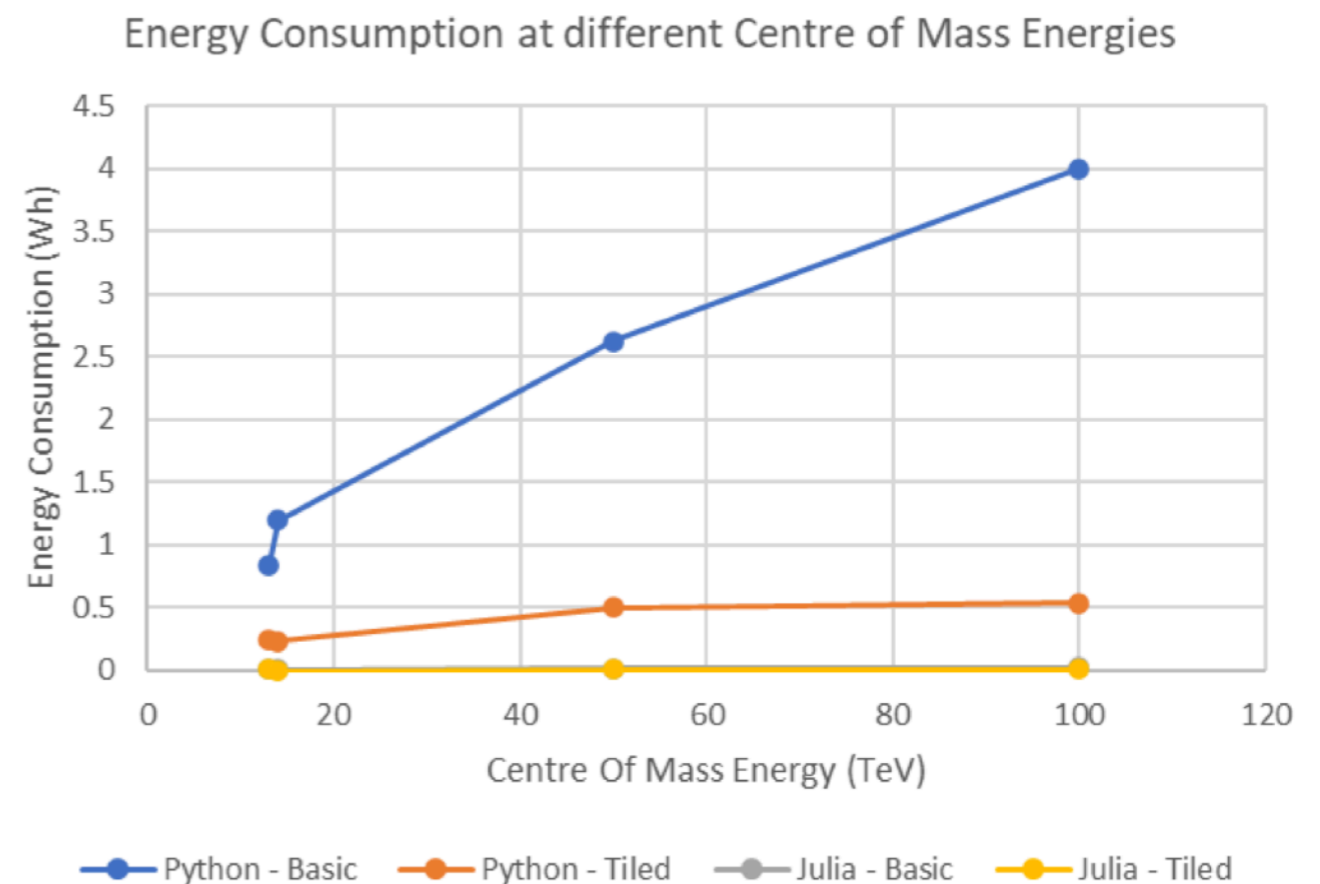
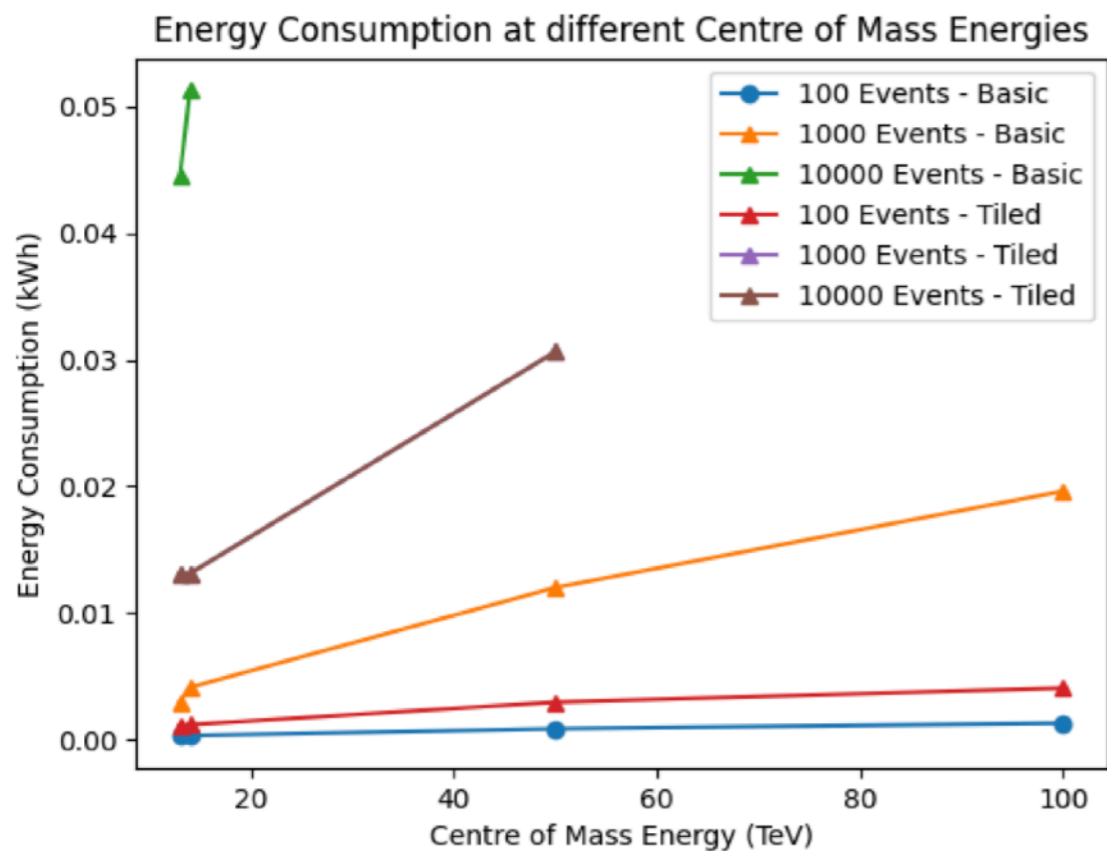
A practical example from a summer project

Akshat
Gupta



Estimated energy consumption and scaling for widely used jet algorithm (FastJet AKT4) run on different CoM samples (Pythia) in different languages (Python, Julia)

(More recent work: analysed top tagging ML algorithms, to be public soon!)

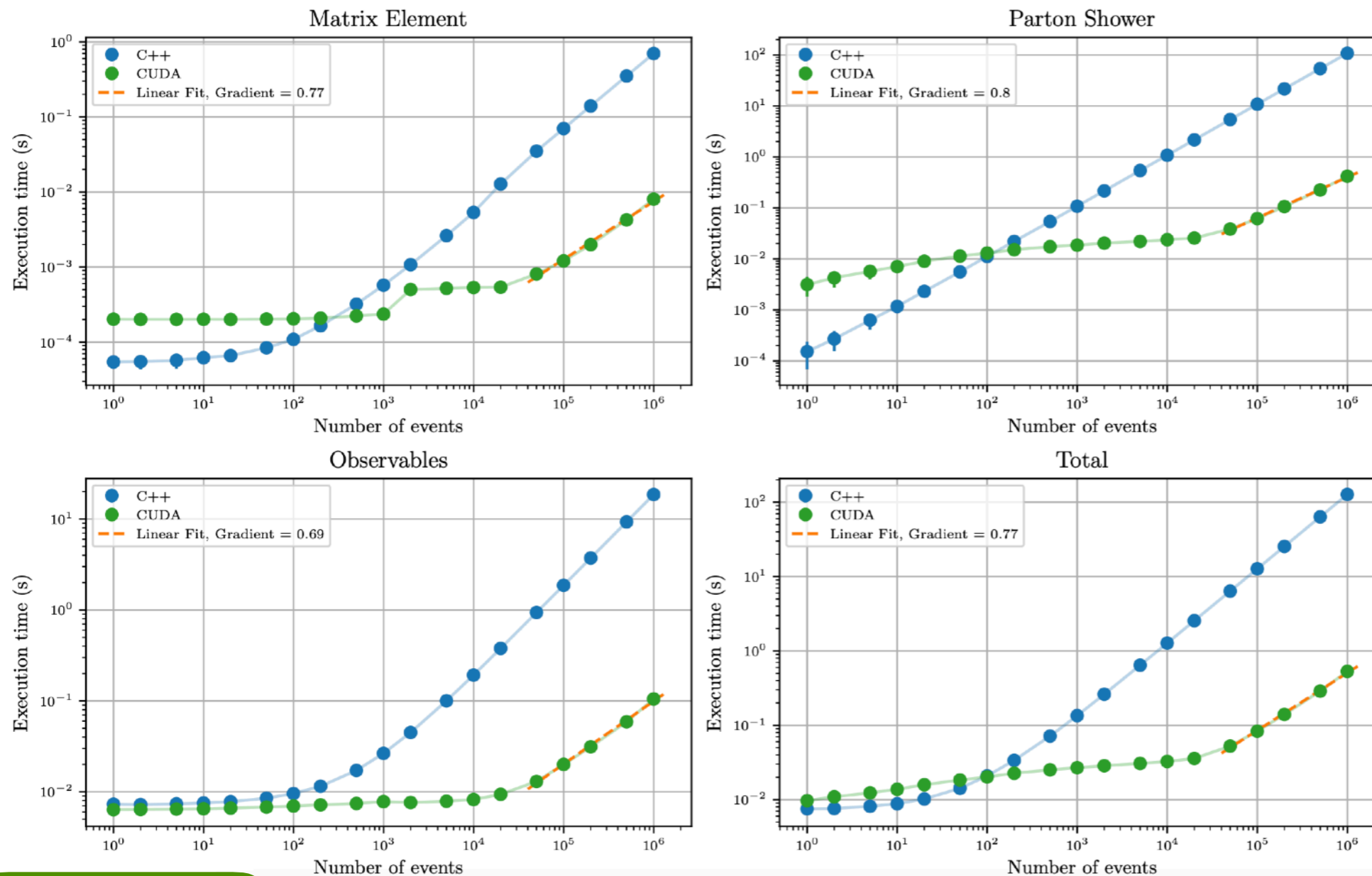


A practical example from yesterday's poster sessions

An Algorithm to Parallelise Parton Showers on a GPU

<https://arxiv.org/abs/2403.08692>

Michael H. Seymour and Siddharth Sule*



Also event generators using GPUs (Madgraph, Sherpa, most likely others I don't know about!) as well as HEP experiment trigger / reco software frameworks...

Energy
efficiency

MANCHESTER
1824

The University of Manchester

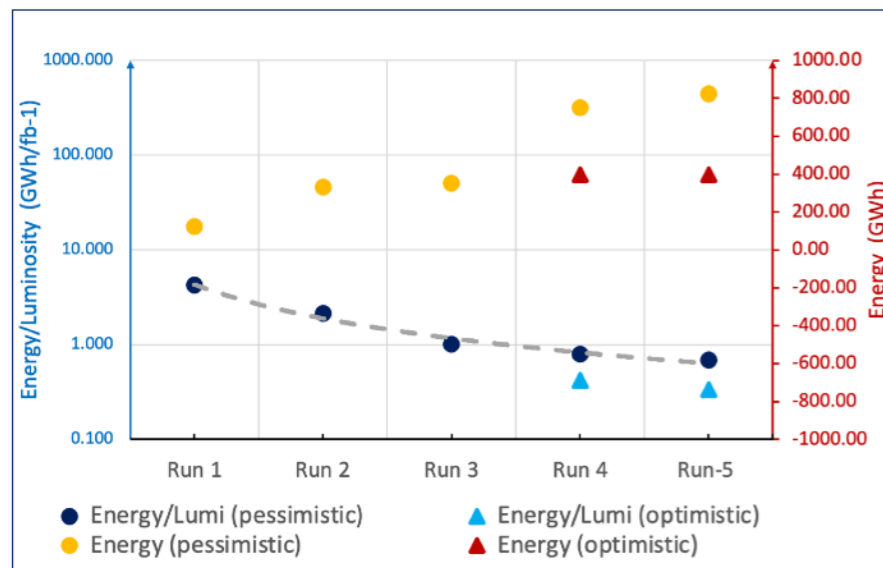
Not in this talk: Worldwide LHC Computing Grid

Proposal by S. Campana, D. Britton, T. Boccali at HEP Software Foundation / WLCG workshop 2024
Talk by S. Campana, D. Britton, B. Panzer at CHEP 2023

In WLCG $\text{GWh}/\text{fb}^{-1}$ represent the energy needed to **analyse** the data

The scale on the right (**RED**) shows the energy and the scale on the left (**BLUE**) shows $\text{GWh}/\text{fb}^{-1}$ (log!)

Energy needs in Run-4 and Run-5: +100% compared to Run-2 in the **pessimistic** scenario, only +10% in the **optimistic** scenario



Proposal

A lot of work happening in our community in the direction of environmental sustainability (experiments, sites, software community)

There are many aspects to consider: requires different kind of expertise

As a first step we propose to organise a workshop to present to the whole community the ongoing efforts. This will allow to identify synergies and gaps. Target period: Fall 2024

The outcome of the workshop should include:

- Agreeing on the metrics
- Indicating the main areas of work
- Identifying opportunities to contribute

The progress should be regularly tracked according to the agreed metrics

Energy efficiency

Carbon awareness

Hardware efficiency

Measurements



Take home point 3:

It's a good time to look into energy consumption / carbon intensity as a metric for HEP software and hardware
also out of necessity, see next part...

A user story by E. Chadwick

Slides from [Talk by E. Chadwick at the HEP Software Foundation + Swift-HEP C++ course in Manchester](#)

Unsustainable Software

Two students collaborate to model telescope observations of a specific type of astronomical event.

- They start together, but their code diverges as they each study a different telescope
- As they prepare to submit a paper, one of them introduces a major bug and only notices it at the last minute while doing a manual check
- Six months later, changes are requested – but they no longer remember which code files worked and which were broken
- Replicating their own results is tough
- After the paper is accepted, both students move onto other projects, and don't touch the code again

Unsustainable Software - Consequences

- The code is difficult to understand and use
 - Effectively nobody can replicate the results
- The code for the telescope models cannot be applied to other types of astronomical event, even though the maths is generalised
 - A later student must start over from scratch
- The code was never reviewed for bugs
 - The published results could contain major errors
- The code is abandoned after the paper is published, and the Python version it used falls out of support soon after
 - Using the code in any way becomes harder over time

Note: there is no shame in sharing code with the purpose of improving it!

1. Admit that you have a problem

Hello, I am Eli, and I've written terrible software.
 It's ok to not write perfect code
 It's ok to write very imperfect code
 Just don't deny that your code could be better out of pride or shame

I have been there, and I have the feeling that most of us (even among the lecturers) have been there too!



The University of Manchester

Is *your* code sustainable (in the FAIR sense)?

Slido link
slido.com #3524741



- Is easy to evolve and maintain
- Fulfils its intent over time
- Survives uncertainty
- Supports relevant concerns (political, economic, social, technical, legal, environmental)

Meanwhile, let's try with ... [we need a brave MC generator lecturer volunteering]: <https://fairsoftwarechecklist.net/v0.2/>

More complex (there is a human behind this):

<https://www.software.ac.uk/resources/online-sustainability-evaluation>



Software Sustainability Institute: practical suggestions

Slide/recommendations from [Talk by E. Chadwick at the HEP Software Foundation + Swift-HEP C++ course in Manchester](#)

1. Admit you have a **problem** = your code is not *perfect*
2. Introduce a **repository and version control** system
3. Make **changes little and often**
4. Get your code to **compile, build and run** on a machine that isn't yours
5. Invest some time in **automating** and **formalising tests** on your code
6. Make your code **modular**, build it up from simple interacting components
7. **Share** your code: both when it's not ready (to get feedback) and when it's ready (to make it Open Source & citable)
8. Join a **community of practice** of people who discuss similar questions/challenges for domain-specific quality checks

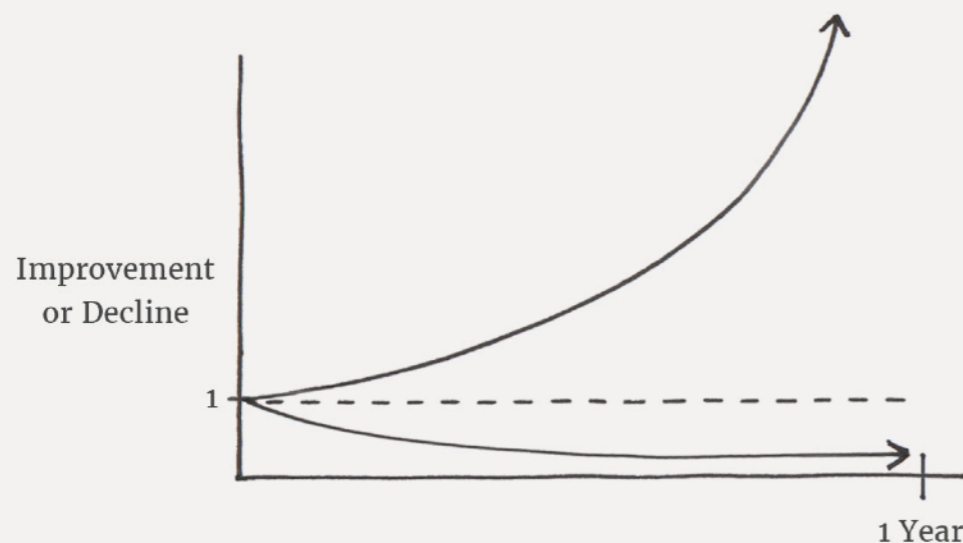


The right mindset to make software sustainable

Slide/recommendations from [Talk by E. Chadwick at the HEP Software Foundation + Swift-HEP C++ course in Manchester](#)

The Power of Tiny Gains

1% better every day $1.01^{365} = 37.78$
1% worse every day $0.99^{365} = 0.03$



JamesClear.com

- Don't do everything at once
- You can pick and choose...
 - And start with the lowest hanging fruit!
- Strive to make *small improvements every time you write a new piece of code*



What should the incentives be?

Let's have a look at your answers...

<https://devblogs.microsoft.com/sustainable-software/philosophy/>

Two philosophies of Sustainable Software Engineering



Asim Hussain

August 17th, 2020 | 0 | 0

*Everyone has a part to play
in the climate solution*

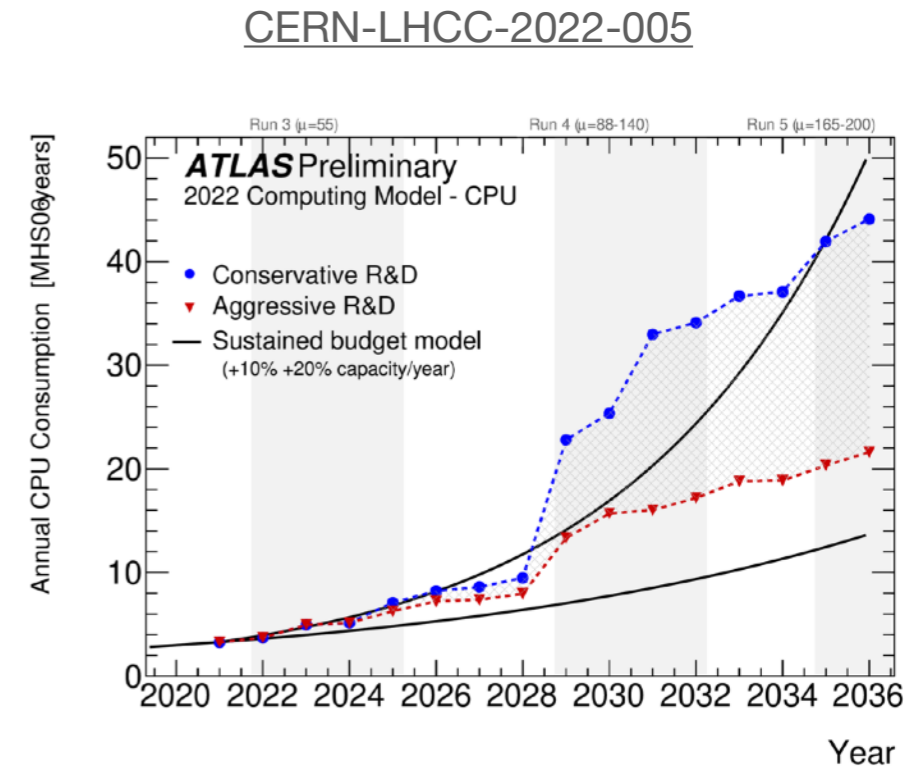
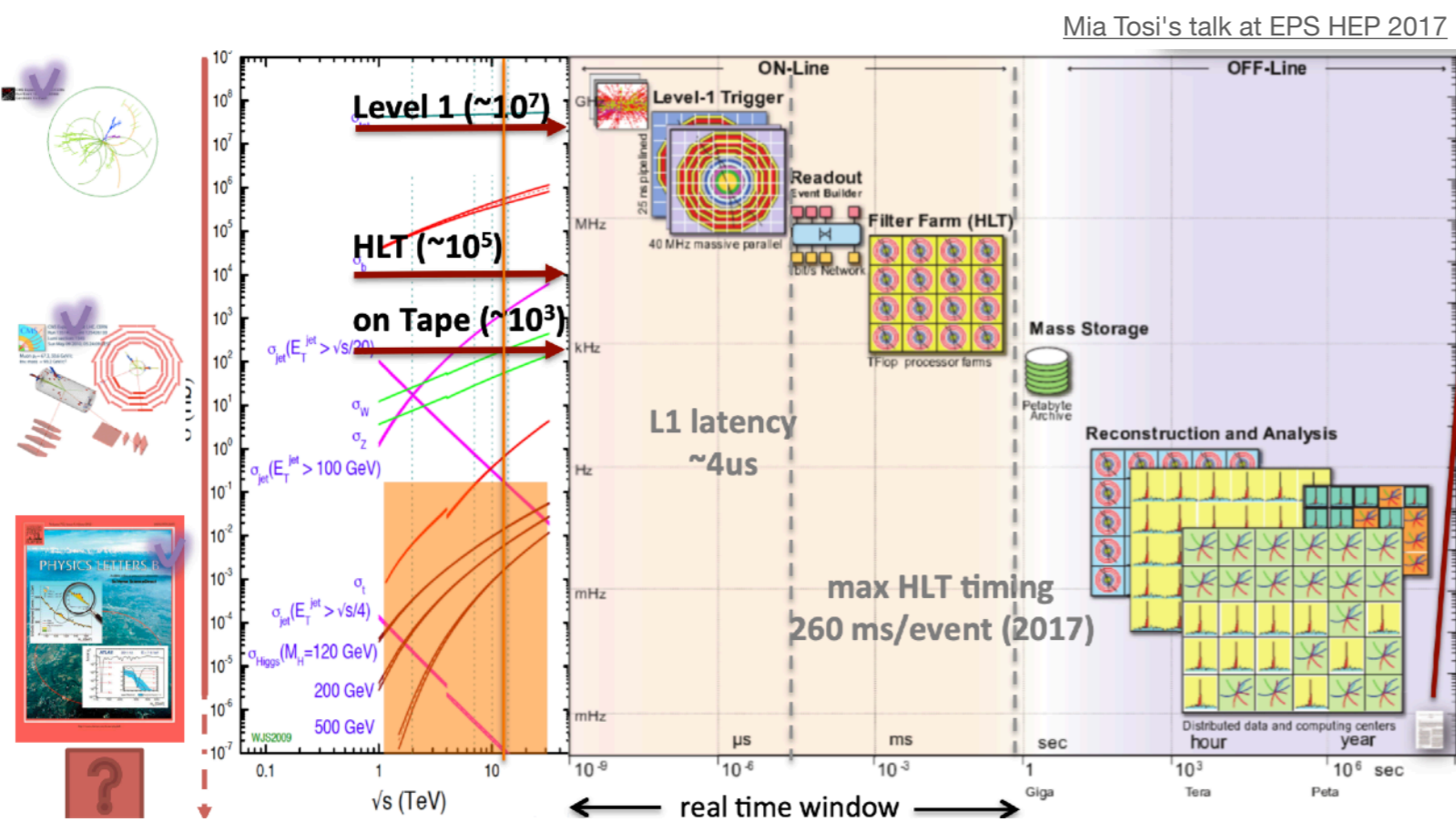
There is a global movement/
community who cares and takes
action through connected changes

*Sustainability is enough,
all by itself, to justify our work.*

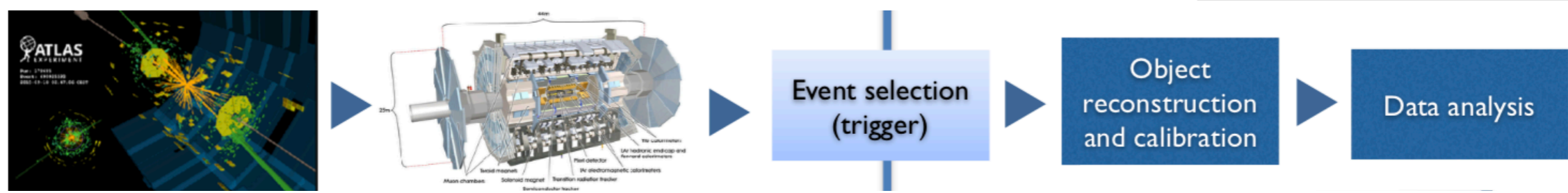
There are many advantages of
sustainable / energy efficient / green
software...see next slide for HEP



Incentives for HEP: we can't do all the physics with inefficient software!



This also includes generation of **large MC samples**



CPU for selecting & processing events **large but limited processing power**

For similar problems shared w/industry:



The Software Sustainability Institute advocates for:

<https://www.software.ac.uk>

About us

The Software Sustainability Institute is the first organisation in the world that was dedicated to improving software in research. We help people build better software and more sustainable research software to enable world-class research.

Read more >



Recognition of software as a research output



Software skills



Recognition of the role of RSEs



Reproducible research

More career-related incentives needed in our field!

(Including towards those who do training)



The University of Manchester

Take home point 4:

Writing sustainable software (in the broadest sense) is worth doing & necessary for our field, but we need to improve incentives in terms of career paths and funding

The next 3 slides will have too many acronyms

Very little time left, but wanted to close with a brief list of initiatives around sustainable software in HEP that you can join (that I know about / I'm involved in - please email me if you know more / want to join!)

HSF = High Energy Physics (HEP) Software Foundation

- Forum for physicists with interest in software for HEP
 - ...and beyond: contacts and shared meetings with nuclear physics, accelerator, DM experiments
 - Interacts with LPCC on community software
 - Many sub-topics, including *trigger & reconstruction*, *MC generators*, *analysis software* and *training*
- In progress: **community software / projects affiliation scheme**



[Website](#)
[Discussion Forums](#)

Other software-related national projects / resources (with HEP focus)

[IRIS-HEP \(US\)](#)

[Swift-HEP \(UK\)](#)

[PUNCH4NFDI \(DE\)](#)

Training center from Germany



ESCAPE/EOSC-Future Dark Matter Science Project

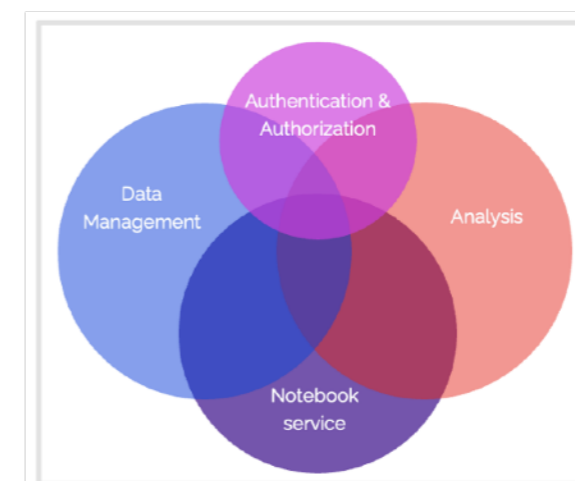
Can we make entire HEP/astro
(dark matter) software pipelines FAIR?
(*You can try yours out as well!*)

<https://eoscfuture.eu/data/dark-matter/>
+ links there

- ESCAPE is an **Open Collaboration** (previously EU-funded) that brings together different research infrastructures
 - 10 ESFRI (CTA, EST, FAIR, **HL-LHC**, **KM3NeT**, **SKAO**, LSST, VIRGO, ESO, JIVE)
 - 2 pan-European International Organisations (**CERN**, ESO)
 - 4 supporting European consortia (APPEC, ASTRONET, ECFA, NuPECC)
 - *Position statement of all clusters (including ESCAPE) on operational commitment to EOSC and Open Research can be found at <https://zenodo.org/records/10732049>*
- ESCAPE services tied into European Open Science Cloud (EOSC) through the EOSC-Future project (now concluded)
- Main workhorse for running workflows: **ESCAPE Virtual Research Environment (VRE)**



Virtual Research Environment



Two thematic **Science Projects (SP)** within EOSC-Future have data and workflows on the ESCAPE VRE (EVERSE has representatives from the Dark Matter SP)

VRE: open-source analysis platform with a single entry point

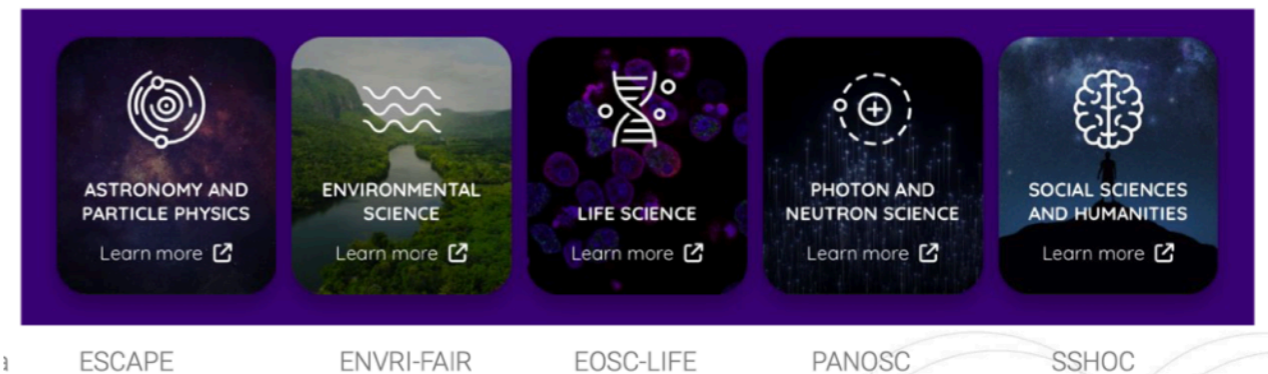
(Mostly) compliant with FAIR principles

Caterina Doglioni - 2024/06/10 - MCNet School @ CERN



EVERSE (EU and beyond) <https://everse.software/>

Can we create a **toolkit of resources for software excellence** that is useful for different scientific communities (European Science Clusters)?
[includes reward/training angles] →



Goals

- Build a **community-led structure** for improving for improving the quality of research software and code
- Leverage existing tools and processes** within the research community
- Establish a sustainable and collaborative **ecosystem to ensure research software and code quality**
- Provide a **framework** to ensure **recognition, reward** and **career development** for researchers and RSEs

Other reproducibility/FAIR-related national projects / resources (with HEP+ML focus)

[FAIR4HEP \(US\)](#)

[FAIROS-HEP \(US\)](#)

[EuCAIF \(WG3\) \(EU\)](#)

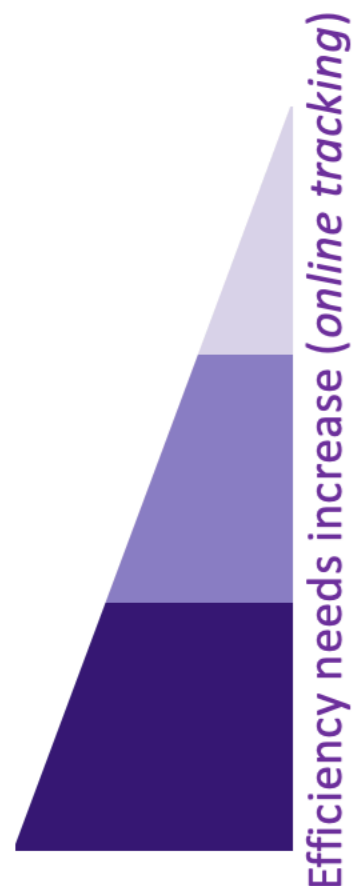
(Still a WIP, but coming soon!)



EVERSE (EU and beyond) <https://everse.software/>

First ESCAPE pilots loosely inspired by the 3-tiers of software

4



Software to reconstruct particle tracks (on CPU and GPU) [C++/CUDA, Sycl]

- efficiency AND sustainability crucial, hard constraints on execution time
- UofM will collaborate with University of Huddersfield RSE (C. Venters) on a "software quality" paper

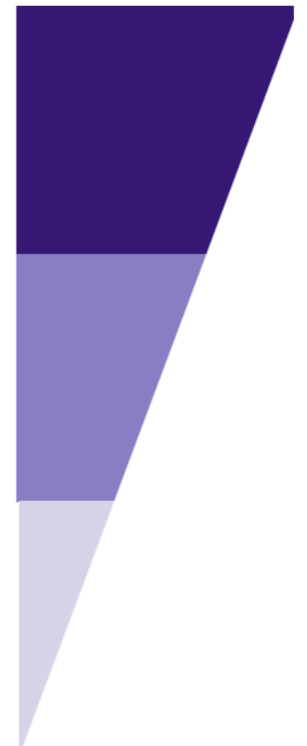
ML-based data compression software [Python]

- Early-Stage Researcher-led open source software project
- already on VRE as a Jupyter Notebook

Framework to analyse LHC data [C++ / Python]

- Developed for analysis of o(TB) of LHC collision events
- undergoing rewrite / refactoring in the next year
- good timing for software quality improvements

Number of (concurrent) users decreases



Mostly representing particle physics community

More pilots welcome! New call for funding in Fall 2024 at <https://oscars-project.eu/>



Take home point 5:

You can get involved in software sustainability using your own test cases, work with very nice and competent people (this is generally my policy!)

Conclusions

- Various definitions of **software sustainability**:
 - **FAIR** software principles
 - **Environmental** sustainability
- Making software sustainable is:
 - A **necessity** (we don't have infinite resources, computing & people)
 - A **drive** (reproducibility \leftrightarrow scientific method, climate responsibility)
- Making software sustainable requires **small improvements** rather than overhauling everything
- Many **ongoing initiatives** on making software more sustainable & rewarding the effort