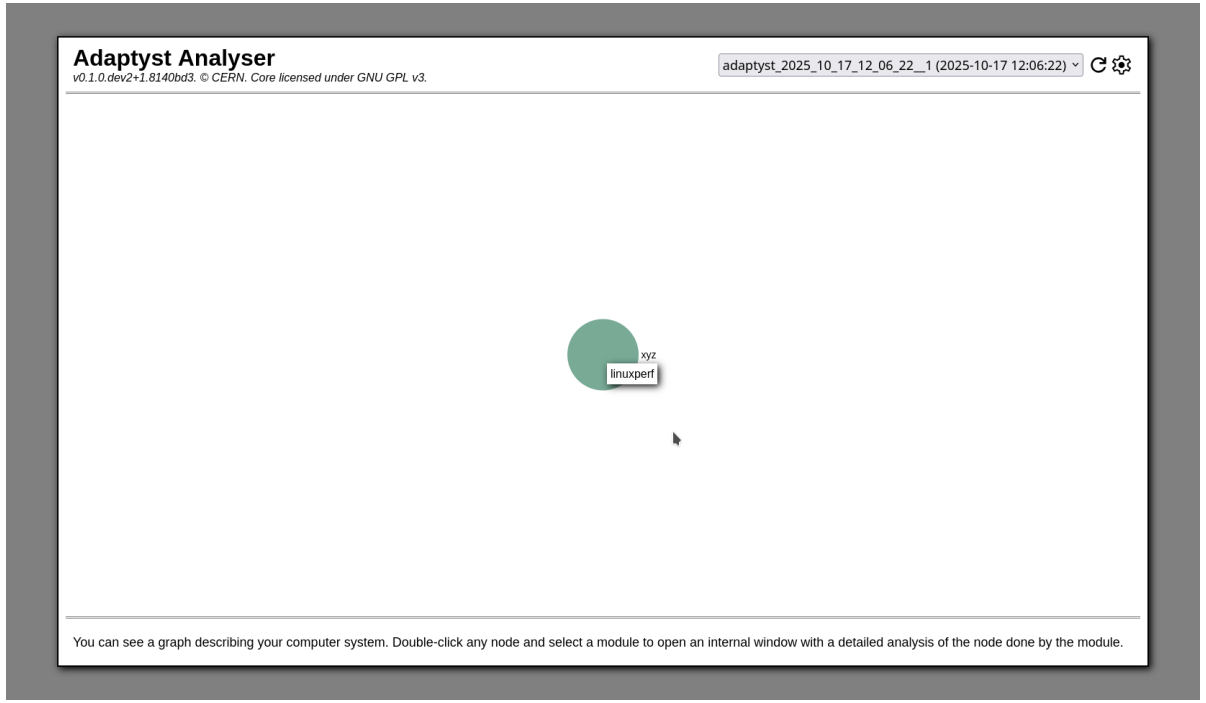
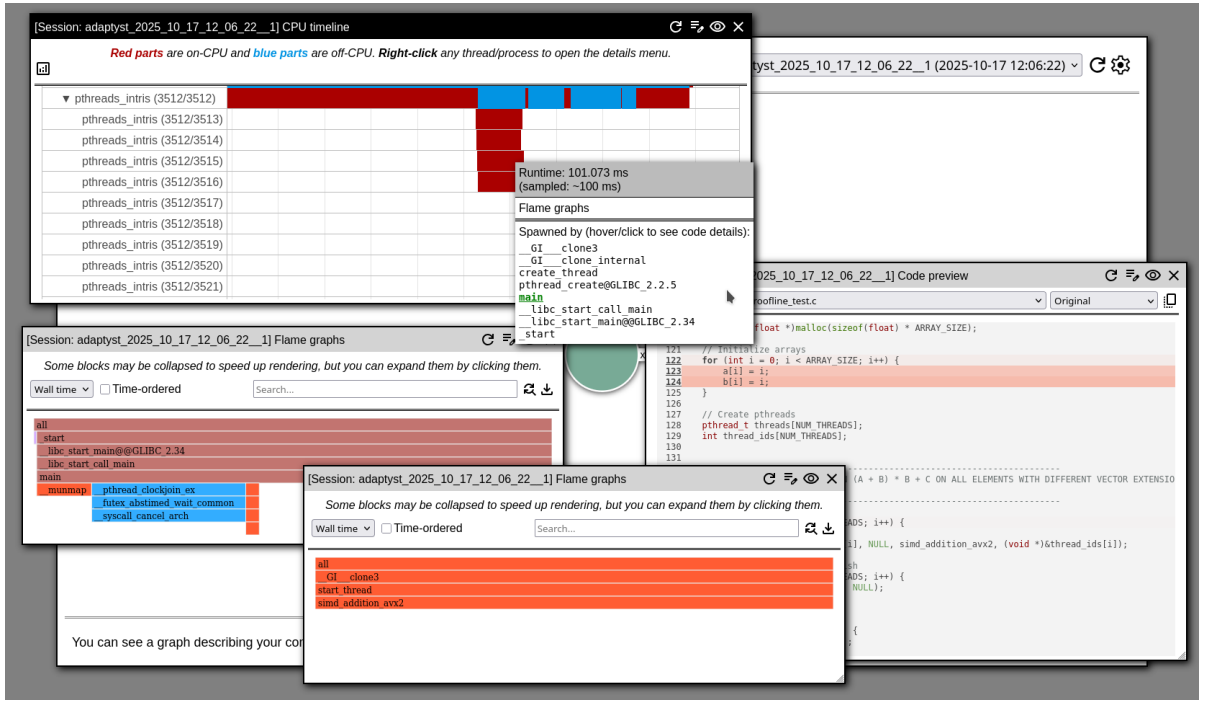




Towards unified full-stack performance analysis and automated computer system design at CERN with Adaptyst

Maksymilian (Maks) Graczyk, CERN

FOSDEM, 1 February 2026

With thanks to Anton Alkin, Giovanni Iadarola, Maarten Litmaath, Daniele Massaro, Sebastien Ponce, Lorenzo Rossi, and Sioni Summers

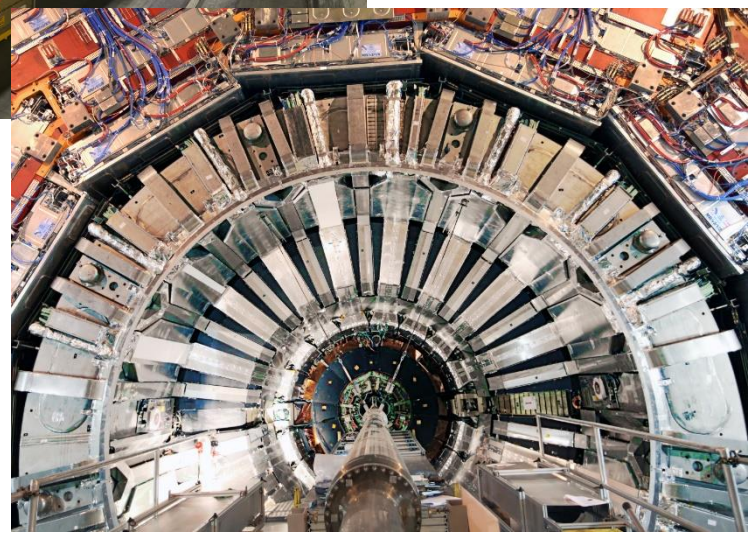
Hello!

- I am Maks, currently a junior performance research engineer working for nearly 3 years at CERN and starting a PhD at CERN and ETH Zürich in April this year.
- My background is computer science: I graduated with an integrated Master's degree from Imperial College London in 2022.
- Everything at the software-hardware boundary is what excites me the most, especially in the context of applications to physics and space research.
- This is my first FOSDEM ever: apart from presenting Adaptyst, I am here also to learn and network :)

What is CERN?

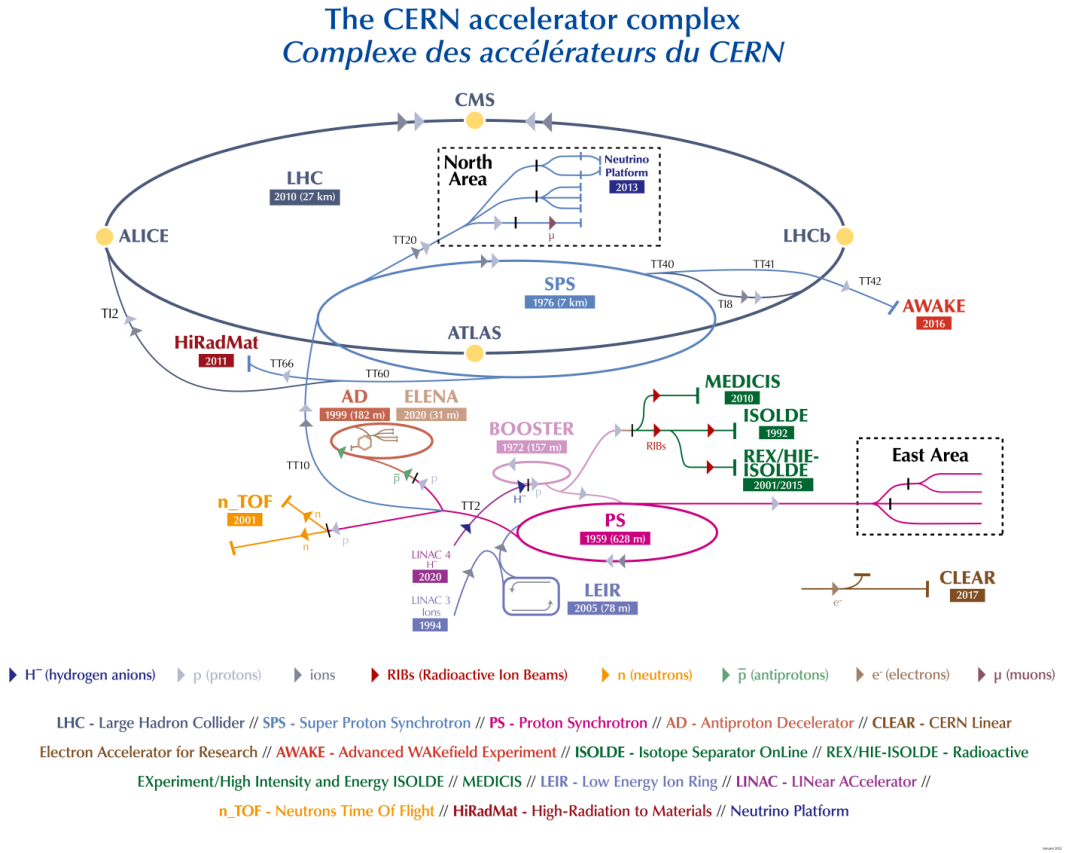
- CERN (European Laboratory for Particle Physics) is the world-leading particle physics facility located in Geneva, Switzerland.
- It is best known for the Large Hadron Collider (LHC), the place of the Higgs boson discovery.
- Apart from particle physics, research carried out at CERN includes astrophysics and cosmology.

Credit: CERN



Computing at CERN

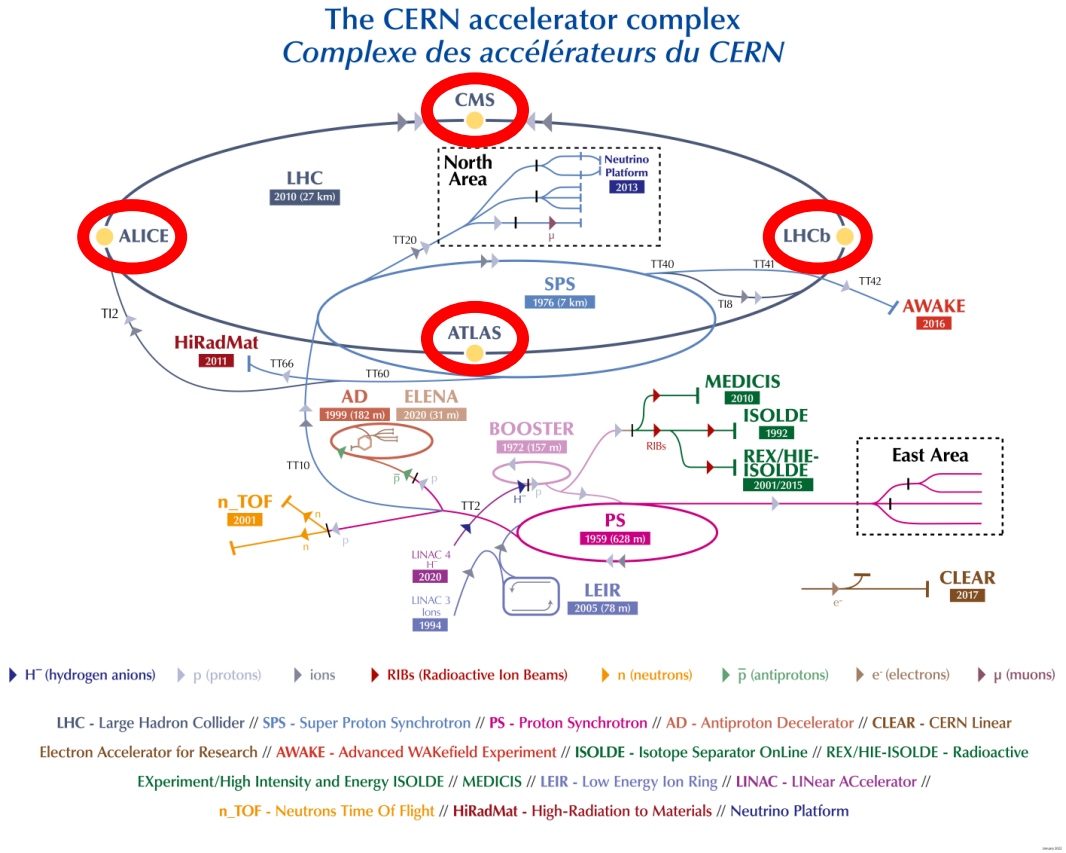
- Computing is one of the three main pillars at CERN, alongside accelerators and detectors.
- The complexity of the CERN accelerators and the amount of data gathered by experiments hosted there create multiple computing challenges at all levels.
- The flagship collider (LHC) features four major experiments at collision points: ATLAS, CMS, LHCb, ALICE.



Credit: CERN

Computing at CERN

- Computing is one of the three main pillars at CERN, alongside accelerators and detectors.
- The complexity of the CERN accelerators and the amount of data gathered by experiments hosted there create multiple computing challenges at all levels.
- The flagship collider (LHC) features four major experiments at collision points: ATLAS, CMS, LHCb, ALICE.



Credit: CERN

Performance challenges

What are the problems we encounter? **A non-exhaustive list in no particular order:**

- Performance analysis tool fragmentation or lack of such tools
- Necessity of optimising/porting codes per architecture
- Inability of analysing everything in one go or at all due to too complex systems or a mix of programming languages/models
- Dealing with resource limitations
- Strict time constraints in real-time systems
- Optimisations (across the stack) we're not aware of

Not just runtime, but also
e.g. energy efficiency

Performance challenges

What are the problems we encounter? **A non-exhaustive list in no particular order:**

- Performance analysis tool fragmentation or lack of such tools
- Necessity of optimising/porting codes per architecture
- Inability of analysing everything in one go or at all due to too complex systems or a mix of programming languages/models
- Dealing with resource limitations
- Strict time constraints in real-time systems
- Optimisations (across the stack) we're not aware of

Not just runtime, but also e.g. energy efficiency

Performance challenges

Only algorithmic, only hardware, or only (operating) system optimisations are not enough in isolation!

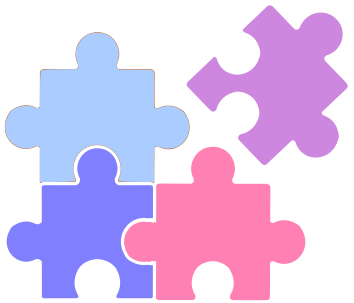
What are the problems we encounter? **A non-exha**
no particular order:

- Performance analysis tool fragmentation or lack of such tools
- Necessity of optimising/porting codes per architecture
- Inability of analysing everything in one go or at all due to too complex systems or a mix of programming languages/models
- Dealing with resource limitations
- Strict time constraints in real-time systems
- Optimisations (across the stack) we're not aware of

What is Adaptyst?

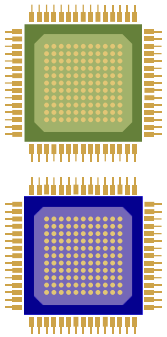
An <u>early-phase</u> comprehensive and architecture-agnostic performance analysis tool addressing your software, hardware, and system needs. Both today and tomorrow.

Born as part of the SYCLOPS project funded by the European Union.



Modular design

Adaptyst will always be up-to-date with the market thanks to the modular design based on system/HW modules (and workflow plugins soon) developed by external developers from the SW and HW worlds.



Architecture-agnostic

Intel? AMD (CPU)? ARM? RISC-V? NVIDIA GPU? FPGA? Something else? Adaptyst gets you covered.

All-in-one analysis

Thanks to the linuxperf module, Adaptyst analyses on-CPU and off-CPU activity for all threads and processes of your code along with low-level SW-HW interactions if possible.

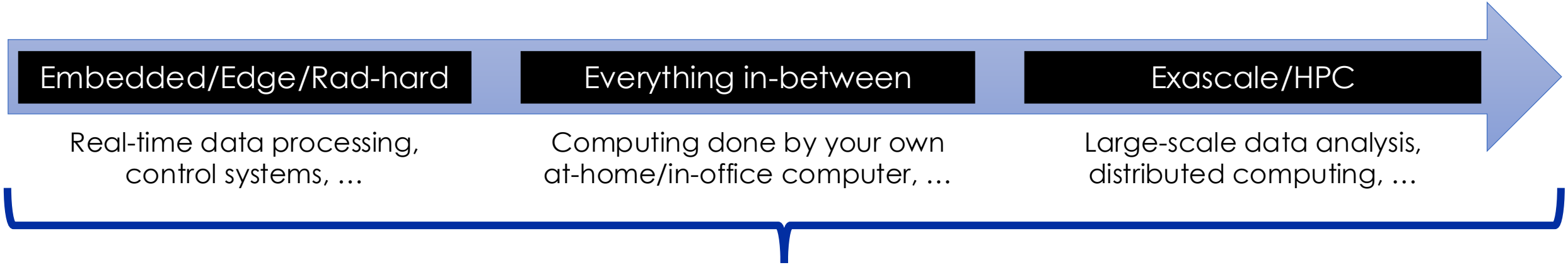
Open-source from CERN

Adaptyst is an open-source project developed at the leading particle physics laboratory for the benefit of all.

Icons based on the ones designed by Freepik except for the CERN logo.

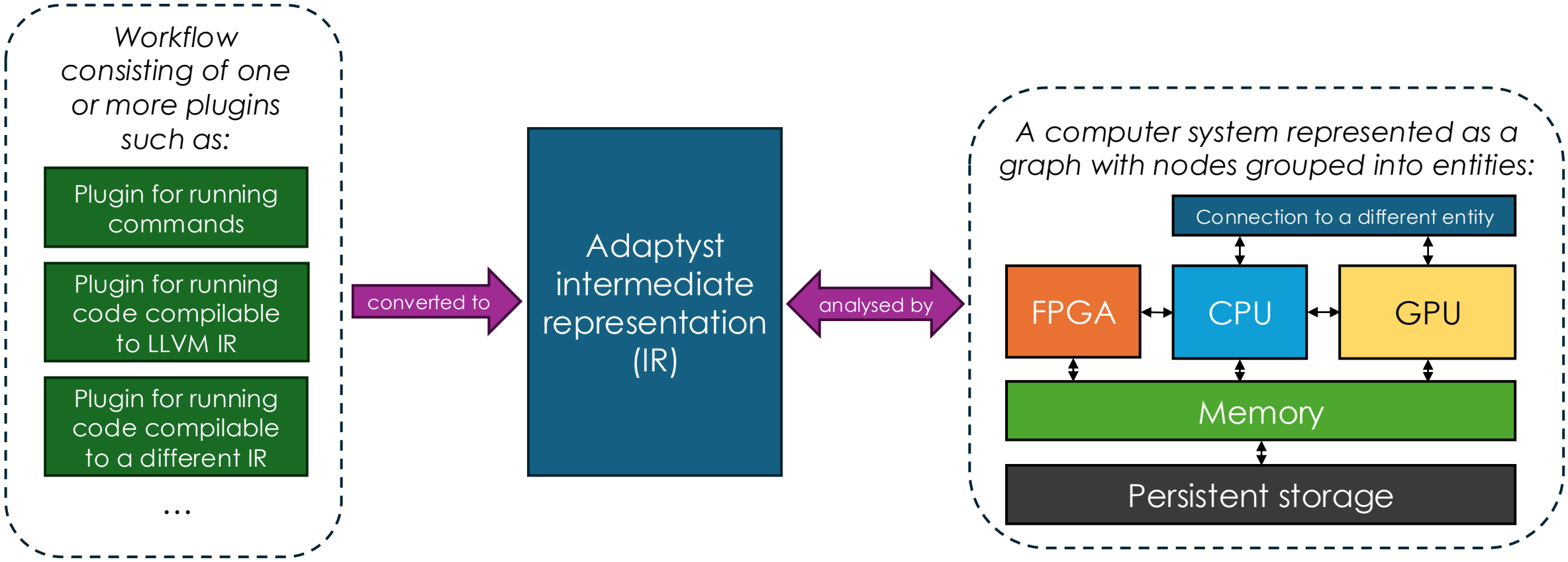
Adaptyst goals

- We want Adaptyst to tackle the aforementioned challenges and help advance computing at CERN + beyond.
- Given the variety of platforms (CPUs with ISAs, GPUs, FPGAs, ASICs etc.) and use cases, the goal of Adaptyst is becoming **a full-stack system design/compilation and SW-HW co-design tool <u>across the entire spectrum</u>**:



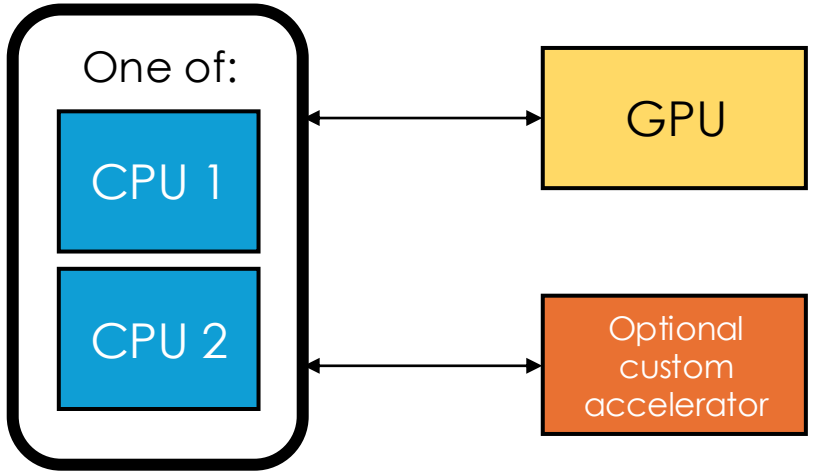
To be covered by Adaptyst!

The shown structure and used work such as IR may change as the R&D progresses.



System graph

- A system graph describes a computer system where a given user workflow should be executed. It consists of nodes grouped into entities:
  - A node can be imagined as a computer peripheral (CPU, GPU, FPGA etc.).
  - An entity can be imagined as a computer server.
- Each node has one or more modules inside. A module is responsible for modelling/profiling a given system component and describing how its performance analysis results should be displayed afterwards.
- The graph will be able to be **constructed automatically by Adaptyst** with hints from a user.



Modules provided by us (currently Linux only)

- **linuxperf:** CPU activity analysis based on "perf" with the custom patches (moving to an in-house equivalent based on `perf_event_open(2)` and/or eBPF soon)
  - Samples both on-CPU and off-CPU activity of every spawned thread and process
  - Minimises risk of broken profiled stacks for programs compiled with frame pointers
  - Performs cache-aware roofline profiling using the CARM Tool from INESC-ID
  - Supports custom sampling-based "perf" events for profiling SW-HW interactions
- **nvgpu:** NVIDIA GPU activity analysis based on CUPTI
  - Traces CUDA API calls in a basic way (runtime and/or driver)
  - More features coming as time passes and users provide their feedback!
- **cusmet (soon):** analysis of any metrics obtainable from external tools
- **soc-baremetal (soon):** analysis of bare-metal software running on SoC's

Performance metrics

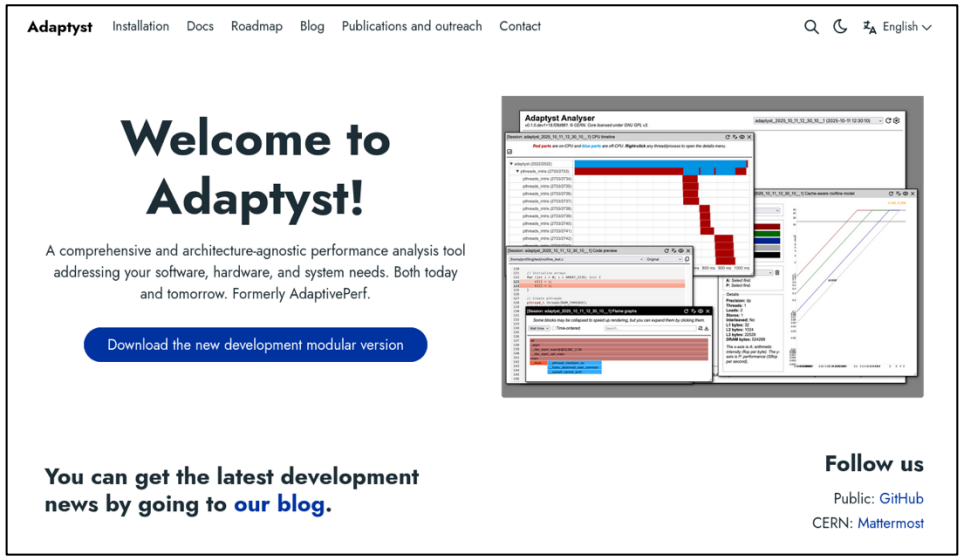
What are the things we (want to) analyse at CERN? **A non-exhaustive list in no particular order:**

- Latency, throughput etc.
- Usage of memory and other resources
- Parallelisation opportunities
- Utilisation of low-level hardware features
- Performance of code compilation + JIT if used
- Energy/Power consumption

State of Adaptyst features:
Currently supported
Already planned or can be added in the future

How to download Adaptyst?

- It's open-source and you can get it for free from our website along with the documentation: https://adaptyst.web.cern.ch (the repositories are on GitHub).
- Adaptyst is available **as an early development version**, in form of a source code (other variants such as container images and pre-built binaries are coming soon!).



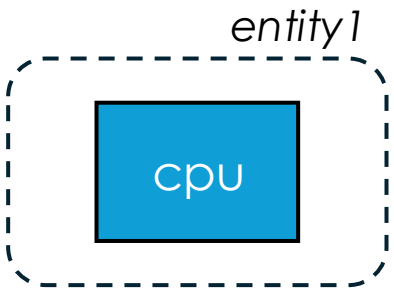
Website

GitHub

Quick start with Adaptyst

Define a simple one-entity system graph with a CPU node to be analysed by the linuxperf module and save it to **system.yml**:

```
entities:
  entity1:
    options:
      handle_mode: local
      processing_threads: 1
    nodes:
      cpu:
        modules:
          - name: linuxperf
```



Quick start with Adaptyst

```
$ adaptyst -s system.yml -d -- <command to be profiled>
...
...
==> Done in 10.815 s in total!
-> The results are available in /home/profiling/test/adaptyst_2026_01_19_15_55_44__1
$ mkdir result_dir && mv /home/profiling/test/adaptyst_2026_01_19_15_55_44__1 result_dir/ &&
adaptyst-analyser <optional options> result_dir
...
...
...
...
[2024-10-12 13:57:52 +0200] [2192] [INFO] Listening at: http://127.0.0.1:8000 (2192)
...
```

Run this command (**no root***) and wait until it finishes.

Afterwards, start Adaptyst Analyser as shown.

Open the website in your web browser. Done!

*Under certain circumstances.

Adaptyst Analyser demo video:
https://cernbox.cern.ch/s/x7Ks6WIFxwDbHEA
(also in the talk resources at fosdem.org)

Side note: Codes analysed in the demo video involve the ROOT data analysis framework widely used at CERN.

ROOT website

How does Adaptyst compare to other similar and maintained profilers?

	Hardware-vendor-portable	Runs on RISC-V	Analyses software-hardware interaction[1]	Open-source	Off-CPU profiling	Flexible support of heterogeneous and custom architectures	Flexible support of multi-node systems
Adaptyst	**Yes**	**Yes**	**Yes**	**Yes**	**Yes**	**Yes**	**WIP[2]**
Original "perf"	Yes	Yes	Yes	Yes	Limited	No	No
Intel VTune Profiler	No	No	Yes	No	Yes	Intel only, NF[3]	MPI only, NF[3]
AMD µProf	No	No	Yes	No	Yes	AMD GPUs only, NF[3]	MPI only, NF[3]
valgrind	Yes	Yes, as a fork	No	Yes	No	No	No
gprof	Yes	Yes	No	Yes	No	No	No
gperftools	Yes	Yes	No	Yes	No	No	No
NVIDIA profilers	No	No	Yes	No	Yes	NVIDIA GPUs only, NF[3]	NF[3]
TAU	Yes	Yes	Yes	Yes	Yes	GPUs only, NF[3]	MPI only, NF[3]
HPCToolkit	Yes	No	Yes	Yes	Yes	GPUs only, NF[3]	MPI only, NF[3]

[1] If supported by a user's hardware architecture.
[2] Work in progress.
[3] Non-flexible because any architectural updates can be implemented only by the profiler developers or via code contributions if open-source.

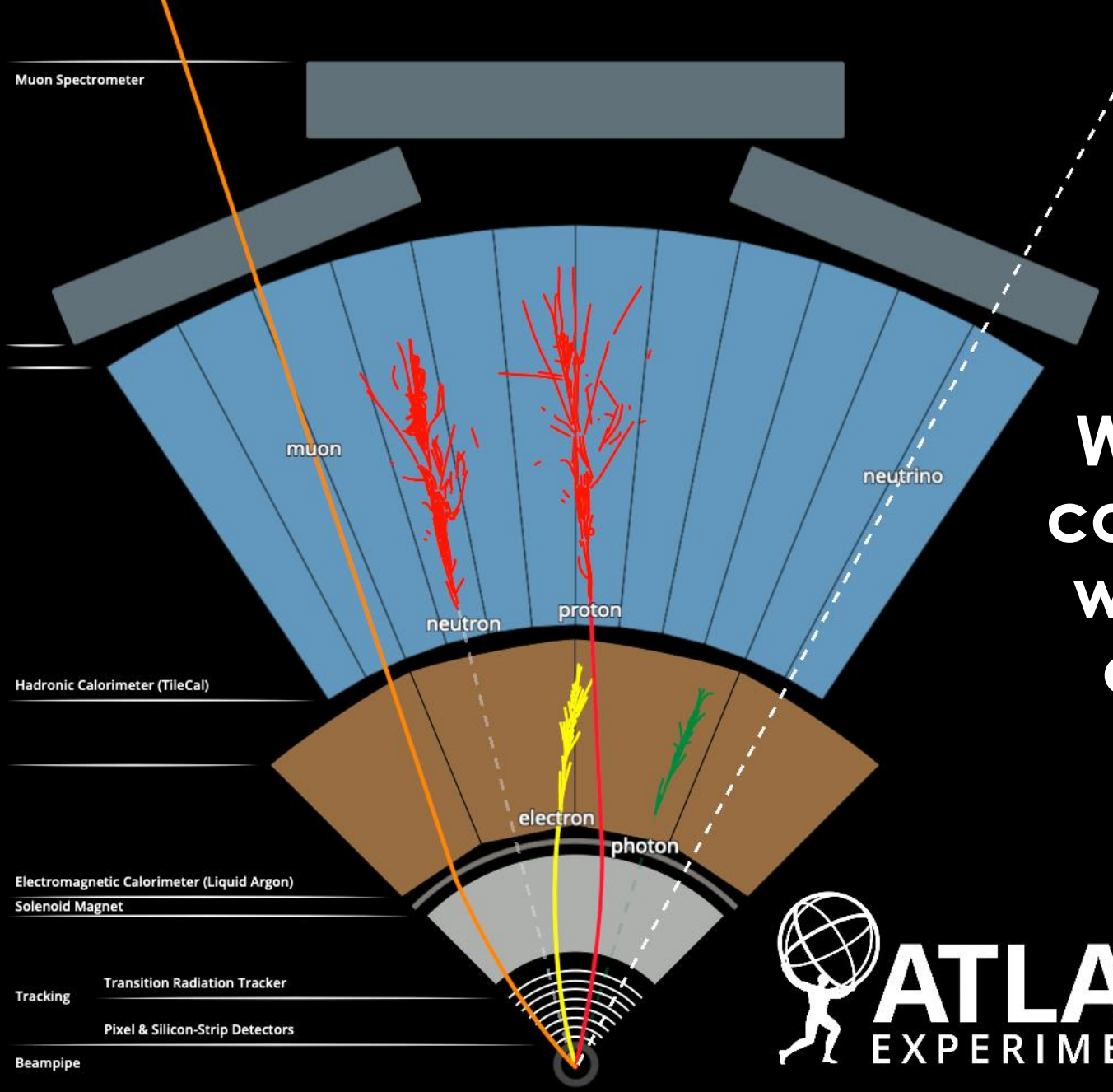
Contributing to Adaptyst

- Performance encompasses all of computing, so we're sure Adaptyst can help with your work: **please try it out and give us feedback!** We'll also be happy to see code contributions :)
- Your help will push forward R&D on automated software-hardware co-design and in case of modules, increase visibility of system/hardware products amongst users such as CPUs, GPUs, custom accelerators etc.
- By contributing to Adaptyst, you will also help address CERN performance problems and therefore help science!

Contributing to Adaptyst

- You don't know what/how to contribute beyond trying the tool out? Here are some ideas:
  - Writing modules for various system and hardware components
  - Integrating external profiling viewers with Adaptyst Analyser
  - Adding collaborative features to Adaptyst Analyser
  - Making a logo for Adaptyst :)
- Interested? Talk to me at the conference or send us a message: adaptyst-contact@cern.ch.

Credit: NASA/ESA and The Hubble Heritage Team (STScI)

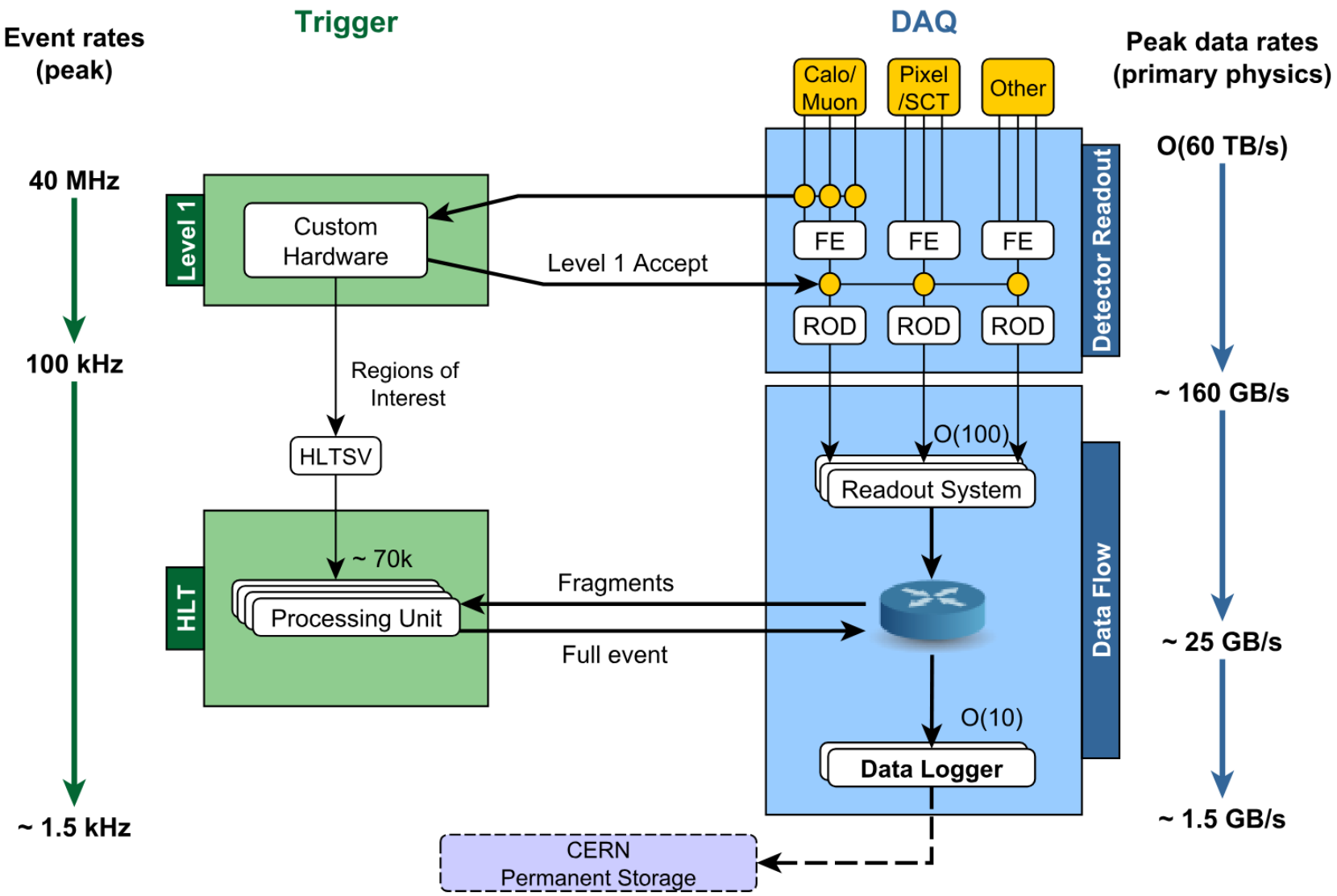


What happens computing-wise when particles collide in the LHC? (ATLAS experiment)

ATLAS EXPERIMENT

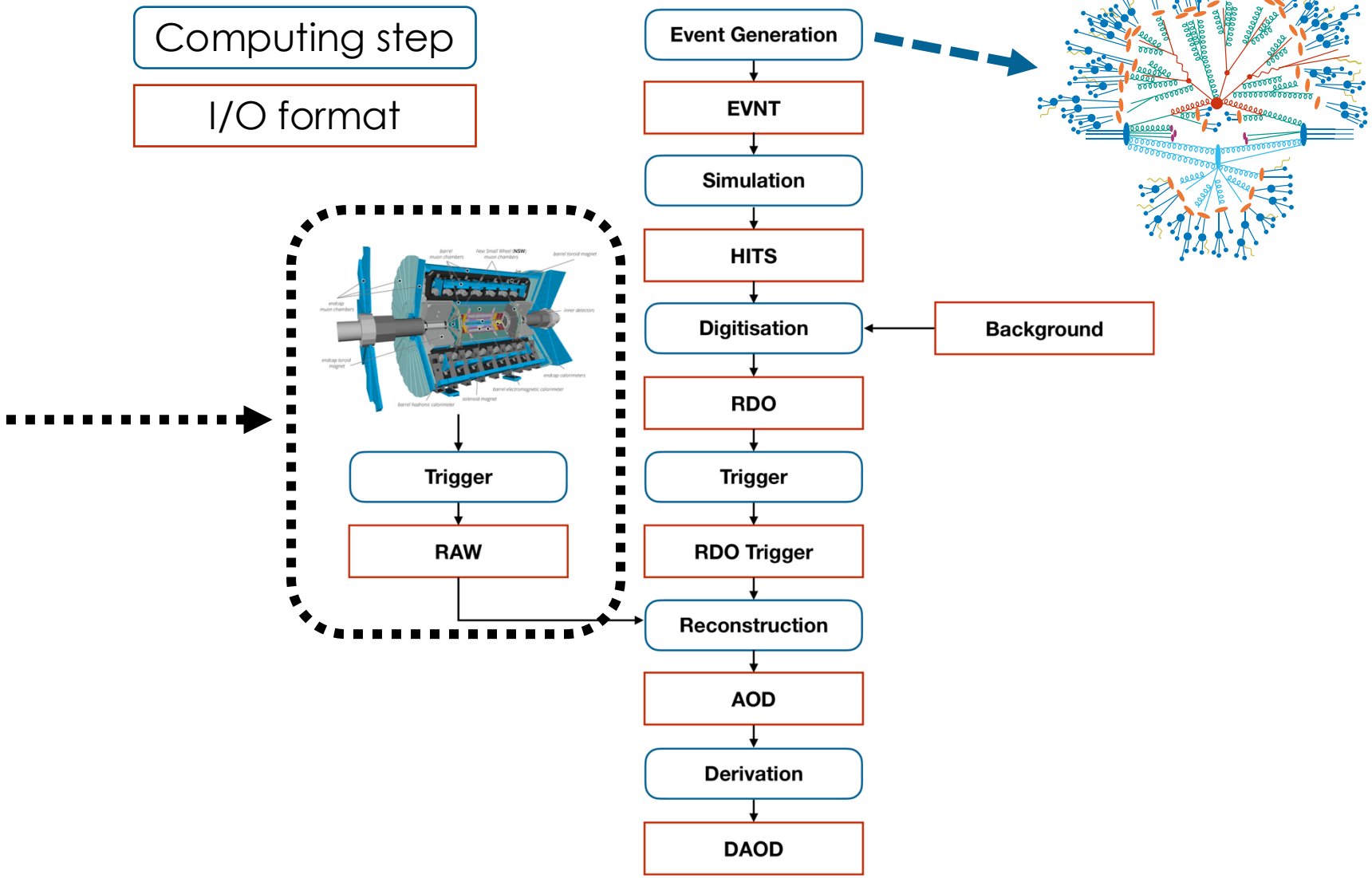
Credit: https://cds.cern.ch/record/2770815

What happens computing-wise when particles collide in the LHC? (ATLAS experiment)



Credit: A Borga et al. "The ATLAS readout system for LHC runs 2 and 3". In: Journal of Instrumentation 18.08 (2023), P08022

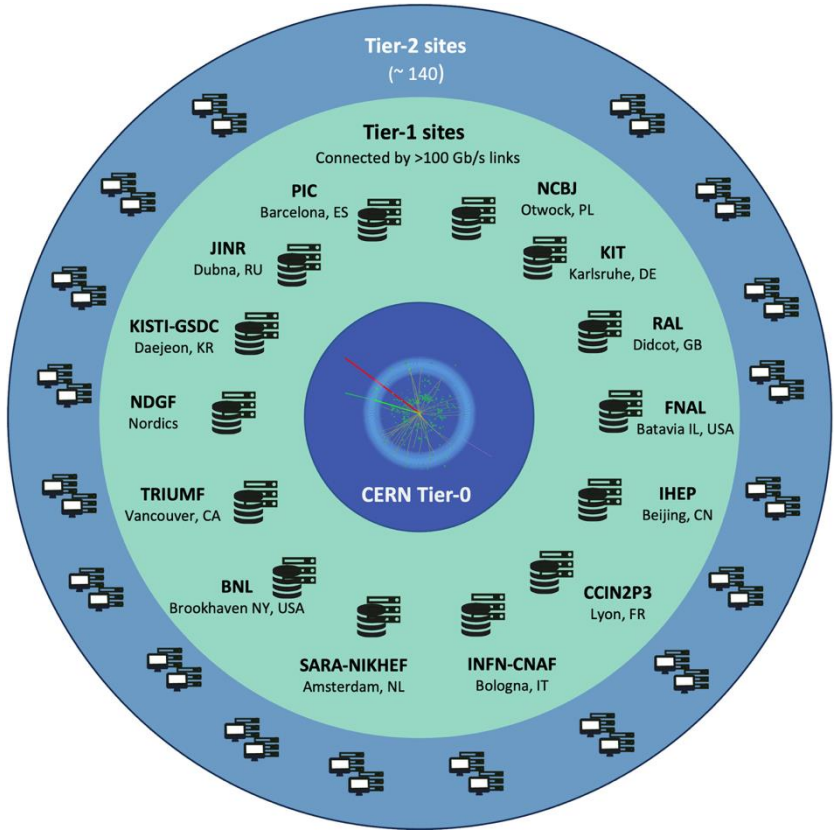
What happens computing-wise when particles collide in the LHC? (ATLAS experiment)



Credit: https://atlas.web.cern.ch/Atlas/GROUPS/PHYSICS/PAPERS/SOFT-2022-02 and the SHERPA collaboration

Worldwide LHC Computing Grid

- The amount of saved data (**500+ PB / year in 2024-2025**) means that at least **~1 million** CPU cores are needed for their processing, not to mention I/O and storage.
- These are arranged as a grid of distributed computers with dedicated networking:
  - Tier 0: CERN data centres
  - Tier 1: 14 sites
  - Tier 2: 130+ sites
  - Tier 3

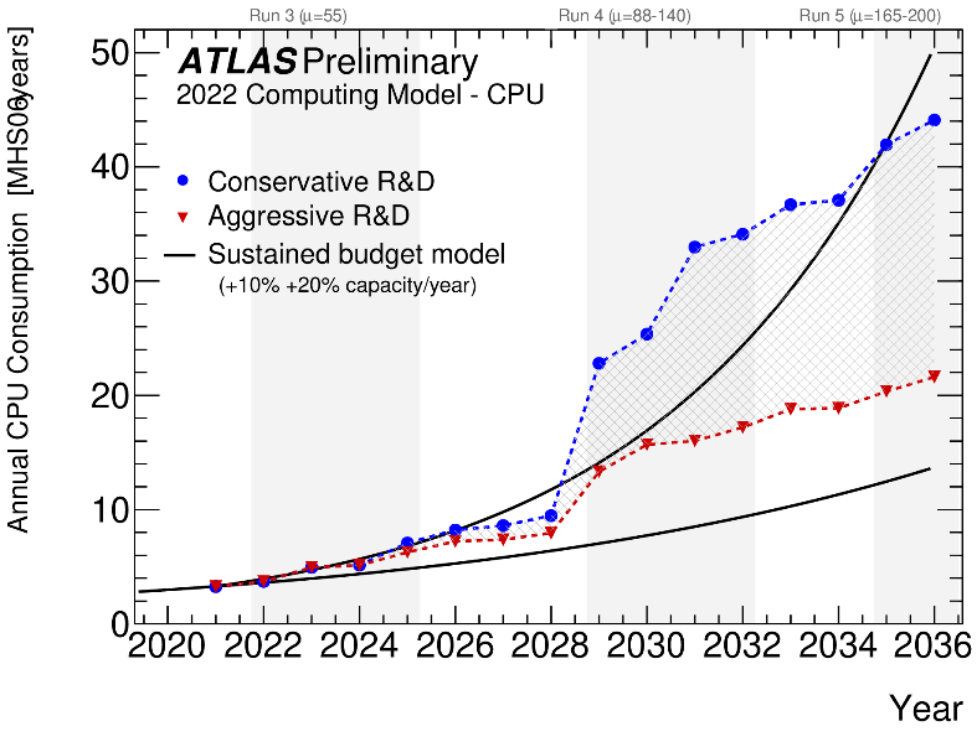


Credit: CERN

wlcg-public.web.cern.ch

Performance challenges

- LHC is going to be upgraded to High Luminosity LHC, increasing the number of particle collisions **by a factor of 5 to 7.5**.
- This **will** raise **significantly** the computing demands!
- Also: how to make sure that our algorithms/systems are **optimal under given performance and budgetary constraints**?
- Don't forget about recent trends in the technology world outside…



Credit: https://cds.cern.ch/record/2802918

Performance challenges

Summary of control systems

Xsuite docs

Other areas with performance-related issues or constraints (not exhaustive):

- **Accelerator control systems for steering and monitoring the CERN accelerators in real time**
- **Simulation of beams inside the accelerators with Xsuite**

Credit: https://cds.cern.ch/record/2915506, figure 1.2

Insufficient performance

⬇

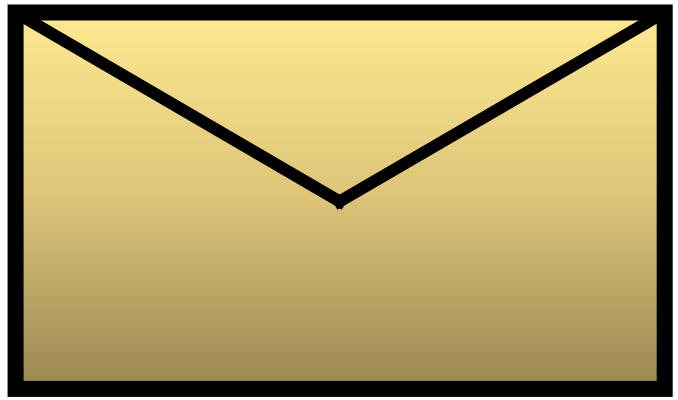
Tasks/Upgrades can't be finished on time or at all

⬇

Fewer and delayed physics results!

Thank you!

adaptyst.web.cern.ch



adaptyst-contact@cern.ch

Adaptyst development is funded until the end of March 2026 by the European Union Horizon programme: projects SYCLOPS (grant 10109287), EVERSE (grant 10112974), and OpenWebSearch.eu (grant 101070014). Funding beyond this date has already been secured from other sources.

Extra slides

Preliminary ideas

Full-stack system compilation / Software-hardware co-design

- A system component module will take an IR and any assumptions (e.g. the average number of iterations of a given unbounded loop) as the input and return a mathematical function g for all IR regions along with information of interest for a user, e.g. profiling results.
- The match (a real number between 0 and 1 inclusive) will be calculated for various categories **per node**:

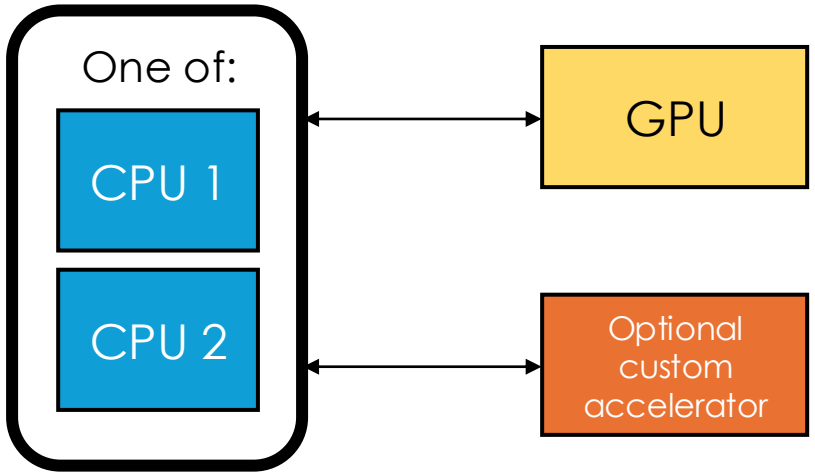
$$M_{category} = h_{category}\left(\sum_i \alpha_i \, g_i(X, Y), \beta\right)$$

X: optimisable software parameters
Y: optimisable hardware parameters
α_i: weight of the i-th module of a node, where $0 \leq \alpha_i \leq 1$ and $\sum \alpha_i = 1$
β: user constraints, e.g. maximum power consumption

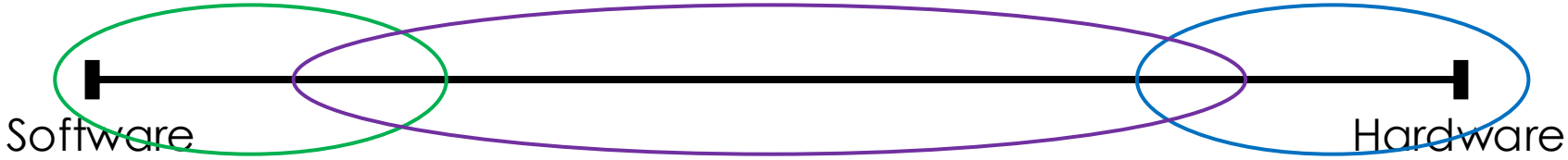
Preliminary ideas

Full-stack system compilation / Software-hardware co-design

- Software-hardware co-design will be achieved by optimising X and Y such that the matches of interest are maximised. **This will ultimately include automatic system component matching.**
- After determining the most optimal system, a final user workflow code will be generated that can be compiled with already-existing tools for the suggested platform.



Related work: software-hardware co-design grouped by the customisation target



Transparent compilers, portable and vendor-specific parallel programming models, standard compilers:

- Significant promise in terms of code portability (except vendor-specific models) and performance programming facilitation
- No hardware optimisations
- Explicit accelerated code needed, potentially with a different paradigm than the rest of the code (except transparent compilers)
- Lack of the bigger picture
- Limited language support
- Limited automatic hardware selection / automatic accelerated code selection

Software-hardware co-design frameworks and modular compilers:

- The best of both worlds: effortless programming, the potential for optimising both software and hardware, better scalability across languages
- Lack of the bigger picture
- Limited automatic hardware selection / automatic accelerated code selection

Adaptyst will bridge the gaps here!

High-level synthesis:

- Takes hardware optimisations into account
- Assuming that custom hardware must be generated for an entire code
- Lack of the bigger picture
- Hardware-like thinking required
- Limited language support

Bibliography: *See references made in section 4 of the Adaptyst introductory paper on arXiv.*

Related work: design space exploration

- In terms of design space exploration, a variety of methods have been investigated before [1, 2, 3, 4, 5, 6, 7, 8, 9] with a high applicability potential for the project.
- However, the work there has limitations in terms of supported workflows, hardware, and/or optimisations, for example:
  - high-level synthesis only
  - dataflow programs only
  - approximate computing only
  - CPU-FPGA or CPU-GPU systems only
  - hardware architecture design only
  - software-side or software-to-hardware mapping optimisations only

Bibliography:

1. Sepide Saeedi et al. "A Survey on Design Space Exploration Approaches for Approximate Computing Systems". In: Electronics 13.22 (2024), p. 4442.
2. Benjamin Carrion Schafer and Zi Wang. "High-level synthesis design space exploration: Past, present, and future". In: IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems 39.10 (2019), pp. 2628–2639.
3. Meena Belwal and TSB Sudarshan. "A survey on design space exploration for heterogeneous multi-core". In: 2014 International Conference on Embedded Systems (ICES). IEEE. 2014, pp. 80–85.
4. Malgorzata Michalska et al. "High-precision performance estimation for the design space exploration of dynamic dataflow programs". In: IEEE Transactions on Multi-Scale Computing Systems 4.2 (2017), pp. 127–140.
5. Jeronimo Castrillon, Rainer Leupers, and Gerd Ascheid. "MAPS: Mapping concurrent dataflow applications to heterogeneous MPSoCs". In: IEEE Transactions on Industrial Informatics 9.1 (2011), pp. 527–545.
6. Andy D Pimentel, Cagkan Erbas, and Simon Polstra. "A systematic approach to exploring embedded system architectures at multiple abstraction levels". In: IEEE transactions on computers 55.2 (2006), pp. 99–112.
7. Maxime Pelcat et al. "An open framework for rapid prototyping of signal processing applications". In: EURASIP journal on embedded systems 2009.1 (2009), p. 598529.
8. Constantino Gómez et al. "Design space exploration of next-generation HPC machines". In: 2019 IEEE International Parallel and Distributed Processing Symposium (IPDPS). IEEE. 2019, pp. 54–65.
9. Lana Scravaglieri et al. "Compiler, Runtime, and Hardware Parameters Design Space Exploration". In: IPDPS 2025-39th IEEE International Parallel and Distributed Processing Symposium. 2025.