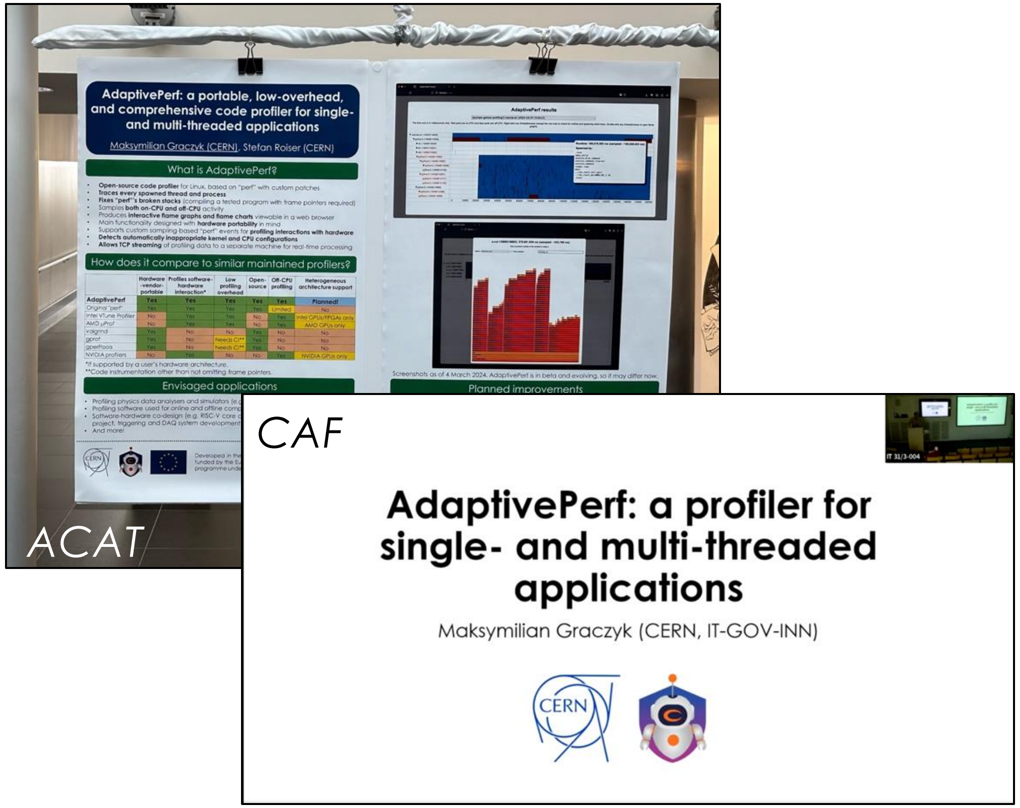
AdaptivePerf: an architecture-agnostic code profiler

Maksymilian Graczyk, Stefan Roiser
(CERN, IT-FTI-PSE)

You may have heard this before…

- This year: ACAT, CERN Compute & Accelerator Forum (CAF), RISC-V Summit Europe, …
- If that’s the case for you, **there have been a few significant updates, so stay tuned for this talk!**
- Updates are coloured in green.



AdaptivePerf: a portable, low-overhead, and comprehensive code profiler for single- and multi-threaded applications

Maksymilian Graczyk (CERN), Stefan Roiser (CERN)

What is AdaptivePerf?

- **Open-source code profiler** for Linux, based on "perf" with custom patches
- **Traces every spawned thread and process**
- **Fixes "perf"'s broken stacks** (compiling a tested program with frame pointers required)
- Samples **both on-CPU and off-CPU** activity
- Produces **interactive flame graphs and flame charts** viewable in a web browser
- Main functionality designed with **hardware portability** in mind
- Supports custom sampling-based "perf" events for **profiling interactions with hardware**
- **Detects automatically inappropriate kernel and CPU configurations**
- **Allows TCP streaming** of profiling data to a separate machine for real-time processing

How does it compare to similar maintained profilers?

	Hardware-vendor-portable	Profiles software-hardware interaction*	Low profiling overhead	Open-source	Off-CPU profiling	Heterogeneous architecture support
AdaptivePerf	**Yes**	**Yes**	**Yes**	**Yes**	**Yes**	**Planned!**
Original "perf"	Yes	Yes	Yes	Yes	Limited	No
Intel VTune Profiler	No	Yes	Yes	No	Yes	Intel GPUs/FPGAs only
AMD μProf	No	Yes	Yes	No	Yes	AMD GPUs only
valgrind	Yes	No	No	Yes	No	No
gprof	Yes	No	Needs CI**	Yes	No	No
gperftools	Yes	No	Needs CI**	Yes	No	No
NVIDIA profilers	No	Yes	No	No	Yes	NVIDIA GPUs only

*If supported by a user's hardware architecture.
**Code instrumentation other than not omitting frame pointers.

Envisaged applications

- Profiling physics data analysers and simulators (e.g
- Profiling software used for online and offline comp
- Software-hardware co-design (e.g. RISC-V core c project, triggering and DAQ system development
- And more!

Developed in the funded by the Eu programme unde

Screenshots as of 4 March 2024. AdaptivePerf is in beta and evolving, so it may differ now.

Planned improvements

AdaptivePerf: a profiler for single- and multi-threaded applications

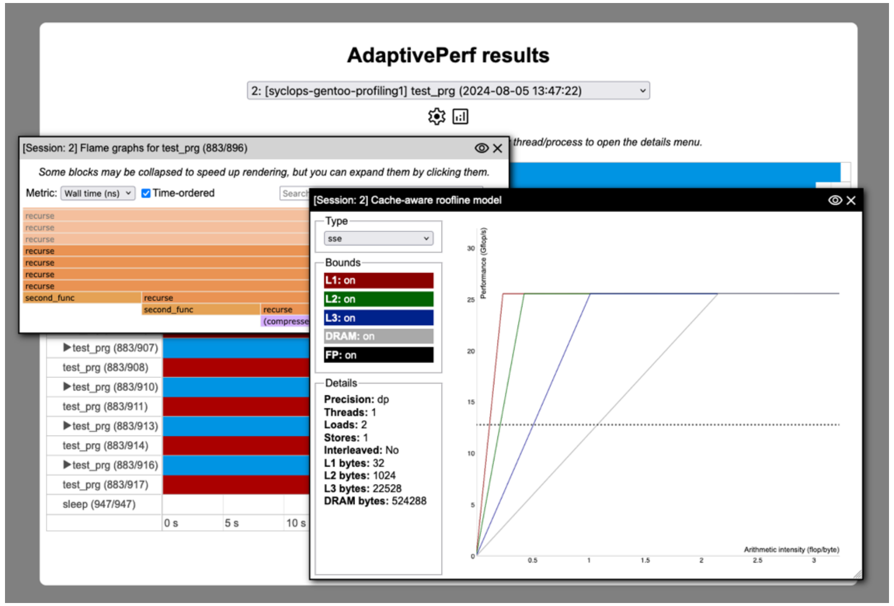
Maksymilian Graczyk (CERN, IT-GOV-INN)

What is AdaptivePerf?

NB: AdaptivePerf is **not** a continuous profiler! It profiles single commands and is not meant for 24/7 monitoring.

The main target audience is SW and HW developers optimising their software and/or hardware, also as part of software-hardware co-design for specific applications.

- **Open-source code profiler** for Linux, based on "perf" with custom patches
- Samples **both on-CPU and off-CPU** activity
- **Traces every spawned thread and process**
- **Minimises risk of broken profiled stacks** for programs compiled with frame pointers by detecting inappropriate kernel and CPU configurations automatically
- Produces data for rendering **interactive flame graphs and charts** in a web browser with a new GUI and cache-aware roofline plots produced by the CARM tool from INESC-ID
- Main functionality designed with **hardware portability** in mind (tested on x86-64, arm64, and RISC-V)
- Supports custom sampling-based "perf" events for **profiling interactions with hardware**
- **Allows TCP streaming** of profiling data to a separate machine for real-time processing



AdaptivePerf results

2: [syclops-gentoo-profiling1] test_prg (2024-08-05 13:47:22)

thread/process to open the details menu.

[Session: 2] Flame graphs for test_prg (883/896)

Some blocks may be collapsed to speed up rendering, but you can expand them by clicking them.

Metric: Wall time (ns) ☑ Time-ordered

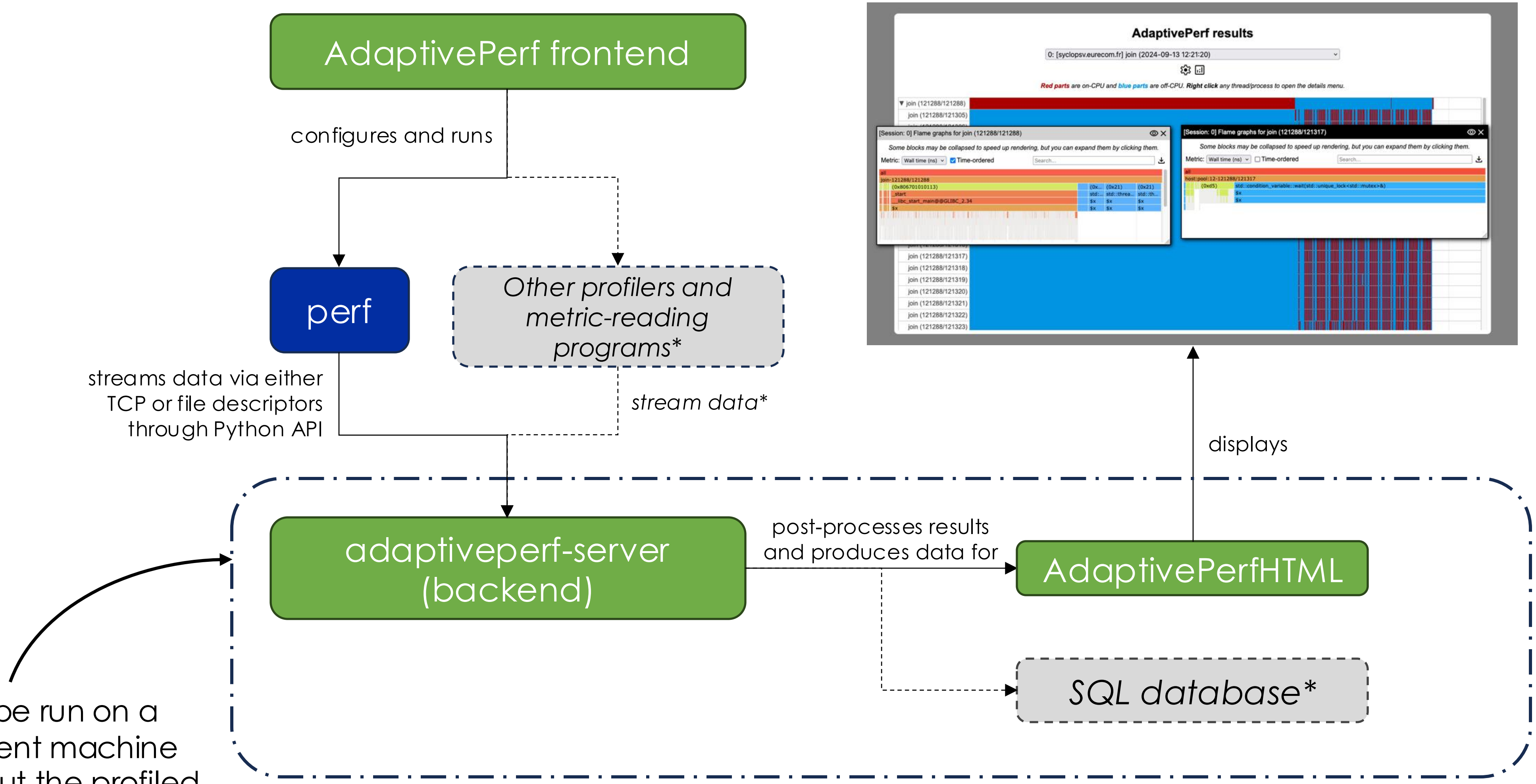
[Session: 2] Cache-aware roofline model

Type: sse

Bounds: L1: on, L2: on, L3: on, DRAM: on, FP: on

Details
Precision: dp
Threads: 1
Loads: 2
Stores: 1
Interleaved: No
L1 bytes: 32
L2 bytes: 1024
L3 bytes: 22528
DRAM bytes: 524288

How does AdaptivePerf work under the hood?



AdaptivePerf frontend
configures and runs
perf
Other profilers and metric-reading programs*
streams data via either TCP or file descriptors through Python API
stream data*
adaptiveperf-server (backend)
post-processes results and produces data for
AdaptivePerfHTML
SQL database*
displays

AdaptivePerf results

0: [syclopsv.eurecom.fr] join (2024-09-13 12:21:20)

Red parts are on-CPU and blue parts are off-CPU. Right click any thread/process to open the details menu.

▼ join (121288/121288)
join (121288/121305)
join (121288/121317)
join (121288/121318)
join (121288/121319)
join (121288/121320)
join (121288/121321)
join (121288/121322)
join (121288/121323)

[Session: 0] Flame graphs for join (121288/121288)

Some blocks may be collapsed to speed up rendering, but you can expand them by clicking them.

Metric: Wall time (ns) ☑ Time-ordered Search...

[Session: 0] Flame graphs for join (121288/121317)

Some blocks may be collapsed to speed up rendering, but you can expand them by clicking them.

Metric: Wall time (ns) □ Time-ordered Search...

all
host:pool:12-121288/121317
(0xd5)
std::condition_variable::wait(std::unique_lock<std::mutex>&)
$x

Can be run on a different machine without the profiled programs and debug info!

*Mostly implemented and therefore coming very soon.

What is SYCLOPS?

- A project funded from the European Union HE research and innovation programme (grant agreement No 101092877) about hardware acceleration with open standards using SYCL and RISC-V: https://www.syclops.org
- CERN project tasks:
  1. Implementing SYCL support in ROOT and cling + demonstrating it on a Lorentz vector calculation example.
  2. Benchmarking and profiling + integration testing of all use cases envisaged in SYCLOPS (ROOT, genomics analysis, and autonomous systems).
- AdaptivePerf is part of task 2, but its applications and development plans extend beyond SYCLOPS!

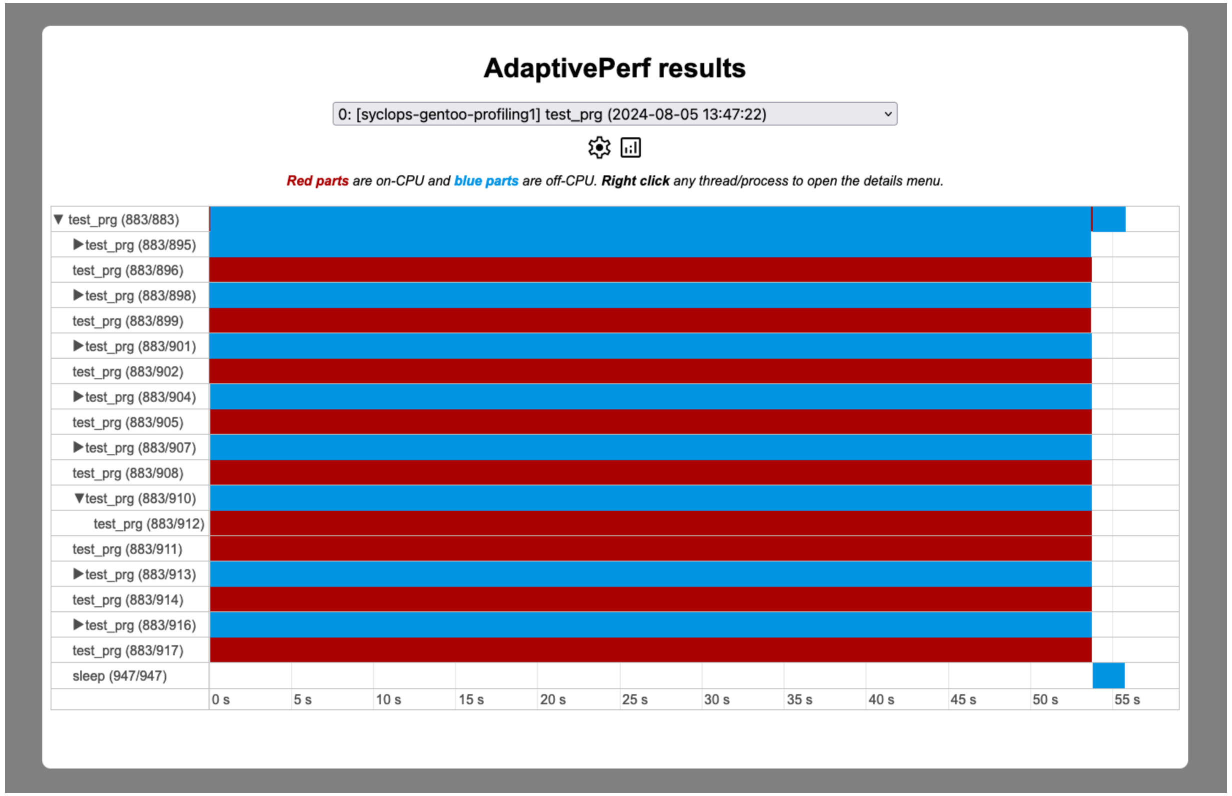
How to download AdaptivePerf?

- It's open-source and you can get it for free from our GitHub: https://github.com/AdaptivePerf.
- AdaptivePerf is available **as an early development version**, in form of a source code, a Gentoo-based VM image, and Docker + Apptainer images with frame pointers. DEB + RPM, a CVMFS setup, and non-Linux server binaries will follow soon.
- There are 2 parts:
  - AdaptivePerf: the main program which is the command-line profiling tool (frontend) and server (backend), licensed under GNU GPL v2 **only**. It also includes the patched "perf".
  - AdaptivePerfHTML: the web server for displaying profiling results as an interactive website, licensed under GNU GPL v3.

Quick start with AdaptivePerf

- Run `adaptiveperf -- <command to be profiled>` and wait until it finishes and produces the "results" directory. The command doesn't need to be quoted and under certain circumstances, root rights are not required. Partial profiling will also be possible very soon!
- Set the `FLASK_PROFILING_STORAGE` environment variable to the "results" path.
- Run Flask (a Python web framework) and point it to AdaptivePerfHTML: `adaptiveperf.app:app`.
- Open the website in your web browser. Done!

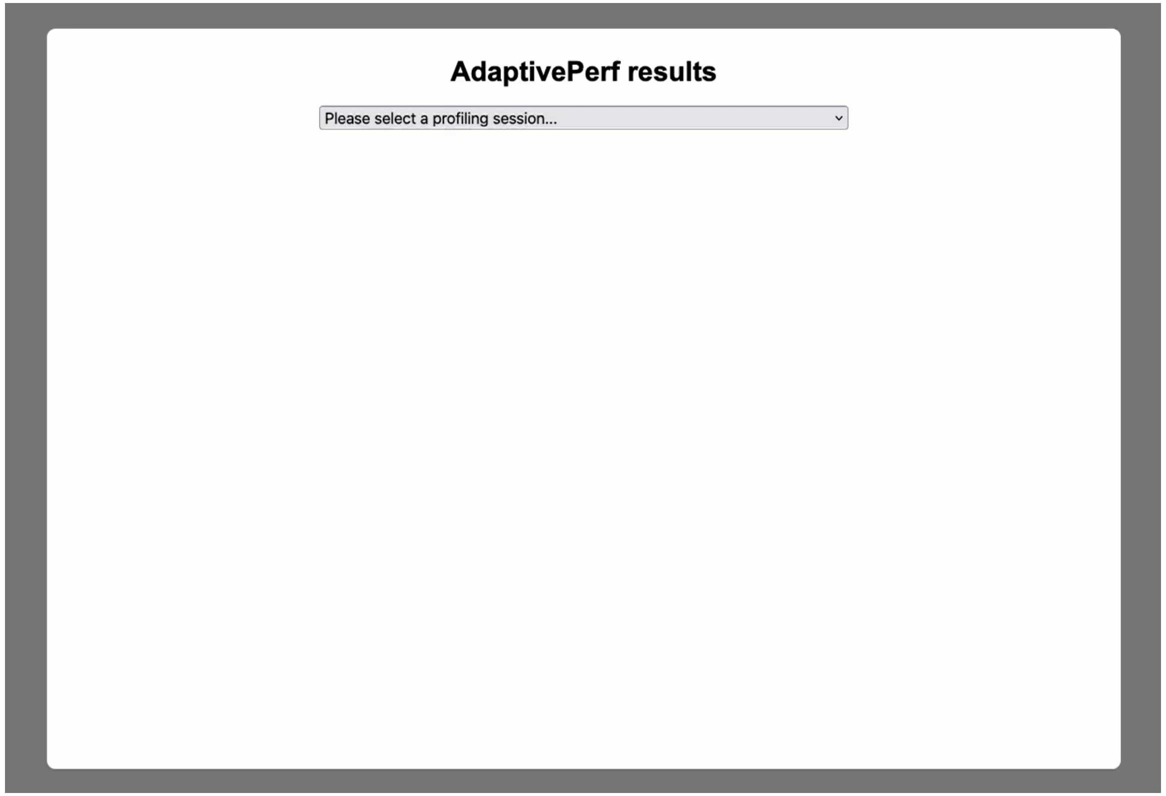
Demo



AdaptivePerf results

0: [syclops-gentoo-profiling1] test_prg (2024-08-05 13:47:22)

Red parts are on-CPU and *blue parts* are off-CPU. **Right click** any thread/process to open the details menu.



AdaptivePerf results

Please select a profiling session...

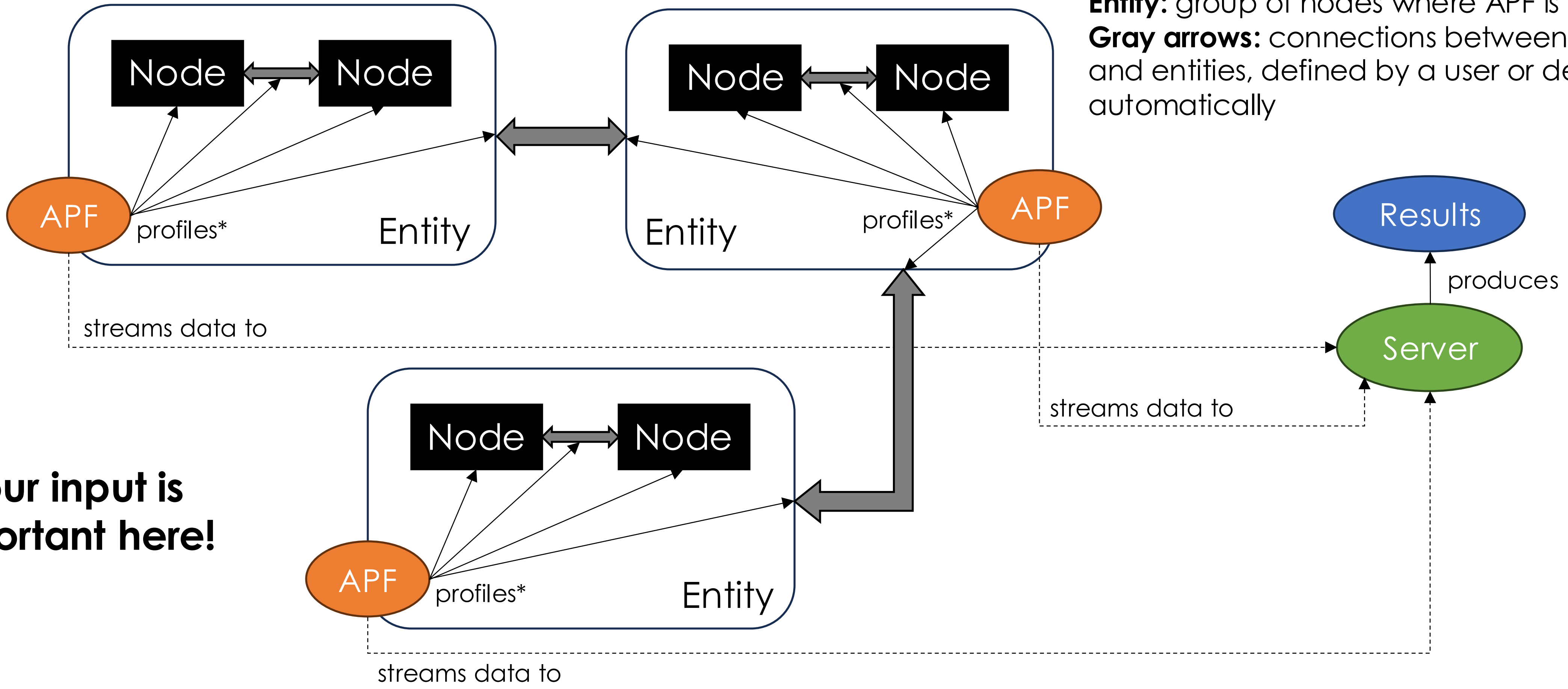
Work in progress: profiling heterogeneous and custom architectures

- In the context of the changing technological landscape, we want AdaptivePerf to help push forward computing at CERN and other physics experiments and therefore allow more scientific discoveries.
- Given the variety of platforms (CPUs with different ISAs, GPUs, FPGAs, ASICs etc.), the goal of AdaptivePerf is **delivering a single profiling-based platform for comparing and customising software and hardware architectures scaling from embedded/edge to exascale.**
- This is how we envisage profiling heterogeneous and custom computer systems in AdaptivePerf:

Work in progress: profiling heterogeneous and custom architectures

APF: AdaptivePerf frontend instance
Node: arbitrary hardware unit defined by a user or detected automatically
Entity: group of nodes where APF is deployed
Gray arrows: connections between nodes and entities, defined by a user or detected automatically

*using built-in methods or methods defined by a user



Your input is important here!

Future plans

- Looking into profiling memory usage by threads/processes
- Adding profiling on a lower level and with more debug info, e.g. showing line numbers, going down to LLVM IR / MLIR / assembly etc.
- Downloading debug symbols automatically if not present
- Removing or weakening the frame pointer compilation requirement

Thank you!

Any questions or comments?

Feel free to reach out to me at the conference or by e-mail:
maksymilian.graczyk@cern.ch

Extra slides

How does AdaptivePerf compare to other similar and maintained profilers?

	Hardware-vendor-portable	Runs on RISC-V	Profiles software-hardware interaction*	Low profiling overhead	Open-source	Off-CPU profiling	Heterogeneous architecture support
AdaptivePerf	**Yes**	**Yes**	**Yes**	**Yes**	**Yes**	**Yes**	**Work in progress!**
Original "perf"	Yes	Yes	Yes	Yes	Yes	Limited	No
Intel VTune Profiler	No	No	Yes	Yes	No	Yes	Intel GPUs/FPGAs only
AMD µProf	No	No	Yes	Yes	No	Yes	AMD GPUs only
valgrind	Yes	Yes, as a fork	No	No	Yes	No	No
gprof	Yes	Yes	No	Needs CI**	Yes	No	No
gperftools	Yes	Yes	No	Needs CI**	Yes	No	No
NVIDIA profilers	No	No	Yes	No	No	Yes	NVIDIA GPUs only

*If supported by a user's hardware architecture.

**Code instrumentation other than not omitting frame pointers.

How to tackle the frame pointer compilation requirement?

- For example, by DWARF processing whenever frame pointers cannot be used, see: https://www.polarsignals.com/blog/posts/2022/11/29/profiling-without-frame-pointers (this is more compact than what “perf” currently does).
- Full removal may be unnecessary, see: https://brendangregg.com/blog/2024-03-17/the-return-of-the-frame-pointers.html.