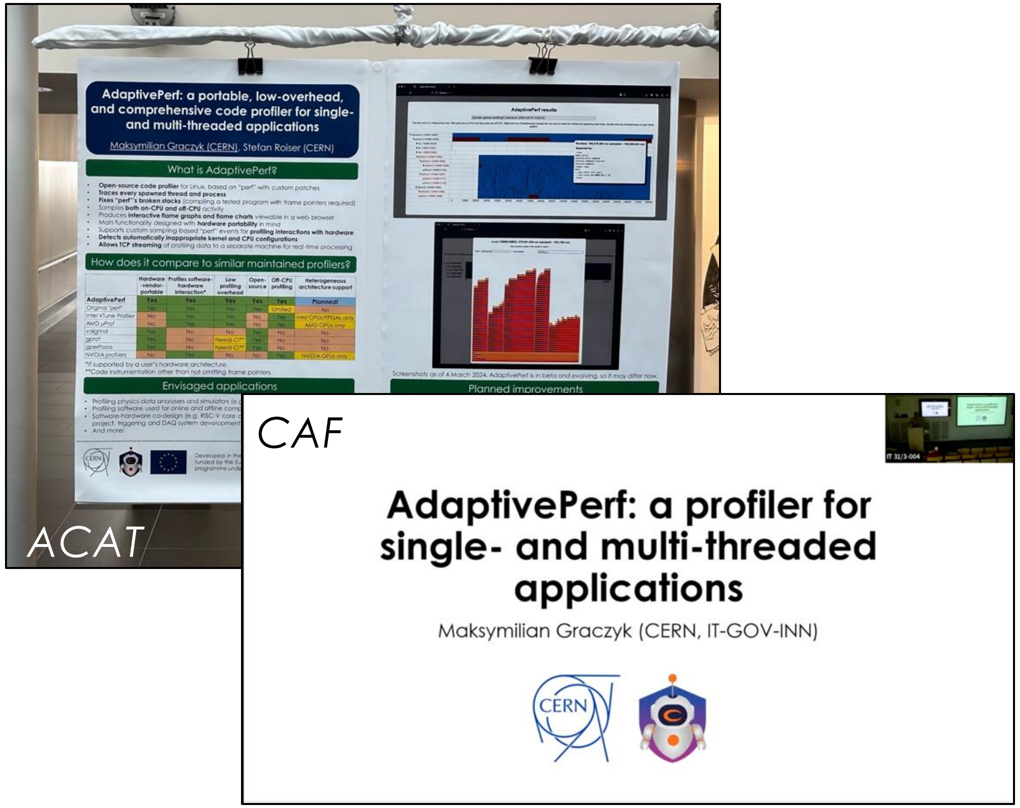
AdaptivePerf: an architecture-agnostic code profiler

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

CHEP'24, Kraków, 22 October 2024

You may have heard this before…

- This year: ACAT, CERN Compute & Accelerator Forum (CAF), RISC-V Summit Europe, …
- **There have been exciting developments, so please keep listening!**
- 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

AdaptivePerf results

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

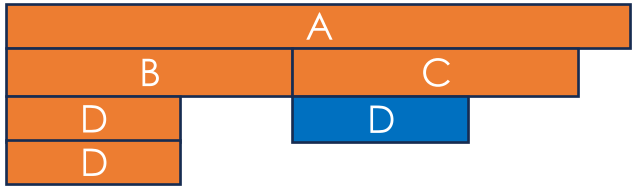
Maksymilian Graczyk (CERN, IT-GOV-INN)

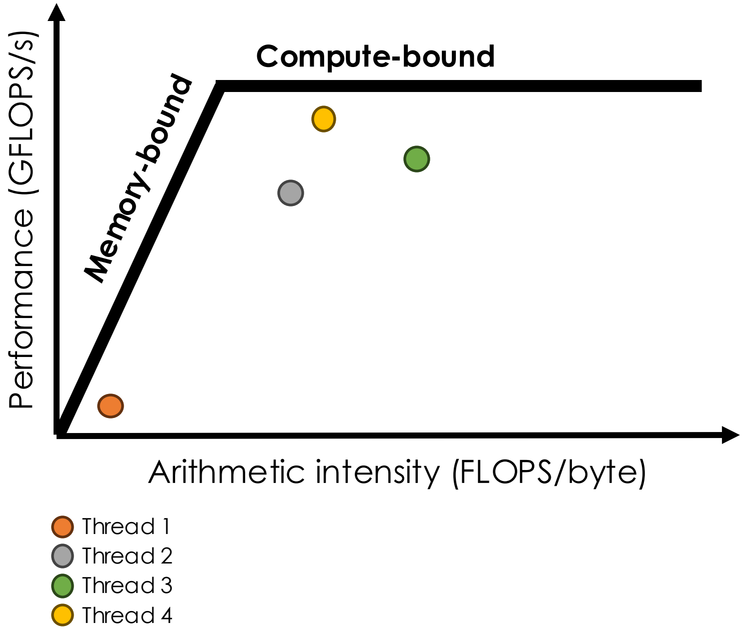
Intro: SYCLOPS

- 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 SYCLOPS use cases (ROOT, genomics analysis, and autonomous systems).
- AdaptivePerf is part of task 2, but its applications and development plans **extend beyond SYCLOPS**.

Intro: Flame graphs and roofline modelling

- **Flame graphs:** plots showing hierarchical data as merged stacked blocks, with their width being proportional to the corresponding data frequency. The x-axis can be either non-time-ordered or time-ordered.
- **Roofline modelling:** a benchmarking and profiling method which uses machine-specific performance bounds, program arithmetic intensity, and program floating-point operations per second to determine whether a code is memory- or compute-bound.



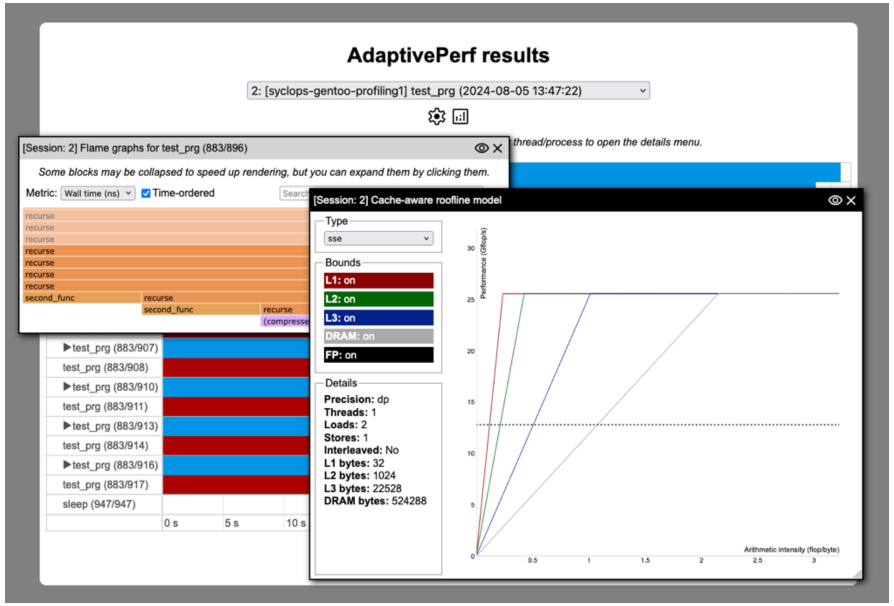


What is AdaptivePerf?

NB: AdaptivePerf is **not** a continuous profiler! It profiles single workloads 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 and language-agnostic code profiler** for Linux, originally built on top of "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

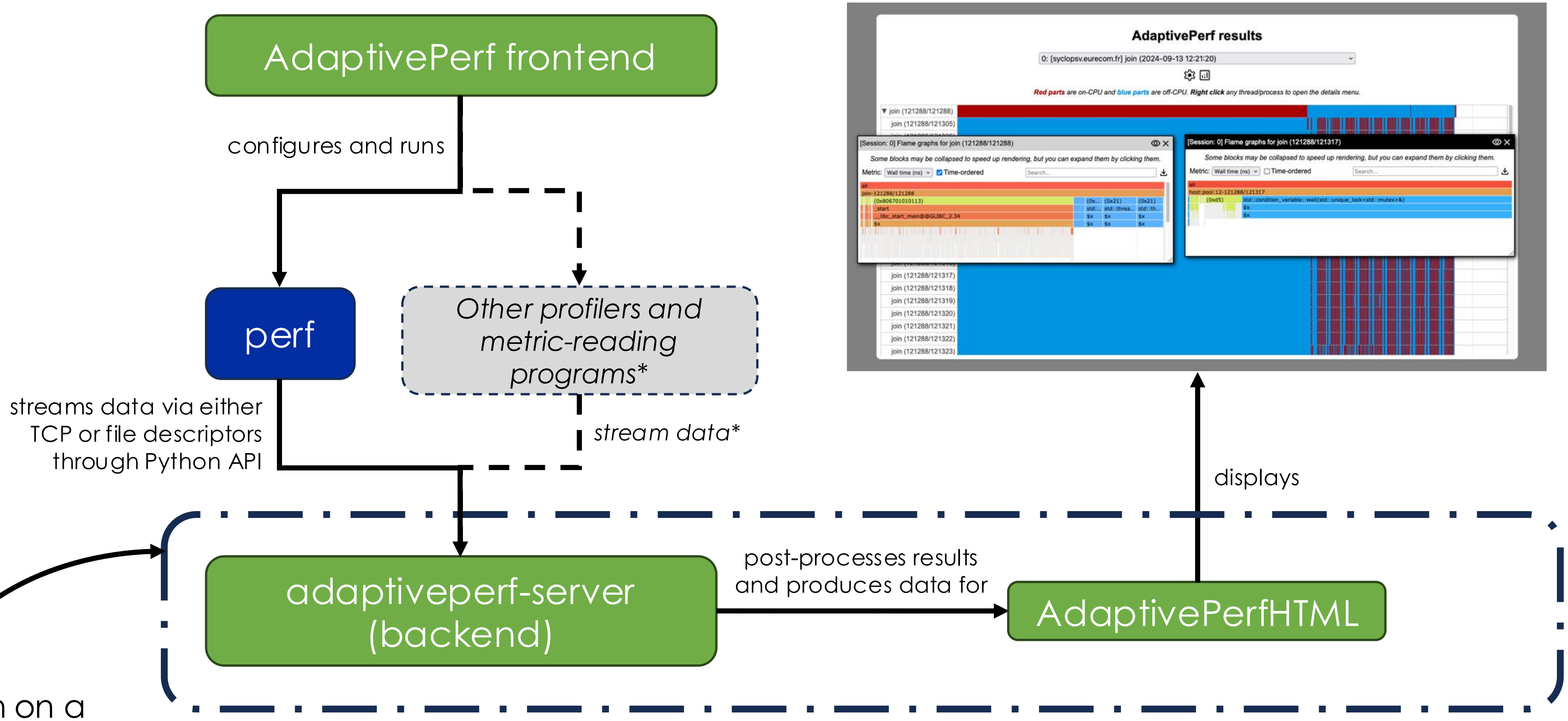
Type: sse

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
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.

[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

[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

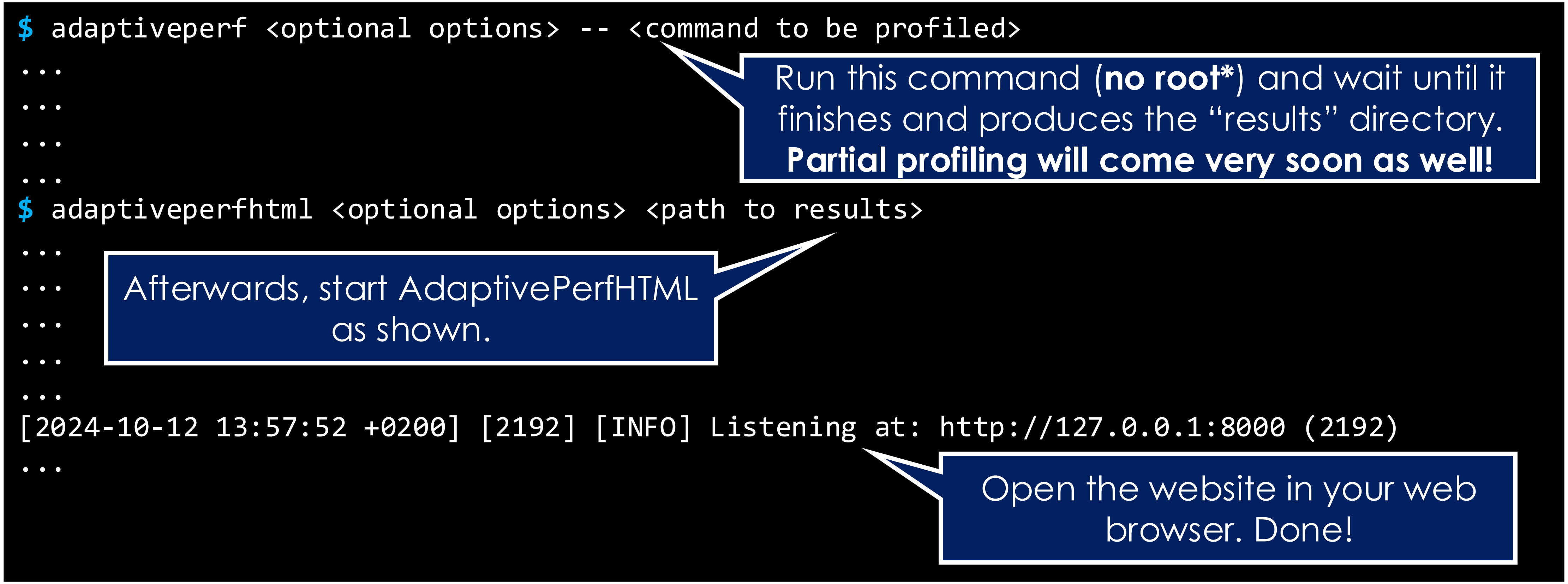
Can be run on a different machine without the profiled programs and debug info!

*Mostly implemented and therefore coming very soon.

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 (FPs).
- DEB + RPM, a CVMFS setup, and pre-built non-Linux server binaries will follow soon.
- There are 2 parts:
  - AdaptivePerf: the main program (frontend + 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



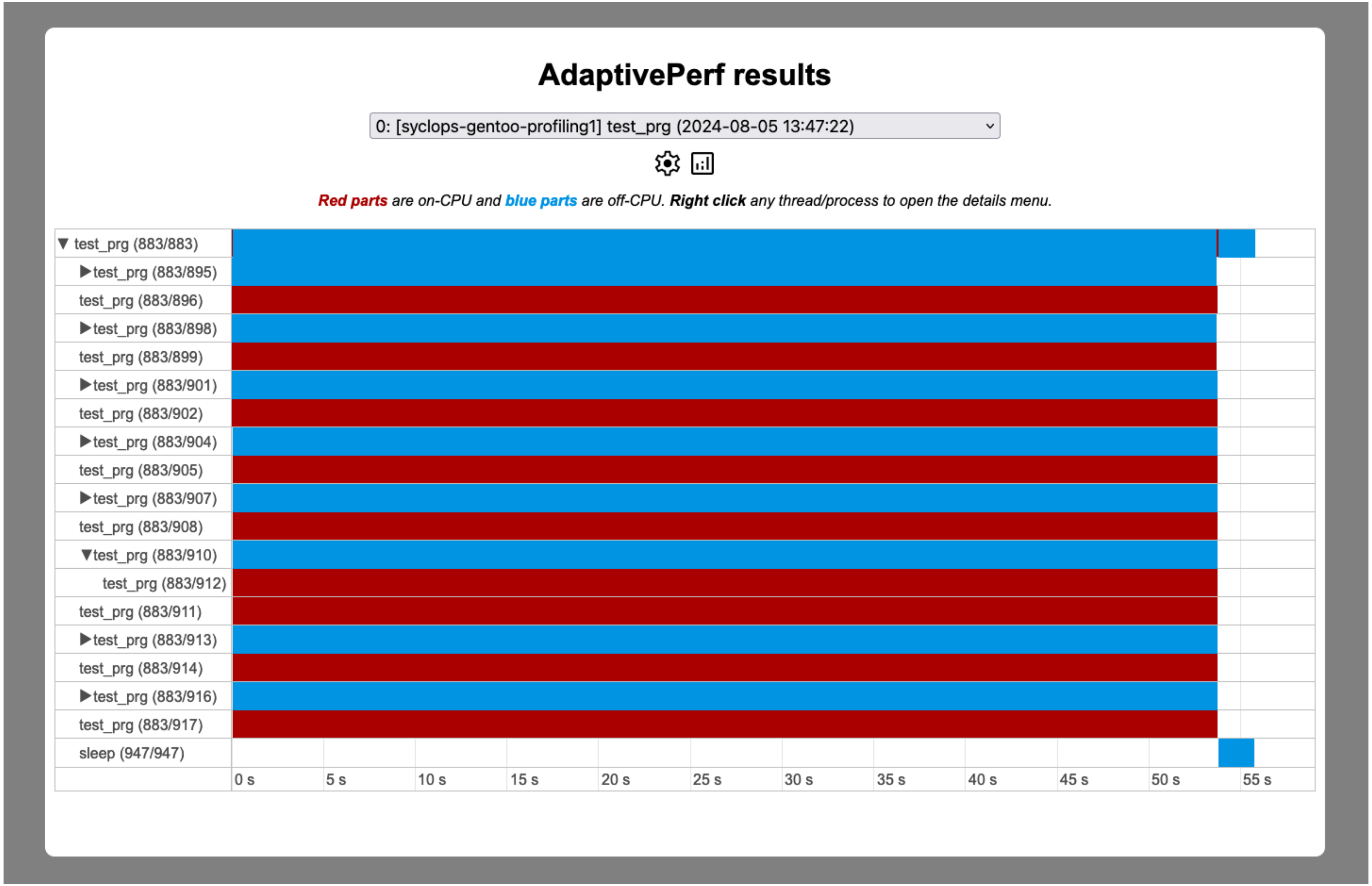
```
$ adaptiveperf <optional options> -- <command to be profiled>
...
...
...
...
$ adaptiveperfhtml <optional options> <path to results>
...
...
...
...
...
[2024-10-12 13:57:52 +0200] [2192] [INFO] Listening at: http://127.0.0.1:8000 (2192)
...
```

Run this command (**no root***) and wait until it finishes and produces the "results" directory. **Partial profiling will come very soon as well!**

Afterwards, start AdaptivePerfHTML as shown.

Open the website in your web browser. Done!

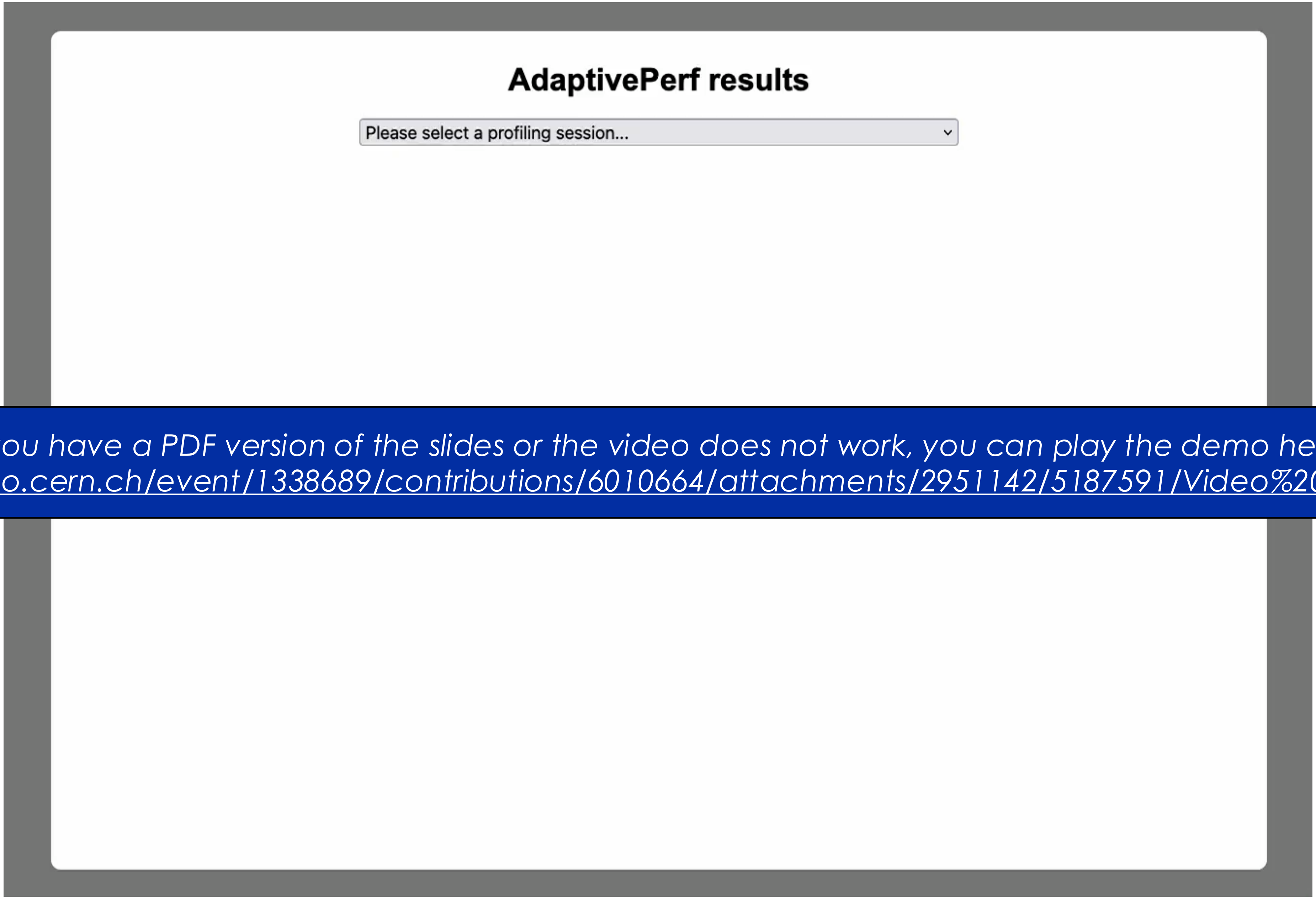
*Under certain circumstances.



AdaptivePerf results

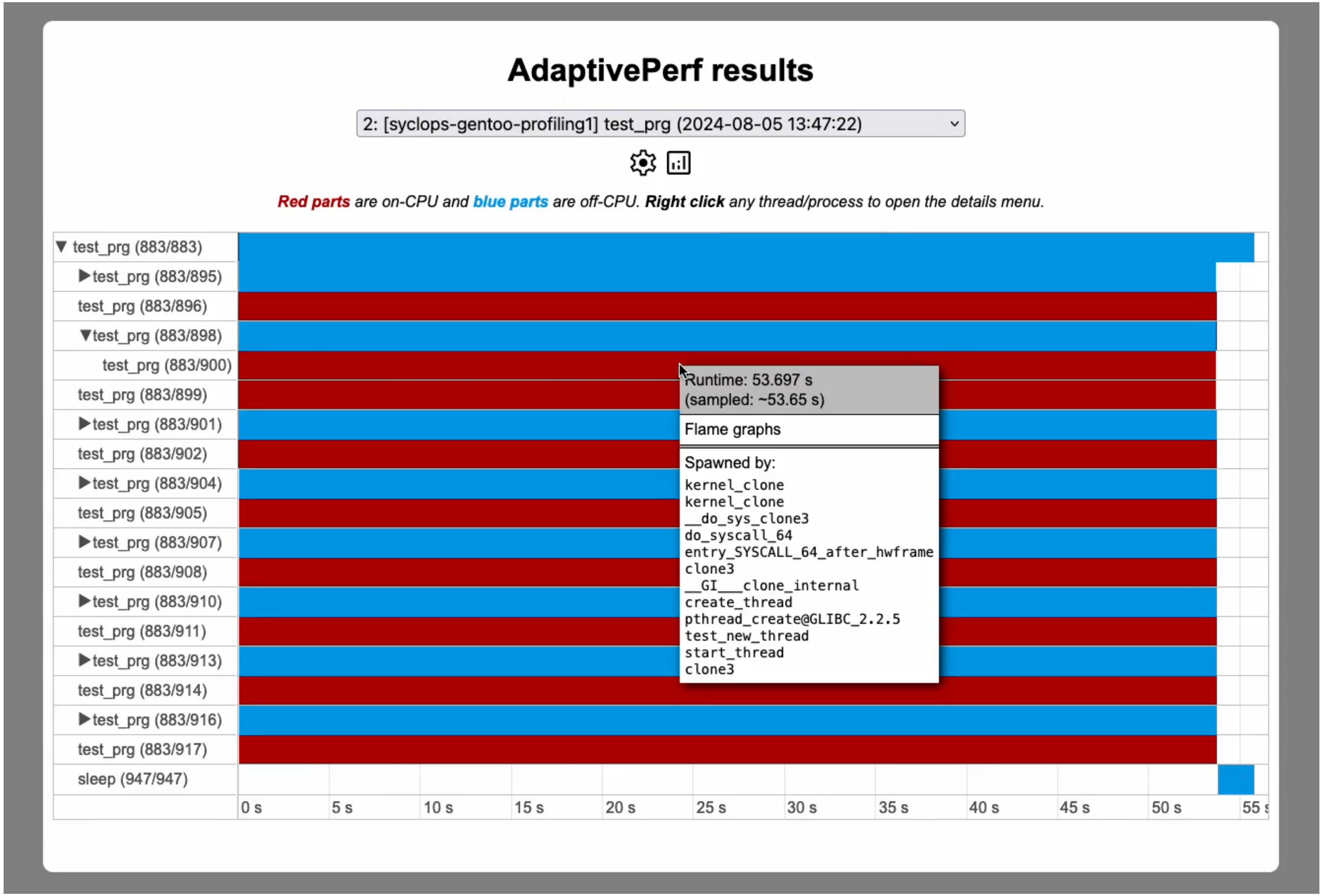
0: [syclops-gentoo-profiling1] test_prg (2024-08-05 13:47:22)

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



AdaptivePerf results

If you have a PDF version of the slides or the video does not work, you can play the demo here:
https://indico.cern.ch/event/1338689/contributions/6010664/attachments/2951142/5187591/Video%20demo.mp4



AdaptivePerf results

2: [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.*

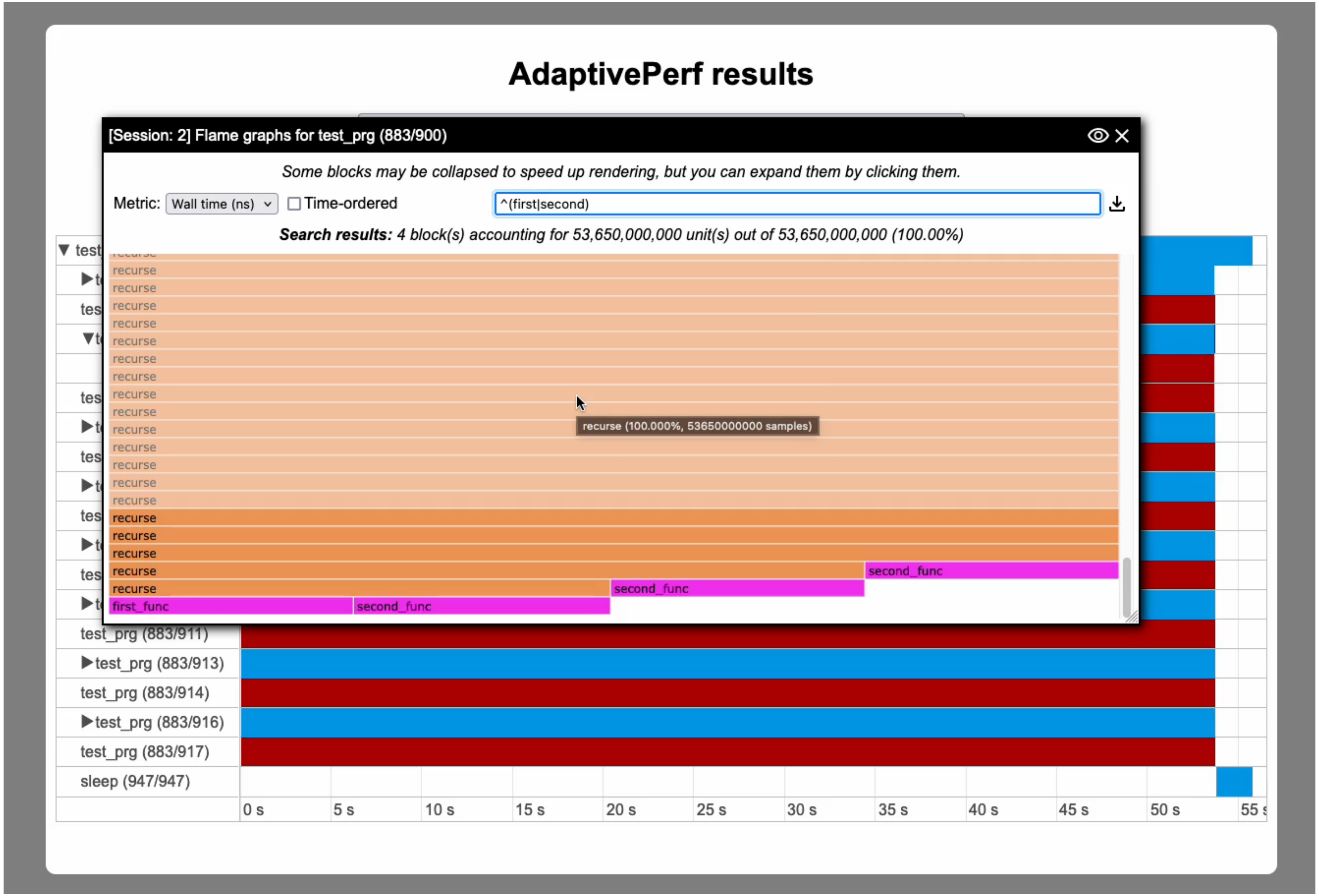
Runtime: 53.697 s
(sampled: ~53.65 s)

Flame graphs

Spawned by:

```
kernel_clone
kernel_clone
__do_sys_clone3
do_syscall_64
entry_SYSCALL_64_after_hwframe
clone3
__GI___clone_internal
create_thread
pthread_create@GLIBC_2.2.5
test_new_thread
start_thread
clone3
```

AdaptivePerf results



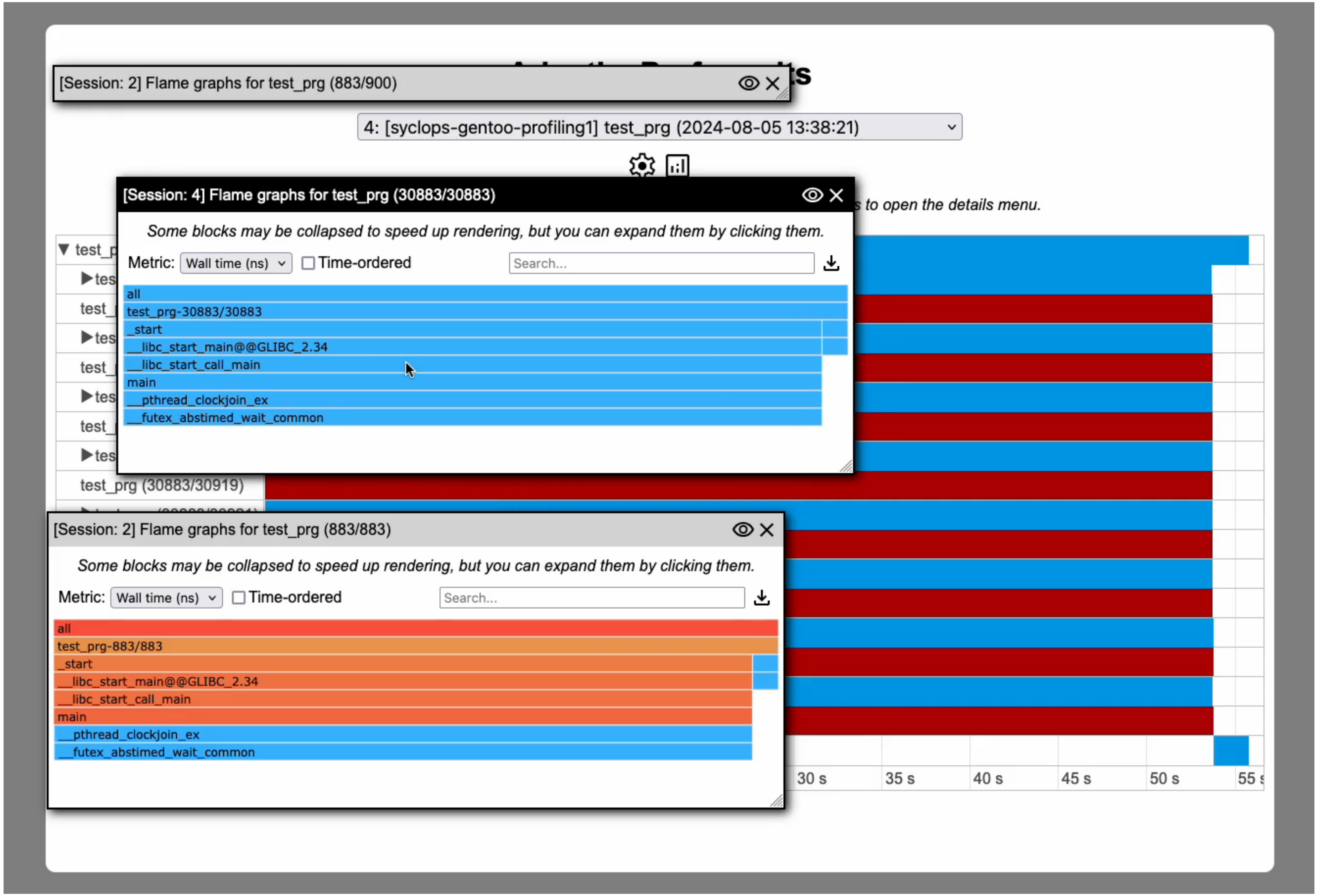
[Session: 2] Flame graphs for test_prg (883/900)

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

Metric: Wall time (ns) □ Time-ordered ^(first|second)

***Search results:** 4 block(s) accounting for 53,650,000,000 unit(s) out of 53,650,000,000 (100.00%)*

recurse (100.000%, 53650000000 samples)



[Session: 2] Flame graphs for test_prg (883/900)

4: [syclops-gentoo-profiling1] test_prg (2024-08-05 13:38:21)

[Session: 4] Flame graphs for test_prg (30883/30883)

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

Metric: Wall time (ns) □ Time-ordered

```
all
test_prg-30883/30883
_start
__libc_start_main@@GLIBC_2.34
__libc_start_call_main
main
__pthread_clockjoin_ex
__futex_abstimed_wait_common
```

to open the details menu.

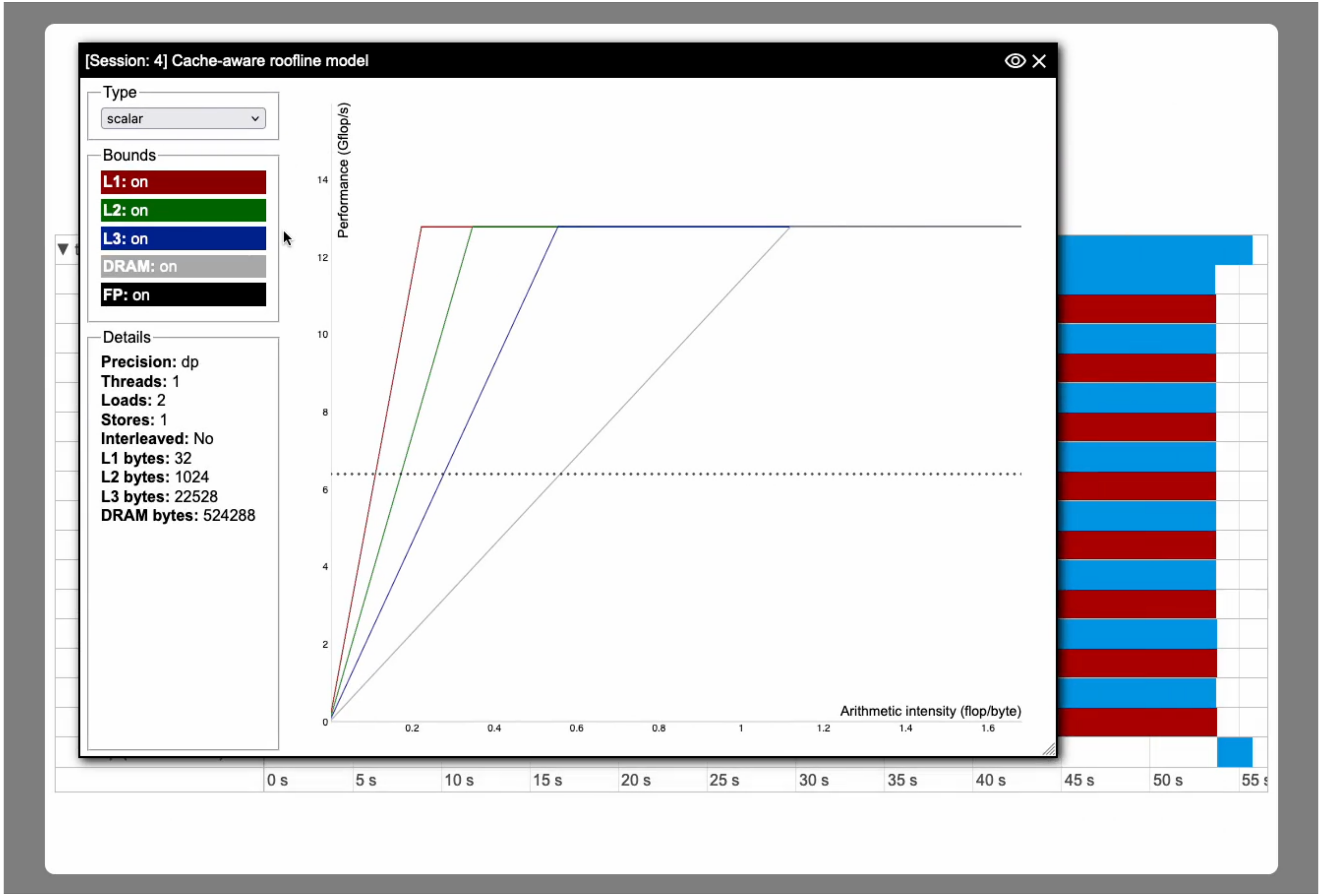
test_prg (30883/30919)

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

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

Metric: Wall time (ns) □ Time-ordered

```
all
test_prg-883/883
_start
__libc_start_main@@GLIBC_2.34
__libc_start_call_main
main
__pthread_clockjoin_ex
__futex_abstimed_wait_common
```



[Session: 4] Cache-aware roofline model

Type: scalar

Bounds

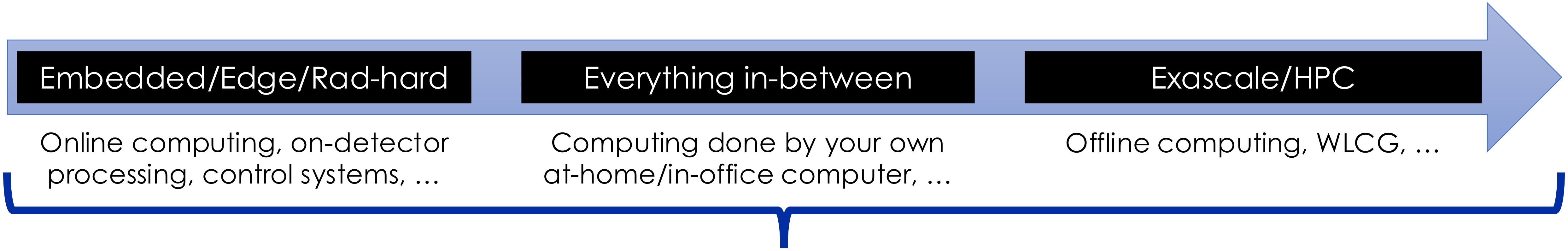
- L1: on
- L2: on
- L3: on
- DRAM: on
- FP: on

Details

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

Work in progress: profiling heterogeneous and custom architectures

- In the context of the changing tech landscape, we want AdaptivePerf to help advance computing at CERN and other physics experiments.
- Given the variety of platforms (CPUs with ISAs, GPUs, FPGAs, ASICs etc.), the goal of AdaptivePerf is **delivering a single and unified profiling-based platform for comparing and customising off-the-shelf and in-house software and hardware architectures across the entire spectrum:**



Embedded/Edge/Rad-hard

Online computing, on-detector processing, control systems, …

Everything in-between

Computing done by your own at-home/in-office computer, …

Exascale/HPC

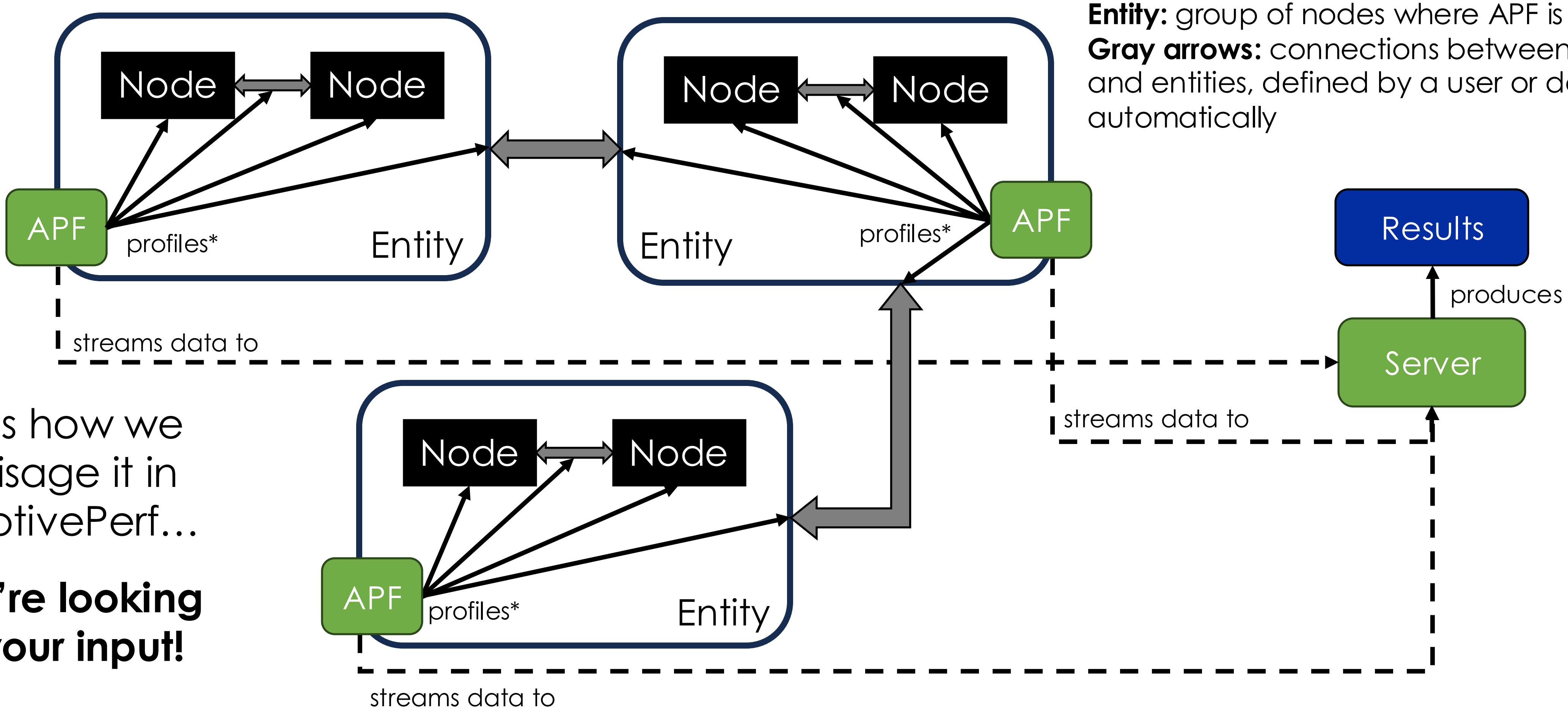
Offline computing, WLCG, …

To be covered by 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



This is how we envisage it in AdaptivePerf…

…we're looking for your input!

Future plans

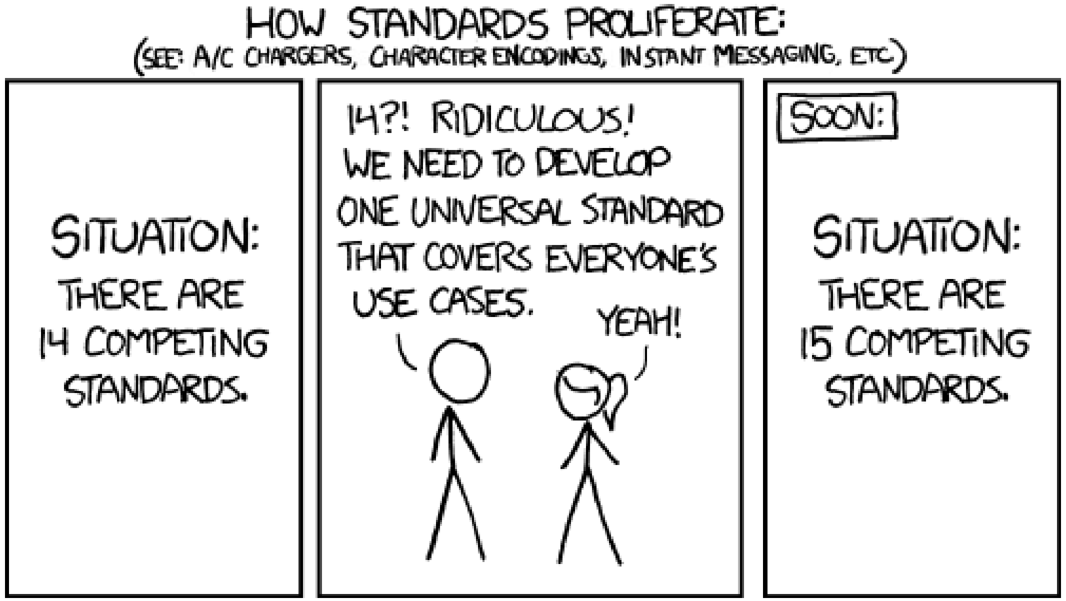
- Exporting profiling results to a well-documented SQL database
- Looking into profiling memory usage by threads/processes
- Adding profiling on a lower level and with more debug info, e.g. showing line numbers, going down to LLVM IR / MLIR / assembly etc.
- Downloading debug symbols automatically if 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

Sounds familiar?


HOW STANDARDS PROLIFERATE:
(SEE: A/C CHARGERS, CHARACTER ENCODINGS, INSTANT MESSAGING, ETC)
SITUATION: THERE ARE 14 COMPETING STANDARDS.
14?! RIDICULOUS! WE NEED TO DEVELOP ONE UNIVERSAL STANDARD THAT COVERS EVERYONE'S USE CASES.
YEAH!
SOON:
SITUATION: THERE ARE 15 COMPETING STANDARDS.

From https://imgs.xkcd.com/comics/standards.png

AdaptivePerf will **NOT** be the (n + 1)-th competing profiler (to some extent). It will be a unified platform where existing profilers can be integrated easily and any remaining gaps can be filled in by AdaptivePerf itself or a user.

How does AdaptivePerf compare to other similar and maintained profilers?

	Hardware-vendor-portable	Runs on RISC-V	Profiles software-hardware interaction[1]	Low profiling overhead	Open-source	Off-CPU profiling	Flexible support of heterogeneous and custom architectures	Flexible support of multi-node systems
AdaptivePerf	**Yes**	**Yes**	**Yes**	**Yes**	**Yes**	**Yes**	**WIP[2]**	**WIP[2]**
Original "perf"	Yes	Yes	Yes	Yes	Yes	Limited	No	No
Intel VTune Profiler	No	No	Yes	Yes	No	Yes	Intel only, NF[3]	MPI only, NF[3]
AMD µProf	No	No	Yes	Yes	No	Yes	AMD GPUs only, NF[3]	MPI only, NF[3]
valgrind	Yes	Yes, as a fork	No	No	Yes	No	No	No
gprof	Yes	Yes	No	Needs CI[4]	Yes	No	No	No
gperftools	Yes	Yes	No	Needs CI[4]	Yes	No	No	No
NVIDIA profilers	No	No	Yes	No	No	Yes	NVIDIA GPUs only, NF[3]	NF[3]
TAU	Yes	No	Yes	Yes	Yes	Yes	GPUs only, NF[3]	MPI only, NF[3]
HPCToolkit	Yes	No	Yes	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.

[4] 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.