# Advanced Linux Power Management Evaluation using Perf

Hagen Paul Pfeifer <hagen@jauu.net>

# Perf Power Analyzer

**perf power-analyzer == perf script, bundled with perf, developed in kernel tree**

**Debian: apt-get install linux-perf**

**Widely available, even on embedded systems like Yocto or Buildroot. No complex installation, which is often difficult on embedded systems**

Not yet committed upstream, still needs some last polish. URL for testing at the last slide

```
$ perf script record power-analyzer -a -- sleep 60
[ perf record: Woken up 77 times to write data ]
[ perf record: Captured and wrote 23,165 MB perf.data (246395 samples) ]

$ perf script report power-analyzer --help
usage: power-analyzer.py [-h] [-m
[{idle-cluster,task,timer,frequency,wakeups-timemap,idle-governor,all}]] [-C CPU] [--extended]
[-v] [--file-out]
```

**Info: record power-analyer will capture all required events for all modes!**

# Supported Modes

**Functionality are grouped into "modes"**

- Modules can be used as required, focused and with minimal overhead.
- Different modes require different tracepoints. It is often more efficient to manually record the events that are actually required for the specific mode, via `perf record -e <event>`

**Available Modes (subset):**

### Timer

User & kernel space analysis of timers with impact on wakeups as well as timer tick analysis for tickless systems

### Frequency

Creates a chronological overview of the P-States across all cores. Also maps this data to the executed tasks.

### Wakeup-Timemap

Simple chronological overview of wakeups per time unit for tasks, applied at task or CPU level

### Idle-Governor

Provides characteristics of the selected idle governor, for example a residency analysis. Enables the governors to be benchmarked against each other

# Power Analyzer Modes & Post-Processing

## Use Case: Timer Module

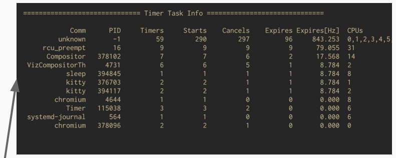
### 1. Recording

```
perf script record
power-analyzer
or
perf record -e 'timer:*'
```

Required events are documented

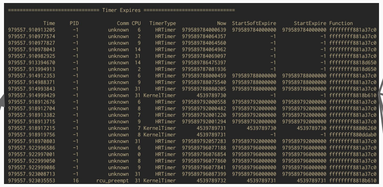
### 2. Analyzation

```
perf script report
power-analyzer -m timer
```



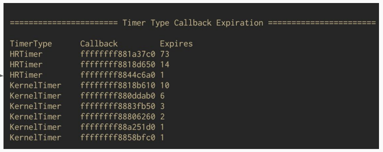
```
============================== Timer Task Info ==============================

           Comm       PID     Timers     Starts    Cancels    Expires Expires[Hz]  CPUs
        unknown        -1         59        290        297         96     843.253  0,1,2,3,4,5
    rcu_preempt        16          9          9          9          9      79.055  31
     Compositor    378102          7          7          6          2      17.568  14
VizCompositorTh      4731          6          6          5          1       8.784  2
          sleep    394845          1          1          1          1       8.784  8
          kitty    376703          2          2          1          1       8.784  1
          kitty    394117          2          2          1          1       8.784  2
       chromium      4644          1          1          0          0       0.000  8
          Timer    115038          3          3          2          0       0.000  6
 systemd-journal       564          1          1          0          0       0.000  6
       chromium    378096          2          2          1          0       0.000  0
```



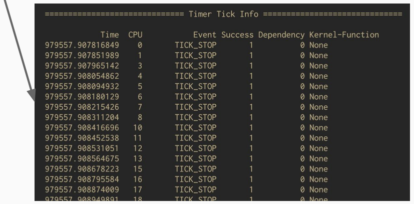
```
============================ Timer Expires ============================

           Time      PID          Comm CPU   TimerType            Now  StartSoftExpire      StartExpire Function
979557.910913205      -1       unknown   6     HRTimer 979589784000639 979589784000000 979589784000000 ffffffff881a37c0
979557.910977574      -1       unknown   2     HRTimer 979589784064357              -1               -1 ffffffff881a37c0
979557.910977827      -1       unknown   9     HRTimer 979589784064560              -1               -1 ffffffff881a37c0
979557.910978043      -1       unknown  14     HRTimer 979589784064962              -1               -1 ffffffff881a37c0
979557.910982925      -1       unknown  31     HRTimer 979589784069097              -1               -1 ffffffff881a37c0
979557.913394670      -1       unknown  14     HRTimer 979589786475397              -1               -1 ffffffff8818d650
979557.913994913      -1       unknown   2     HRTimer 979589787081936              -1               -1 ffffffff8818d650
979557.914912353      -1       unknown   6     HRTimer 979589788000459 979589788000000 979589788000000 ffffffff881a37c0
979557.914988371      -1       unknown   9     HRTimer 979589788075540 979589788000000 979589788000000 ffffffff881a37c0
979557.914993843      -1       unknown  31     HRTimer 979589788080205 979589788000000 979589788000000 ffffffff881a37c0
979557.914999429      -1       unknown  31 KernelTimer      4539789730              -1               -1 ffffffff8818b610
979557.918912676      -1       unknown   6     HRTimer 979589792000558 979589792000000 979589792000000 ffffffff881a37c0
979557.918912704      -1       unknown   8     HRTimer 979589792000482 979589792000000 979589792000000 ffffffff881a37c0
979557.918913382      -1       unknown   7     HRTimer 979589792001220 979589792000000 979589792000000 ffffffff881a37c0
979557.918913715      -1       unknown   9     HRTimer 979589792001294 979589792000000 979589792000000 ffffffff881a37c0
979557.918917215      -1       unknown   7 KernelTimer      4539789731       4539789730       4539789730 ffffffff88806260
979557.918919756      -1       unknown   8 KernelTimer      4539789731              -1               -1 ffffffff880ddab0
979557.918970803      -1       unknown  31     HRTimer 979589792057283 979589792000000 979589792000000 ffffffff881a37c0
979557.922996586      -1       unknown   6     HRTimer 979589796077188 979589796000000 979589796000000 ffffffff881a37c0
979557.922997001      -1       unknown   0     HRTimer 979589796076854 979589796000000 979589796000000 ffffffff881a37c0
979557.922999050      -1       unknown   8     HRTimer 979589796077860 979589796000000 979589796000000 ffffffff881a37c0
979557.922999086      -1       unknown   9     HRTimer 979589796077841 979589796000000 979589796000000 ffffffff881a37c0
979557.923008713      -1       unknown  31     HRTimer 979589796087399 979589796000000 979589796000000 ffffffff881a37c0
979557.923035553      16   rcu_preempt  31 KernelTimer      4539789732       4539789731       4539789731 ffffffff8818b610
```



```
====================== Timer Type Callback Expiration ======================

TimerType      Callback         Expires
HRTimer        ffffffff881a37c0 73
HRTimer        ffffffff8818d650 14
HRTimer        ffffffff8844c6a0 1
KernelTimer    ffffffff8818b610 10
KernelTimer    ffffffff880ddab0 6
KernelTimer    ffffffff8883fb50 3
KernelTimer    ffffffff88806260 2
KernelTimer    ffffffff88a251d0 1
KernelTimer    ffffffff8858bfc0 1
```



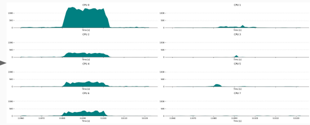
```
============================== Timer Tick Info ==============================

           Time  CPU          Event Success Dependency Kernel-Function
979557.907816849    0      TICK_STOP       1          0 None
979557.907851989    1      TICK_STOP       1          0 None
979557.907965142    3      TICK_STOP       1          0 None
979557.908054862    4      TICK_STOP       1          0 None
979557.908094932    5      TICK_STOP       1          0 None
979557.908180129    6      TICK_STOP       1          0 None
979557.908215426    7      TICK_STOP       1          0 None
979557.908311204    8      TICK_STOP       1          0 None
979557.908416696   10      TICK_STOP       1          0 None
979557.908452538   11      TICK_STOP       1          0 None
979557.908531051   12      TICK_STOP       1          0 None
979557.908564675   13      TICK_STOP       1          0 None
979557.908678223   15      TICK_STOP       1          0 None
979557.908795584   16      TICK_STOP       1          0 None
979557.908874009   17      TICK_STOP       1          0 None
979557.908949891   18      TICK_STOP       1          0 None
```

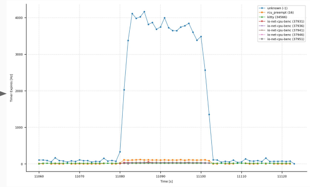
Command line output is clear and easy to read - even in target. It provides a quick and detailed overview at the same time.

### 3. Post-Processing

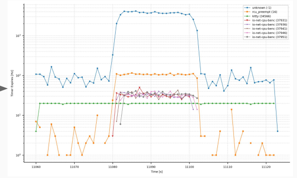
`timer-expires-cpu.py`



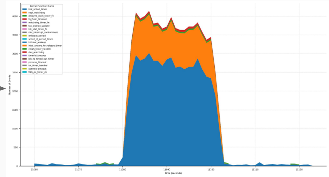
`timer-expires.py`



`timer-expires-log.py`

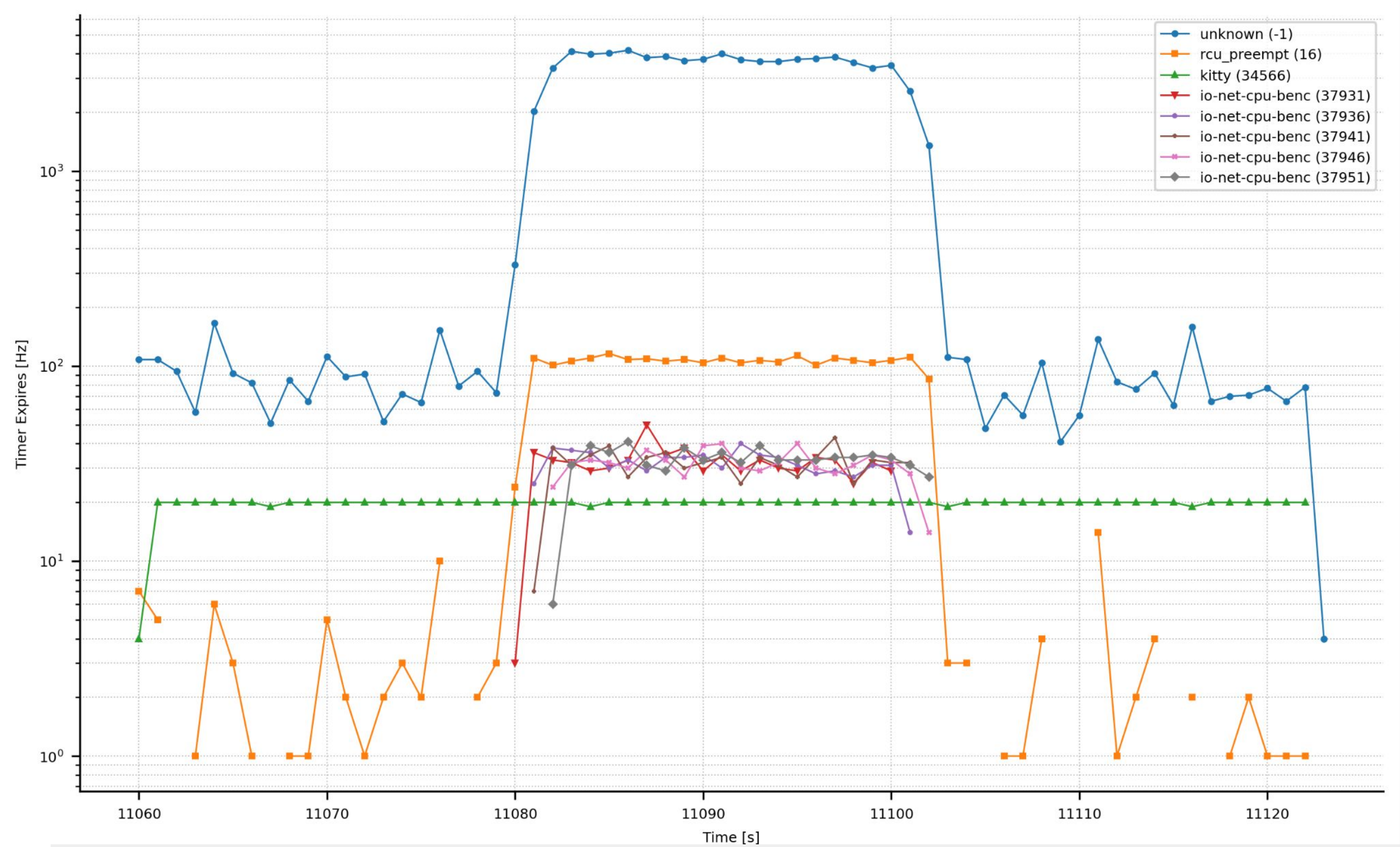


`timer-expires-unknown.py`



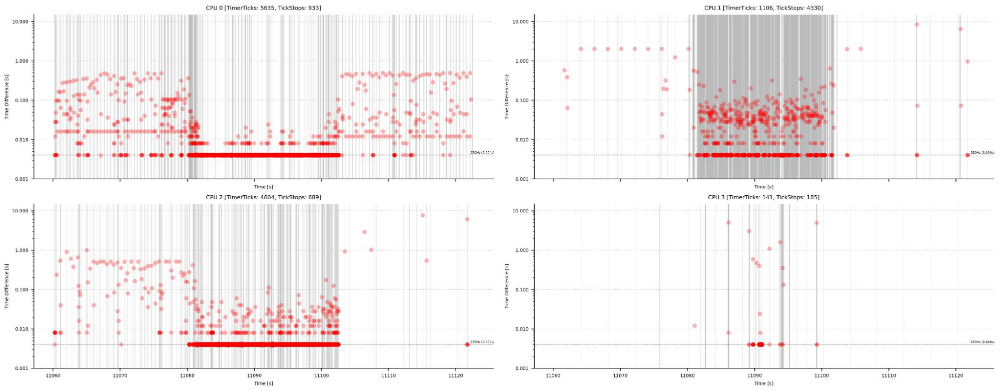
Post-processing helps with visualization, big data analysis or filtering

# Showcase Timer Module

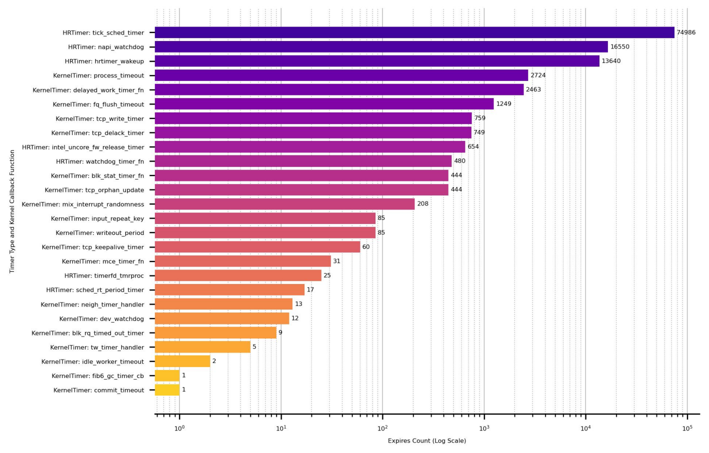


Timer expires over time, mapped to PID, unknown classified timer expires are kernel timer (high resolution timer or classic kernel timer, see next slide)

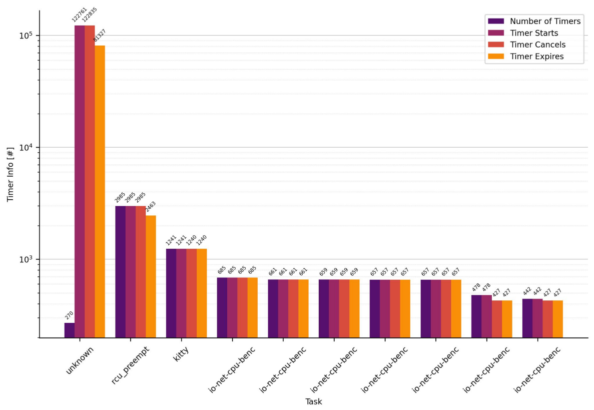
# Showcase Timer Module II



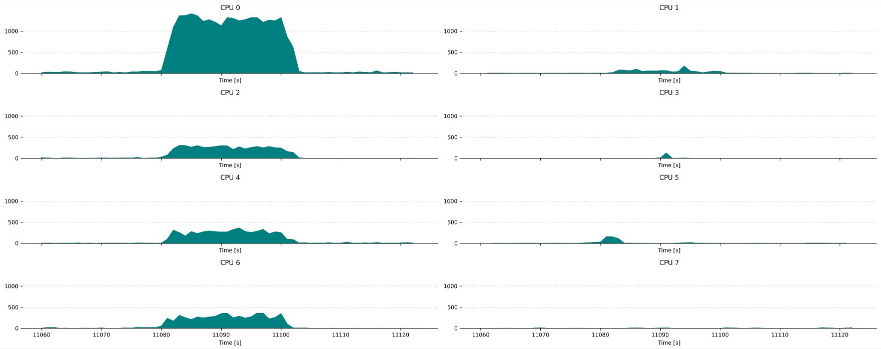
Timer tick behavior on a tickless (CONFIG_NO_HZ_FULL) system over time. Red: time difference between ticks. Gray: tick stop commands



Kernel Timer expires frequency. Here: timer tick, followed by NAPI during NIC load, hrtimer wakeups (e.g. timer expired),



Timer statistics per process, including number of timers, starts, cancel and expirations.



Time expires (not cancels) per CPU core

# General Options

**--extended**

Some of the analyses are computationally intensive and are not always required immediately ("expert analysis"). To switch these on, add --extended

**--cpu <n>**

Often you want to limit yourself to one CPU - e.g. for C-state analysis - to limit the output, this option can be used. Note: it is often better to limit the recording during recording time (perf record -C <n>)

**--file-out**

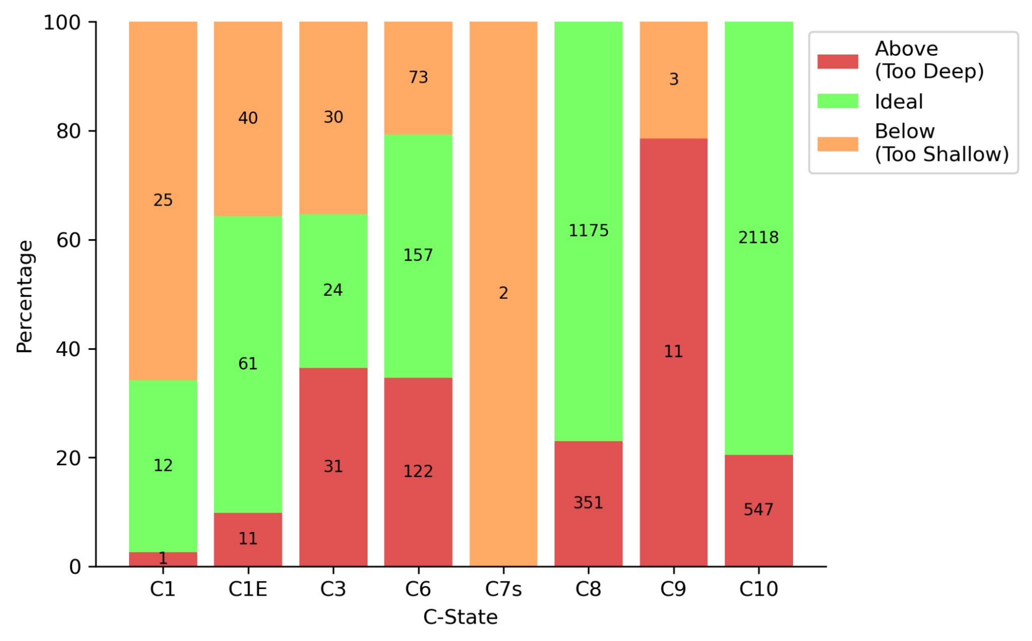
Some of the analyses call for post-processing. For example in numpy and matplotlib. To simplify this, the data is written to dedicated files

In order to make post-processing easy to use, the design was based on the requirement that the data can always be read in via pandas `pd.read_csv(PATH, delim_whitespace=True)`. This makes post-processing quite easy (at least reading the data ;-)

# Modules

**Many other analyses are still possible with timer module**

**The remaining modules such as idle-governor, idle-cluster, frequency enable analyses in completely different areas - but crucial for power management analysis and optimization**



**One last sneak-peak:** it shows the C-State choice of the Idle Governor (here menu, Intel based system) and evaluates the correctness of the decision.

Further analyses show the ideal C-states and also point out possible causes of the "wrong" decision

# Thank You!

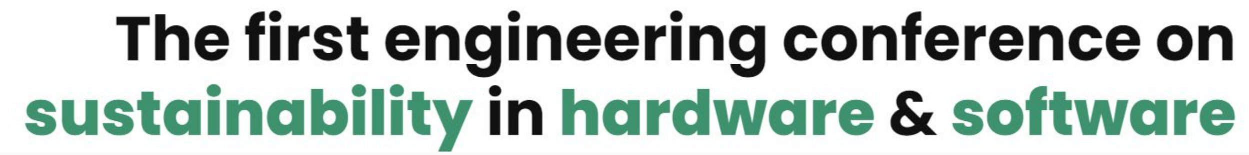
**Questions?**

**Source Code and Project Home:**

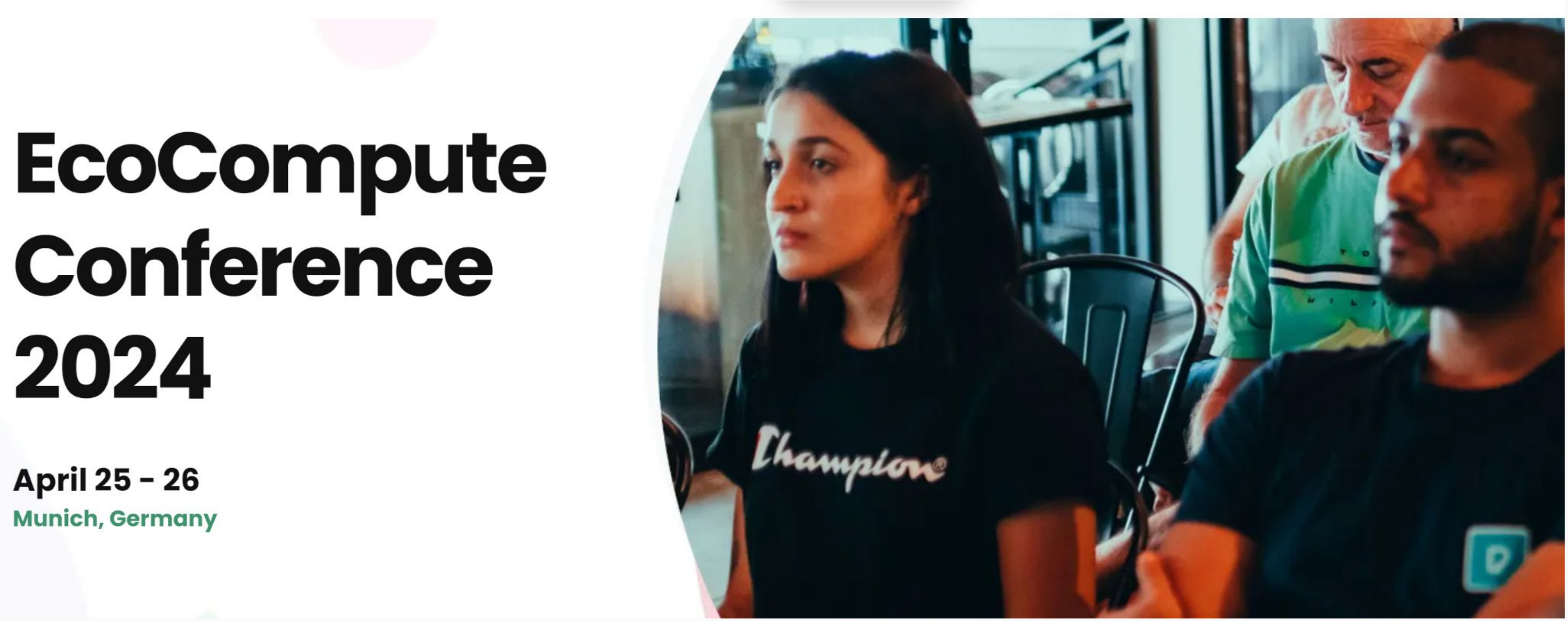
https://github.com/protocollabs/linux-kernel-perf (branch: perf-powerstat)

**Post Processing Scripts:**

https://github.com/hgn/perf-power-analyzer-post



The first engineering conference on **sustainability** in **hardware** & **software**



# EcoCompute Conference 2024

**April 25 - 26**

**Munich, Germany**

**https://www.eco-compute.io/**

# Legal Disclaimer

This presentation is for informational purposes only and is not intended as an endorsement or promotion of the products or brands mentioned herein. The trademarks, logos, and service marks (collectively the “Trademarks”) displayed in this presentation are registered and unregistered Trademarks of their respective owners.

No affiliation or endorsement by the trademark owners is intended or should be inferred. The use of these Trademarks in this presentation does not imply any affiliation with or endorsement by their respective owners.

The views and opinions expressed in this presentation are those of the presenter and do not necessarily reflect the official policy or position of the brands mentioned. This presentation is not sponsored, endorsed by, or associated with any of the brands whose products are mentioned.

All information in this presentation is provided on an “as is” basis with no guarantees of completeness, accuracy, usefulness, or timeliness, and without any warranties of any kind whatsoever, express or implied.