

Increasing Awareness of Energy Consumption in Jupyter Notebooks

presented by Marcel



Why?

- Data science and Machine Learning are very power-hungry disciplines
- Jupyter Notebooks are a popular data science tool
- Hides energy consumption
 - Not necessarily local: Server-client architecture
 - Often offered as a service (e.g., Google Colab)



Existing works

- jupyter-resource-usage^[0]
- Uses `psutil` to measure **RAM** and **CPU** usage
- Other resources not supported (not even GPU)^[1]

[0]: github.com/jupyter-server/jupyter-resource-usage

[1]: github.com/jupyter-server/jupyter-resource-usage/issues/12



A screenshot of the JupyterLab interface. The top bar shows 'jupyter hello-world' and 'Last Checkpoint: 11/20/2021 (autosaved)'. The right side of the top bar has a 'Logout' button and a Python logo. Below the top bar is a menu bar with 'File', 'Edit', 'View', 'Insert', 'Cell', 'Kernel', and 'Help'. To the right of the menu bar is a 'Trusted' status indicator and a dropdown menu showing 'Python 3 (ipykernel)'. Below the menu bar is a toolbar with icons for file operations, running, and cell management. On the right side of the toolbar, the text 'Memory: 140.7 MB' is displayed and circled in yellow. The main content area shows a notebook titled 'A simple "Hello, world!" program' with the text 'This notebook demonstrates something:'.

A simple "Hello, world!" program

This notebook demonstrates something:

How to track energy usage

- Intel CPUs offer a **Running Average Power Limit** (RAPL)
- Each trackable power metric has **several files** in `/sys/bus/event_source/devices/power/events`
 - `.../event`
 - `.../event.scale`
 - `.../event.unit`
- `perf_event_open` syscall returns a file descriptor
- Reading from that file returns **cumulative ticks** since tracking started

How to track energy usage

```
#include <linux/perf_event.h>

struct perf_event_attr attr;
memset(&attr, 0, sizeof(attr));
attr.type    = type;    // from /sys/bus/event_source/devices/power/type
attr.size    = sizeof(attr);
attr.config  = config;  // from /sys/bus/event_source/devices/power/events/<event>

long fd = syscall(
    __NR_perf_event_open, // perf_event_open syscall
    &attr,
    -1, // restricting to a pid is not supported
    0,  // not running on a particular cpu
    -1, // group_fd is not supported
    0   // no flags
);
```

How to track energy usage

```
long int ticks_before;
read(fd, &ticks_before, sizeof(long int));

sleep(10);

long int ticks_after;
read(fd, &ticks_after, sizeof(long int));

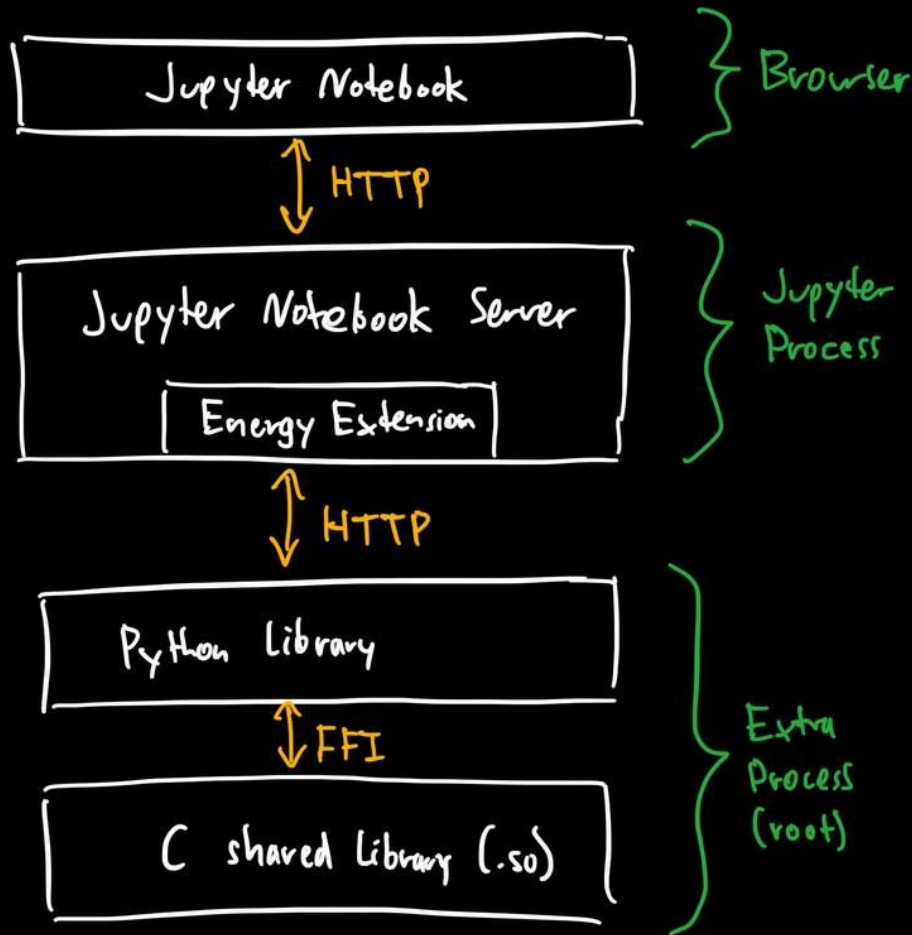
close(fd);

// scale from /sys/bus/event_source/devices/power/events/<event>.scale
double used_energy = (double)(ticks_after - ticks_before) * scale;

// unit from /sys/bus/event_source/devices/power/events/<event>.unit
printf("In 10 seconds, you used %0.3f %s.\n", used_energy, unit);
// In 10 seconds, you used 3.458 Joules.
```

How this integrates with Jupyter Notebooks

- Temporary!
- Start root process automatically on startup
- Allow hardware event listening for non-roots via `/proc/sys/kernel/perf_event_paranoid`

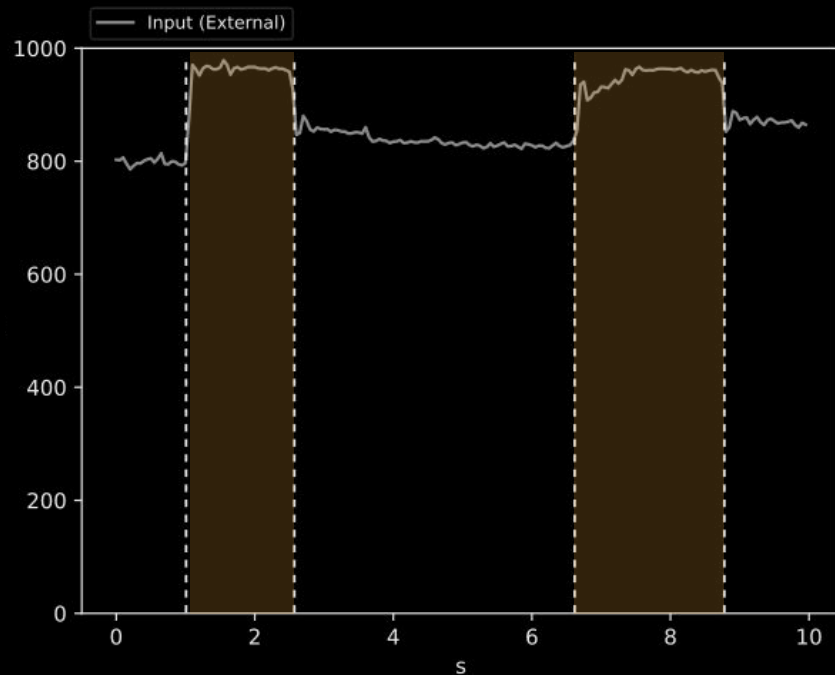


Demo!



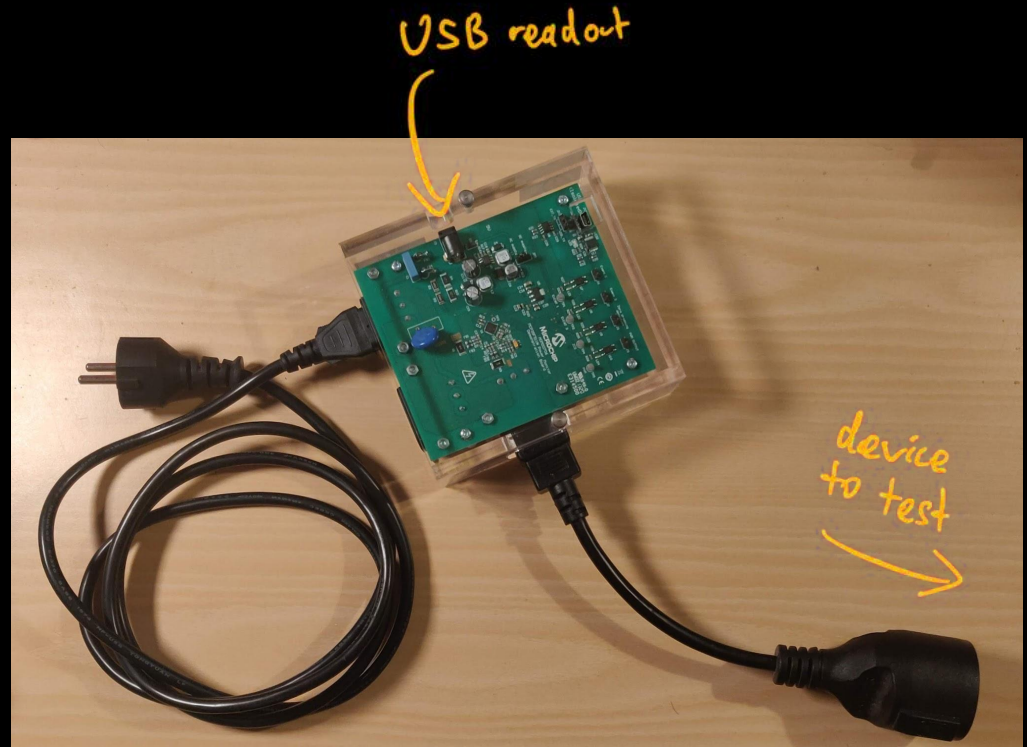
Next steps

- **Comparisons:**
You could toast a toast!
- **Graph** over time
 - Including: When was the Notebook executing code vs idle energy usage?
- Support more **sources**
 - CPU, GPU, Memory, ...
 - Jetson Counter
- **Measure** and assess accuracy



Next steps: Measure

- Assess if recorded energy consumption is accurate
- Measure-ception: Measure the overhead of measuring energy
- Measure cool comparison values



github.com/MarcelGarus/jupyter-energy
mgar.us/energy-slides

