CNRS - INP - UT3 - UT1 - UT2J

Institut de Recherche en Informatique de Toulouse

ENERGUMEN

MojitO/S

AN OPEN SOURCE SYSTEM, ENERGY AND NETWORK MONITORING TOOLS AT THE O/S LEVEL

Monitoring landscape for HPC/Cloud

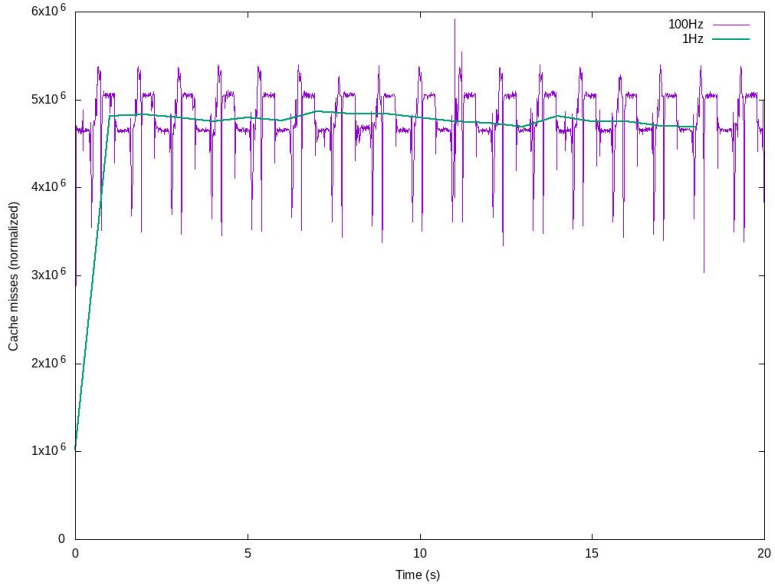
Several independent tools

Ganglia/Nagios/…

- Low overhead, high interoperability, plugin-based, distributed
- No synchronicity between sensors and focus on low throughput

Telemetry/Ceilometer

- Focus on administration/billing
- Low throughput and no synchronicity



Monitoring for researchers

Synchronicity is important

Understand the behavior of applications

- Impact on energy of a particular function?
- Links between cache management and network?
- Machine learning
  - Needs correlation between all measurements

Choices

Open to discussion

Min frequency 1Hz

Synchronized on the second (to enable external free synchronisation)

- Example: at 2Hz
  - 18.00
  - 18.50
  - 19.00
  - …

MojitO/S

https://sourcesup.renater.fr/mojitos

Capabilities

What to measure ?

Network

- Received/Transmitted packets/bytes

Performance counters

- Hardware (cache_misses, …), Software (context_switches, …)

RAPL (Running Average Power Limit)

- Energy consumption (CPU, Core, RAM, …)

Capabilities (cont'd)

When to measure

Regularly

- At a frequency during a duration
- System-level monitoring for network, performance counters, RAPL

Linked to an application

- Total for the whole life of the application
- Application-level for performance counters
- System-level monitoring for network and RAPL

Example of system-level monitoring

Classical usage for HPC systems

Monitoring for RAPL and cpu_cycle : 5 seconds, 1Hz

```
$ ./mojitos -t 5 -f 1 -p cpu_cycles -r
#timestamp cpu_cycles package-00 core0 dram0
1036988.197227391 162214 19898 4944 1586
1036989.000151326 332613664 2513116 379577 1115171
1036990.000116433 482150700 3321341 587218 1380673
1036991.000182835 525984292 3592582 691221 1385982
1036992.000165117 397678789 2770561 444030 1375729
```

Example of application-level monitoring

Classical usage for Cloud/Virtualized systems

Monitoring for RAPL and cpu_cycle : 5 seconds, 1Hz

```
$ ./mojitos -p instructions -r -o output -e script.sh
stress-ng: info:  [9361] dispatching hogs: 8 cpu
stress-ng: info:  [9361] successful run completed in 4.06s

$ cat output
#timestamp instructions package-00 core0 dram0
4.066199942 74705 16052 9155 1465
```

MojitO/S overhead

https://sourcesup.renater.fr/mojitos

Global overhead

For one measure (nova cluster on g5k Lyon) :

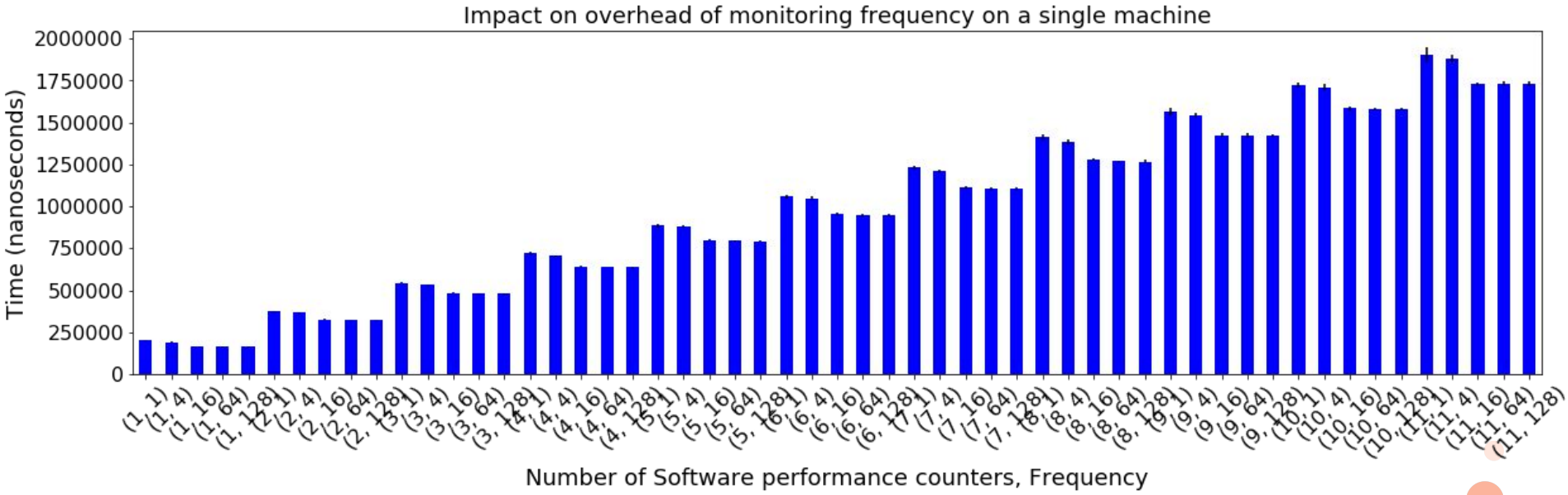
$$10\mu s + 26\mu s * Rapl + 160\mu s * (Soft + HardB4) + 53\mu s * HardO4 + 42\mu s * Net$$

With

- Rapl == 1 if rapl enabled
- Soft == number of software performance counters
- HardB4 == number of hardware performance counters (max 4)
- HardO4== number of hardware PC strictly over 4
- Net == 1 iif Network monitoring is enabled

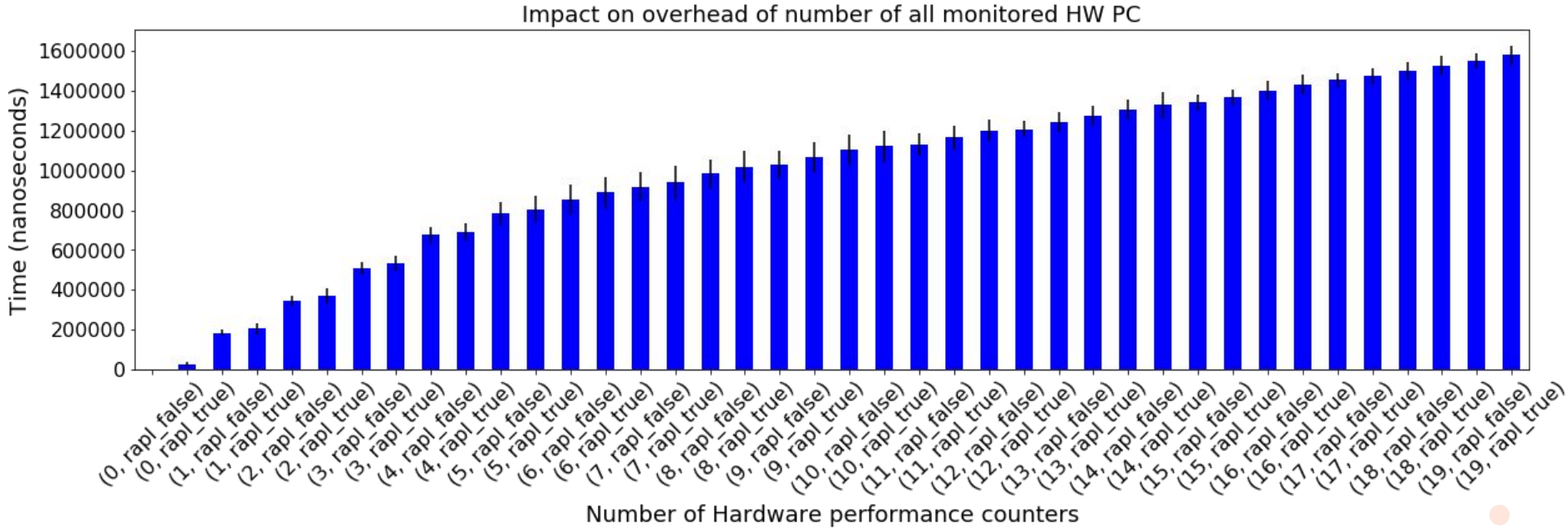
Impact of frequency

The faster we go, the lower it costs ! Thanks to the caches !



Zoom on Hardware Performance counters

Architecture with only 4 actual hardware performance counters registers

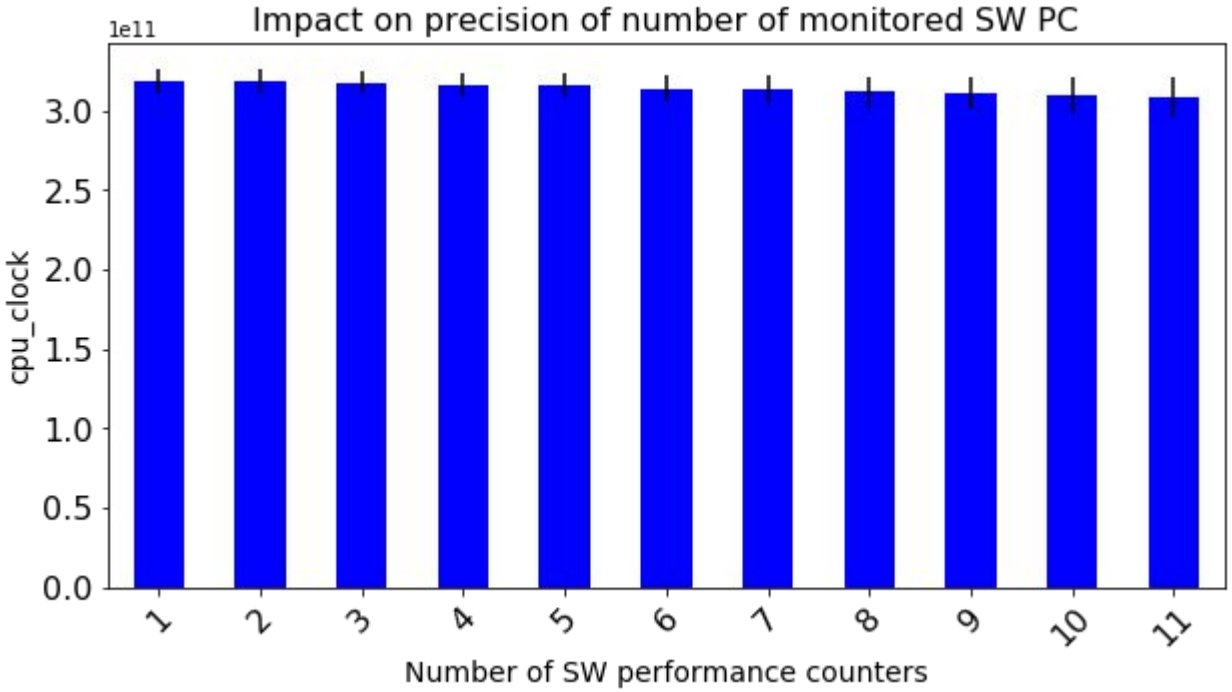


MojitO/S precision

https://sourcesup.renater.fr/mojitos

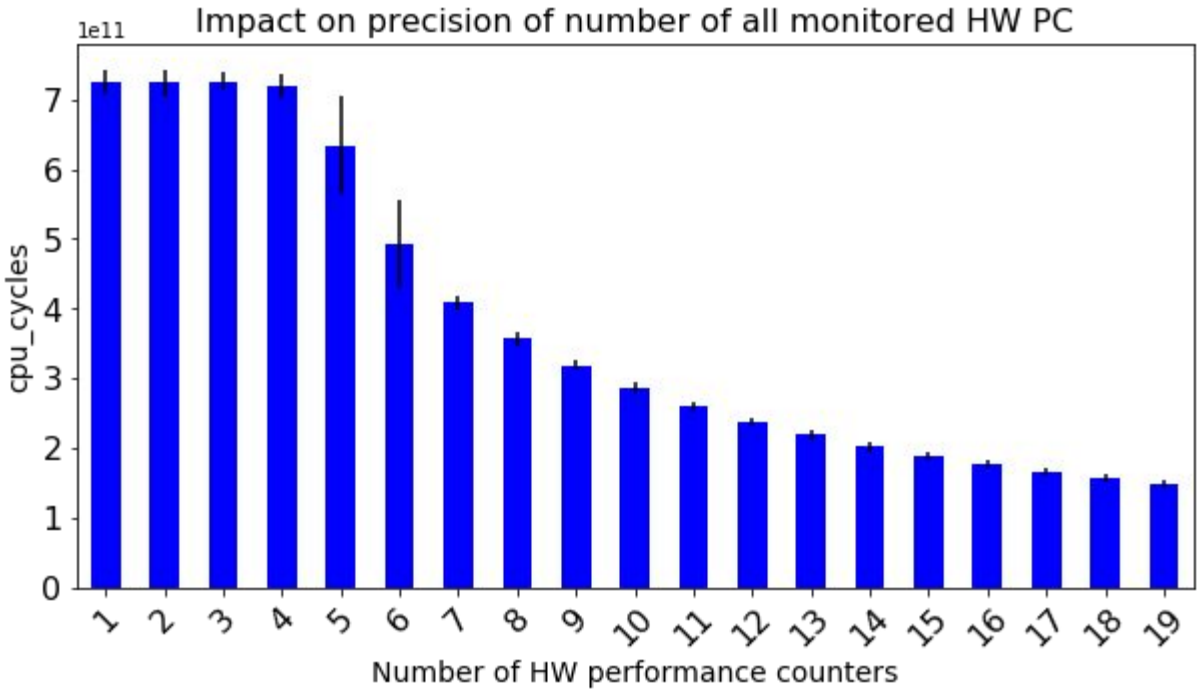
Impact of number of software PC

No impact between the number and the precision



Impact of number of hardware PC

Actually 4 hardware registers plus multiplexing



MojitO/S usage example

https://sourcesup.renater.fr/mojitos

RAPL monitoring

RAPL values during 2 seconds with a frequency of 2 Hz

```
$ ./mojitos -t 2 -f 2 -r
#timestamp package-00 core0 dram0
1036389.135659868 10986 2869 1526
1036389.500183551 1291440 255736 515562
1036390.000754048 1333553 228393 689513
1036390.500113978 1581967 267944 701536
```

Performance counters monitoring

Performance counters (cpu_cycle, cache_ll and page_faults) during 4 seconds with a frequency of 1Hz

```
$ ./mojitos -t 4 -f 1 -p cpu_cycles,cache_ll,page_faults
#timestamp cpu_cycles cache_ll page_faults
1036846.351749455 571199 1232 0
1036847.001098880 348173344 2451387 872
1036848.000166158 388112961 2509305 791
1036849.000191883 402255979 2625283 799
```

Network monitoring

Network values with no time limit with a frequency of 1Hz

```
$ ./mojitos -t 0 -f 1 -d enp0s25
#timestamp rxp rxb txp txb
1036559.277376027 0 0 0 0
1036560.000161101 4 581 2 179
1036561.000083968 178 268675 55 4954
1036562.000076162 11 1010 5 510
1036563.000069724 17 1643 12 3602
1036564.000113394 990 1493008 369 27299
```

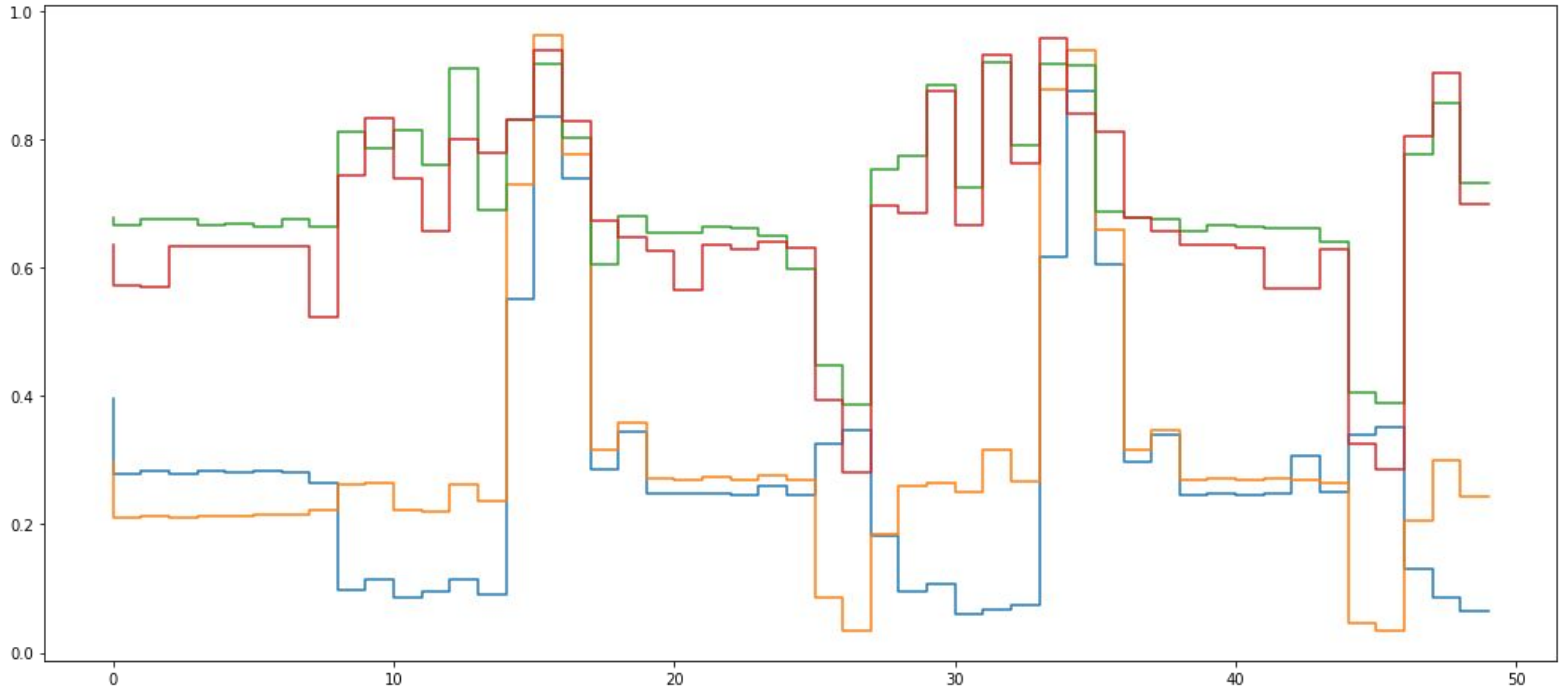
Overhead

Overhead of the monitoring for RAPL and cpu_cycle

```
$ ./mojitos -t 5 -f 1 -p cpu_cycles -r -s
#timestamp cpu_cycles package-00 core0 dram0 overhead
1036988.197227391 162214 19898 4944 1586 149612
1036989.000151326 332613664 2513116 379577 1115171 739573
1036990.000116433 482150700 3321341 587218 1380673 315719
1036991.000182835 525984292 3592582 691221 1385982 272182
1036992.000165117 397678789 2770561 444030 1375729 510379
```

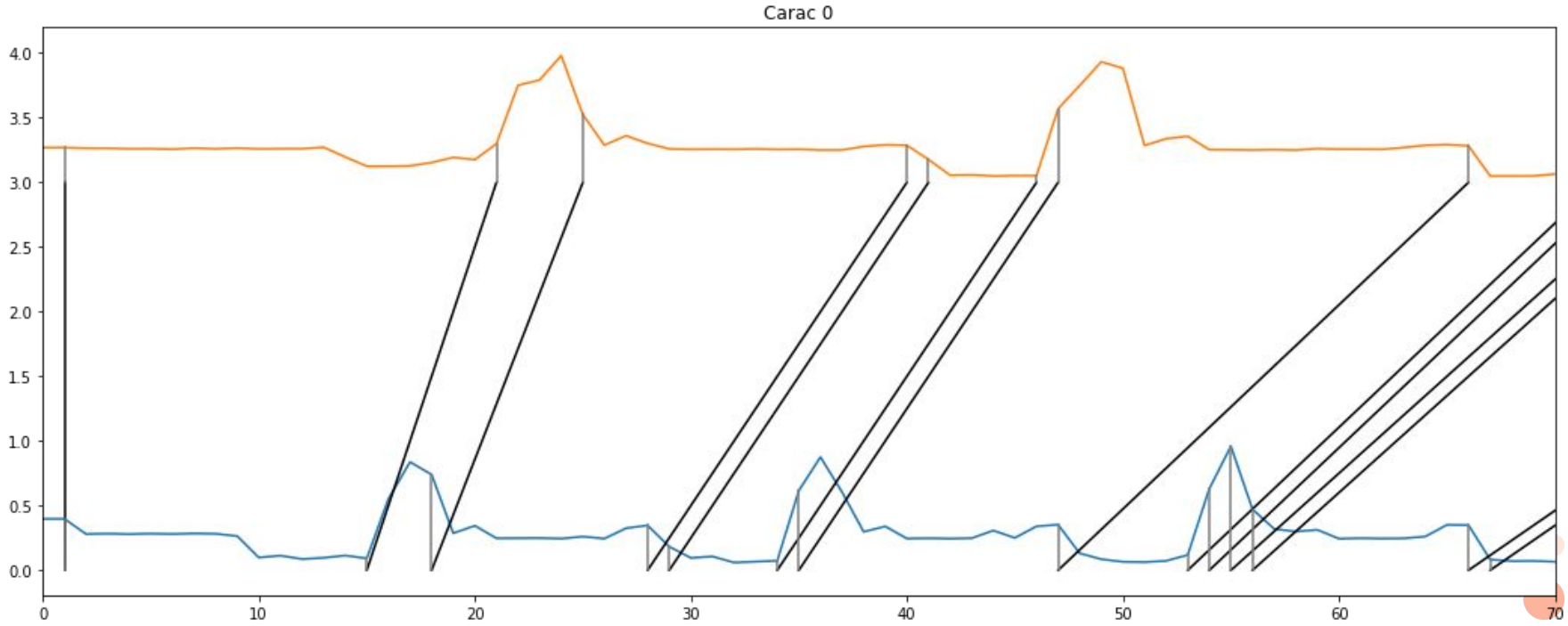
IS (Integer Sort), 100Hz

If you gaze long into an abyss, …



IS (Integer Sort), DVFS

If you gaze long into an abyss, the abyss also gazes into you



Conclusion/Discussion

https://sourcesup.renater.fr/mojitos

Conclusion/Discussion

Starting project, open to comment

MojitO/S

- Command line tool for Linux under GPLv3
- Measures RAPL, Performance counters, network
- Provides library hooks for all monitoring elements
- In the context of Energumen ANR project

Perspectives

- Other sensors?
- Change the ‘second’ synchro?
- …