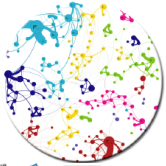
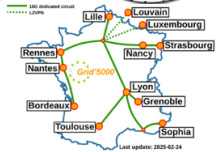
CNRS - INP - UT3 - UT1 - UT2J
Institut de Recherche en Informatique de Toulouse





Energy measures on Grid'5000

Georges Da Costa
Workshop: Environmental impact of HPC
November, 12, 2025

ENERGUMEN
FRANCE 2030 PROGRAMME DE RECHERCHE NUMÉRIQUE POUR L'EXASCALE
FRANCE 2030 PROGRAMME DE RECHERCHE CLOUD
slices RI
slices FR

georges.da-costa@irit.fr

Plan

Energy and Power

$Energy(J) = Power(W) * Time(s)$

Classical use

- If t[i] is the series of power every 1/10s then $E = \sum(t[i]/10)$

Classical models

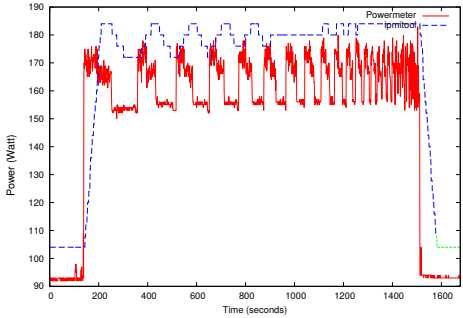
- Server: $Power = Static + \sum(components)$
- CPU: $Power\ \alpha\ Load$
- RAM
- Storage/Network: $Power \simeq Constant$

Leverages

- DVFS (Dynamic Voltage and Frequency Scaling) : decreasing frequencies (CPU or GPU or RAM) increases times and reduces power
- PowerCap : bounds the maximum power of a subsystem

Measure

- IPMI or BMC : classically used for protection against electrical problem at server or rack level
  - Low precision / frequency / reactivity
- e-Pdu : outlet level, usually one per server
  - Medium precision, frequency up to 1Hz
- RAPL (or equivalent) : power measure (and model) inside the CPU/RAM/GPU
  - High precision, frequency up to 1kHz, can go down to each core



Processor level measure (RAPL)

Several existing tools : MojitO/S*, PowerAPI, PAPI, ...
Access specific register in the processor

- Model based : uses internal sensors to compute a consumption for processor, core, RAM
- Obtaining it uses 16μs on a Intel(R) Xeon(R) Gold 5220 CPU @ 2.20GHz
- Can also be used for power cap
- Information are in `/sys/class/powercap/intel-rapl/subsystem/`

```
cat /sys/class/powercap/intel-rapl/subsystem/*/name
cat /sys/class/powercap/intel-rapl/subsystem/*/energy_uj
```

* https://gitlab.irit.fr/sepia-pub/mojitos

Example of RAPL monitoring

```
$ sudo-g5k mojitos -r -f 100 -o /tmp/data.csv -- stress -c 32 -t 1
stress: info: [30592] dispatching hogs: 32 cpu, 0 io, 0 vm, 0 hdd
stress: info: [30592] successful run completed in 1s

gdacosta@gros-49:~/$ head /tmp/data.csv
#timestamp package-00 dram0
1751826254.017109001 0 0
1751826254.020102944 154846 24281
1751826254.030106390 520934 76026
1751826254.040108688 738279 74771
1751826254.050101430 868833 75031
1751826254.060104144 945310 75292
1751826254.070102863 1034361 74541
1751826254.080062131 1118222 75123
1751826254.090056943 1155087 74670
```

How to use these information

	#timestamp	package-00	dram0
count	1.010000e+02	1.010000e+02	101.000000
mean	1.751826e+09	1.127220e+06	73092.039604
std	2.928773e-01	1.750058e+05	9034.836839
min	1.751826e+09	0.000000e+00	0.000000
25%	1.751826e+09	1.164487e+06	74220.000000
50%	1.751826e+09	1.169125e+06	74541.000000
75%	1.751826e+09	1.173642e+06	74771.000000
max	1.751826e+09	1.288327e+06	82023.000000

- `package-00` is the processor, `dram0` the memory
- Values are in micro-joules, and frequency was 100Hz
- Total processor energy is $mean/10^6 * count$, ie 114J for the processor
- Processor mean power is $mean/10^6 * 100$, ie 113W for the processor

External power measures

Direct kwollect*method

- Access to a database
- List of resources + start/stop timestamp
- OAR_ID

Precision depends on the cluster

- 5 sec on Chetemi, $\approx$ 14W of precision
- 1 sec on Taurus, $\approx$ .1W of precision

Some cluster have high precision mode

- Explicit during node reservation
- One measure every 20 milli-seconds

* https://www.grid5000.fr/w/Monitoring_Using_Kwollect

Example of kwollect usage

```
geo@fnancy:~$ curl "https://api.grid5000.fr/stable/sites/lyon/metrics?
               nodes=taurus-10&metrics=wattmetre_power_watt&
               start_time=$(date -d '10 sec ago' +%s)"
```


```
[{"timestamp":"2025-07-06T22:36:02+02:00","device_id":"taurus-10","metric_id":"wat
 {"timestamp":"2025-07-06T22:36:03+02:00","device_id":"taurus-10","metric_id":"wat
 {"timestamp":"2025-07-06T22:36:04+02:00","device_id":"taurus-10","metric_id":"wat
 {"timestamp":"2025-07-06T22:36:05+02:00","device_id":"taurus-10","metric_id":"wat
 {"timestamp":"2025-07-06T22:36:06+02:00","device_id":"taurus-10","metric_id":"wat
 {"timestamp":"2025-07-06T22:36:07+02:00","device_id":"taurus-10","metric_id":"wat
 {"timestamp":"2025-07-06T22:36:08+02:00","device_id":"taurus-10","metric_id":"wat
 {"timestamp":"2025-07-06T22:36:09+02:00","device_id":"taurus-10","metric_id":"wat
 {"timestamp":"2025-07-06T22:36:10+02:00","device_id":"taurus-10","metric_id":"wat
 {"timestamp":"2025-07-06T22:36:11+02:00","device_id":"taurus-10","metric_id":"wat
```

Tools for accessing kwollect information

- Several tools: Expetator*
- Limit : sensor name depends on the cluster

```
def get_g5k_target_metric(cluster_name=None):
    if cluster_name in ['grisou', 'graoully', 'grimoire',
                        'gros', 'gruss', 'paravance']:
        return 'pdu_outlet_power_watt'
    if cluster_name in ['troll', 'yeti', 'gemini', 'neowise',
                        'orion', 'pyxis', 'sagittaire', 'taurus']:
        return 'wattmetre_power_watt'

    return 'bmc_node_power_watt'
```

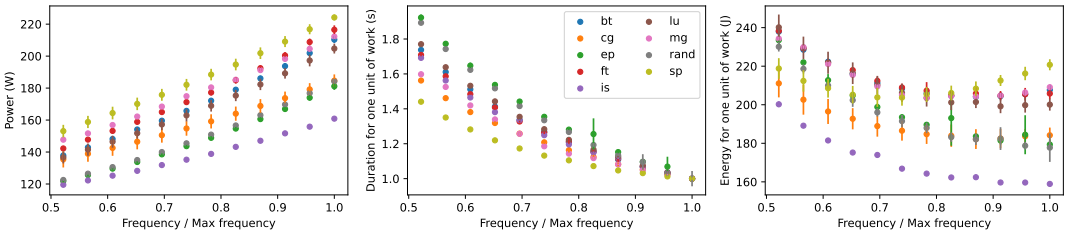
* https://gitlab.irit.fr/sepia-pub/expetator/

Example of usage of Expetator

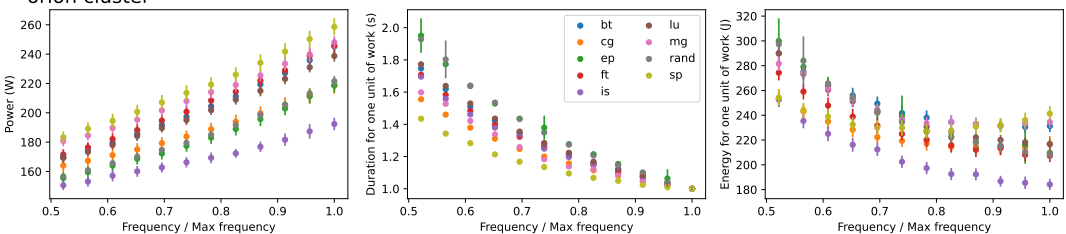
```
from expetator.benchmarks import SleepBench
from expetator.monitors import kwollect
import expetator.experiment as experiment

MONITORS = [ kwollect.Power(metric=kwollect.get_g5k_target_metric())]
BENCHMARKS = [SleepBench(default_time=10)]

experiment.run_experiment('/tmp/power_demo', benchmarks = BENCHMARKS,
                          monitors = MONITORS)

$ cat /tmp/power_demo_montcalm-9.toulouse.grid5000.fr_1751879397_power/
                        montcalm-9.toulouse.grid5000.fr_sleep-10_175187940
[["montcalm-9", [1751879403.685383, 1751879408.68714], [131, 131]]]
```

High accuracy for some clusters

taurus cluster

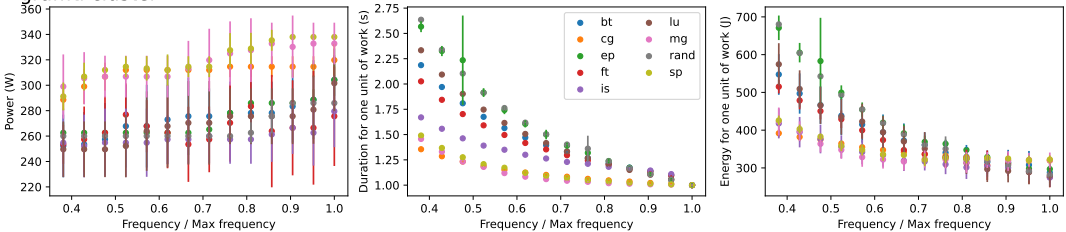


orion cluster

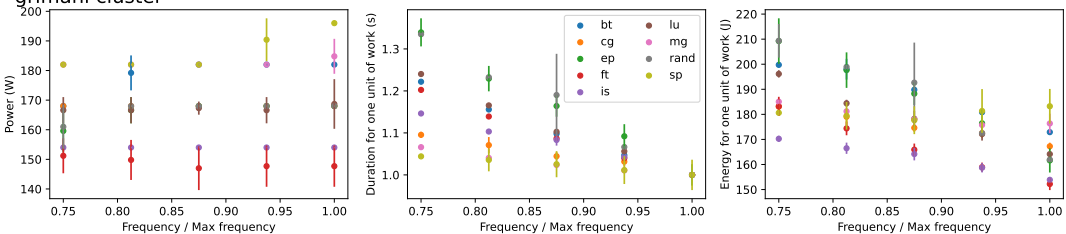


Lower accuracy for other clusters

graffiti cluster



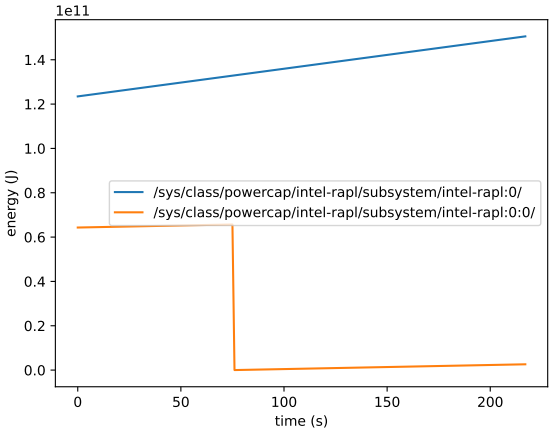
grimani cluster



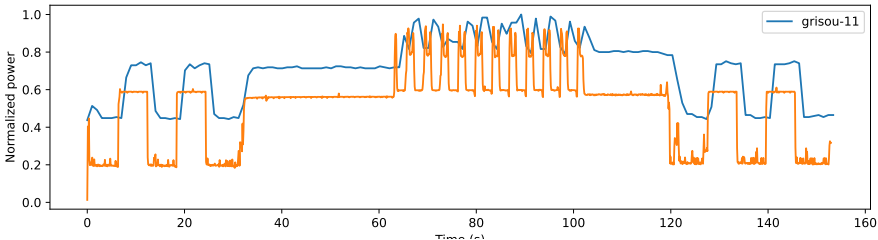
Plan

Limits of direct access of /sys/class file for RAPL

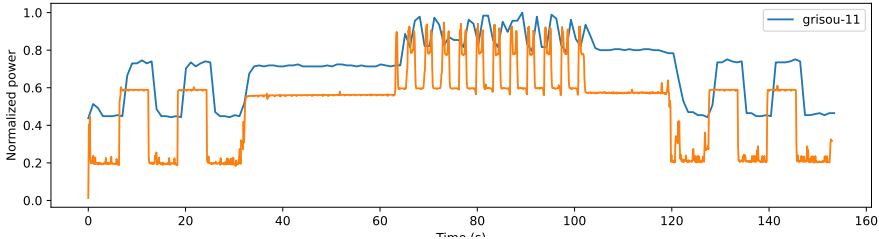
- Maximum value for each sensor : `max_energy_range_uj`
- Different values for memory and processor
- Orange line : memory

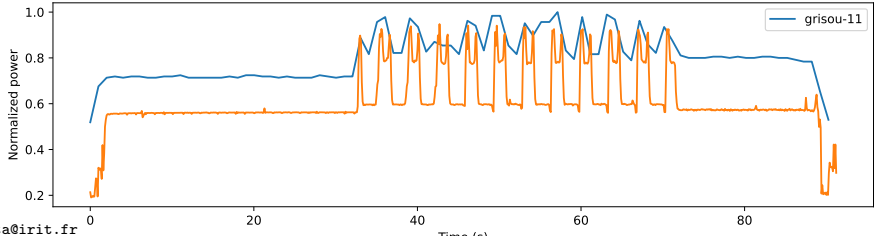


Synchronization between multiple monitoring sources



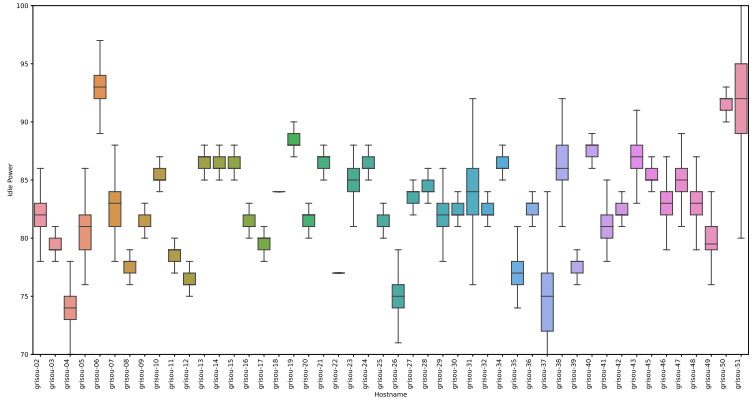
Synchronization between multiple monitoring sources





Heterogeneity of hardware

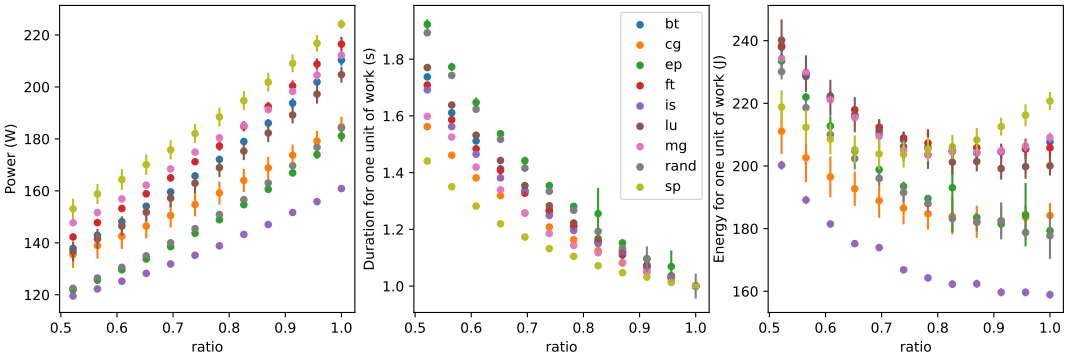
Homogeneous servers consume differently. Here same servers, idles, one measure per second during 5 minutes.



Heterogeneity of software

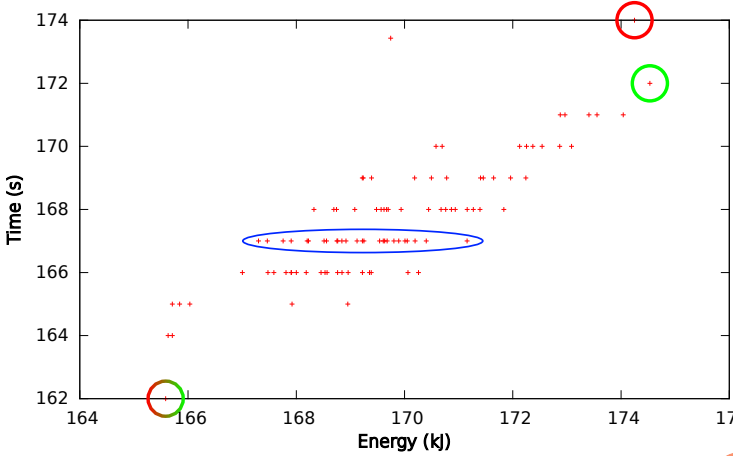
Different applications consume different power and react differently to leverages (here DVFS)

taurus cluster

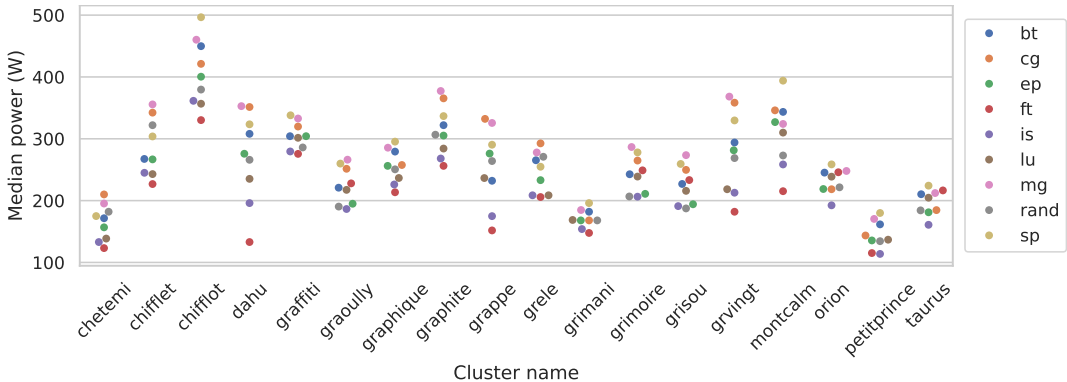


Stability of measures

100 experiments, same 4 servers, same application

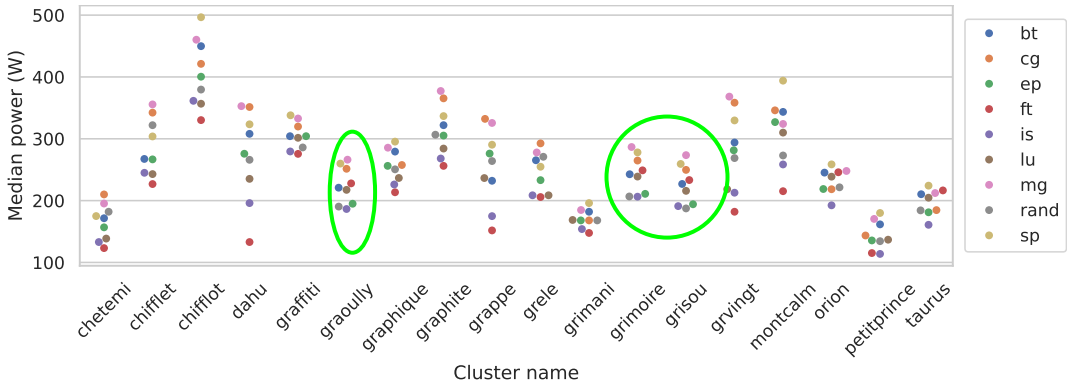


Application power ranking depends on the hardware



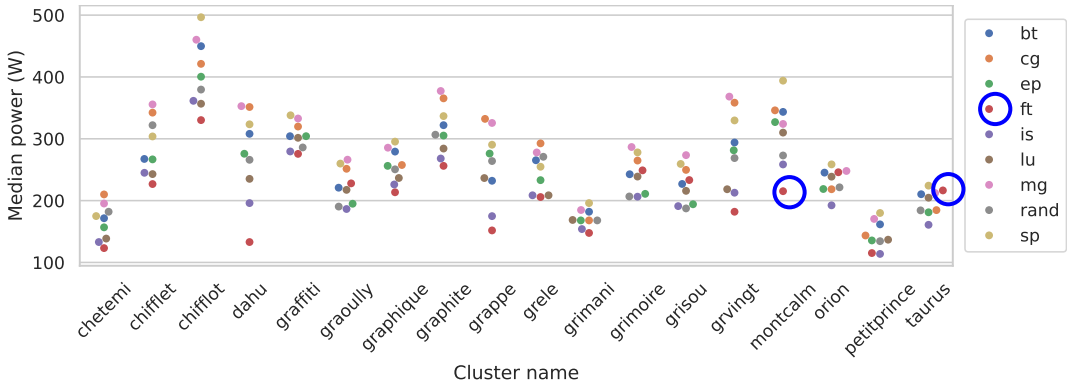
Execution at maximum frequencies for each cluster.

Application power ranking depends on the hardware



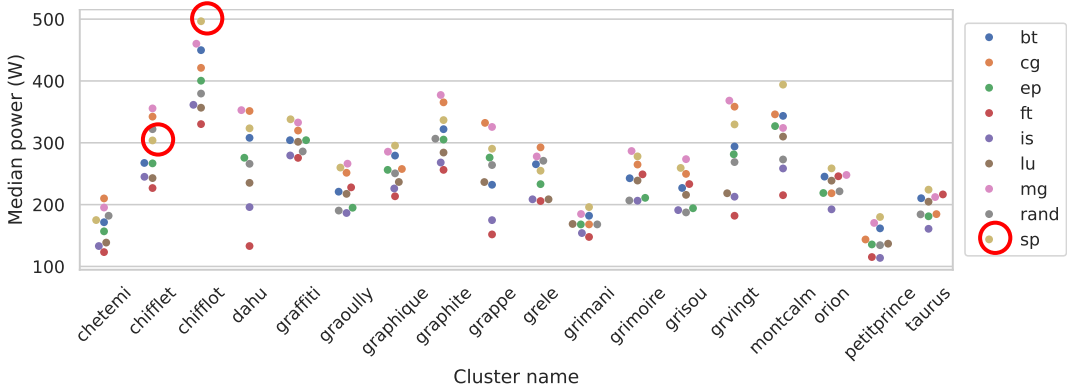
Execution at maximum frequencies for each cluster.
Same architecture: 2 x Intel Xeon E5-2630 v3

Application power ranking depends on the hardware



Execution at maximum frequencies for each cluster.
FT can be first or last

Application power ranking depends on the hardware



Execution at maximum frequencies for each cluster.
SP can be the overall first or in the middle

Plan

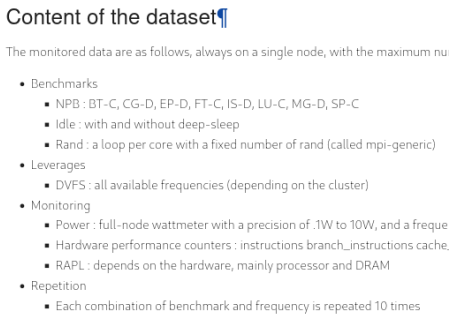
Impact of measures (mean values in μs)

	0.486420
-d X	20.600630
-c	0.390500
-i X	655.981350
-u	81.919890
-m	3.264560
-r	**90.589450**
-a	20.336900
-p cpu_cycles	1790.206120
-p cpu_cycles,instructions	3257.242360
-p cpu_cycles,instructions,cache_references	4147.999970
-p cpu_cycles,instructions,cache_references,cache_misses	4835.197450
-p cache_l1d_r_m	1836.312220
-p cache_l1d_r_m,cache_ll_w_m	3081.448620
-p cache_l1d_r_m,cache_ll_w_m,cache_dtlb_r_a	4023.425780
-p cache_l1d_r_m,cache_ll_w_m,cache_dtlb_r_a,cache_node_w_a	4426.887100

Open-data is the way

- Producing dataset consume large amount of energy, require time, ...
- F.A.I.R. Principle
  - **Findability, Accessibility, Interoperability**, Reusability
- But also Re-productible

Example:
`https://zenodo.org/records/14914799`



Content of the dataset¶

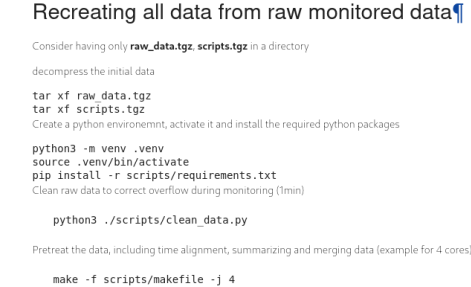
The monitored data are as follows, always on a single node, with the maximum nu

- Benchmarks
  - NPB : BT-C, CG-D, EP-D, FT-C, IS-D, LU-C, MG-D, SP-C
  - Idle : with and without deep-sleep
  - Rand : a loop per core with a fixed number of rand (called mpi-generic)
- Leverages
  - DVFS : all available frequencies (depending on the cluster)
- Monitoring
  - Power : full-node wattmeter with a precision of .1W to 10W, and a freque
  - Hardware performance counters : instructions branch_instructions cache,
  - RAPL : depends on the hardware, mainly processor and DRAM
- Repetition
  - Each combination of benchmark and frequency is repeated 10 times

Open-data is the way

- Producing dataset consume large amount of energy, require time, ...
- F.A.I.R. Principle
  - Findability, Accessibility, Interoperability, **Reusability**
- But also Re-productible

Example:
`https://zenodo.org/records/14914799`



Recreating all data from raw monitored data¶

Consider having only **raw_data.tgz**, **scripts.tgz** in a directory

decompress the initial data

```
tar xf raw_data.tgz
tar xf scripts.tgz
```

Create a python environemnt, activate it and install the required python packages

```
python3 -m venv .venv
source .venv/bin/activate
pip install -r scripts/requirements.txt
```

Clean raw data to correct overflow during monitoring (1min)

```
python3 ./scripts/clean_data.py
```

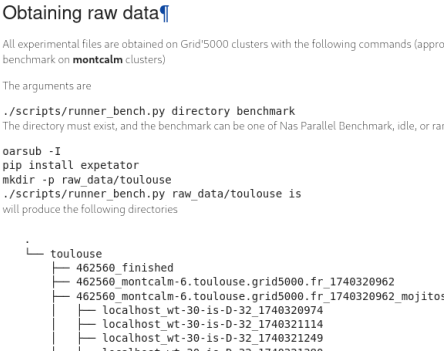
Pretreat the data, including time alignment, summarizing and merging data (example for 4 cores) :

```
make -f scripts/makefile -j 4
```

Open-data is the way

- Producing dataset consume large amount of energy, require time, ...
- F.A.I.R. Principle
  - Findability, Accessibility, Interoperability, Reusability
- But also **Re-productible**

Example:
`https://zenodo.org/records/14914799`



Obtaining raw data¶

All experimental files are obtained on Grid'5000 clusters with the following commands (appro
benchmark on **montcalm** clusters)

The arguments are

```
./scripts/runner_bench.py directory benchmark
```

The directory must exist, and the benchmark can be one of Nas Parallel Benchmark, idle, or ra

```
oarsub -I
pip install expetator
mkdir -p raw_data/toulouse
./scripts/runner_bench.py raw_data/toulouse is
```

will produce the following directories

```
.
└── toulouse
    ├── 462560_finished
    ├── 462560_montcalm-6.toulouse.grid5000.fr_1740320962
    ├── 462560_montcalm-6.toulouse.grid5000.fr_1740320962_mojitos
    │   ├── localhost_wt-30-is-D-32_1740320974
    │   ├── localhost_wt-30-is-D-32_1740321114
    │   ├── localhost_wt-30-is-D-32_1740321249
```

Plan

Experimental Framework

- Benchmarks: 8 Nas Parallel Benchmarks (NPB); `rand`: loop of call to rand function
- Platform: 18 clusters from Grid5000, processors from 2012 to 2021
- Grid5000 Power monitoring, DVFS management, experiment management: Expetator*
- Hardware performance counters, RAPL: MojitO/S†
- Raw data ‡ 7Go, cleaned data 5.6Go

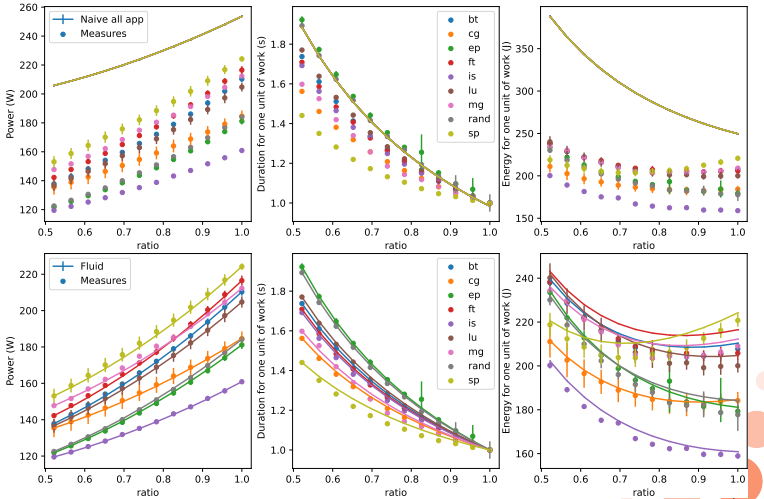
Experiments: All benchmarks, all available frequencies, on all clusters, 10 times

*`https://gitlab.irit.fr/sepia-pub/expetator`
†`https://gitlab.irit.fr/sepia-pub/mojitos`
‡`https://zenodo.org/records/10982239`

Example of result

Classical model:

Improved model:



Plan

Some pointers

Measure tools

- CPU/RAM/GPU-level measure : MojitO/S
  `https://gitlab.irit.fr/sepia-pub/mojitos`

Dataset (4Go)

- 10.5281/zenodo.10982238

Articles

- *Hardware and application aware performance, power and energy models for modern HPC servers with DVFS*. Sustainable Computing : Informatics and Systems, 2025
  - DOI: `https://doi.org/10.1016/j.suscom.2025.101106`
  - Artefact: `https://gitlab.irit.fr/sepia-pub/open-science/ppe-models-for-hpc-servers-dvfs`
- Measure tools Comparison: 10.1109/CCGrid57682.2023.00020
- Methodology of measure : 10.1016/j.suscom.2017.05.003