RÉPUBLIQUE FRANÇAISE
Liberté
Égalité
Fraternité

FRANCE 2030
PROGRAMME DE RECHERCHE
NUMÉRIQUE POUR L'EXASCALE

Fine-grain energy consumption modeling of HPC task-based programs

AG Numpex 2025

Jules RISSE, François TRAHAY, Amina GUERMOUCHE

24/09/2025

INRIA, Institut Polytechnique de Paris

Introduction

Why should we care about energy ?

El Capitan, the fastest supercomputer in the world (top500):

- 30 MW,
- power consumption of 55,000 French households

Electricity cost estimation:
$\$1\text{M}/\text{MW}_{\text{peak}}/\text{year}$

Annual electricty bill: around $30M

Figure 1: El Capitan cluster
(11,136 nodes based on AMD MI300A APUs)

How to use HPC clusters ?

Supercomputer nodes are accelerated and heterogeneous (CPUs, GPUs, APUs, diverse memory tiers, interconnects).

Classic approach: GPU offloading.

Challenges: coordination, data movement, idle host CPU.

Possible solution: express computations as task graphs, let a runtime schedule tasks across resources.
task-based runtime systems

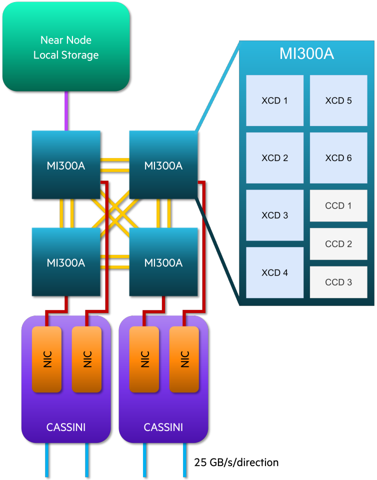


Figure 2: El Capitan node diagram

The StarPU runtime system

Task programming library for heterogeneous architectures.

- Developers express their application as a directed acyclic graph (DAG) of tasks and data dependencies.
- Tasks can have multiple implementations (CPU, CUDA/HIP).
- StarPU handles optimized scheduling, task dependencies, data transfers and replication.

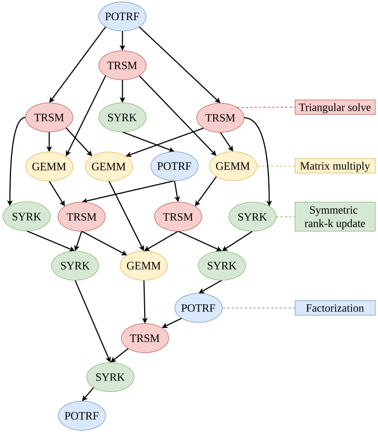
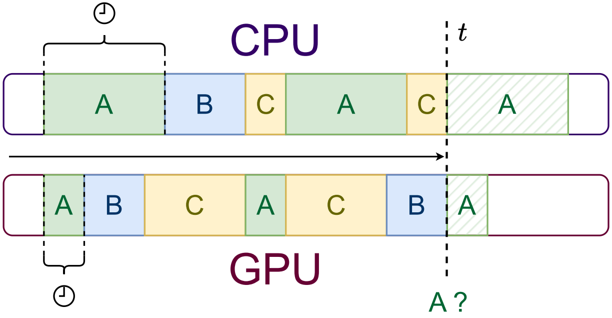


Figure 3: DAG of a 5x5 matrix Cholesky decomposition

Scheduling in StarPU

In StarPU, schedulers can use a history based task performance model.

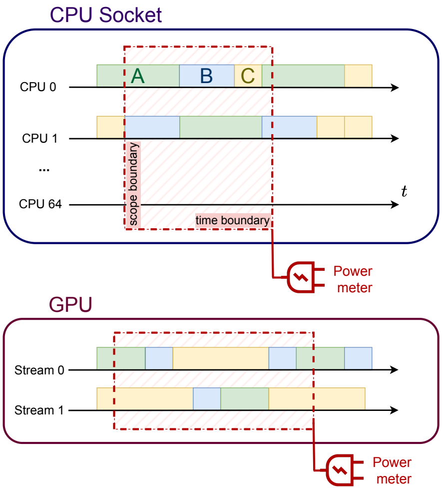
- Task duration is measured at runtime on all devices.
- StarPU schedulers can use timing values for decisions.
- Users can define their own scheduling policies.



Problem: StarPU does not consider the energy consumption of tasks.

How to measure the energy consumption of tasks ?

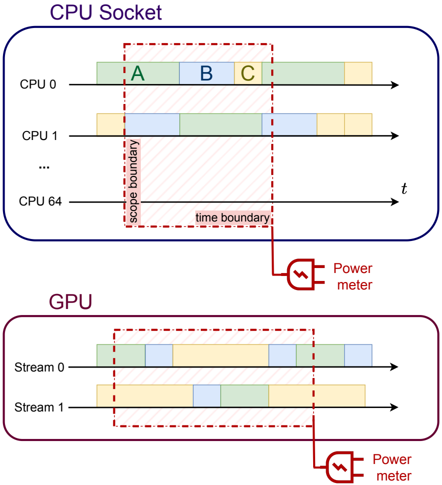
Not directly possible because of granularity issues:



How to measure the energy consumption of tasks ?

Not directly possible because of granularity issues:

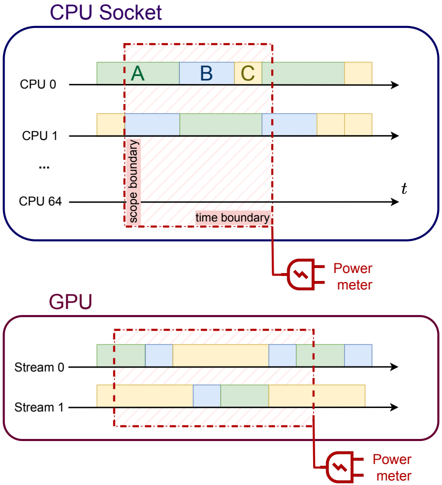
- Time granularity
  - Power meter granularity: $1 \rightarrow 100\,\text{ms}$.
  - Task execution time: $\sim 1\,\text{ms}$.



How to measure the energy consumption of tasks ?

Not directly possible because of granularity issues:

- Time granularity
  - Power meter granularity: $1 \rightarrow 100\,\text{ms}$.
  - Task execution time: $\sim 1\,\text{ms}$.
- Spatial granularity
  - Power meter scope: socket (CPU), device (GPU).
  - Tasks run on individual cores / SM threads in parallel.

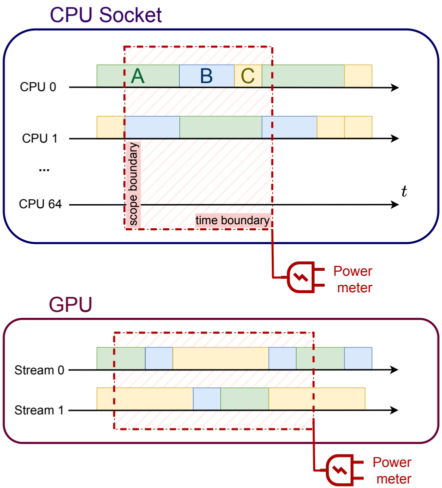


How to measure the energy consumption of tasks ?

Not directly possible because of granularity issues:

- Time granularity
  - Power meter granularity: $1 \rightarrow 100\,\text{ms}$.
  - Task execution time: $\sim 1\,\text{ms}$.
- Spatial granularity
  - Power meter scope: socket (CPU), device (GPU).
  - Tasks run on individual cores / SM threads in parallel.

Objective: model the energy consumption of individual tasks



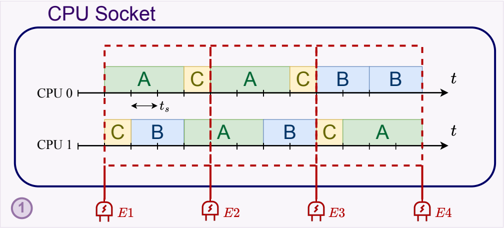
Proposal

Main idea

Use coarse-grain energy measurements and task execution traces to create and solve overdetermined linear systems linking observed energy and fine-grain task energy consumption.

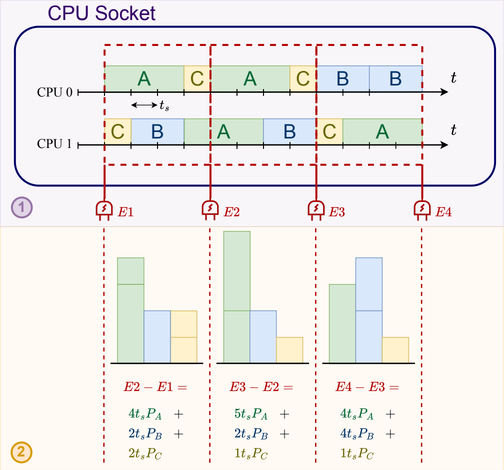
Modeling task energy consumption in StarPU

1. Measure coarse-grain energy consumption and trace task executions at runtime.



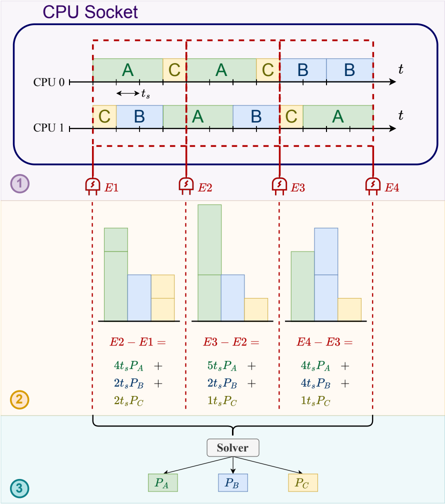
Modeling task energy consumption in StarPU

1. Measure coarse-grain energy consumption and trace task executions at runtime.
2. After execution, group tasks per energy measurement interval to create linear equations.



Modeling task energy consumption in StarPU

1. Measure coarse-grain energy consumption and trace task executions at runtime.
2. After execution, group tasks per energy measurement interval to create linear equations.
3. Solve the resulting linear system to get tasks power consumption.



Implementation

Measuring the energy consumption

energy-reader, a lightweight C library and API for querying software energy counters.

Access monotonic energy counters from:

- AMD and Intel CPUs with Running Average Power Limit (RAPL) counters.
- Nvidia GPUs with NVML.
- AMD GPUs with ROCm-SMI.
- HPE Cray clusters with PM counters.

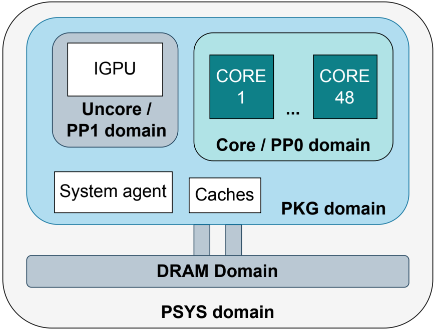


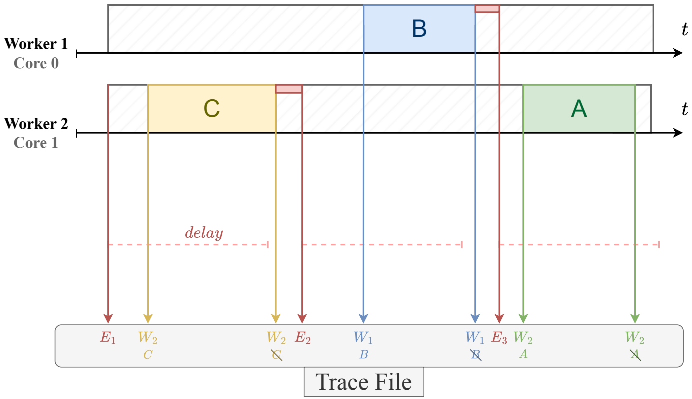
Figure 4: RAPL domains hierarchy

Tracing energy and tasks in StarPU

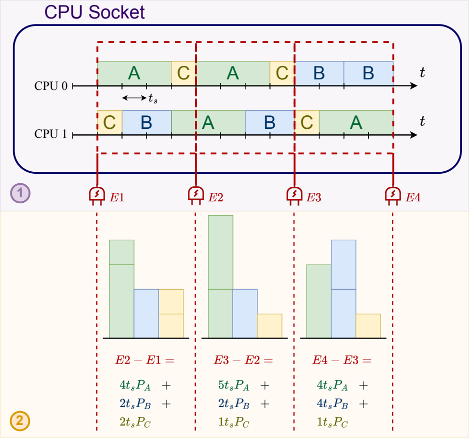
StarPU has a probe based tracing system logging task executions.

We use the active waiting loop of CPU workers to measure and log energy values.

Output: timestamped trace file with energy and task events.



Building the linear system



energy-solver, a python module to build and solve linear systems from StarPU trace files.

Matrix system representation:

$$\begin{pmatrix} 4\mathrm{t_s} & 2\mathrm{t_s} & 2\mathrm{t_s} \\ 5\mathrm{t_s} & 2\mathrm{t_s} & 1\mathrm{t_s} \\ 4\mathrm{t_s} & 4\mathrm{t_s} & 1\mathrm{t_s} \end{pmatrix} \begin{pmatrix} \mathrm{P_A} \\ \mathrm{P_B} \\ \mathrm{P_C} \end{pmatrix} = \begin{pmatrix} \mathrm{E_2 - E_1} \\ \mathrm{E_3 - E_2} \\ \mathrm{E_4 - E_3} \end{pmatrix}$$

Solving the linear system

In practice, the matrix is tall:

$$\begin{pmatrix} 4t_s & 2t_s & 2t_s \\ 5t_s & 2t_s & 1t_s \\ 4t_s & 4t_s & 1t_s \\ \dots & \dots & \dots \\ \dots & \dots & \dots \\ \dots & \dots & \dots \\ \dots & \dots & \dots \end{pmatrix} \begin{pmatrix} P_A \\ P_B \\ P_C \end{pmatrix} = \begin{pmatrix} E_2 - E_1 \\ E_3 - E_2 \\ E_4 - E_3 \\ \dots \\ \dots \\ \dots \\ \dots \end{pmatrix}$$

Given an overdetermined system $\mathbf{Ax} = \mathbf{b}$, a solution can be approached by solving a least squares problem:

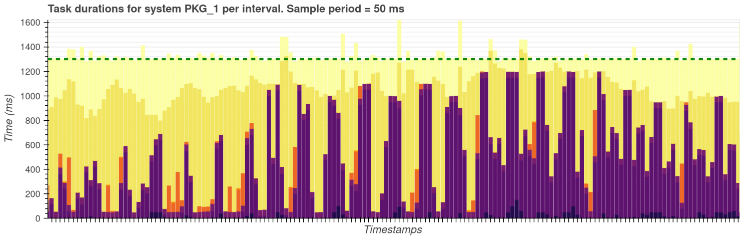
$$\min_{\mathbf{x}} \|\mathbf{Ax} - \mathbf{b}\|^2$$

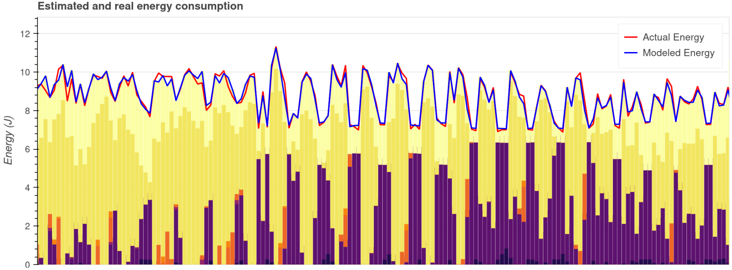
We use ridge regression to solve a regularized version of this problem.

Visualizing linear systems and solutions

Web application for visualization:

- top barchart: time distribution of tasks per measurement interval
- bottom barchart: modeled task energy distribution per interval and measured energy





Validation

Testing Environment

Experiments were run on Grid5000 and PlaFRIM testbeds.

Cluster	CPU	GPU	Memory
enbata	2 × AMD EPYC 9334	2 × AMD MI210 (64 GiB)	384 GiB
grouille	2 × AMD EPYC 7452	2 × Nvidia A100 (40 GiB)	128 GiB
suroit	2 × AMD EPYC 9224	–	256 GiB
paradoxe	2 × Intel Xeon Gold 5320	–	384 GiB

We test applications from dense and sparse linear algebra libraries:

- chameleon: LU solver (dgesv_nopiv), matrix inversion (dpoinv)
- qr_mumps: QR solver (dqrm_spgeqr)

Energy prediction accuracy: method

Each application is run 30 times on each node under three StarPU schedulers:

- dmdas: history-based model
- lws: local work stealing
- random: random scheduling

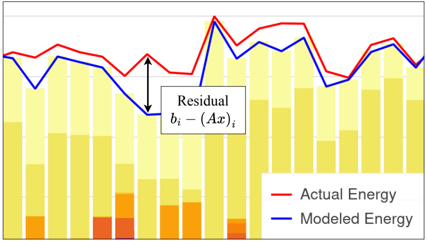
We assess prediction accuracy for one run with the mean absolute percentage error:

$$\mathrm{MAPE} = \frac{100}{m}\sum_{i=1}^{m}\left|\frac{b_i - (A\mathbf{x})_i}{b_i}\right|$$

b_i : measured energy of interval i

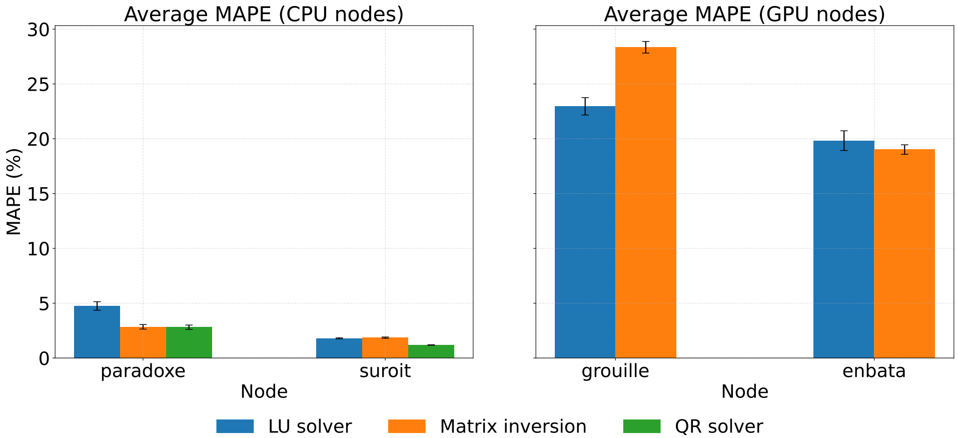
$(A\mathbf{x})_i$: predicted energy for interval i

m : number of intervals



Energy prediction accuracy - results

Model is accurate on CPU (95-99%) but not on GPU (72-81%).



Model portability - method

Can a model obtained from a single run be used to predict the energy consumption of other runs ?

Method:

- create a solution from a single run (training);
- apply the solution to the 29 other program runs (testing);
- compute accuracy metrics (MAPE).

Model portability - results

A model can be used to predict the energy consumption of other runs on CPU with high accuracy and high precision:

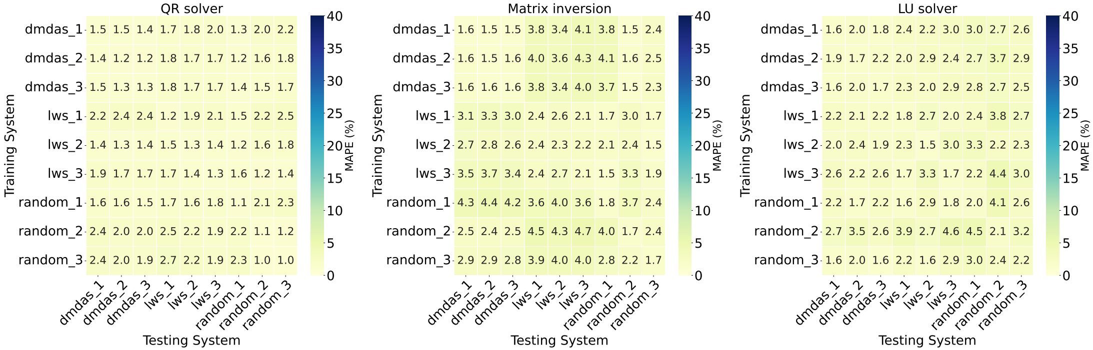


Figure 5: Model portability MAPE matrix for node suroit (AMD CPU)

Model portability - results

On GPU, results show low accuracy but high precision:

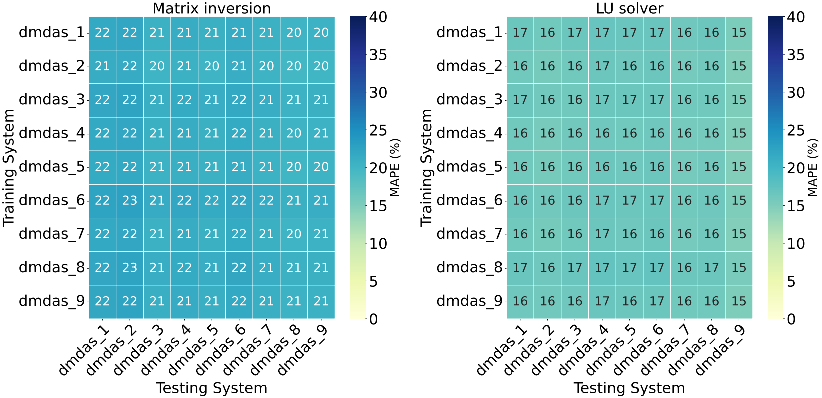


Figure 6: Model portability MAPE matrix for node enbata (AMD GPU)

GPU results overview — NVIDIA A100

Potential improvements:

- NVML energy counters may be erroneous due to subsampling.[1]
- Improve GPU kernel execution tracing.

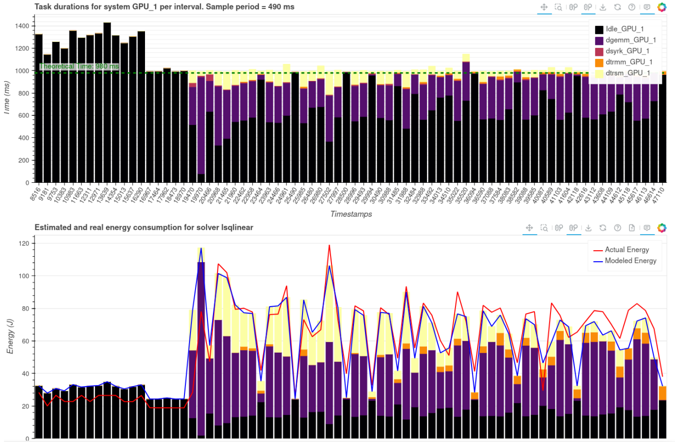


Figure 7: Chameleon matrix inversion run barcharts on grouille (A100).

[1] Z. Yang, K. Adamek, and W. Armour, Accurate and Convenient Energy Measurements for GPUs: A Detailed Study of NVIDIA GPUs Built-In Power Sensor, SC24, Nov. 2024.

GPU results overview - AMD MI210

We observe 2 phases on AMD GPUs runs:

1. Constant frequency, lower power consumption variability.
2. Highly variable frequency with large impact on power.

Need to account for variable frequency in the model.

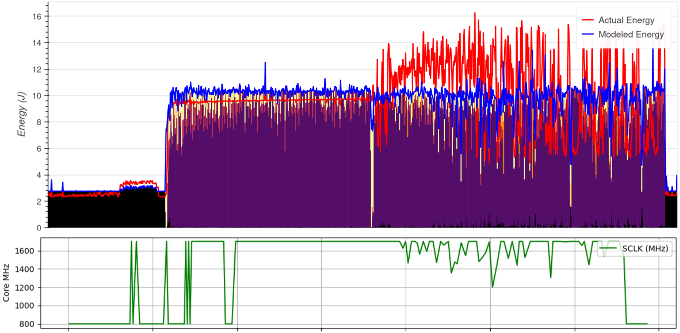


Figure 8: visualization of a chameleon matrix inversion run on enbata (MI210)

RÉPUBLIQUE FRANÇAISE
Liberté
Égalité
Fraternité

FRANCE 2030

PROGRAMME DE RECHERCHE
NUMÉRIQUE POUR L'EXASCALE

Conclusion

Conclusion

Fine-grain task energy consumption modelisation based on:

- Software power meters measurements.
- StarPU execution traces.
- Solving a least square problem.

Note: does not require additional hardware counters.

Achieves accurate and portable task energy consumption model on CPU (95-99 % accuracy), with room for improvements on GPU (72-81 % accuracy).

Future works

Improve GPU task power modeling.

Goal: implement energy aware scheduling policies to StarPU.

Numpex perspectives:

- On multi CPU, multi GPU nodes, powercap devices and create task energy profiles per device (Albert).
- Extend the methodology to other programming paradigm and trace models (Catherine).

Thank you

Paper data and results:
https://gitlab.com/JulesRisse/cluster-paper

Contributions:

energy-reader

energy-solver

Appendix - RAPL domains

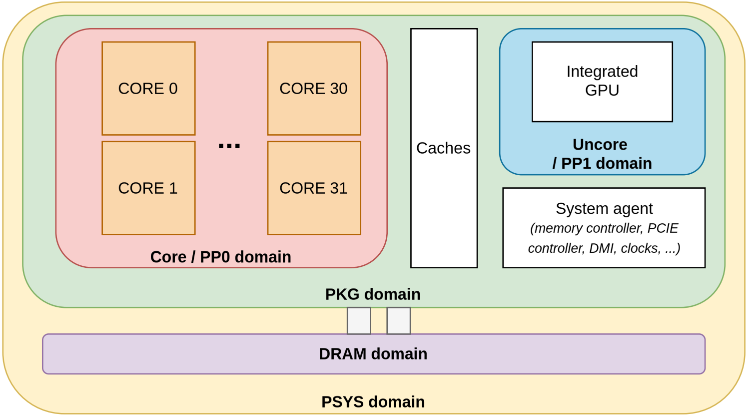
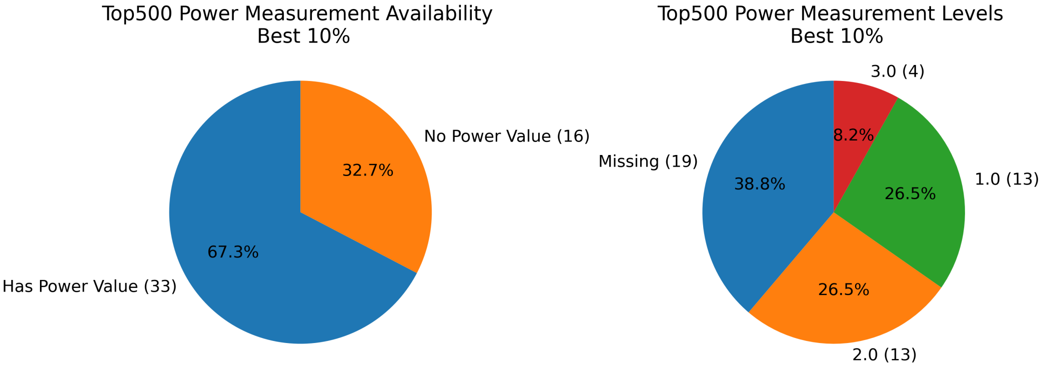


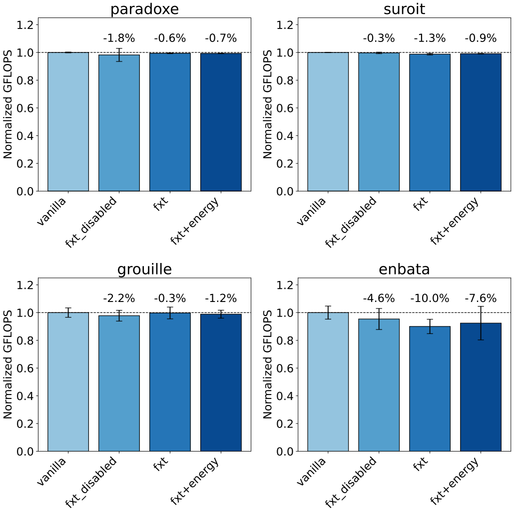
Figure 9: RAPL domains

Appendix - top500 power measurement levels

Method : https://www.top500.org/static/media/uploads/methodology-2.0rc1.pdf



Appendix - StarPU energy measurement overhead



Appendix - energy-reader supported vendors

NVIDIA

Interface :
Nvidia management library (**NVML**)

Scope :
GPU (compute + memory)

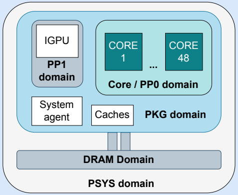
Granularity : model dependant
V100 = 20 ms
A100 = 100 ms
H100 = 100 ms

Latency : ~ 5 ms

intel

Interface :
Running Average Power Limit (**RAPL**)

Scope : RAPL domains



Granularity : 1 ms
Latency : ~ 5 μs

AMD

Interface :
ROCM System Management Interface (**ROCM-SMI**)

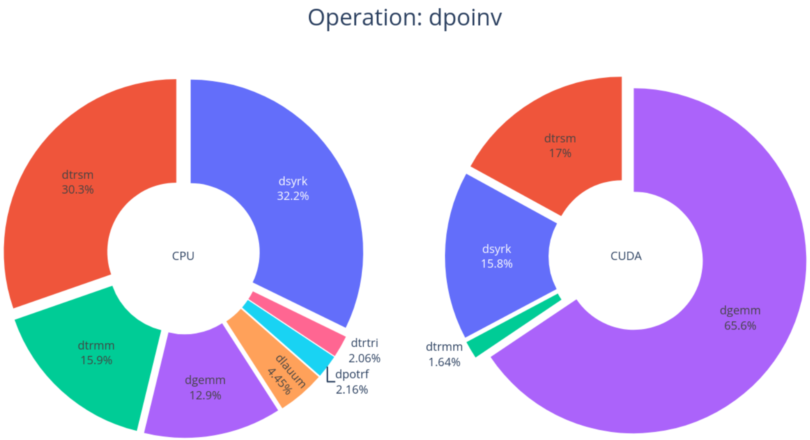
New library : AMD System Management Interface (**AMD-smi**) for CPU and GPU readings

Scope : GPU (compute + memory), CPU (with AMD-smi)

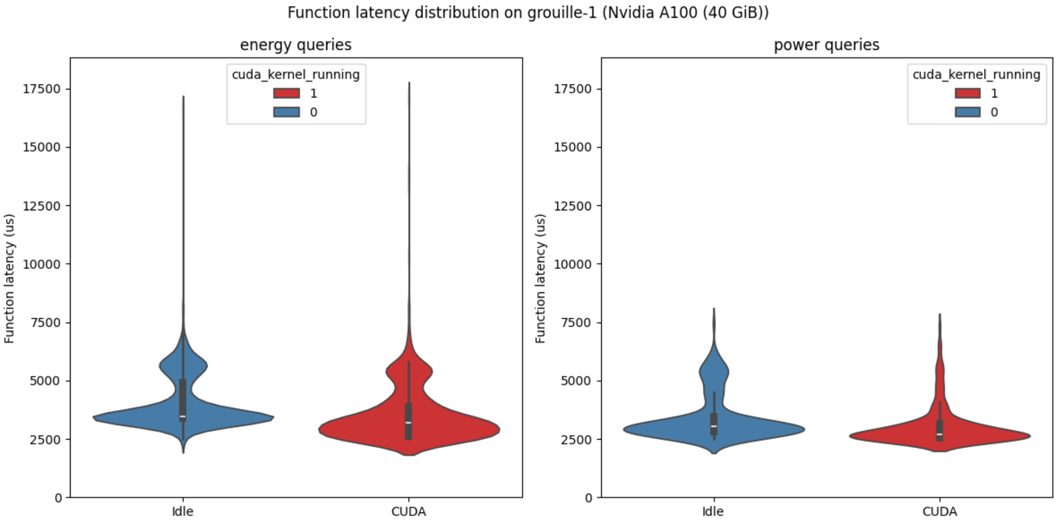
Granularity : 1 ms (MI210)

Latency : ~ 0.5 ms

Appendix - Chameleon kernel distribution



Appendix - NVML latency



Appendix - AMD GPU telemetry