RÉPUBLIQUE FRANÇAISE
Liberté
Égalité
Fraternité

FRANCE 2030
PROGRAMME DE RECHERCHE
NUMÉRIQUE POUR L'EXASCALE

Fine grain energy measurement

Exasoft General Assembly

Jules RISSE

November 8, 2024

Supervisors: François TRAHAY, Amina GUERMOUCHE

Table of contents

Introduction

Why does energy matter ?

Frontier, the fastest supercomputer in the world:

- 22,7 MW
- 40 000 french households
- 30M USD electricity bill

3% diminution = 1M USD savings

How do we use these architectures ?

Problem: Efficient use of heterogeneous multicore architectures is hard.

- Accelerator implementations (CUDA, HIP, FPGA);
- scheduling and data transfers;
- internode communications.

Solution: task based runtime systems

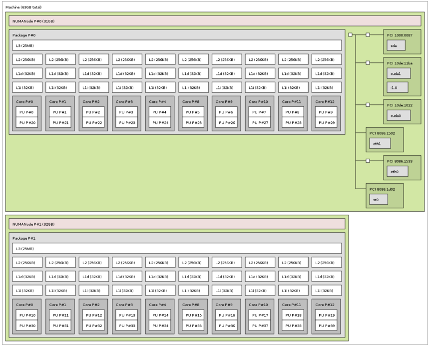


Figure 1: topology of a multicore system with accelerators

The StarPU runtime system

Task based runtime system for heterogeneous hardware.

- View HPC programm as a directed acyclic graph (DAG) of tasks;
- tasks can have multiple implementations (CPU, CUDA, FPGA);
- StarPU handles data transfers, scheduling and efficient task executions.

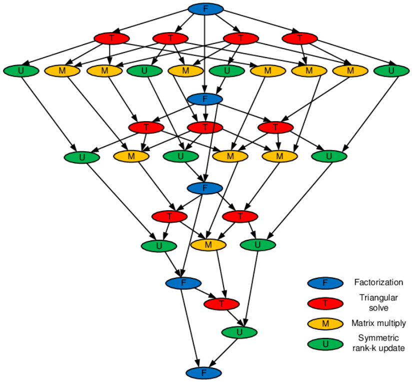
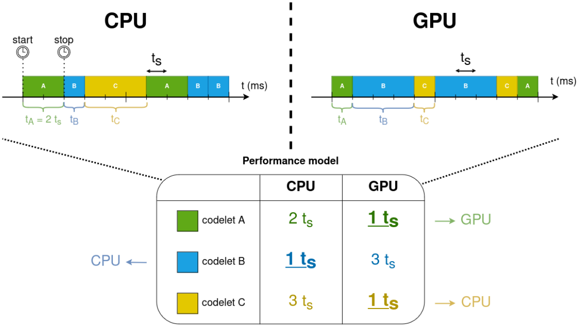


Figure 2: DAG of a 5x5 matrix Cholesky decomposition

Scheduling and performance models

Scheduling policies can use a history based performance model.

- Tasks time are measured during execution;
- they can then be distributed on the fastest processing unit.



Adding energy to the performance model

We quickly run into a granularity issue :

- Tasks execution time: < 1 ms
- Power meter granularity: 1 to 10's of ms

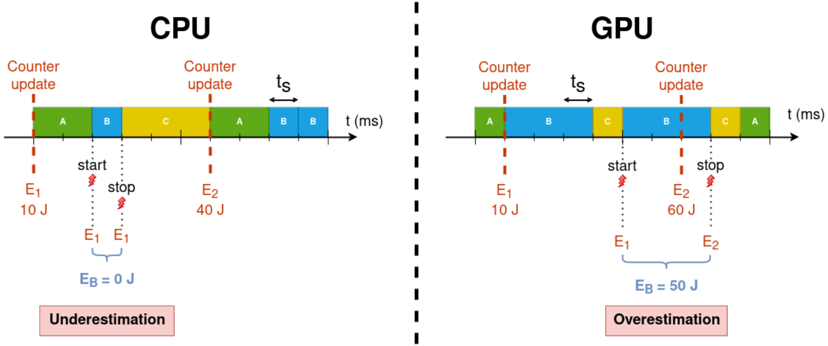


Figure 3: Issues raised by a naïve energy measuring approach

Proposed solution

We regularly measure energy and trace tasks during the StarPU program execution, then create an overdetermined linear system.

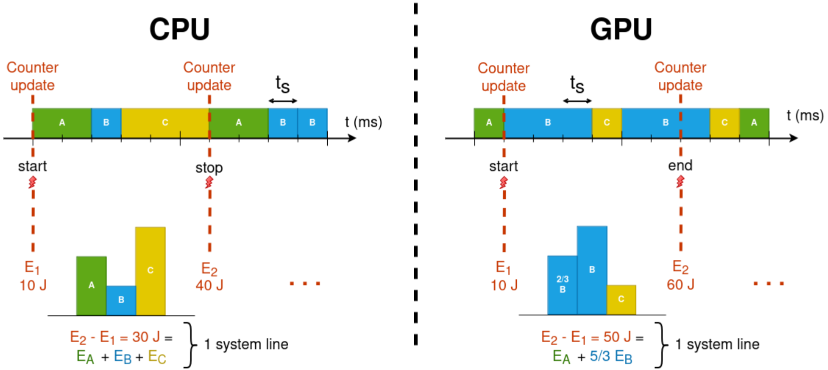


Figure 4: Basic equations linking energy and tasks

RÉPUBLIQUE FRANÇAISE
Liberté
Égalité
Fraternité

FRANCE 2030

PROGRAMME DE RECHERCHE
NUMÉRIQUE POUR L'EXASCALE

cea cnrs Inria

Measuring energy in StarPU

Querying energy counters

Contribution : energy-reader, a lightweight C library and API for querying energy values without elevated privileges.

Type	Method	Granularity	Scope	Cost
CPU	Running Average Power Limits (RAPL)	1 ms	Socket, Cores, RAM, IGPU	in μs
Nvidia GPU	Nvidia Management Library (NVML)	10 - 45 ms	All the GPU	in ms
AMD GPU	ROCm System Management Interface (ROCM-SMI)	x	All the GPU	x

PAPI can also be used with elevated privileges.

Adding energy measurements to StarPU

Using the existing probing mechanism (FxT) tracing tasks executions.
Addition of regular energy probes (RAPL: 25ms, NVML: 100ms).

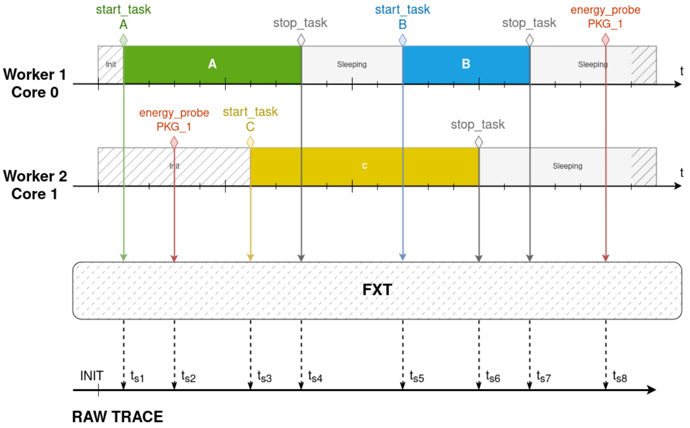


Figure 5: tasks execution and energy measurement probing with FxT

Solving the linear systems

Power system generation

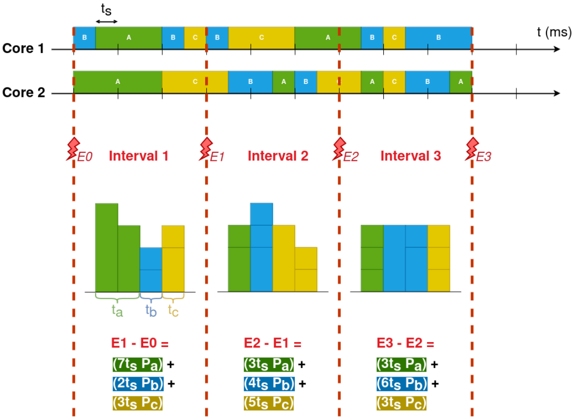


Figure 6: Linear system generation

$$\begin{pmatrix} 7t_s & 2t_s & 3t_s \\ 3t_s & 4t_s & 5t_s \\ 3t_s & 6t_s & 3t_s \end{pmatrix} \begin{pmatrix} P_a \\ P_b \\ P_c \end{pmatrix} = \begin{pmatrix} E_1 - E_0 \\ E_2 - E_1 \\ E_3 - E_2 \end{pmatrix} \tag{1}$$

Solving methods

With an overdetermined system $\mathbf{Ax} = \mathbf{b}$, the Ordinary Least Square (OLS) problem is the following :

$$\min_{\mathbf{x}} \|\mathbf{Ax} - \mathbf{b}\|,$$

the solution of which can be written with the normal equations:

$$\mathbf{x} = (\mathbf{A}^{\mathrm{T}}\mathbf{A})^{-1}\mathbf{A}^{\mathrm{T}}\mathbf{b}.$$

We use 2 methods :

- `StatsModels` OLS regression class
- `Scipy` Linear Least Square method

RÉPUBLIQUE FRANÇAISE
Liberté
Égalité
Fraternité

FRANCE 2030

PROGRAMME DE RECHERCHE
NUMÉRIQUE POUR L'EXASCALE

cea cnrs Inria

Validation

Test program

The test program is a general matrix solver which makes use of a variety of dense linear algebra operations:

- dgemm (matrix multiplication)
- dsyrk (symmetric rank-k update)
- dtrsm (triangular matrix solver)
- dpotrf (Cholesky factorization)

We use the chameleon library for efficient kernel implementations.

Testing environment

We use the grid5000 testbed to try various hardware configurations:

cluster	CPU	GPU	Memory
neowise	1 x AMD EPYC 7642	8 x AMD MI50 (32 GiB)	512 GiB
chifflot	2 x Intel Xeon Gold 6126	2 x Nvidia Tesla V100 (32 GiB)	192 GiB
sirius	2 x AMD EPYC 7742	8 x Nvidia A100 (40 GiB)	1.0 TiB

Result verifications

`chameleon_dtesting` allows batch execution of certain operations. Overall average power consumption is close that operation's consumption.-

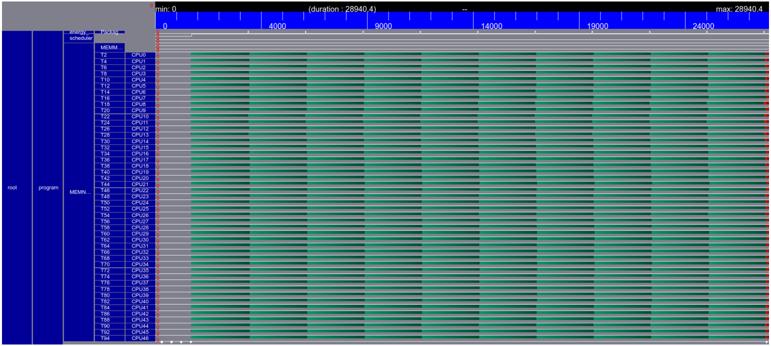
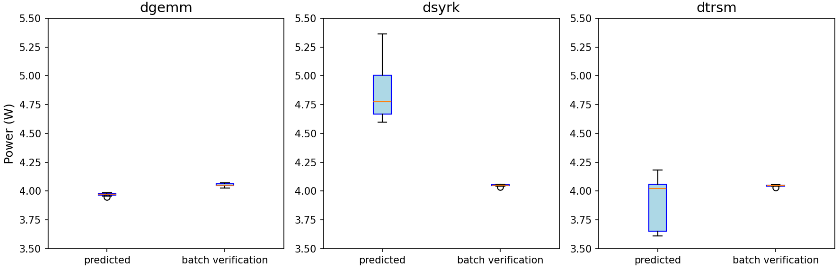


Figure 7: Trace generated by batch testing the dgemm operation on 1 CPU.

Preliminary results

CPU power consumption for dgemm, dsyrk, dtrsm on AMD EPYC 7642 (10 executions):



Results visualization

Saving the results

Every analyzed program execution is saved to a database and results can be viewed in a web application.

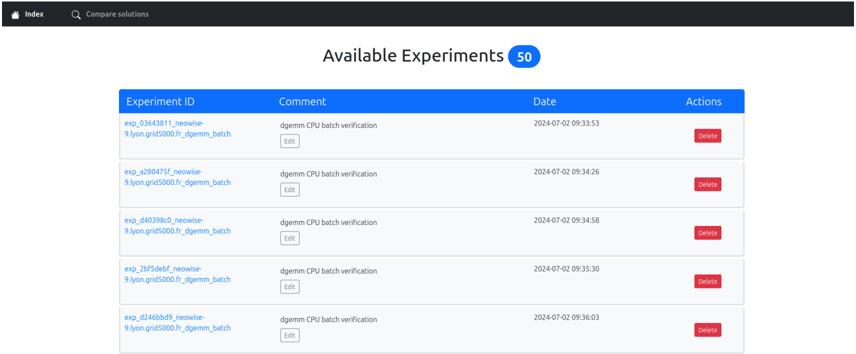
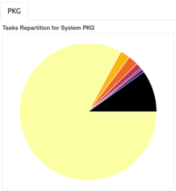


Figure 8: web application dashboard

Visualizing the system



- Fully interactive;
- Each bar is a measurement interval.

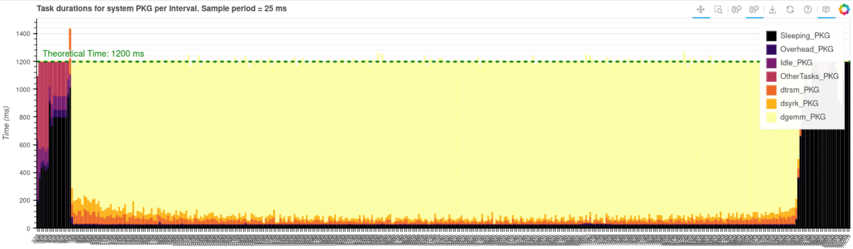


Figure 9: cumulated tasks time pie chart (top) and tasks durations barchart (bottom)

Visualizing predicted and actual tasks energy

We can also select the solver used for prediction.

Figure 10: energy consumption barchart

Visualizing validation metrics for the prediction

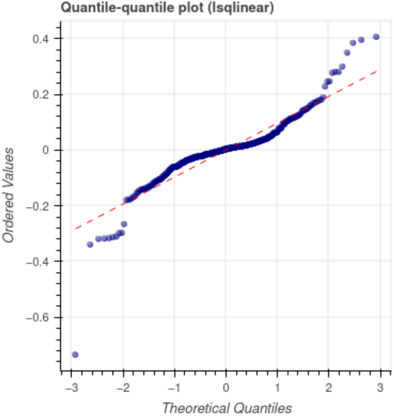


Figure 11: quartile-quartile plot

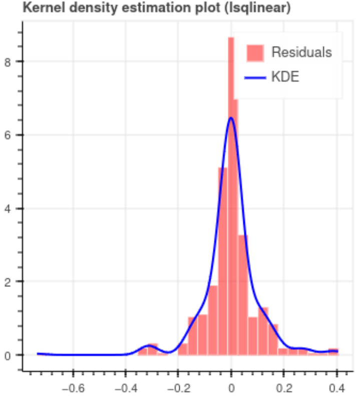


Figure 12: kernel density estimate plot

Comparing a solution with subsequent program runs

System	RMSD	R^2	Residual norm	Residual %
exp_a67620b7_neowise-3.lyon.grid5000.fr_PKG	0.10604092795265399	0.8431039178063354	2.1234679274302786	2.2846347764765977
exp_6f42181f_neowise-3.lyon.grid5000.fr_PKG	0.1121734082792117	0.8226041863090511	2.2462707502691797	2.412166051002955
exp_f989bdc0_neowise-3.lyon.grid5000.fr_PKG	0.1058275393334963	0.8229234846044859	2.1191948236616813	2.272835700050623
exp_e50c391d_neowise-3.lyon.grid5000.fr_PKG	0.08316890452557986	0.8786447194103184	1.6633780905115974	1.7849499280877967
exp_d09c7821_neowise-3.lyon.grid5000.fr_PKG	0.07824147679255933	0.8801645978778135	1.562872274877752	1.6763299852234281
exp_04aaf4c2_neowise-3.lyon.grid5000.fr_PKG	0.07596074768670694	0.9012787399823325	1.5192149537341386	1.6286681535683392
Average	0.094	0.858	1.872	2.010
Standard Deviation	0.015	0.030	0.297	0.320

Figure 13: comparator interface

Conclusion

Conclusion

- Promising results for most used tasks, but solution is not stable nor accurate for less used ones;
- need to try more use cases / hardware + tweaking parameters;
- for Nvidia GPU, high NVML overhead disturbs computations;
- on CPU, tasks tend to have the same power consumption.

Future works

Possible improvements:

- group similar tasks/states in the linear system;
- add RAM power costs;
- use external powermetters (grid5000)
- add more relevant data (p-state, c-state, temperatures);
- explore low consumption core setups.

-> Thesis continuation.

Appendix - RAPL domains

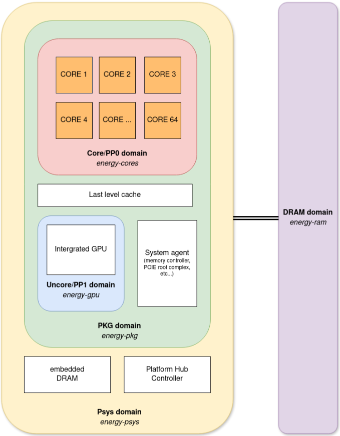


Figure 14: RAPL domains

Appendix - NVML latency

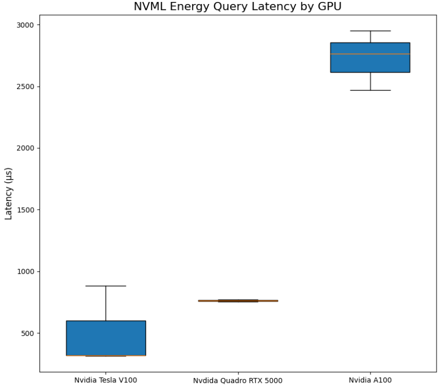


Figure 15: NVML latency boxplot - no extreme values

Appendix - NVML latency 2

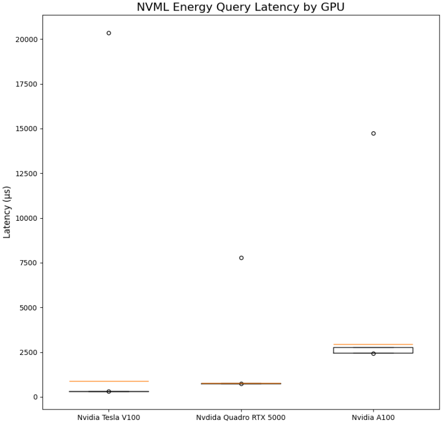


Figure 16: NVML latency boxplot - with extreme values

Appendix - Model applicability

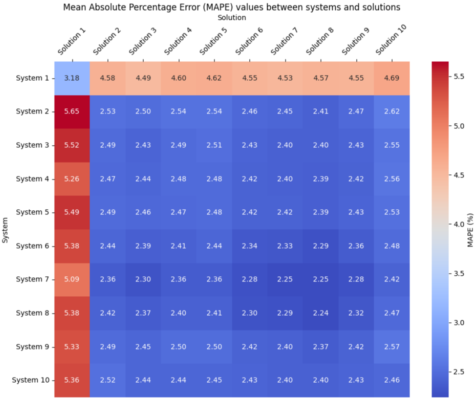


Figure 17: heatmap of mean absolute percentage error using the solution from a program run analysis on subsequent program executions.