CWI
Centrum Wiskunde & Informatica

netherlands eScience center

SC22 Dallas, TX | hpc accelerates.

ASTRON
Netherlands Institute for Radio Astronomy

Going green: optimizing GPUs for energy efficiency through model-steered auto-tuning

Richard Schoonhoven, Bram Veenboer, Ben van Werkhoven, Kees Joost Batenburg

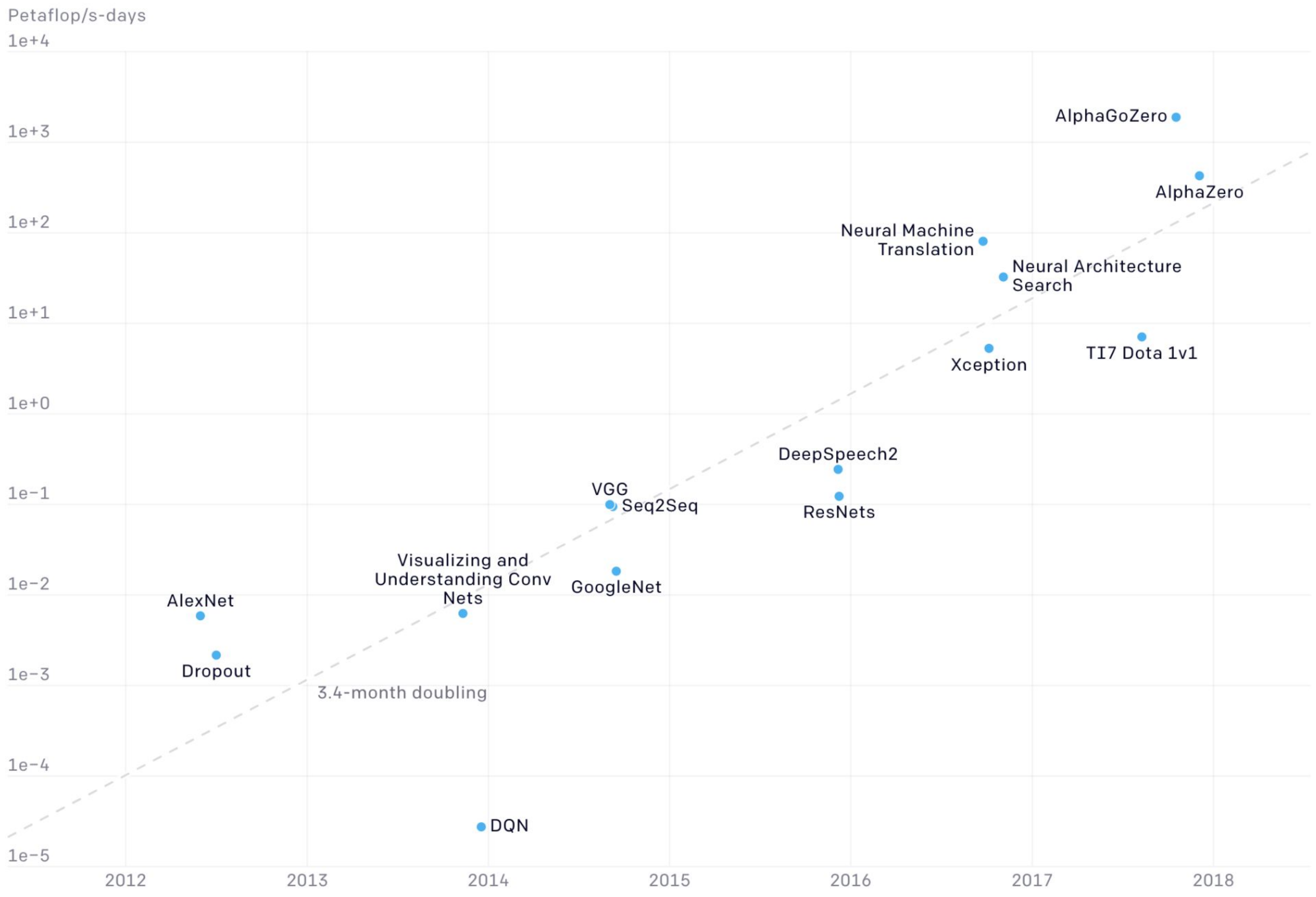
13th IEEE International Workshop on Performance Modeling, Benchmarking and Simulation of High Performance Computer Systems
SC22, Dallas, Texas US
November 14th 2022

richard.schoonhoven@cwi.nl

GPUs consume increasingly more energy

SUMMIT supercomputer: 8.3 out of 13 MW are consumed by GPUs[1].

Computational demands in deep learning have risen 300,000x from 2012 to 2018[2].



1: Stachowski, M., Fiebig, A., & Rauber, T. (2021). Autotuning based on frequency scaling toward energy efficiency of blockchain algorithms on graphics processing units. *The Journal of Supercomputing, 77*(1), 263-291.

2: Schwartz, R., Dodge, J., Smith, N. A., & Etzioni, O. (2020). Green ai. Communications of the ACM, 63(12), 54-63.

Optimizing GPU kernels

Optimizing GPU applications is complex → choosing between many code implementations and parameters

Generic GPU Auto-tuners:

- **Kernel Tuner**[1]
- Kernel Tuning Toolkit[2]
- Auto-Tuning Framework[3]
- CLTune[4]

1: https://github.com/KernelTuner/kernel_tuner

2: https://github.com/HiPerCoRe/KTT

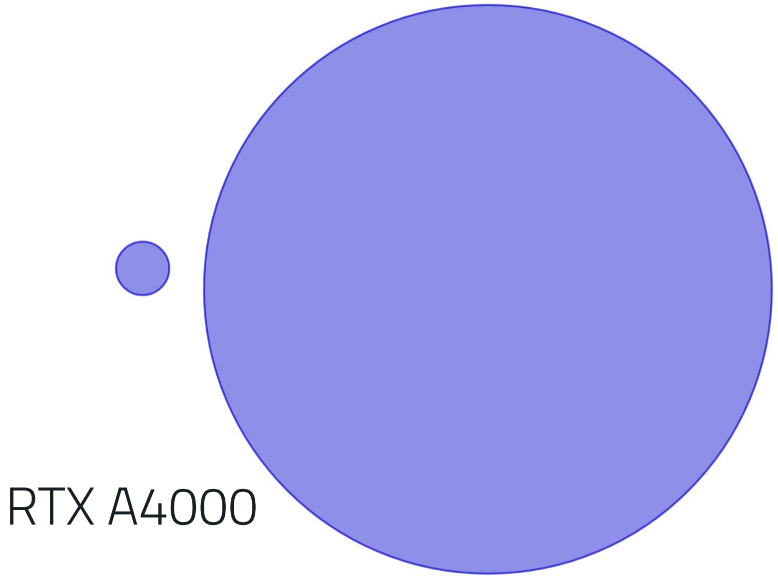
3: Rasch, A., Schulze, R., Steuwer, M., & Gorlatch, S. (2021). Efficient auto-tuning of parallel programs with interdependent tuning parameters via auto-tuning framework (ATF). ACM Transactions on Architecture and Code Optimization (TACO), 18(1), 1-26.

4: https://github.com/CNugteren/CLTune

Hurdles when tuning for energy efficiency

We can additionally tune the core clock frequency, or power limit.

Problem: combinatorially increases the size of the search space.



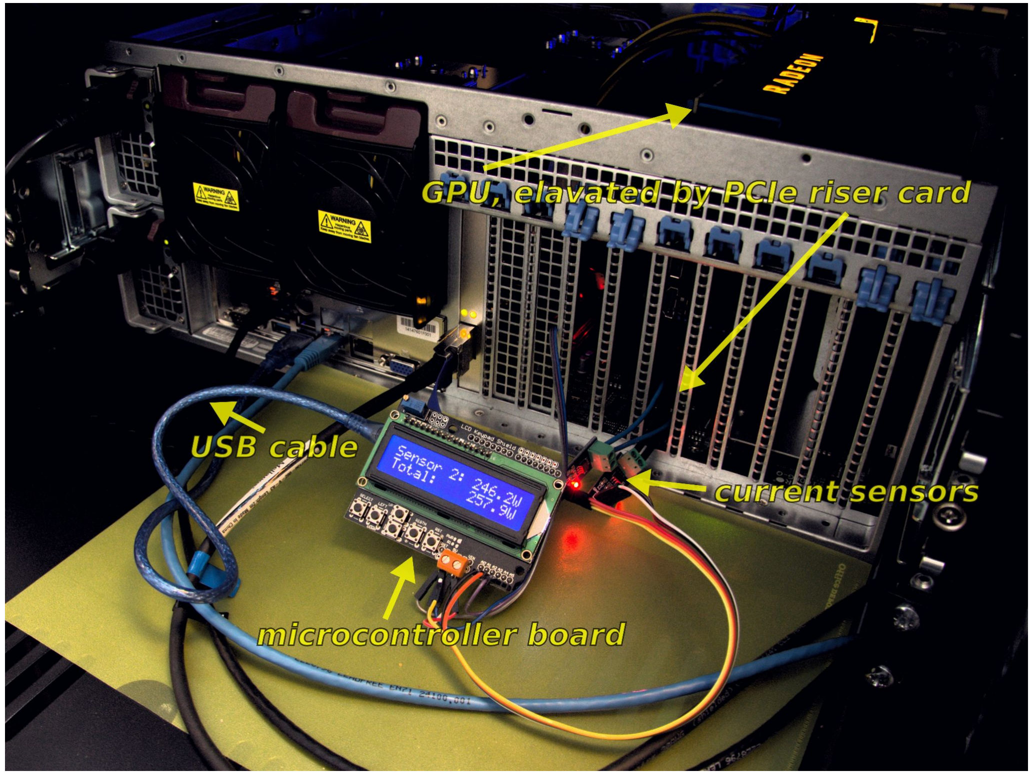
Can we use auto-tuners to improve energy efficiency?

Thus far generic auto-tuners do not have support for measuring power, and optimizing for energy efficiency.

Support for measuring energy

We extended Kernel Tuner with functionality for auto-tuning energy efficiency.

- **Internal:** Using NVIDIA Management Library (NVML)
- **External:** Using PowerSensor2[1]

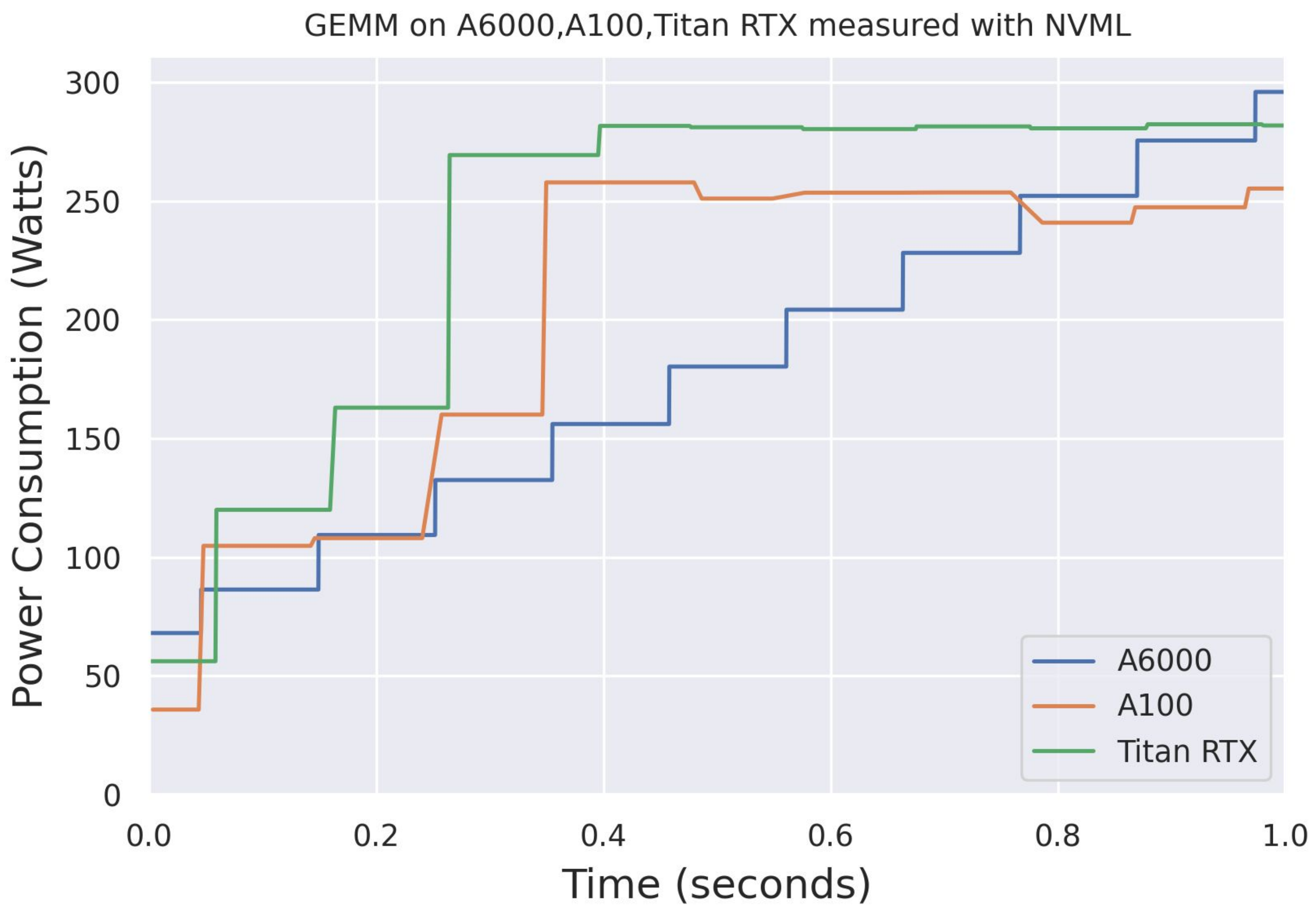


1: Romein, J. W., & Veenboer, B. (2018, April). PowerSensor 2: a fast power measurement tool. In 2018 IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS) (pp. 111-113).

Power measurements with NVML

It can take up to 1 second for NVML power measurements to stabilize.

Kernel Tuner will measure long enough to acquire reliable measurements.



Can we use auto-tuners to improve energy efficiency?

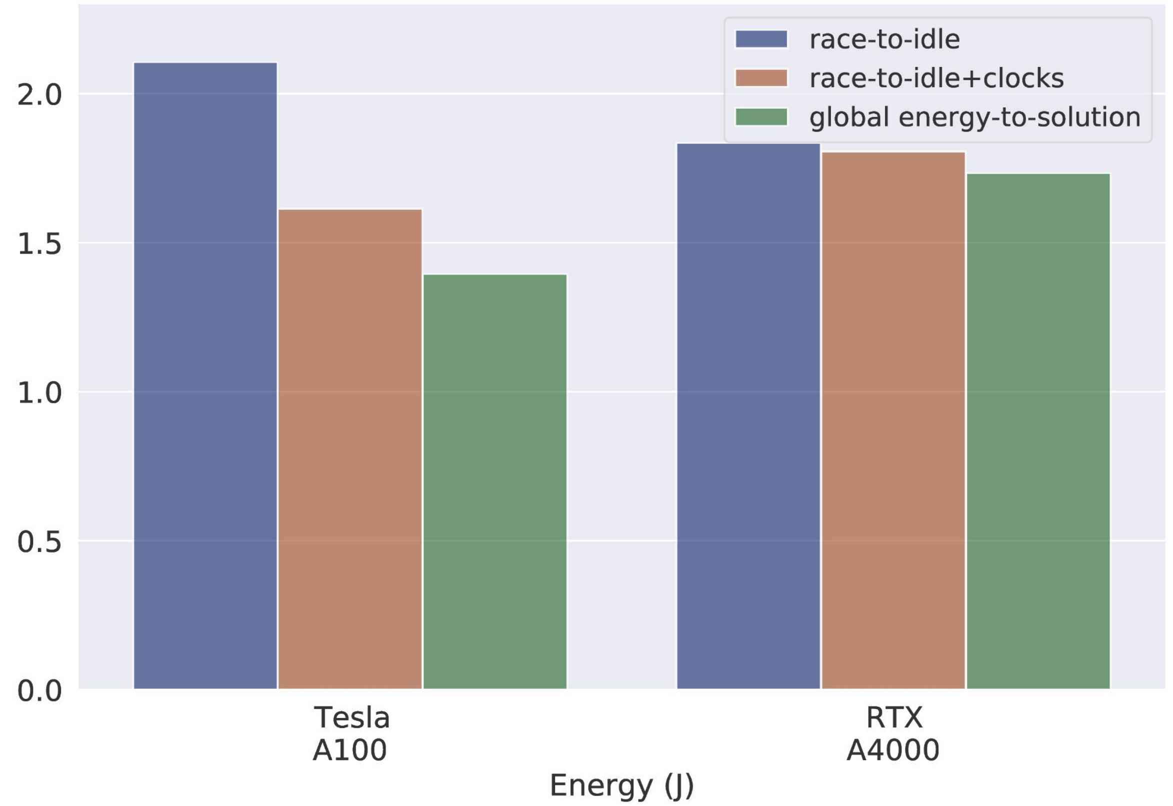
Thus far generic auto-tuners do not have support for measuring power, and optimizing for energy efficiency.

Is this a different optimization problem than tuning for compute performance?

Is tuning for energy efficiency different?

GEMM (matrix multiplication) energy of configuration for:

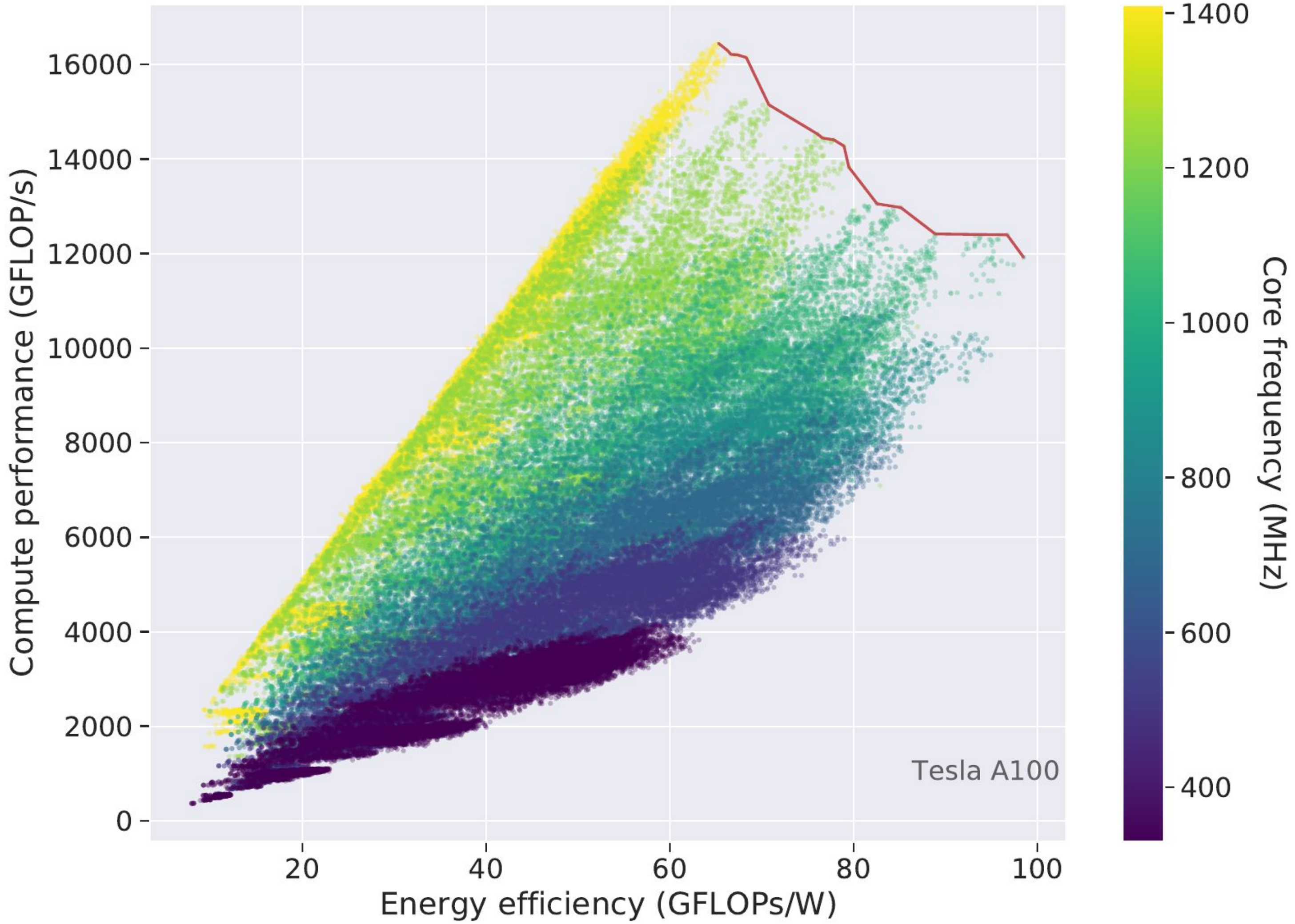
1. race-to-idle,
2. race-to-idle + clock frequency,
3. energy-to-solution.



Energy efficiency vs compute performance

Tesla A100

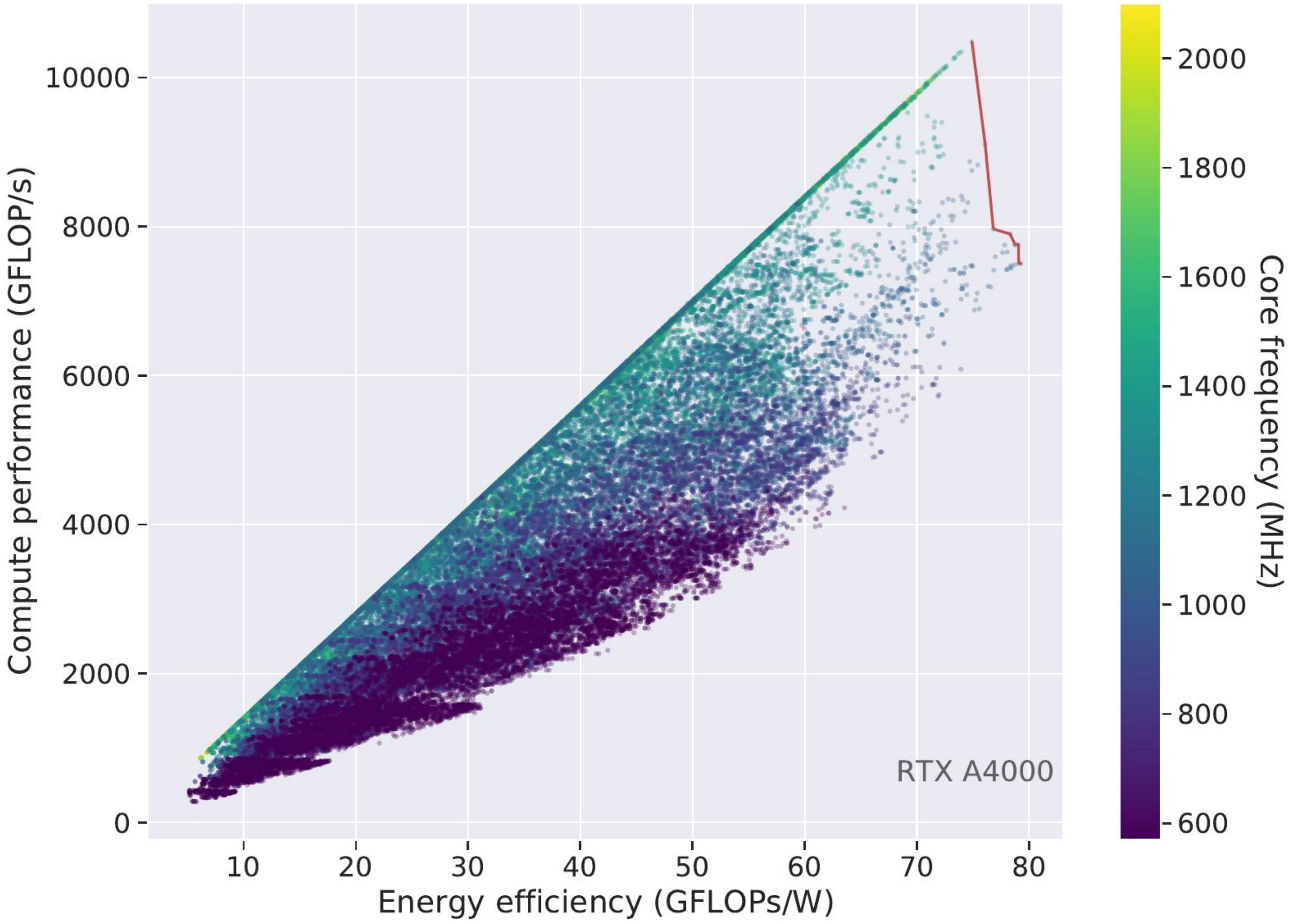
A speed reduction of 27.5% leads to an increase in energy efficiency of 50.9%.



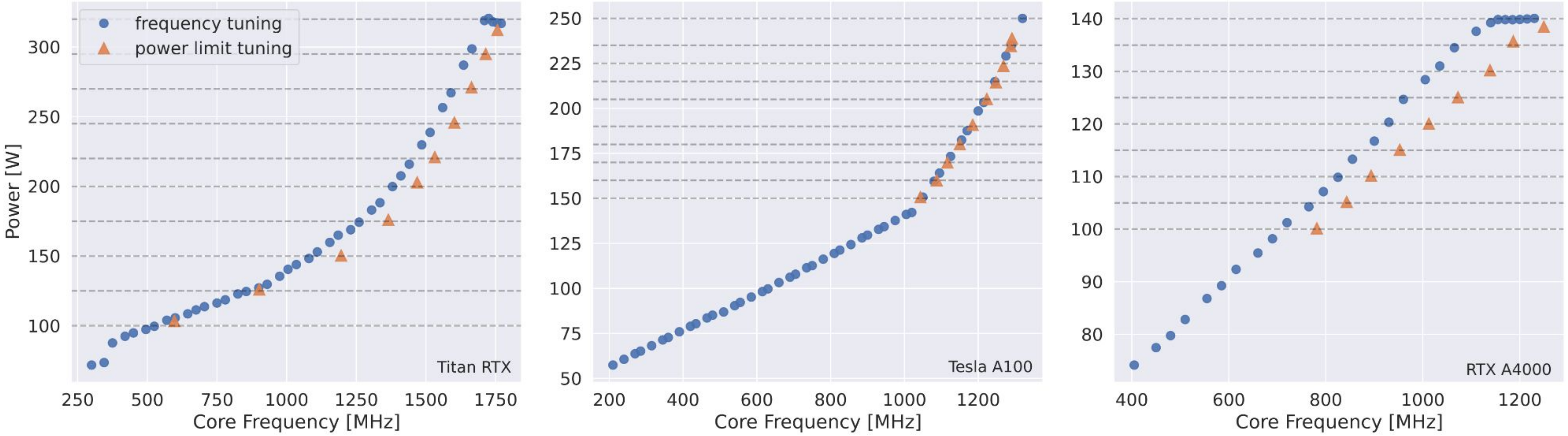
Energy efficiency vs compute performance

RTX A4000

The optimal configuration for performance is close to the optimal configuration for energy use.



Power capping or frequency tuning



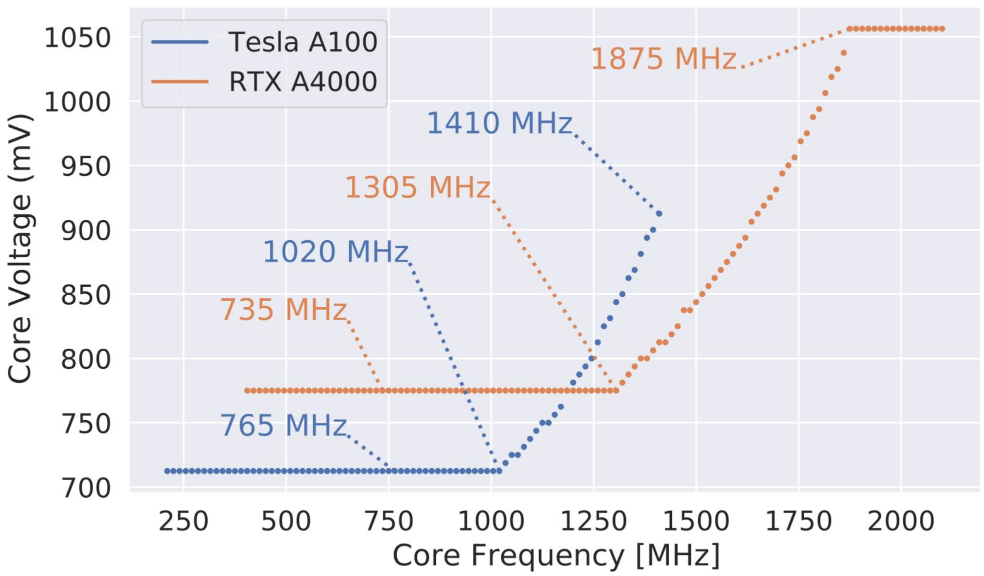
Clock frequency tuning is more predictable for modelling and cover a larger range.

Potentially frequency tuning can result in a more energy efficient configuration at the cost of a large search space.

Power consumption model

The power and energy consumption of a GPU can be modeled[1] as

$$E = \int_{t_0}^{t_1} P(t)\,dt, \qquad P_{gpu} = P_{static} + N_c C f V^2.$$



1: Price, D. C., Clark, M. A., Barsdell, B. R., Babich, R., & Greenhill, L. J. (2016). Optimizing performance-per-watt on GPUs in high performance computing. Computer Science-Research and Development, 31(4), 185-193.

Fitting a model

Fit the model for a synthetic benchmark kernel saturating the computational units

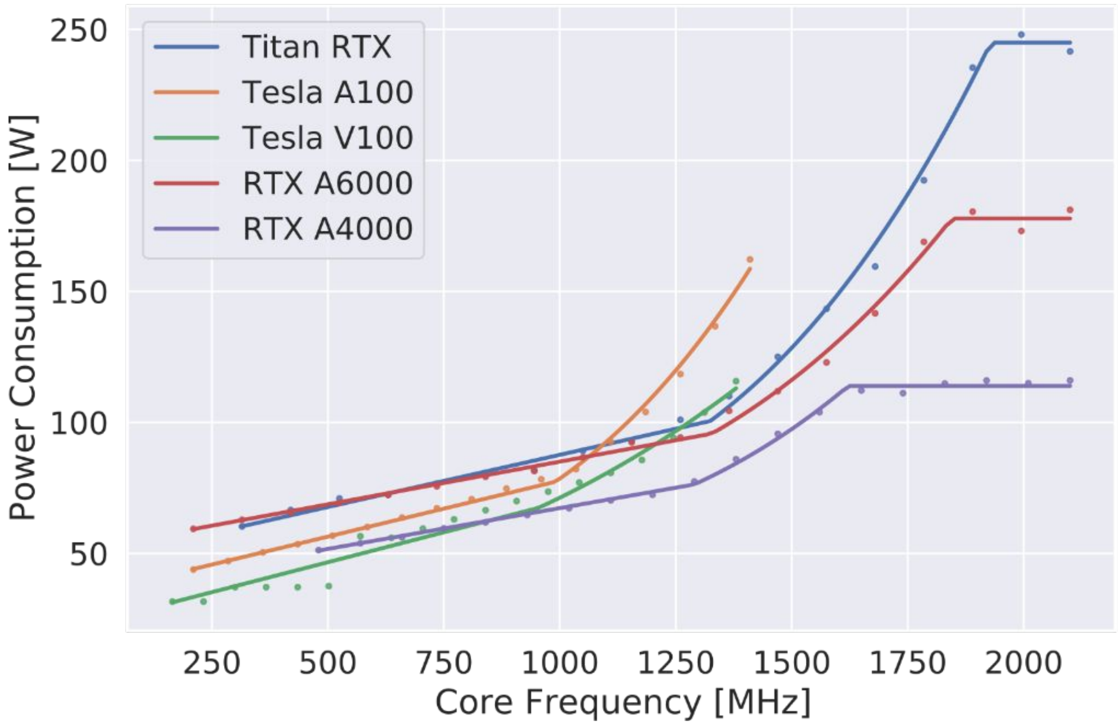
$$P^*_{load} = \min(P_{max}, P^*_{static} + \alpha f v^2)$$

Not all GPUs support voltage readings, so we substitute

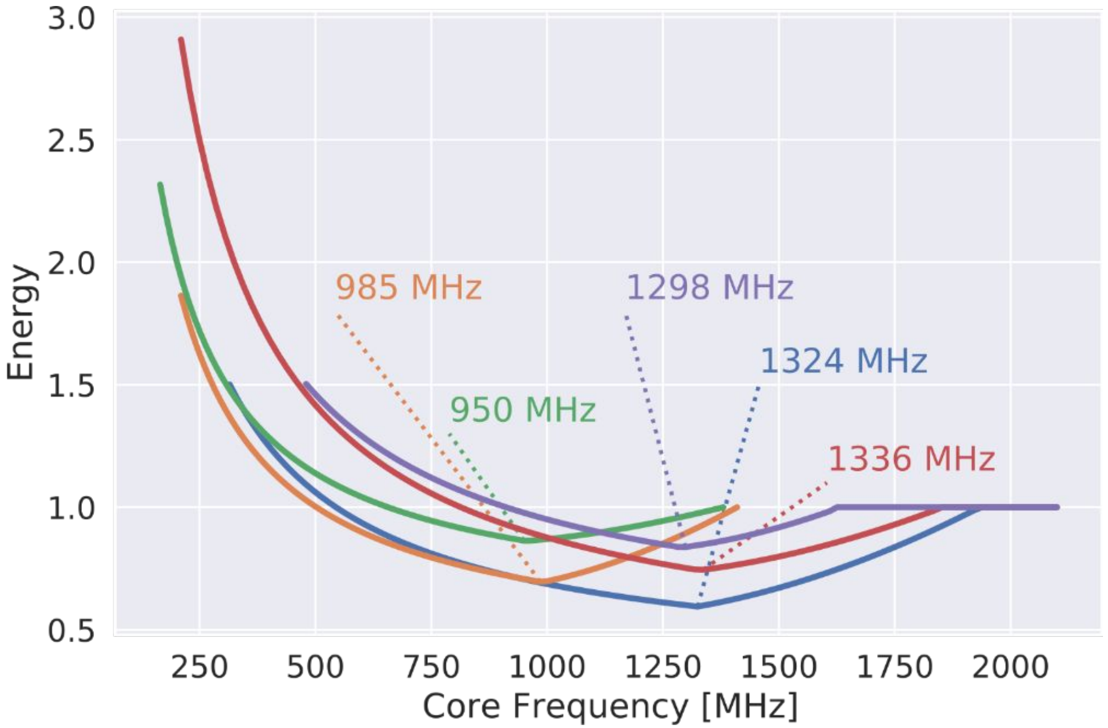
$$v(f) = \begin{cases} 1 & f < \tau \\ 1 + \beta \cdot (f - \tau) & f \geq \tau \end{cases}$$

Experimental results

GPU power consumption measurements (dots), and fitted model (lines).



Estimated energy usage with optimal core frequency.



Optimal frequency for energy efficiency

Energy is proportional to

$$E \propto \frac{P}{f} = \frac{P^*_{static}}{f} + \alpha v^2$$

and has an optimal (minimal) frequency at ridge point.

$$v(f) = \begin{cases} 1 & f < \tau \\ 1 + \beta \cdot (f - \tau) & f \geq \tau \end{cases}$$

Strategy for auto-tuning kernels

Model reduces frequency to one optimal value for kernel that fully loads GPU.

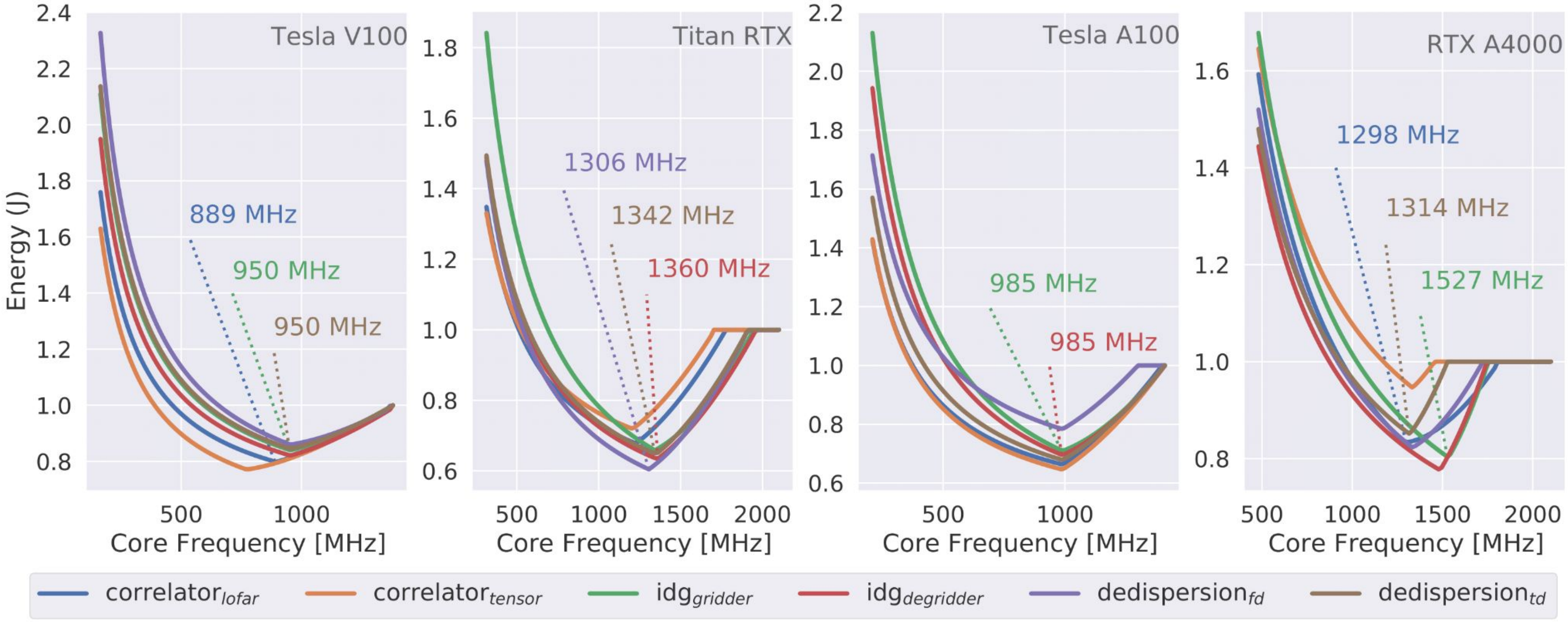
Strategy: Run auto-tuner for ±10% around most energy efficient frequency.

Reduce size of search space by 80%.

Low-Frequency Array (LOFAR)[1]

1: van Haarlem, M. P., Wise, M. W., Gunst, A. W., Heald, G., McKean, J. P., Hessels, J. W., ... & Reitsma, J. (2013). LOFAR: The low-frequency array. Astronomy & astrophysics, 556, A2.

LOFAR: Experimental results



Mean energy efficiency +42.0 ± 24.1%.

Mean runtime -24.3 ± 12.1%.

LOFAR: Experimental results

GPU	Kernel	GOPs/W (before)	GOPs/W (after)	GOPs/W gained	TOP/s (before)	TOP/s (after)	TOP/s gained	Tuned frequency
Tesla A100	Gridder	64.7	102.6	58.6%	16.3	12.0	-26.5%	1035 MHz
	Degridder	59.8	97.5	63.1%	14.5	10.7	-26.2%	1035 MHz
	FD Dedispersion	62.2	92.8	49.1%	9.7	7.3	-24.6%	1035 MHz
	TD Dedispersion	13.3	21.5	61.3%	3.4	2.5	-26.4 %	1035 MHz
	Tensor-Core Correlator	684.8	1264.2	84.6%	148.4	135.2	-8.9%	1035 MHz
	LOFAR Correlator	58.9	125.8	113.8%	12.2	10.7	-12.0%	1035 MHz
RTX A4000	Gridder	77.6	107.5	38.6%	11.0	8.1	-25.8%	1200 MHz
	Degridder	90.8	131.6	44.9%	10.2	9.4	-8.1%	1470 MHz
	FD Dedispersion	77.6	111.9	44.3%	8.3	6.7	-19.2%	1290 MHz
	TD Dedispersion	12.9	17.2	33.0%	1.5	1.1	-22.2%	1200 MHz
	Tensor-Core Correlator	571.2	606.8	6.2%	57.2	55.2	-3.6%	1290 MHz
	LOFAR Correlator	98.9	119.3	20.6%	8.7	8.4	-4.2%	1470 MHz
TITAN RTX	Gridder	55.2	68.6	24.2%	14.3	9.0	-37.2%	1260 MHz
	Degridder	48.4	65.6	35.4%	13.7	8.2	-39.7%	1155 MHz
	FD Dedispersion	39.9	59.9	50.2%	10.2	5.5	-45.4%	1050 MHz
	TD Dedispersion	8.0	12.1	50.7%	2.1	1.3	-40.0%	1050 MHz
	Tensor-Core Correlator	140.5	209.5	49.1%	34.7	23.4	-32.6%	1155 MHz
	LOFAR Correlator	51.5	78.0	51.6%	12.8	7.2	-43.4%	1155 MHz
Tesla V100	Gridder	59.6	73.6	23.6%	11.6	9.5	-18.0%	1110 MHz
	Degridder	61.7	74.2	20.2%	11.0	8.8	-19.9%	1110 MHz
	FD Dedispersion	58.6	69.2	18.1%	7.4	6.0	-19.2%	1110 MHz
	TD Dedispersion	11.6	15.7	34.9%	2.2	1.3	-37.8%	1110 MHz
	Tensor-Core Correlator	260.8	301.5	15.6%	34.2	27.7	-18.9%	1110 MHz
	LOFAR Correlator	74.7	86.8	16.3%	9.9	7.6	-23.5%	1110 MHz

Future work

1. Extend to other manufacturers.
2. Add memory term to power consumption model.
3. System level analysis of impact on performance and energy efficiency.

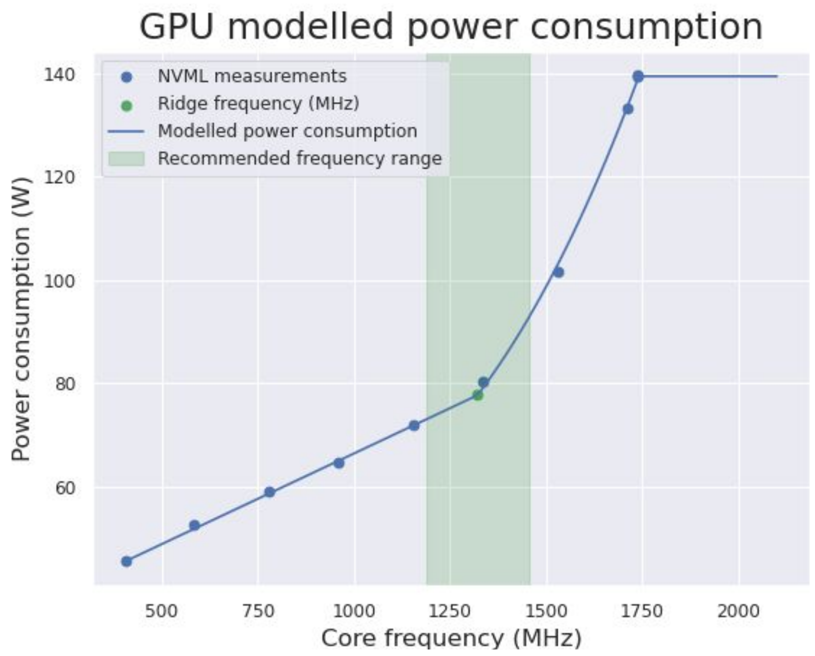
Thank you and try the code!

Try the code with Kernel Tuner:

pip install kernel_tuner[cuda]

Run the Kernel Tuner example (requires rights to set clock frequencies):

examples/cuda/going_green_performance_model.py



Feel free to contact me at:

richard.schoonhoven@cwi.nl