Auto-tuning GPU code for energy efficiency

Ben van Werkhoven
Netherlands eScience Center

Workshop EREC - Energy and resource efficiency of data centers
HPC Asia 2023, February 27, 2023

The World's Fastest Supercomputer: Frontier

Number 1 in TOP500 list (Nov 2022)

- Peak compute performance:
  - 1.6 ExaFLOPS = 1.6×10^{18} floating point operations per second

- 9472 nodes with:
  - 1 CPU (AMD EPYC 64C)
  - 4 GPUs (AMD Instinct MI250X)

Seven of out ten systems in top 10 use GPUs

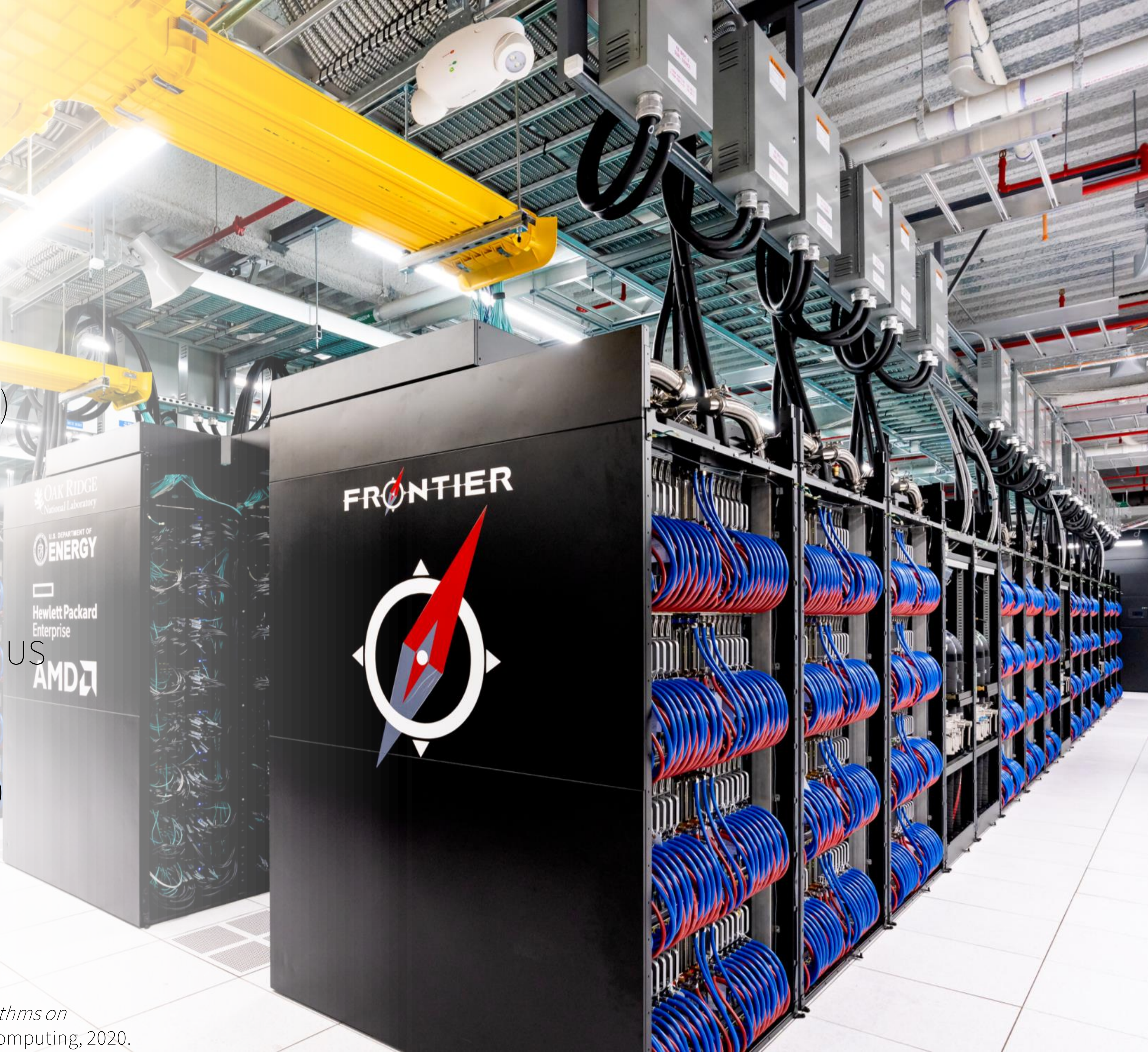
Energy cost of supercomputers

Frontier: #1 in TOP500 list (Nov 2022)

- 20 Megawatt continuously
- $40 million annual electricity bill
- 100,000 metric tons of CO_2 annually
- ~20,000 cars on the road for a year in US

Summit: (#5, Frontier's predecessor)

- 64% of energy is consumed by GPUs



Efficient Computation through Tuned Approximation
David Keyes, SIAG/SC Supercomputing Spotlights 2022
Autotuning based on frequency scaling toward energy efficiency of blockchain algorithms on graphics processing units M. Stachowski, A. Fiebig, and T. Rauber, Journal of Supercomputing, 2020.

Optimizing GPU applications

To maximize GPU code performance, you need to find the best combination of:

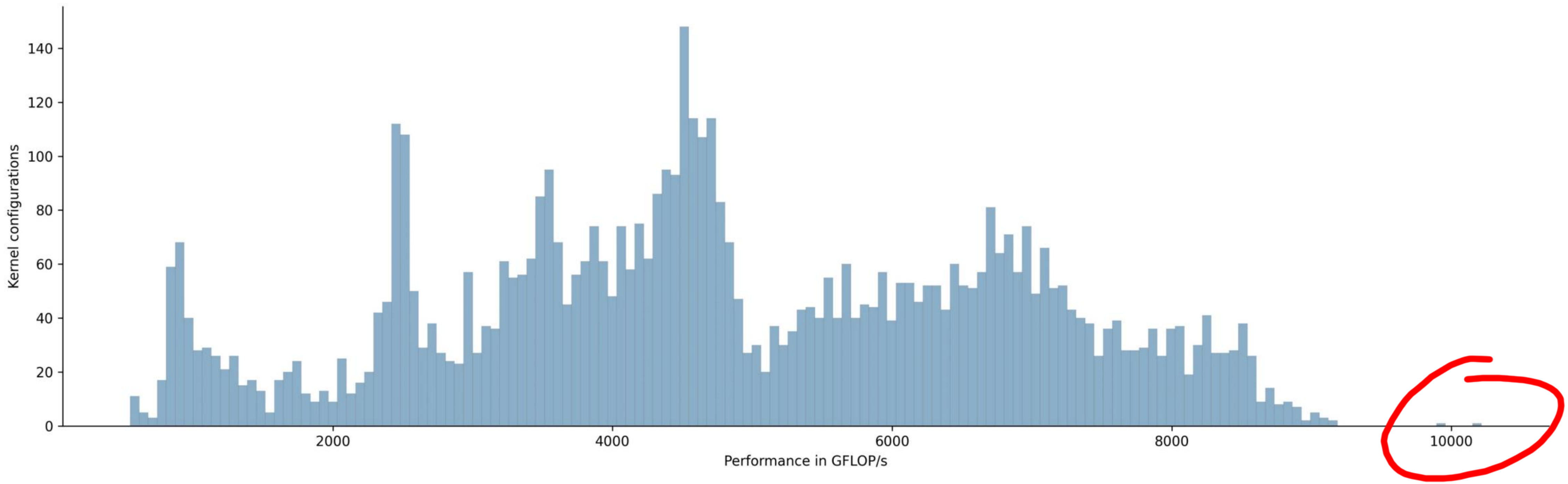
- Different mappings of the problem to threads and thread blocks
- Different data layouts in different memories (shared, constant, ...)
- Different ways of exploiting special hardware features
- Thread block dimensions
- Code optimizations that may be applied or not
- Work per thread in each dimension
- Loop unrolling factors
- Overlapping computation and communication
- ...

Problem:

- Creates a very large design space

Large search space of kernel configurations

Exploring different designs of a Convolution kernel on Nvidia A100



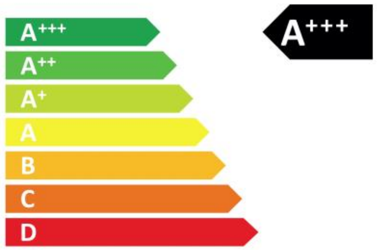
Kernel Tuner – *A tool for auto-tuning GPU kernels*

Supports
- Tuning CUDA, OpenCL, C, Fortran functions
- Tuning both host and device code
- Output checking during tuning
- User-defined metrics and tuning objectives
- 20+ search optimization algorithms
- Extensive documentation and examples
- …

Growing ecosystem:
- Kernel Launcher (library for integrating tuned kernels into C++ applications)
- KTDashboard (live visualizations of tuning runs)
- Kernel Tuner tutorial
- …

https://github.com/KernelTuner/kernel_tuner

Optimizing Energy Efficiency



Minimize energy consumption instead of the execution time

Using Kernel Tuner for energy tuning, challenges:

- How to measure power consumption during tuning?
- Is there a difference between optimizing for time or energy?
- How to effective are optimization algorithms to optimize for energy?
- Can we narrow down the search space?
- Which methods to support? Frequency tuning or power capping?
- How much energy can be gained in practice?

Going Green: optimizing GPUs for energy efficiency through model-steered auto-tuning
Richard Schoonhoven, Bram Veenboer, Ben van Werkhoven, K. Joost Batenburg
PMBS workshop at SC22 2022

Extending Kernel Tuner: Observers

- Observers extend the benchmarking capabilities of Kernel Tuner
- Allows users to subscribe to certain events during benchmarking
- Also used internally for measuring time
- Kernel Tuner implements observers for measuring power consumption:
  - PowerSensorObserver
  - NVMLObserver

class **kernel_tuner.observers.BenchmarkObserver**

Base class for Benchmark Observers

after_finish()

after finish is called once every iteration after the kernel has finished execution

after_start()

after start is called every iteration directly after the kernel was launched

before_start()

before start is called every iteration before the kernel starts

during()

during is called as often as possible while the kernel is running

abstract **get_results**()

get_results should return a dict with results that adds to the benchmarking data

get_results is called only once per benchmarking of a single kernel configuration and generally returns averaged values over multiple iterations.

register_device(*dev*)

Sets self.dev, for inspection by the observer at various points during benchmarking

More information:
https://kerneltuner.github.io/kernel_tuner/stable/observers.html

Nvidia Management Library (NVML)

Observe GPU temperature, core and memory clocks, core voltage, and power

Advantages:

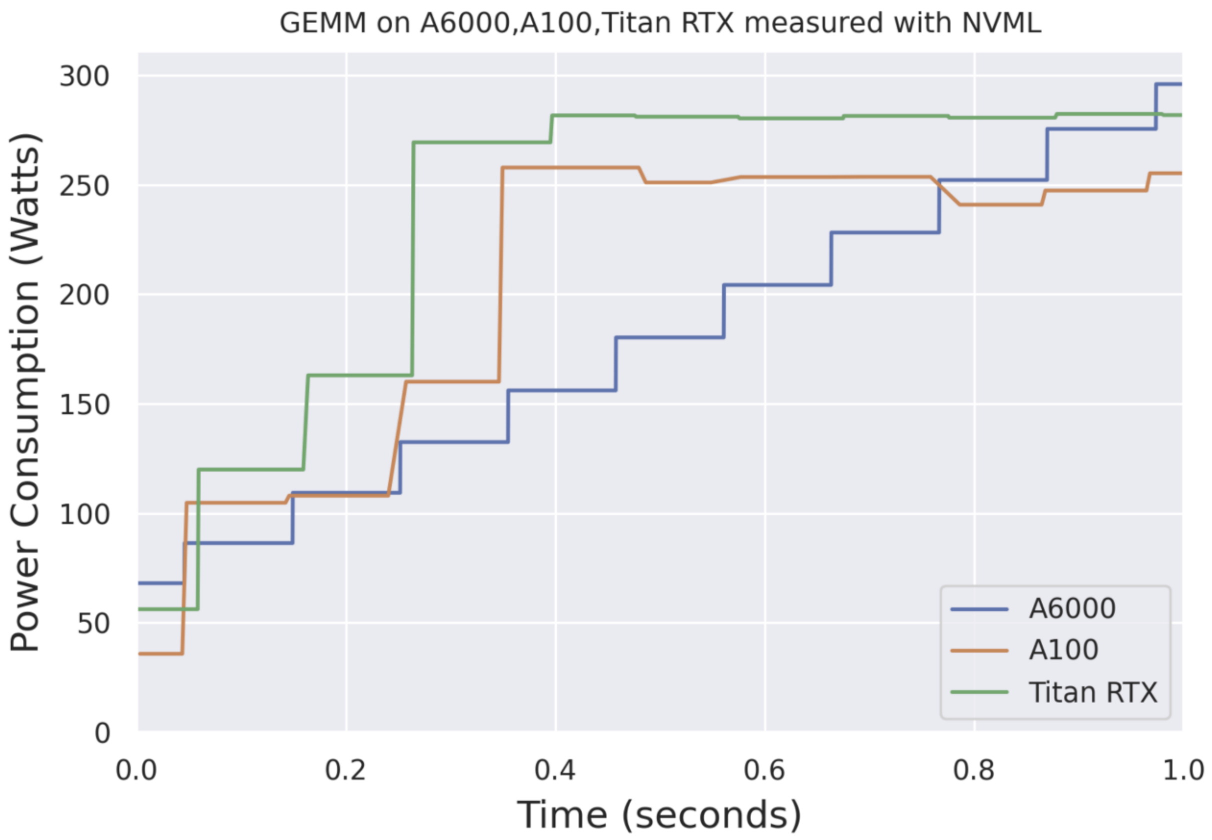
- Highly available

Disadvantages:

- Returns time-averaged power, not instantaneous power consumption
- Limited time resolution

Current solution:

- measure power while continuously running the kernel for one second



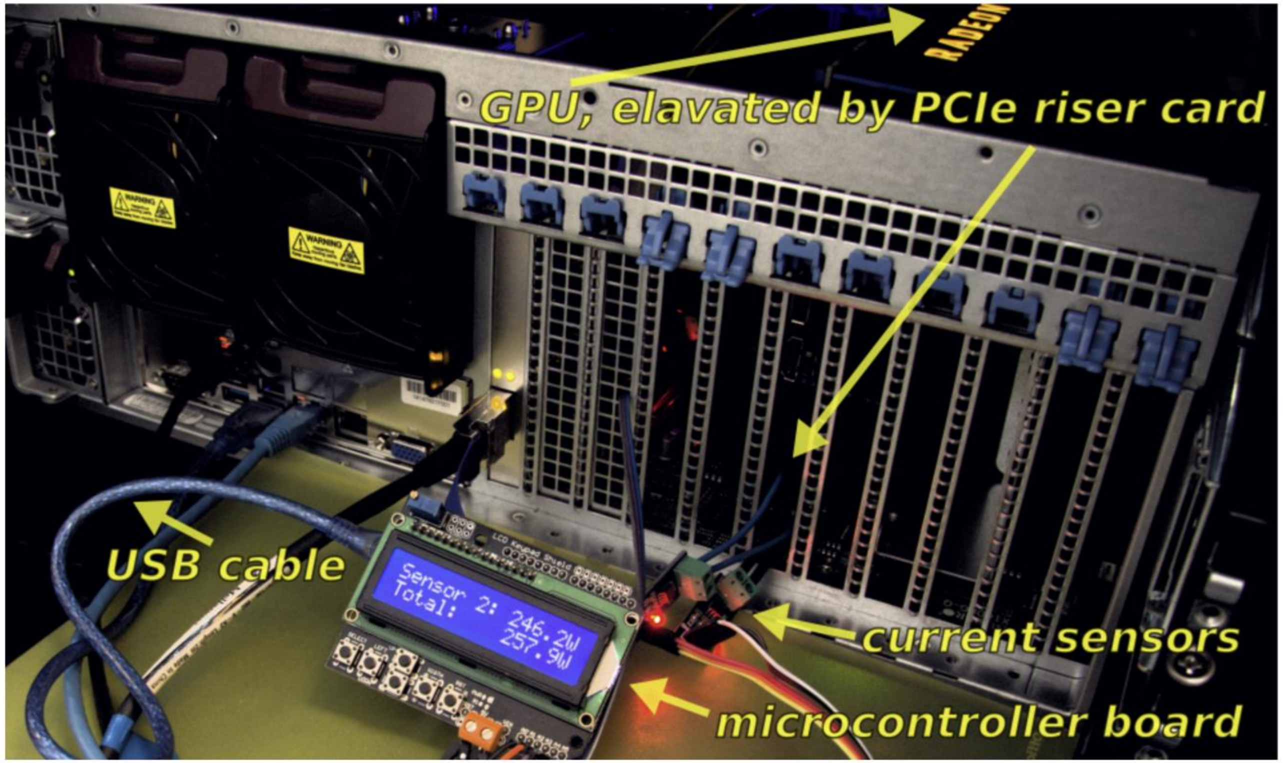
PowerSensor2

Pros:

- Instantaneous power readings
- Time resolution: 2.8 KHz
- Open source: https://gitlab.com/astron-misc/PowerSensor

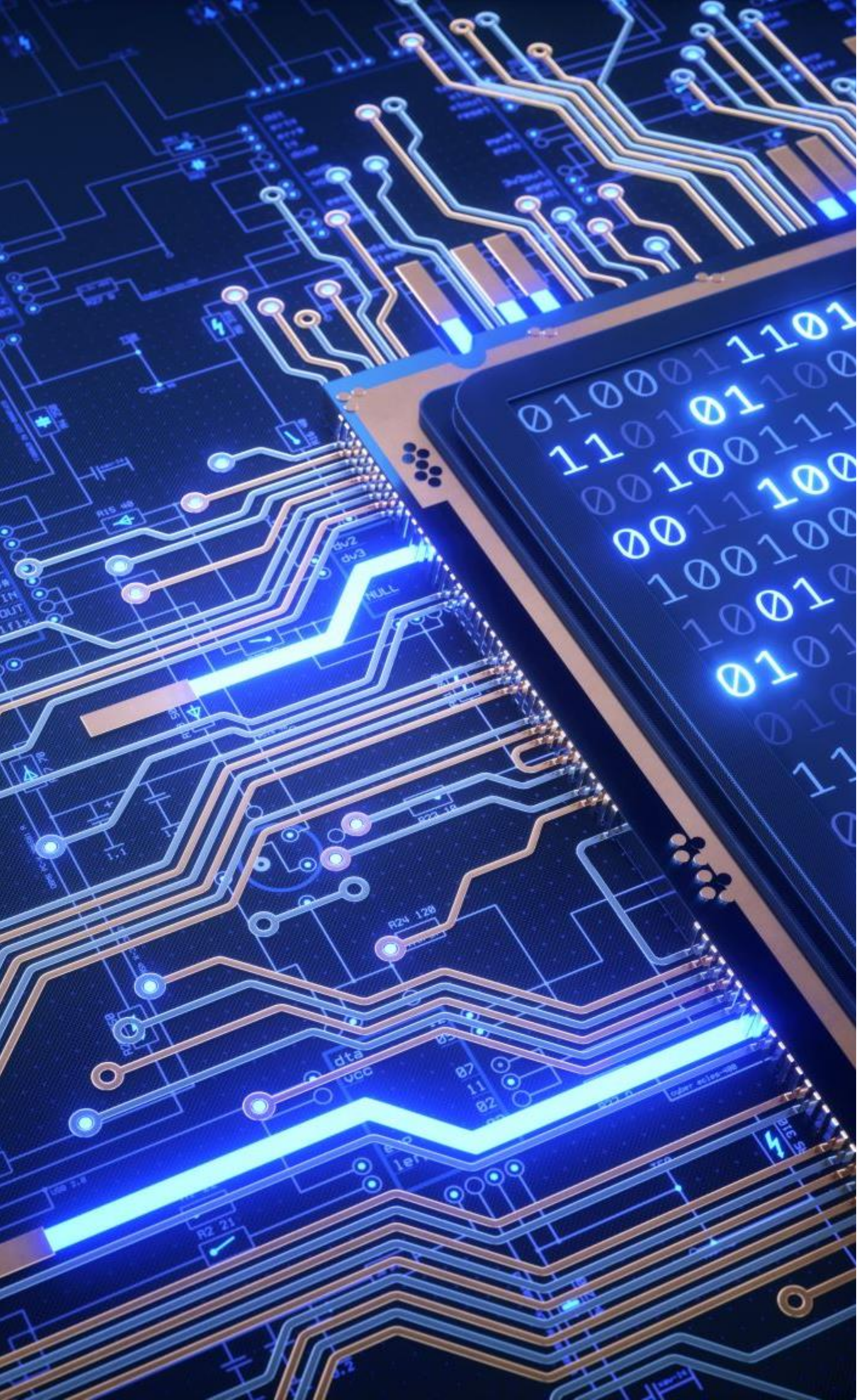
Cons:

- Assembly required



Supported in Kernel Tuner, using PowerSensorObserver

PowerSensor 2: A Fast Power Measurement Tool, John W. Romein and Bram Veenboer, International Symposium on Performance Analysis of Systems and Software (ISPASS 2018)



Power limit vs frequency tuning

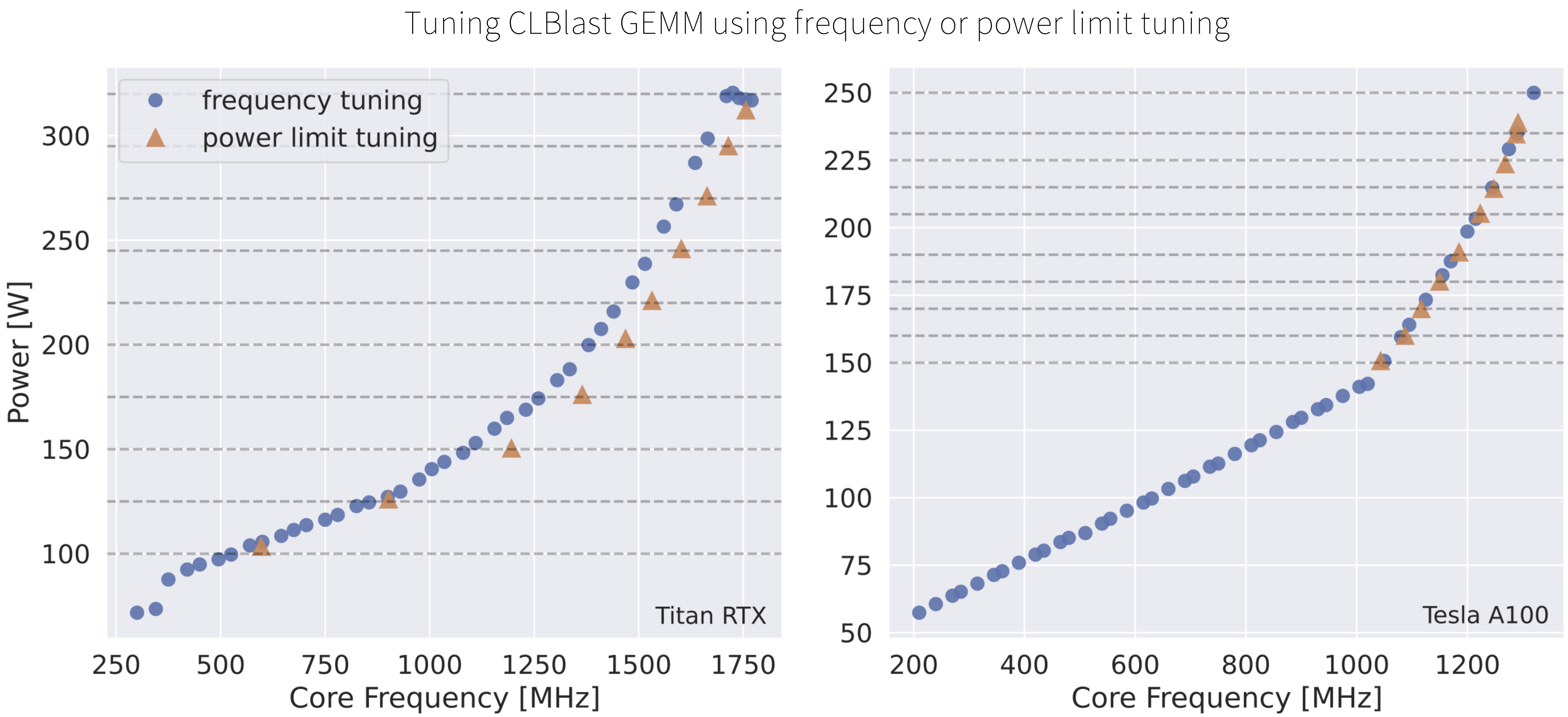
Many tunable parameters affect compute performance and/or energy efficiency

But we can also:

- Limit the GPU clock frequency, allow GPU to vary power consumption
- Limit the GPU power consumption, allow GPU to determine clock frequency

Both methods unfortunately require root privileges for the latest generations of Nvidia GPUs

Frequency-power relation



Frequency tuning vs power capping

Advantages of power capping:

- Potentially more effective, GPU may also lower memory clock
- Reliable method in face of limited power

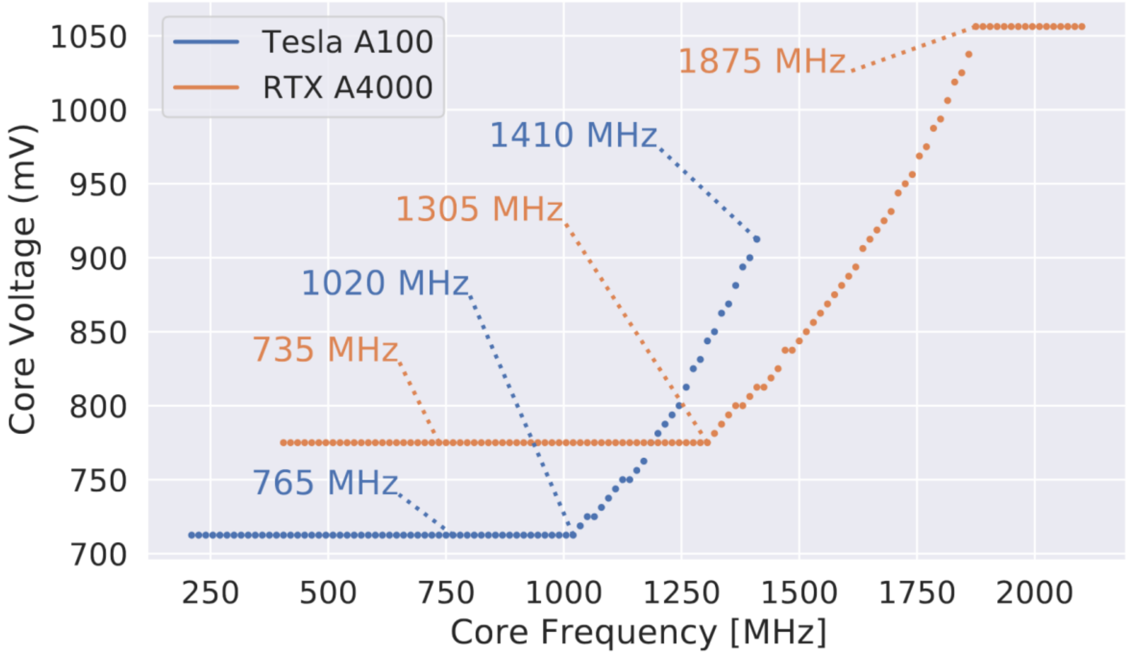
Advantages of frequency tuning:

- Especially on A100, frequency tuning enables a wider power range
- Fixing the clock frequency also improves measurement stability

Frequency Voltage relation

- GPUs rapidly ramp up voltage when clock frequency increases beyond a certain point
- This point appears to be a sweet spot in the trade-off between energy consumption and compute performance
- We call this point the 'ridge point'

Observation: base clock, ridge point, max



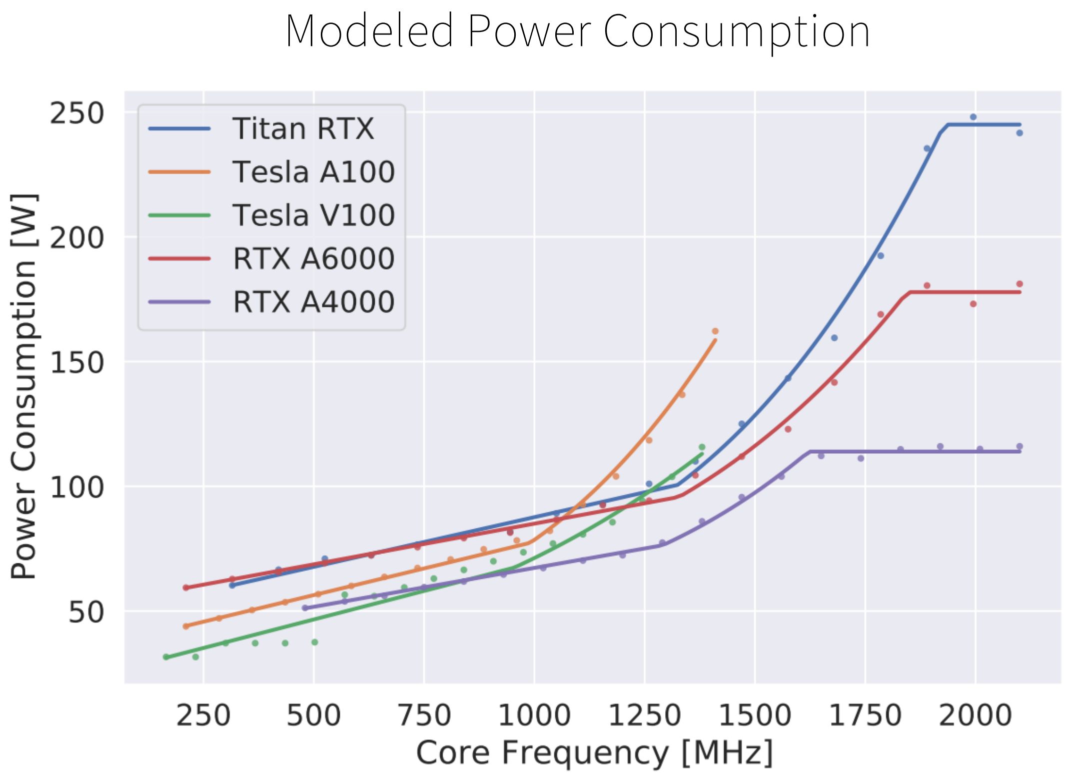
Going Green: optimizing GPUs for energy efficiency through model-steered auto-tuning
Richard Schoonhoven, Bram Veenboer, Ben van Werkhoven, K. Joost Batenburg
PMBS workshop at SC22 2022

A simple power consumption model

- Not every GPU reports core voltages, but we can estimate the voltage using a simple power model
- When we fix all parameters and vary the clock frequency, we can approximate power consumption using:

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

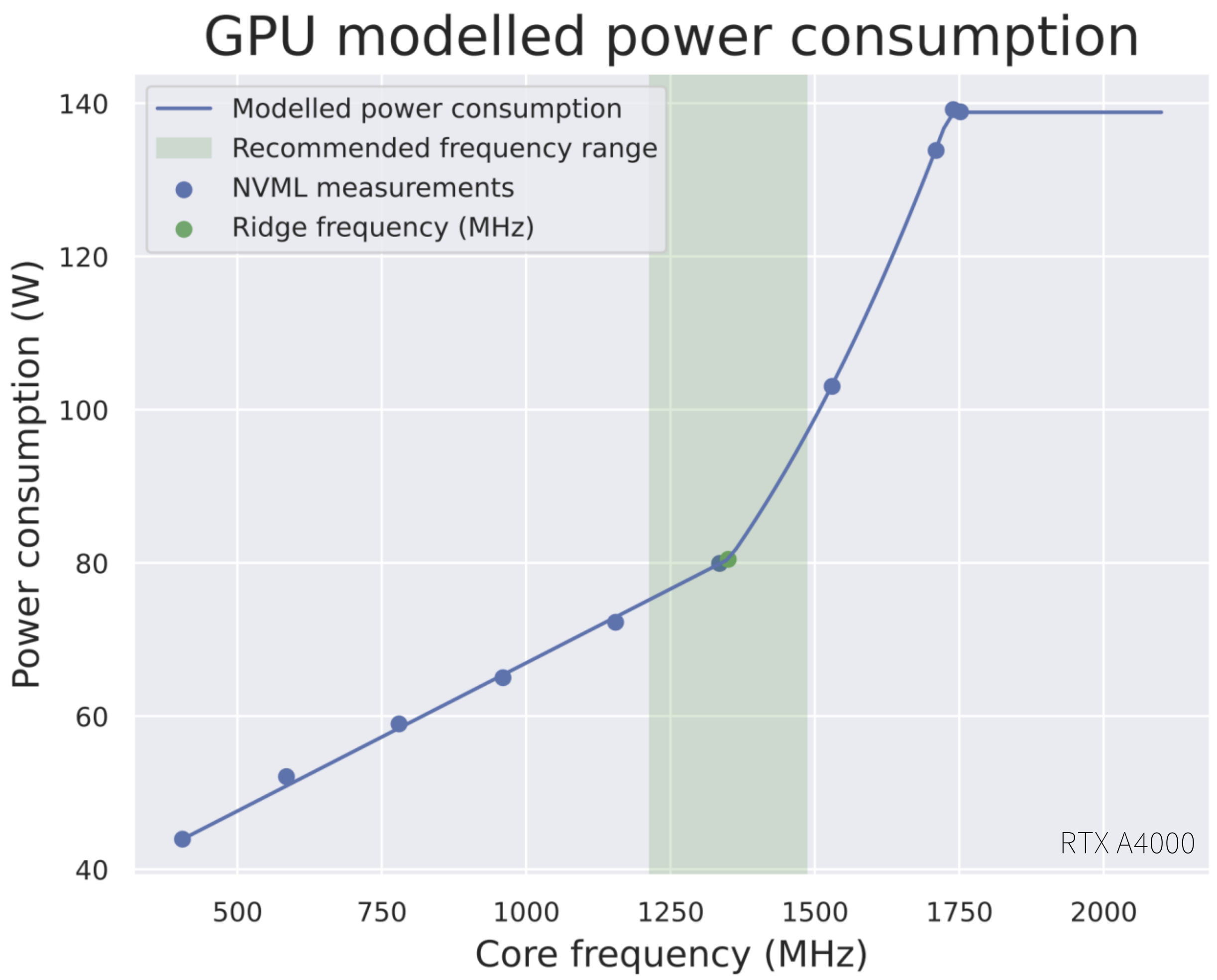
- And identify the GPUs 'ridge point' frequency in this way



Going Green: optimizing GPUs for energy efficiency through model-steered auto-tuning
Richard Schoonhoven, Bram Veenboer, Ben van Werkhoven, K. Joost Batenburg
PMBS workshop at SC22 2022

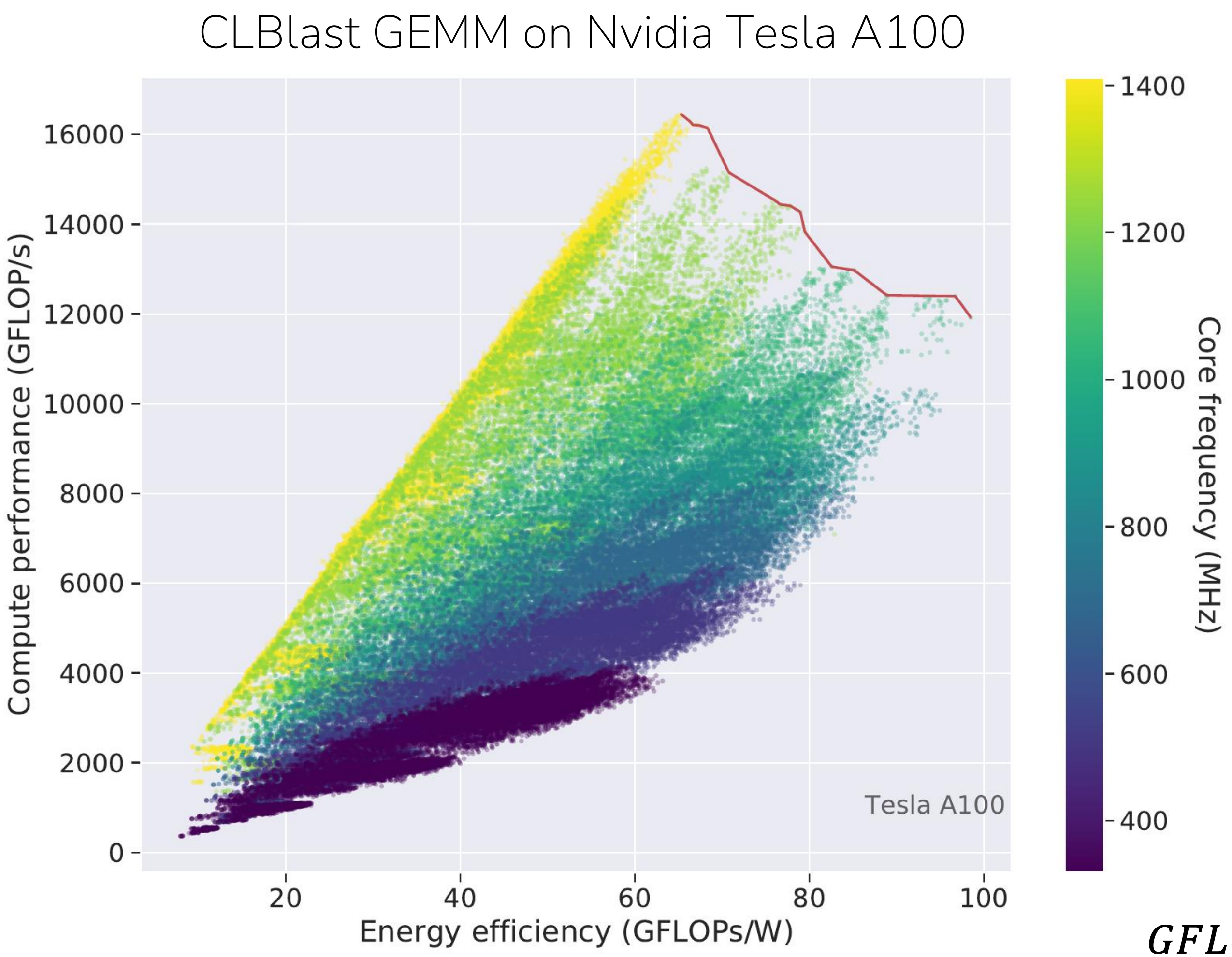
Model-steered auto-tuning

- Use performance model to limit the frequency range for tuning
- Support implemented in Kernel Tuner 0.4.4
- Reduces the search space by ~80% on average



https://github.com/KernelTuner/kernel_tuner/blob/master/examples/cuda/going_green_performance_model.py

Practical energy gains



We can gain 50.9% energy efficiency, by trading in 27.5% performance

$$\frac{GFLOPs}{W} = \frac{GFLOP/s}{J/s} = \frac{GFLOP}{J}$$

Tuning radio astronomy applications

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

Tuned 6 different radio astronomy applications on four GPUs, trading on average 24.3 ± 12.1% compute performance for 42.0 ± 24.1% gain in energy efficiency

Future research directions

- Improving NVML power measurement accuracy and stability
- PowerSensor3
- Multi-objective optimization
- AMD support: HIP/ROCm
- Tuning accuracy and mixed-precision to explore GPU energy-accuracy trade-offs

esiwace
CENTRE OF EXCELLENCE IN SIMULATION OF WEATHER AND CLIMATE IN EUROPE

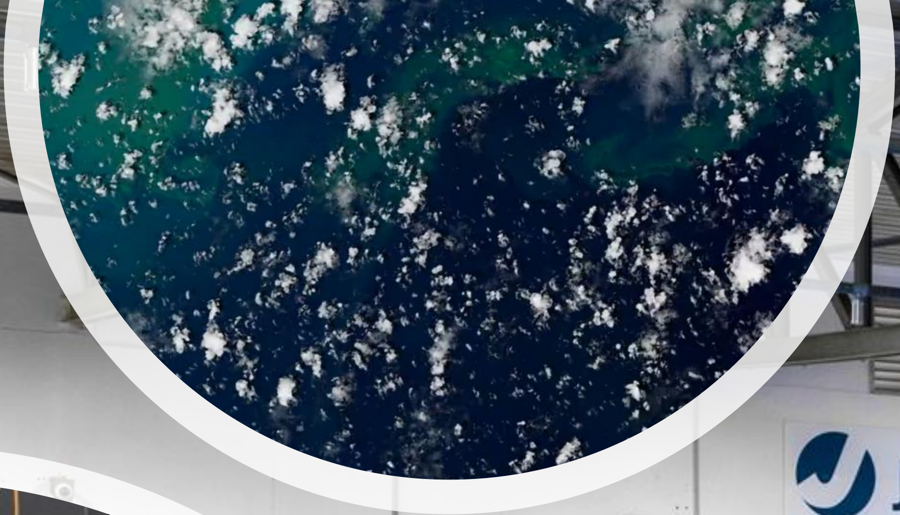
Conclusion

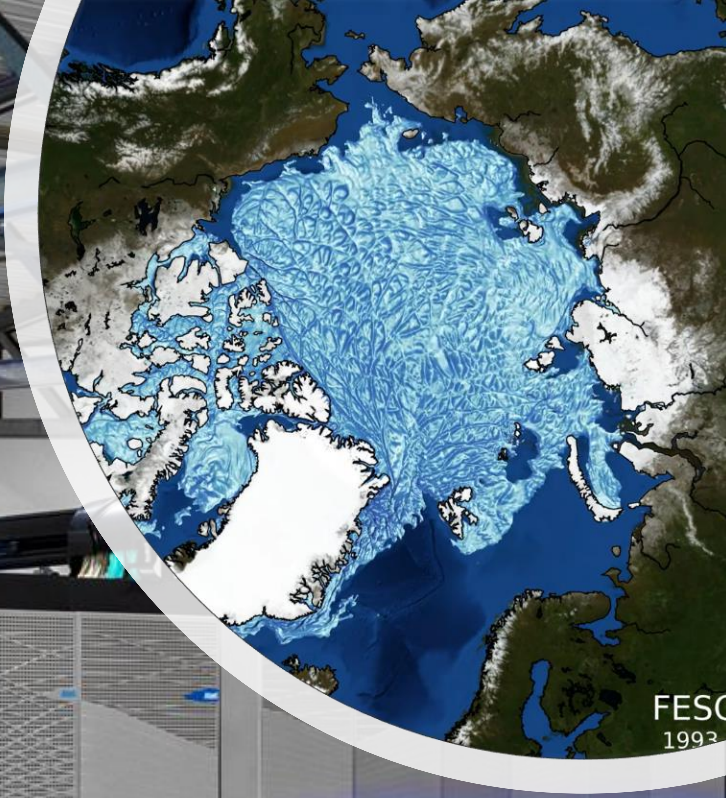
- GPU Energy measurements:
  - NVML (slower, but highly available), PowerSensor2 (faster, but assembly required)
- Energy is clearly a different optimization objective, distinct from time
- Model-steered auto-tuning can effectively optimize GPU energy efficiency
  - on average gain 42.0% energy efficiency, trading in 24.3% compute performance

- All presented methods available in Kernel Tuner:
  - http://github.com/KernelTuner/kernel_tuner
- Full paper:
  - *Going green: optimizing GPUs for energy efficiency through model-steered auto-tuning*
    R. Schoonhoven, B. Veenboer, B. van Werkhoven, K. J. Batenburg
    International Workshop on Performance Modeling, Benchmarking and Simulation of High Performance Computer Systems (PMBS) at Supercomputing (SC22) 2022
  - https://arxiv.org/abs/2211.07260

Related publications

- Going green: optimizing GPUs for energy efficiency through model-steered auto-tuning
  Richard Schoonhoven, Bram Veenboer, Ben van Werkhoven, K. Joost Batenburg
  International Workshop on Performance Modeling, Benchmarking and Simulation of High-Performance Computer Systems (PMBS) at Supercomputing (SC22) 2022
- Benchmarking optimization algorithms for auto-tuning GPU kernels
  Richard Schoonhoven, Ben van Werkhoven, K. Joost Batenburg
  IEEE Transactions on Evolutionary Computation (TEVC) 2022
- Optimization Techniques for GPU Programming
  Pieter Hijma, Stijn Heldens, Alessio Sclocco, Ben van Werkhoven, and Henri Bal
  ACM Computing surveys 2022
- Kernel Tuner: A search-optimizing GPU code auto-tuner
  B. van Werkhoven
  Future Generation Computer Systems 2019





ESiWACE3 Services

Service 1: funds small collaborative projects to port and optimize European weather and climate models to new EuroHPC systems
Service 2: funds small projects to assist in the adoption of HPC tools developed by the ESiWACE3 consortium (includes Kernel Tuner)
Calls for proposals in Service 1 open in 2023, Service 2 in 2024

esiwace.eu
CENTRE OF EXCELLENCE IN SIMULATION OF WEATHER AND CLIMATE IN EUROPE

Funding

The CORTEX project has received funding from the Dutch Research Council (NWO) in the framework of the NWA-ORC Call (file number NWA.1160.18.316).

ESiWACE2 has received funding from the European Union’s Horizon 2020 research and innovation programme under grant agreement No 823988. ESiWACE3 is funded by EuroHPC JU and national co-funding bodies under grant agreement No 101093054.

The ConFu project is funded by the Netherlands eScience Center (file number 00020223-A).