OAK RIDGE National Laboratory

June 13, 2025

Power-Capping Metric Evaluation for Improving Energy Efficiency in HPC Applications

Energy Efficiency with Sustainable Performance: Techniques, Tools, and Best Practices (EESP) Workshop at ISC High Performance

Maria Patrou[1], Thomas Wang[2], Wael Elwasif[1], Markus Eisenbach[1], Ross Miller[1], William Godoy[1] & Oscar Hernandez[1]

[1]*Oak Ridge National Laboratory, Oak Ridge, Tennessee, USA*
[2]*Camas High School, Camas, Washington, USA*

U.S. DEPARTMENT OF ENERGY

ORNL IS MANAGED BY UT-BATTELLE LLC
FOR THE US DEPARTMENT OF ENERGY

EESP WORKSHOP 2025

Overview

- Power-Capping
- Motivation
- Related Work
- HPC Application – LSMS
- Methodology
- Experimental Evaluation
- Conclusion

Power-Capping

- Power management technique that enables setting a maximum power limit on a device and it dynamically adjusts the power utilization to be under the defined limit
- New-generation architectures, such as NVIDIA’s GH200 superchip, regulate both CPU and GPU simultaneously
- It incorporates the automatic power-steering system and dynamically reallocates power between the CPU and the GPU based on their usage; preference on the CPU
- Limits are enforced through Dynamic Voltage and Frequency Scaling (DVFS) for CPU and GPU
- More significant impact on compute-intensive workloads than memory-bound tasks

Overview of the NVIDIA GH200 Superchip

Architecture

Key Features:

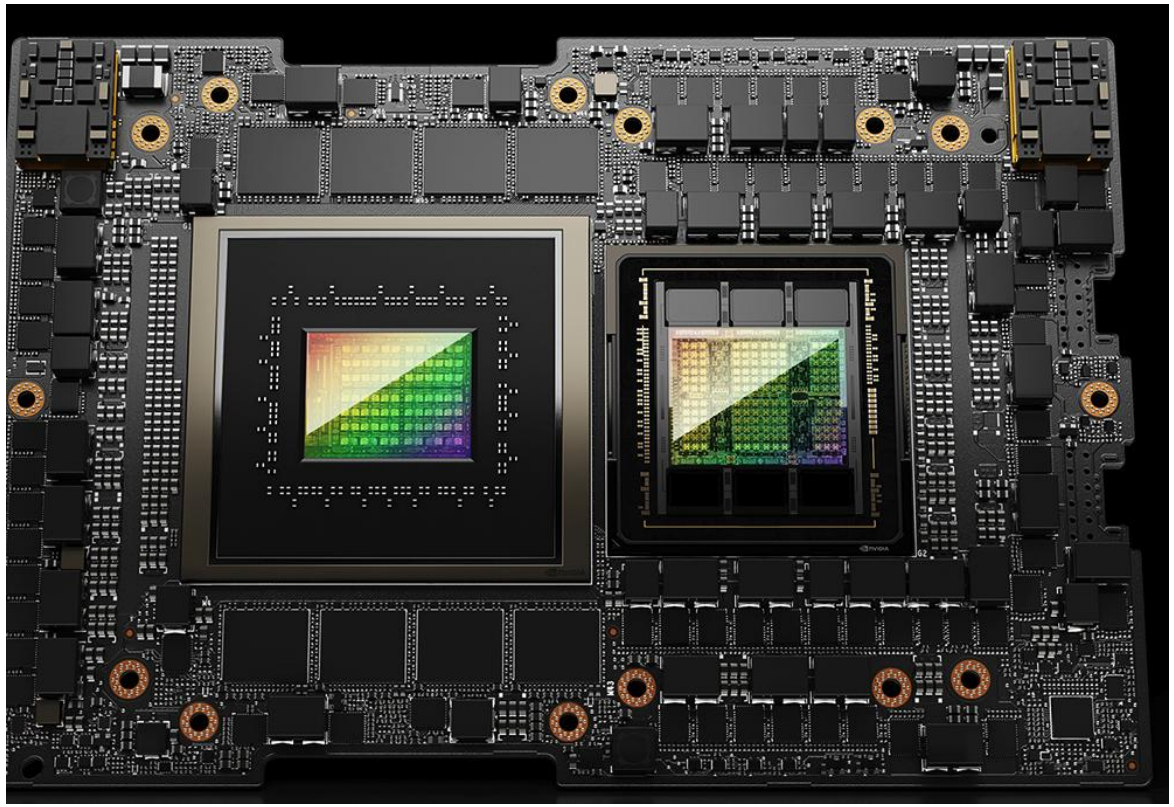
- 72-core Arm Neoverse-V2 CPU (Grace)
- Hopper GPU with 96GB HBM3
- Integrated CPU-GPU memory architecture

Peak Power Consumption: Up to 1kW

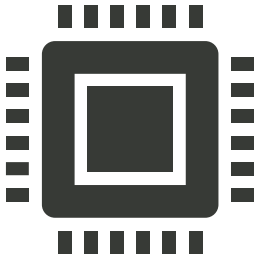
Challenges:

- Managing power across CPU, GPU, & I/O
- High-resolution monitoring for transient power spikes

NVIDIA GH200



Motivation

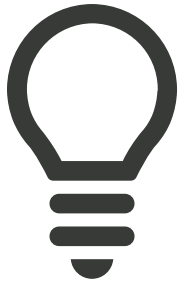


HPC Power Challenges

Increasing power consumption in modern superchips (e.g., NVIDIA GH200)

Need for fine-grained power and energy optimizations

Moore's Law slowdown and efficiency trade-offs



Why Fine-Grained Power Analysis?

Identifying power spikes and inefficiencies

Optimizing application performance and energy use

New architectures have more knobs to control power utilization.

*What **power-capping setting** is more suitable for a **GPU task** to achieve **energy efficiency**?*

Related Work

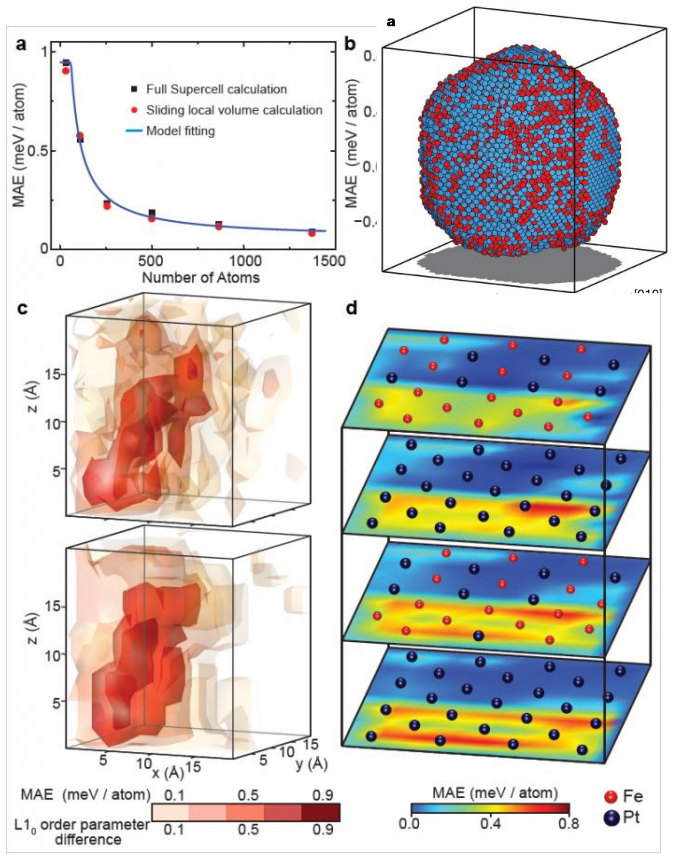
- CPU-GPU power management with heuristic or machine learning methods have been explored
- Greenup, Powerup, and Speedup (GPS-UP) framework to categorize optimizations by their impact on execution time, power, and energy efficiency
- Fine-grained power-capping strategies, such as the Global Extensible Open Power Manager have proven more effective than static limits

- We utilize the automatic power steering of the GH200 when setting a given power cap, and analyze the impact of energy and runtime performance of the most suitable power capping setting as suggested by two decision-makers metrics

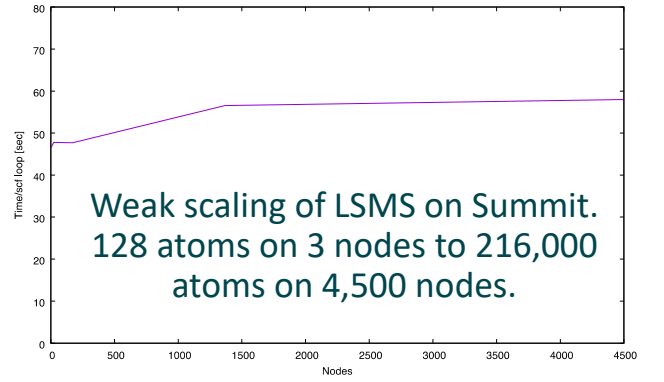
LSMS (Locally Self-consistent Multiple Scattering)
DFT First Principles Calculations for Large Systems

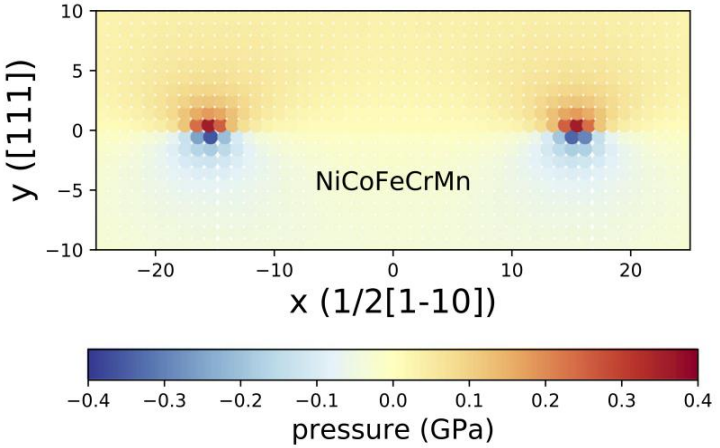
- LSMS solves the Schrödinger or Dirac equation for electrons in solids within Density Functional Theory for alloy large systems and magnetic materials.
- ORNL's exascale application
- One of the most ORNL's power-hungry HPC applications
- Open source:

https://github.com/mstsuite/lsms



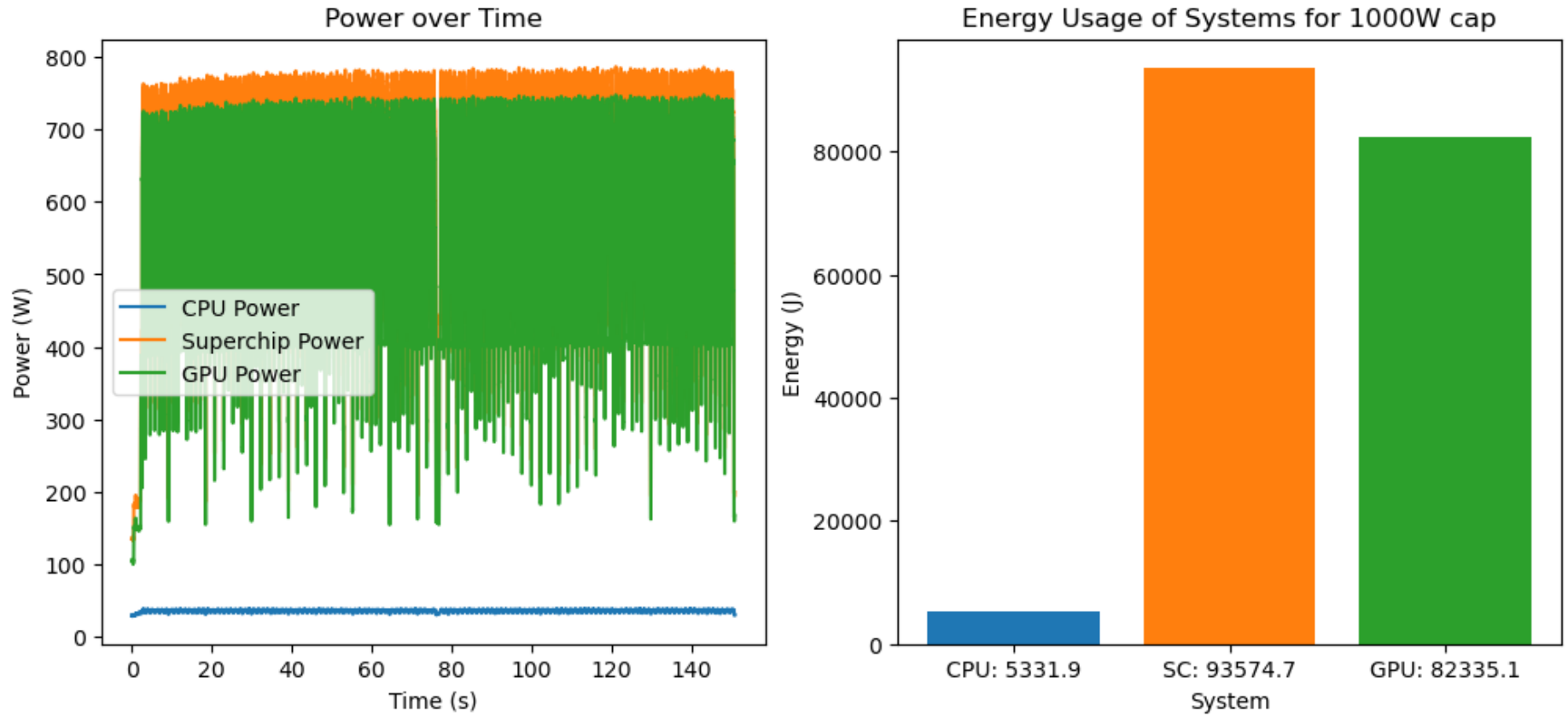
Magnetism in FePt. Nature 542, 75 (2017)





Dislocation core structure in NiCoFeCrMn alloy

HPC Application – LSMS Energy/Power Data



Power and total energy consumption, including CPU and GPU components (no power capping).

HPC Application LSMS – GPU Task Breakdown

Investigation of GPU tasks --- kernels within the application and GPU inactivity phases:

- **sm90_gemm_ts64x64x32**: Compute-intensive cuBLAS zgemm operation; typically compute-bound
- **buildKKRMatrix**: Matrix construction primarily involving memory operations; memory-bound.
- **sm90_gemm_ts32x32x32**: Another compute-intensive cuBLAS zgemm kernel; compute-bound
- **getrf_pivot** kernels: Lower-upper factorization with pivoting; constrained by random memory accesses
- **trsm_left_kernel**: Triangular matrix solve (BLAS); typically memory-bound due to data access patterns
- **gpu compute idle**: Periods in which computations only occur on the CPU

HPC Application LSMS – GPU Task Breakdown

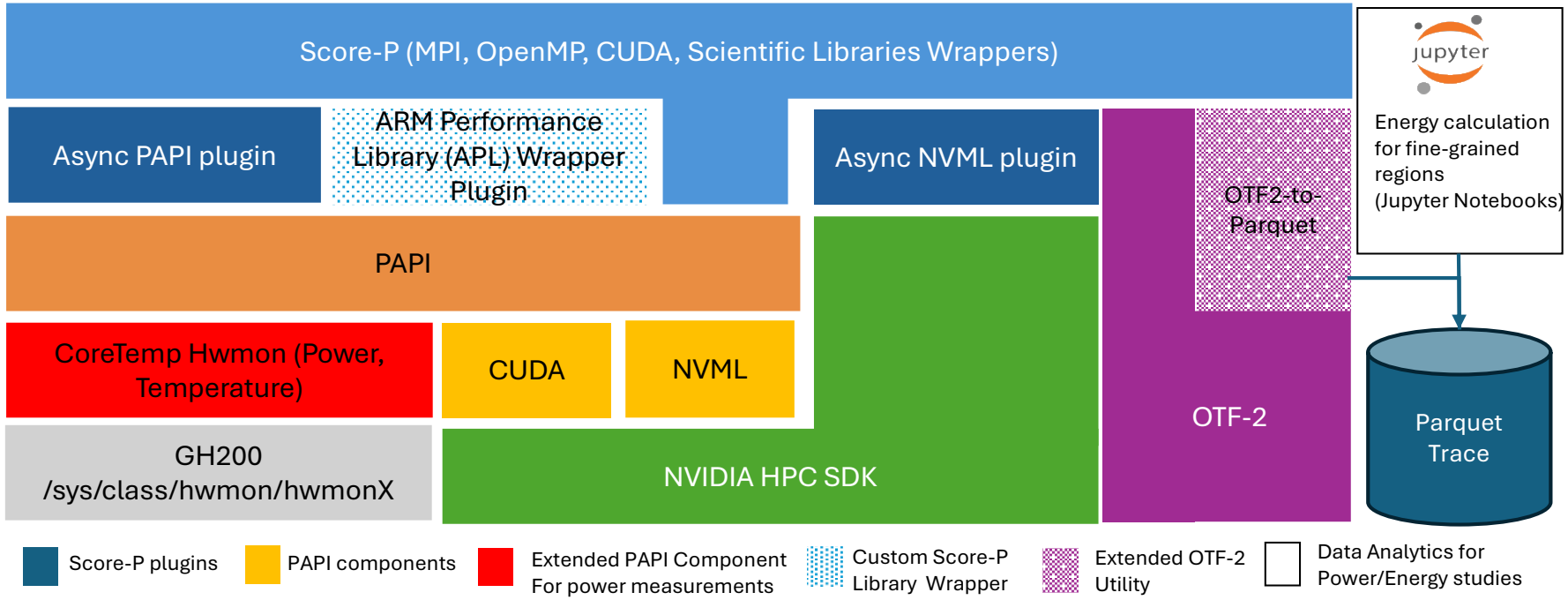
Task	Total Time (s)	# Calls	Total Energy (J)	Avg. Power (W)
sm90_gemm_ts64x64x32	77.89	21,632	35,361.83	454.02
buildKKRMatrix	34.90	128	12,867.73	368.74
sm90_gemm_ts32x32x32	8.03	94,208	4,076.98	507.51
getrf_pivot(1)	4.07	16,384	2,694.54	662.05
getrf_pivot(2)	4.07	30,720	2,670.36	656.11
trsm_left_kernel	3.57	150,272	2,328.26	651.57
getrf_pivot(3)	1.82	8,192	1,146.70	630.06
gpu compute idle	8.83	601,345	2,425.49	274.80

Measurements at the default power setting (1,000 W, no power capping) per GPU kernel and GPU compute idle time.

Methodology

- Measure energy, power and runtime performance at the superchip level for the entire application, and focus on the CPU- and GPU-level per GPU task (kernel and GPU inactivity phase)
- Incremental chip-level power constraints on NVIDIA's GH200 superchip (with automatic power-steering system), ranging from 200 W to the maximum power limit of 1,000 W (default)
- Power data collection with Score-P and custom Performance Application Programming Interface (PAPI) component designed to read power information via the Linux */sys/class/hwmon* interface
- Metrics calculation to decide on the "best" power capping settings per GPU task

The Open-Source Performance Toolkit for Power/Energy



Python Data Analytics

Extended Components

- PAPI extension for hwmon to measure Superchip and Grace CPU power
- Score-P NVML and PAPI plugins readings
- Runtime overhead per run for the power measurements was 1.3%
- OTF2-to-Parquet conversion utility
- Jupyter-based analytics for energy profiling

Supported Architectures

- GH200, potential for AMD Instinct and other platforms

Based on tools from:

VAMPIR

Score-P
Scalable performance measurement infrastructure for parallel codes

icl-utk-edu/papi

NVIDIA

arm

https://dl.acm.org/doi/10.1145/3703001.3724383

Decision-maker Metrics for "best" power capping settings

Speedup-Energy-Delay

- Calculation of speedup energy-delay from the baseline (runtime1 and energy1) - no power capping (1,000W)

$$SPEEDUP_{energy-delay} = \frac{\left[\frac{runtime_1}{runtime_n}\right]}{\left[\frac{energy_n}{energy_1}\right]} = \frac{[runtime_1 * energy_1]}{[runtime_n * energy_n]}$$

Euclidean Distance of normalized energy/runtime

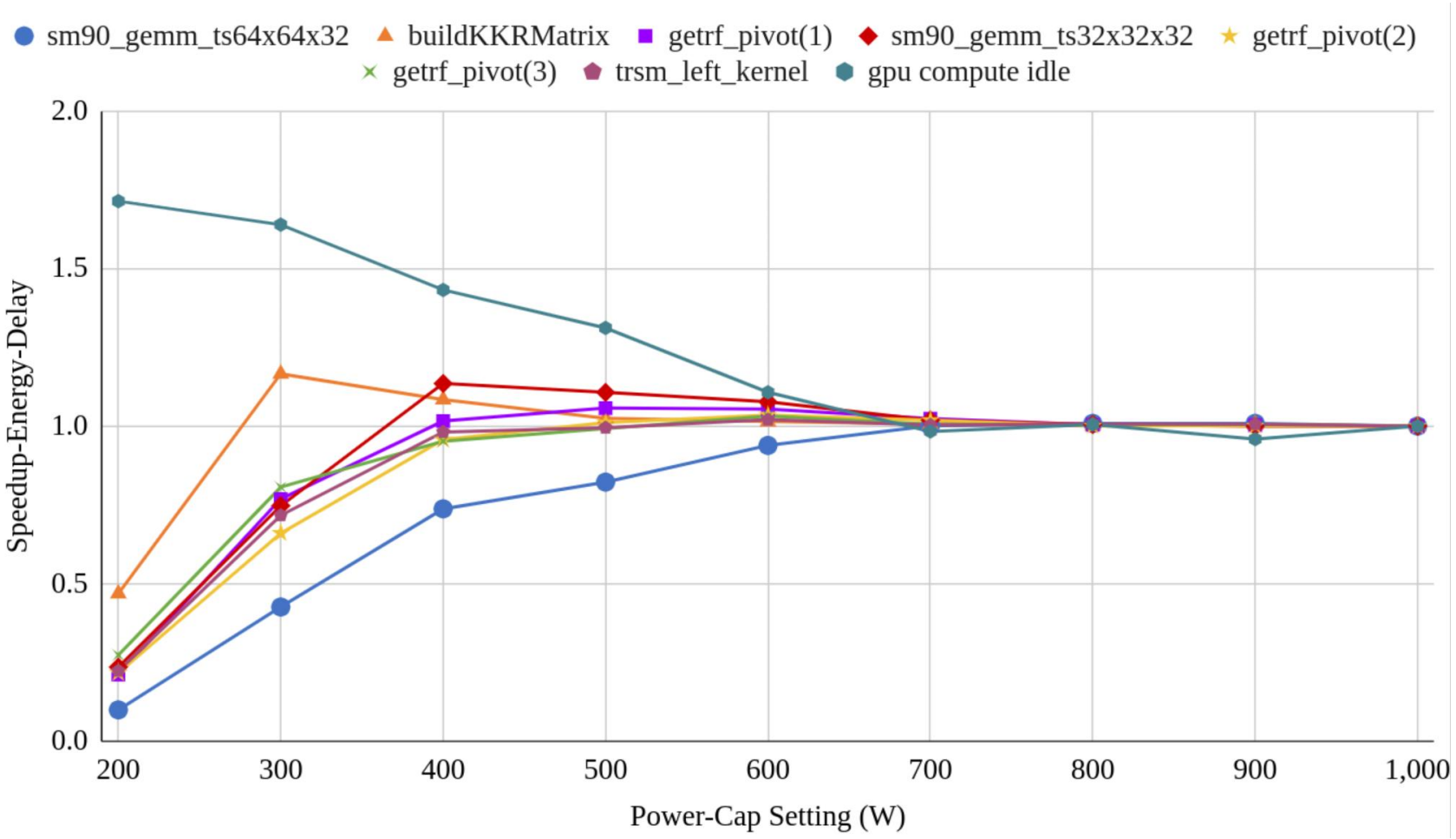
- Normalization of energy and runtime metrics using the feature scaling normalization method

$$n_{energy_{ki}} = \frac{[e_{ki} - emin_k]}{[emax_k - emin_k]}$$

- Euclidean distance of normalized energy and runtimes from point (0,0)

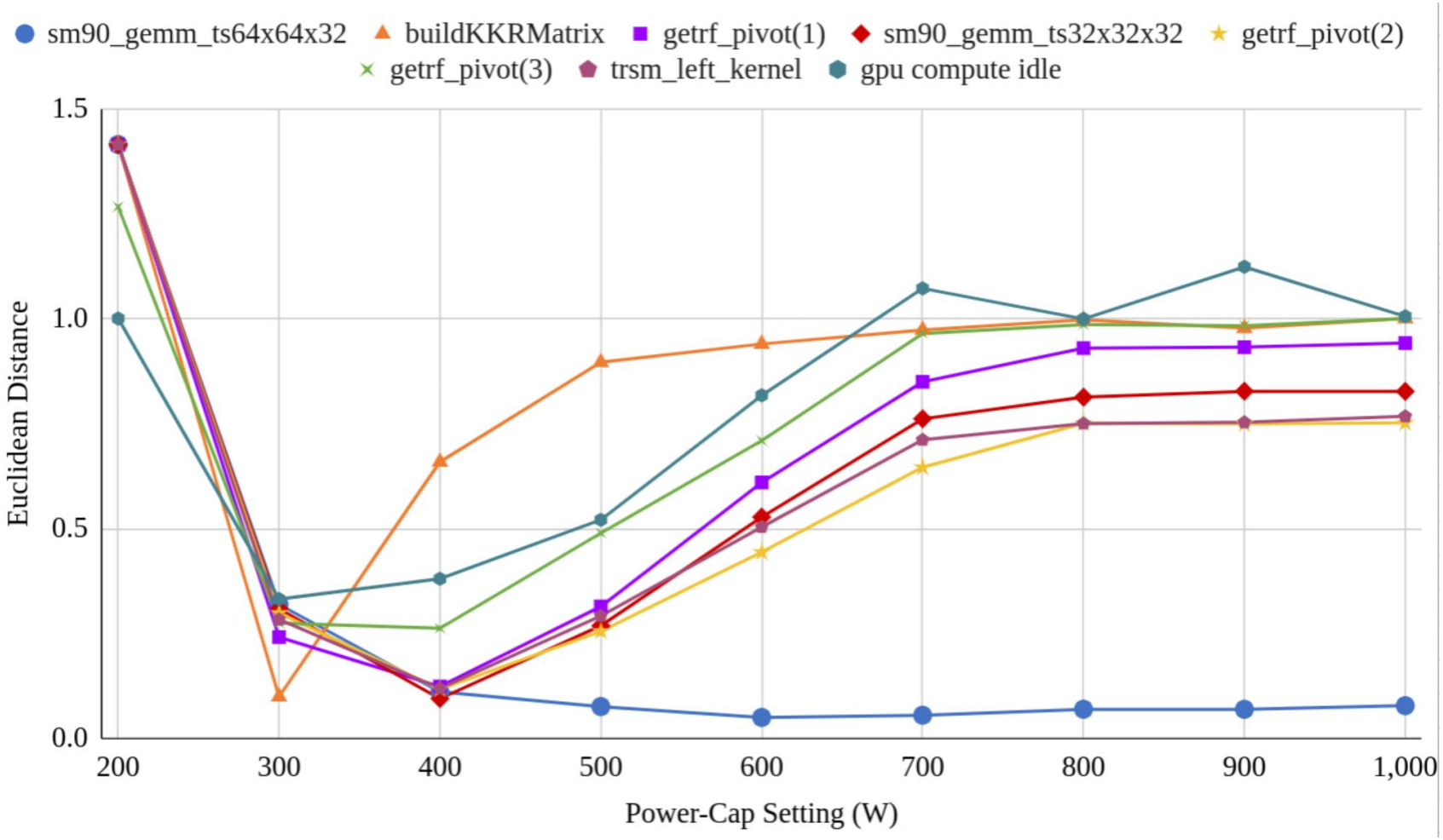
$$distance_{ki} = \sqrt{n^2_{energy_{ki}} + n^2_{runtime_{ki}}}.$$

Experimental Evaluation



Speedup-energy-delay per GPU task and power-cap setting. Higher is better.

Experimental Evaluation



Euclidean distance of normalized energy/runtime per GPU task and power-cap setting. Lower is better.

Experimental Evaluation

* SED = speedup-energy-delay
  ED = Euclidean distance

GPU task	Power Cap (W)		Energy (%)		Runtime (%)	
	SED	ED	SED	ED	SED	ED
sm90_gemm_ts64x64x32	900	600	0.85	3.42	0.00	10.3
buildKKRMatrix	300	300	22.92	22.92	11.30	11.30
sm90_gemm_ts32x32x32	400	400	31.40	31.40	28.42	28.42
getrf_pivot(1)	500	400	20.61	28.50	19.16	37.67
getrf_pivot(2)	600	400	10.05	24.48	7.37	38.41
trsm_left_kernel	600	400	9.02	25.53	7.74	36.85
getrf_pivot(3)	600	400	9.10	24.15	6.59	38.46
gpu compute idle	200	300	46.58	39.69	9.25	1.17

Runtime increase (%) and Energy reduction (%): default power compared with the most suitable power setting identified by speedup-energy-delay (SED) and Euclidean distance (ED) per GPU task.

Conclusion

- Investigation of GPU task-specific power-capping with NVIDIA GH200 superchip's automatic power-steering system
- Evaluation of two metrics—*speedup-energy-delay* and *Euclidean distance–based normalization*—to determine optimal power limits per computational GPU task
- Comparing the optimal power-capping settings suggested by each one, the Euclidean distance–based normalization is affected by the data distribution differently than the speedup-energy-delay; ED more biased towards energy
- Significant energy savings are achievable through fine-grained power management

- In the future, we will extend these methodologies, develop strategies for additional domains, and adaptive power-capping optimizations

Acknowledgments

- This work supported by the US Department of Energy's Office of Science, Advanced Scientific Computing Research program through EXPRESS: 2023 Exploratory Research for Extreme-Scale Science. This research used resources of the Oak Ridge Leadership Computing Facility at Oak Ridge National Laboratory, which is supported by the Office of Science of the US Department of Energy under contract DE-AC05-00OR22725.
- ORNL's ACE testbed program
- Department of Energy (DOE) Advanced Scientific Computing Research
- NVIDIA Corporation for technical support
- Collaborators at ORNL, NVIDIA, and Camas High School (WA)
    - Special thanks to Thomas Wang for working on this as his high-school senior project

Thank you!

U.S. DEPARTMENT OF ENERGY

ORNL IS MANAGED BY UT-BATTELLE LLC
FOR THE US DEPARTMENT OF ENERGY