Sustainable Scientific Computing Workshop
Leiden, NL
October 27, 2025

Friedrich-Alexander-Universität
Erlangen-Nürnberg

Patterns, measurements, guesstimates: How to work with energy metrics in HPC and stay sane

Georg Hager

Erlangen National High Performance Computing Center (NHR@FAU)

Points of view: computational science

Scientist (“nerd”)

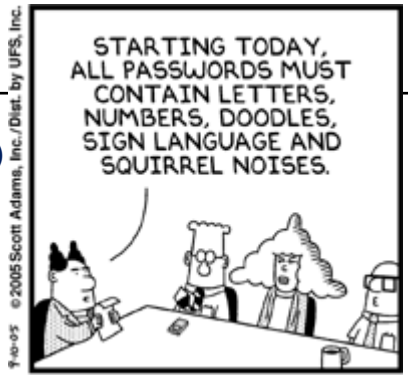
CPU time allocation

Project runtime

Science

Metric: Papers/CPUh

Computing Center (“naggers”)



Hardware & maintenance cost

Energy cost

Infrastructure

Machine lifetime

Science

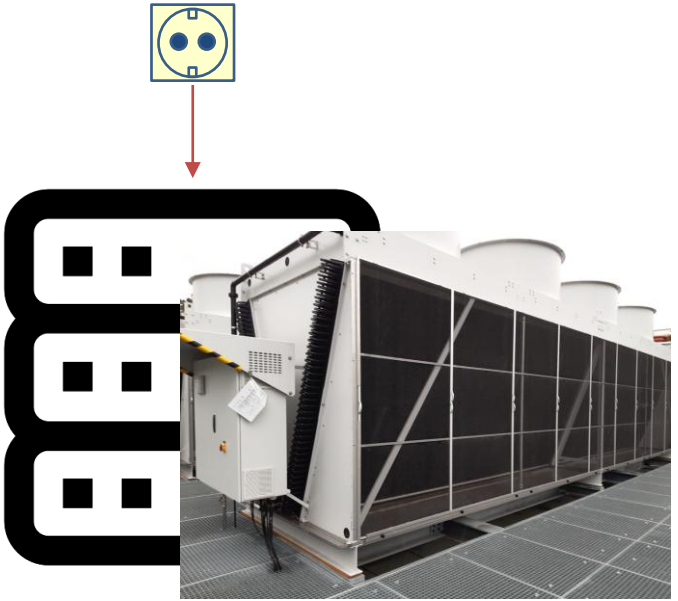
Next machine

Metric: ???

Power and energy of a computer system

Energy “consumption” (goes into heat):

$$E = \int_0^T W(t)\,\mathrm{d}t$$



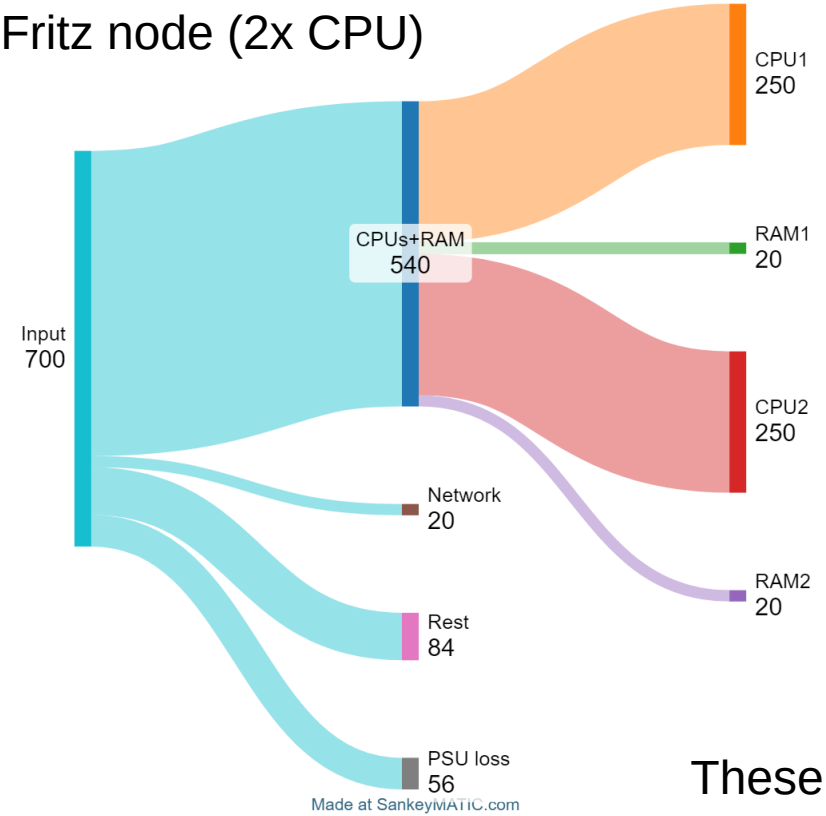
- W: power dissipation [W]
- T: Runtime
- E: Energy [Wh]

≈ 1500

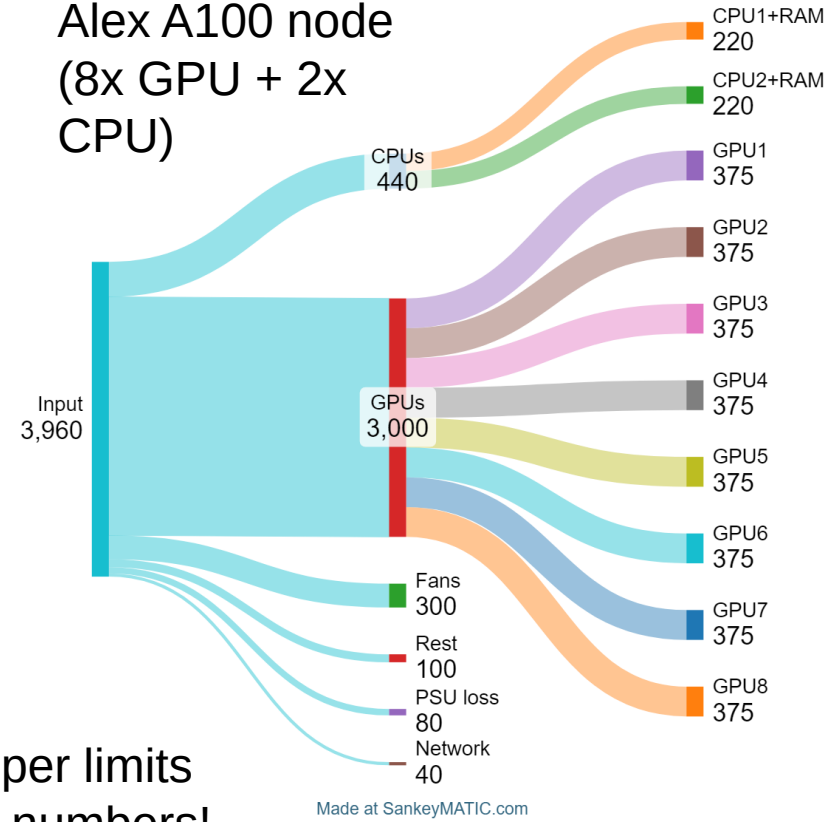
- Example: Fritz cluster at full load 650 kW x 8700 h = 5.7 x 10^6 kWh p.a.

Where does the power go in a system?

Fritz node (2x CPU)



Alex A100 node (8x GPU + 2x CPU)



These are upper limits and eyeballed numbers!

Performance/power/energy/TCO tools

- Tools (CPU)
  - LIKWID tools (based on RAPL)
    - Available as module on NHR@FAU clusters
    - https://github.com/RRZE-HPC/likwid
    - `likwid-perfctr -g ENERGY [-m] -c N:0-71 ./a.out`
    - `likwid-mpirun -g ENERGY [-m] -np 360 ./a.out`
  - ClusterCockpit (based on LIKWID)
    - https://github.com/ClusterCockpit
- Tools (GPU)
  - Nvidia + AMD tools `nvidia-smi` / `rocm-smi`
  - ClusterCockpit (based on `nvidia-smi`)
- TCO tool for cost assessment based on power models (by Ayesha Afzal)
  - https://wattlytics.netlify.app/

ClusterCockpit https://clustercockpit.org/

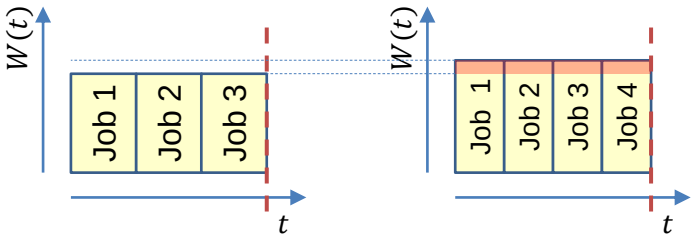
Job-specific monitoring accessible from HPC portal

Energy consumption and CO2 equivalents are reported per job

Reducing the energy per job

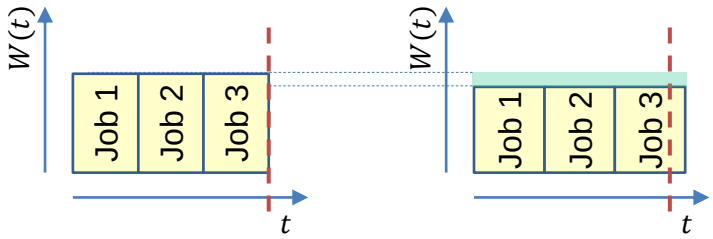
Reduce the job's runtime T

- Better overall use of resources
- Better use of your resource allotment
- More "science per CPUh"
- Probably higher or lower power dissipation of the system



Reduce the job's power $W(t)$

- Lower power dissipation of the system
- Probably some performance loss → probably less "science per CPUh"

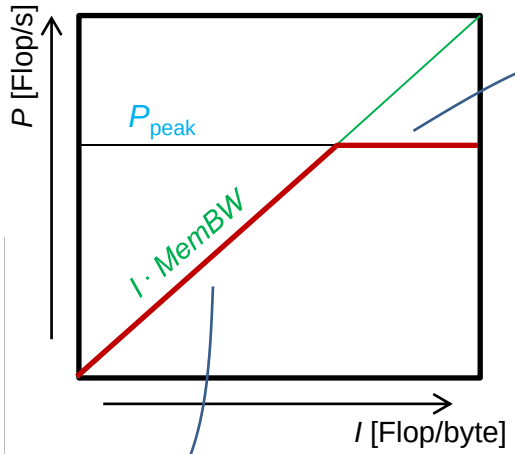


Code optimization for runtime and energy

- Use the best algorithm
  - E.g., $N^2 \rightarrow N \log N$
- Use optimized libraries
  - E.g., OpenBLAS → MKL
- Use aggressive compiler optimization
  - E.g., `-O1` → `-Ofast -xHost`
- Do less work
  - E.g., use sparse matrices instead of dense
- Balance the workload
  - All “devices” finish at the same time
- Transfer less data from far away
  - Cache blocking / register reuse
  - Avoid network communication
- Hide communication overhead
  - Asynchronous/bidirectional network communication
  - Asynchronous GPU data transfers

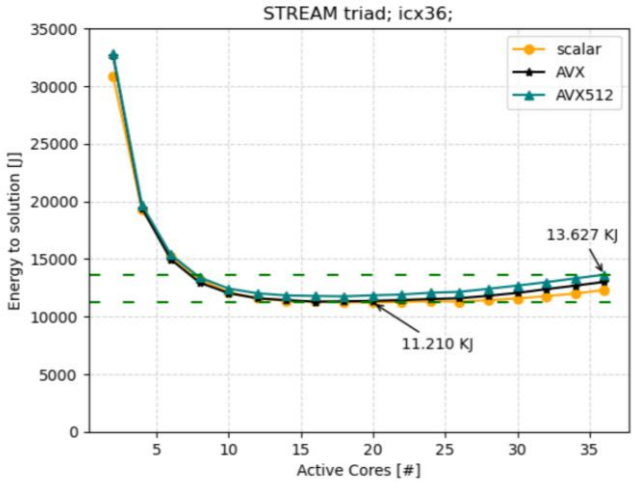
Code characterization with Roofline

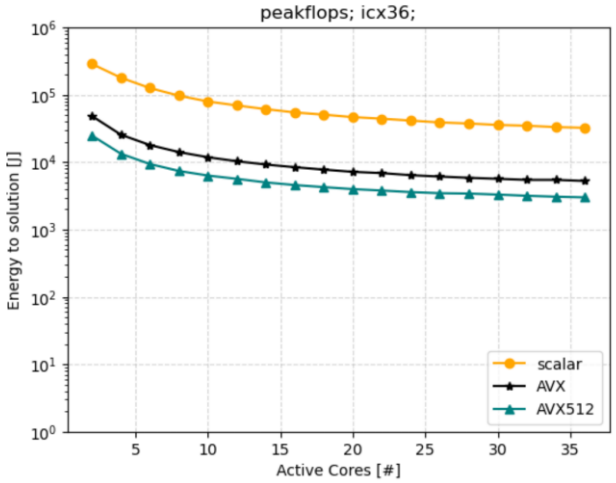
Roofline point of view on a CPU socket level



core bound
(“scalable”)

memory bound
(“saturating”)

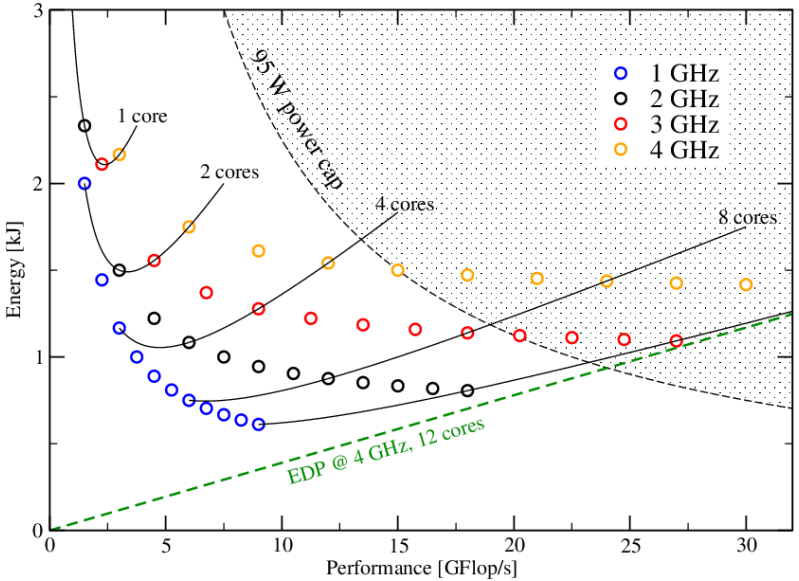


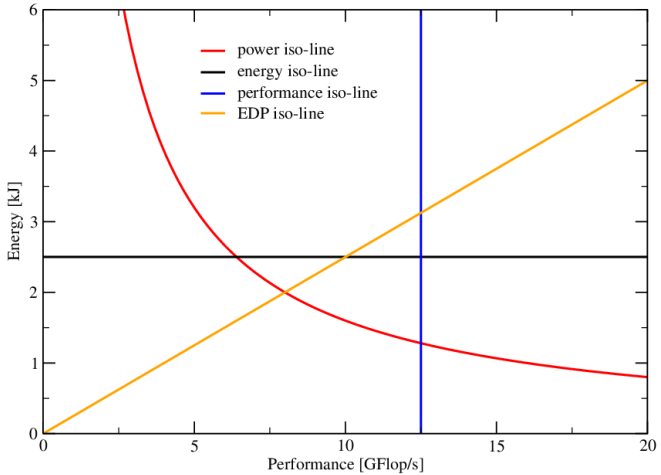


Data: Master thesis Stefan de Souza

Introducing the Z-plot

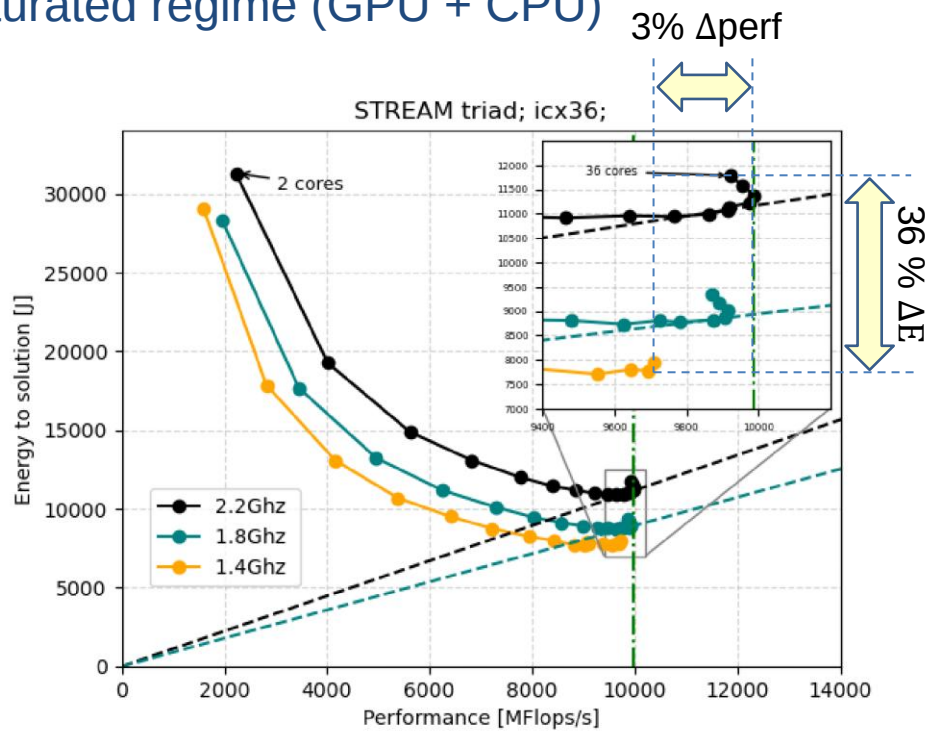
- Energy to solution vs. performance
- “Interesting” loci are straight lines (mostly)
- Plot data in curves with #cores, frequency, or any other interesting parameter





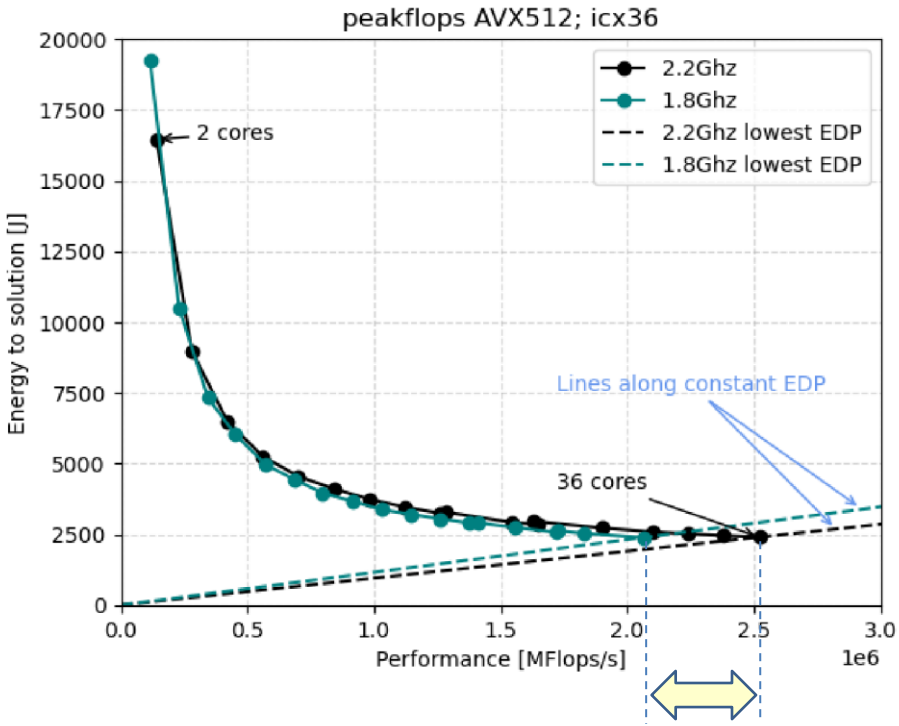
Memory-bound code

- Shows “performance saturation” vs. # of cores on CPUs
- Weak sensitivity to clock frequency in saturated regime (GPU + CPU)
  - Significant energy saving potential
- Optimal “operating point” is crucial
  - # of cores at saturation point → min E
  - Clock speed



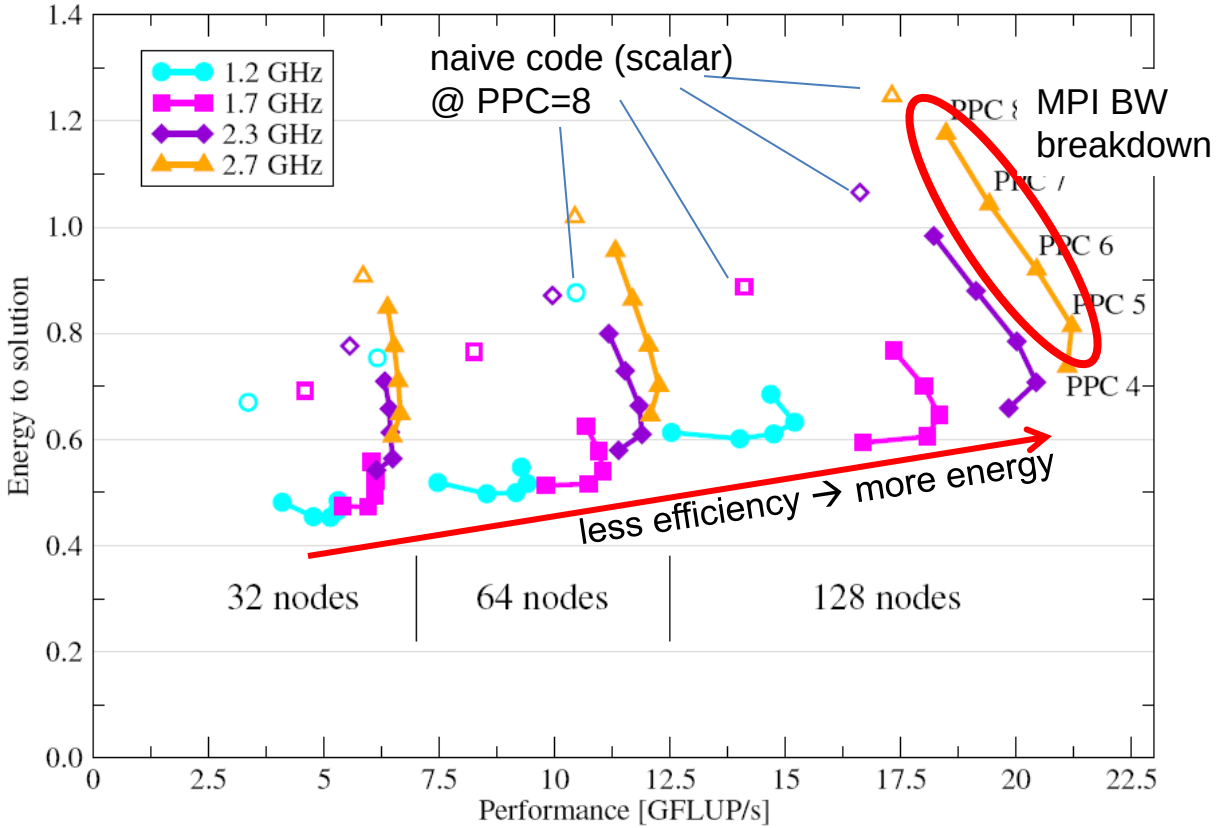
Core-bound (“scalable”) code

- No on-chip scaling bottleneck
- Performance scales with # cores
- The more cores, the lower the energy to solution
- Performance sensitive to clock speed
  - Ideally, proportional
- Power is sensitive to clock speed
  - $W \propto f^{\alpha}$, $\alpha \in [1, \ldots, 3]$
  - There is an optimal frequency for minimal energy to solution
  - … at the price of significant performance loss



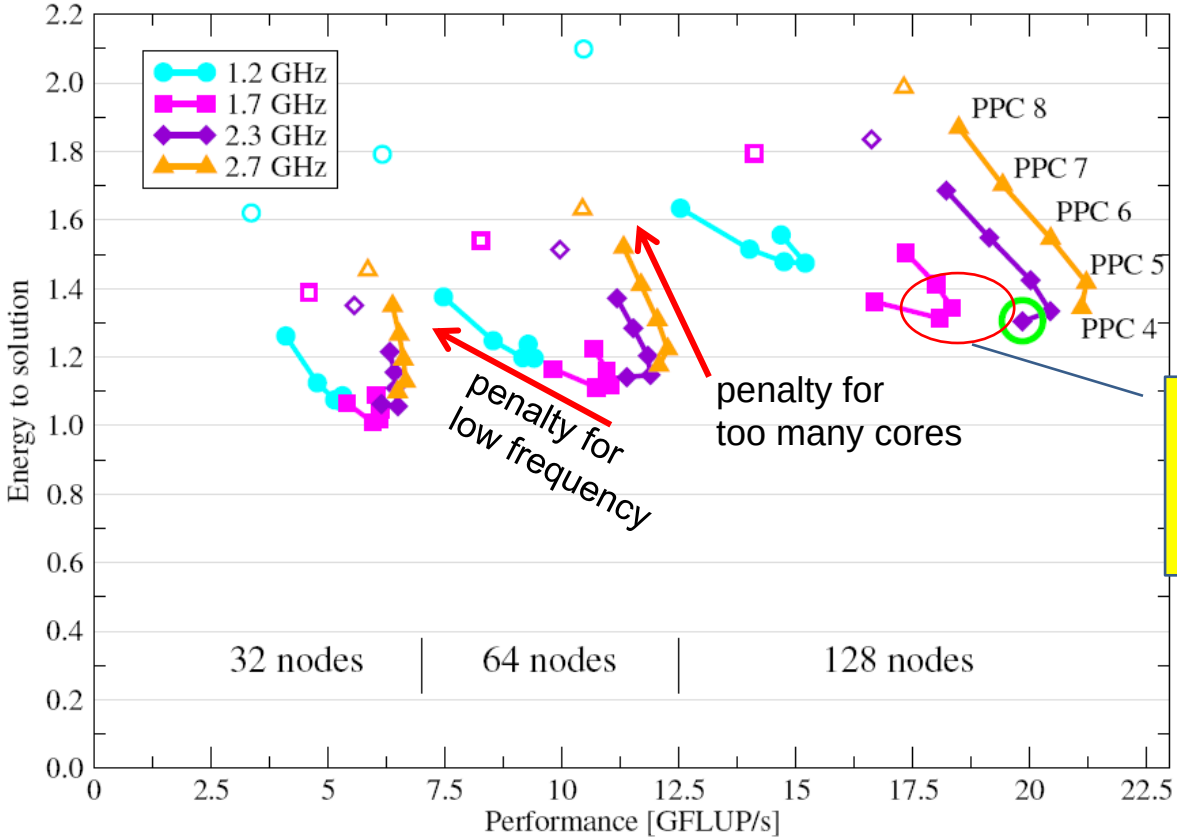
10% Δperf
@ zero ΔE

Analysis with Z-plots: LBM on SuperMUC: CPUs only



Wittmann et al., DOI: 10.1002/cpe.3489

… and taking a realistic W_0? → full node



Large W_0 enforces sharply defined optimal operating point

An energy model for multicore CPUs

- Utilization of memory interface:

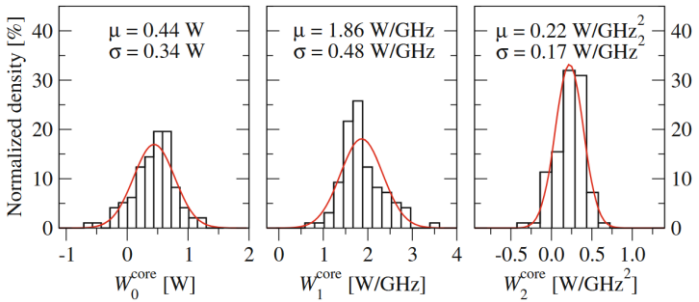
$$u(n) = \min\left(1, \frac{nT_{\mathrm{L3Mem}}}{T_{\mathrm{ECM}} + \bar{T}_{\mathrm{P}}}\right) = \min\left(1, \frac{nT_{\mathrm{L3Mem}}}{T_{\mathrm{ECM}} + (n-1)u(n-1)p_0}\right)$$

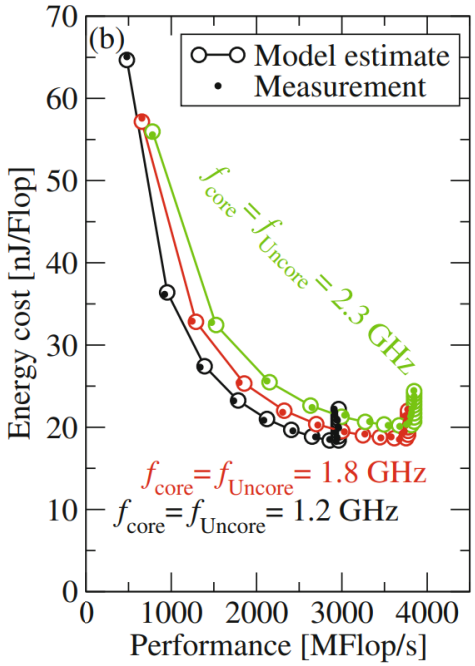
- Chip power:

$$P_{\mathrm{chip}} = P_{\mathrm{base}}(f_{\mathrm{Uncore}}) + nP_{\mathrm{core}}(f_{\mathrm{core}}, n)$$

$$P_{\mathrm{base}}(f_{\mathrm{Uncore}}) = W_0^{\mathrm{base}} + W_1^{\mathrm{base}} f_{\mathrm{Uncore}} + W_2^{\mathrm{base}} f_{\mathrm{Uncore}}^2$$

$$P_{\mathrm{core}}(f_{\mathrm{core}}, n) = \left(W_0^{\mathrm{core}} + W_1^{\mathrm{core}} f_{\mathrm{core}} + W_2^{\mathrm{core}} f_{\mathrm{core}}^2\right) \varepsilon(n)^{\alpha}$$





$$E(n, f_i) = P_{chip}(n) \times T_{min}/u(n)$$

J. Hofmann, G. Hager, D. Fey,
DOI: 10.1007/978-3-319-92040-5_2
ISC Gauss Award 2018

Energy-optimal frequency for scalable code?

- Assume $f_{core} = f_{Uncore} = f$ and set

$$\frac{\partial E(n,f)}{\partial f} = 0 \Rightarrow \qquad f_{\text{opt}} = \sqrt{\frac{W_0^{\text{base}} + nW_0^{\text{core}}}{W_2^{\text{base}} + nW_2^{\text{core}}}}$$

- Linear power-frequency behavior → $f_{opt} = \infty$

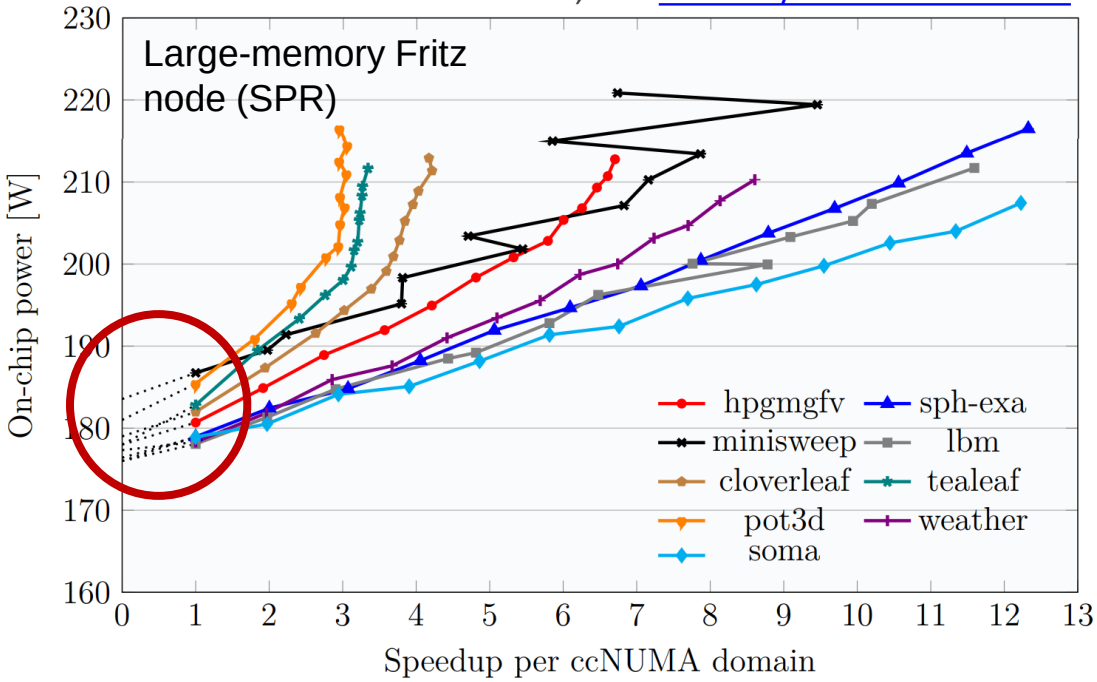
Chip baseline power today

- Much of the CPU/GPU chip power today is "baseline power"
- Extrapolation to "zero concurrency"

- Fritz:
  - ICL socket $W_0 \approx 100$ W (TDP = 250W)
  - SPR socket $W_0 \approx 180$ W (TDP = 350W)

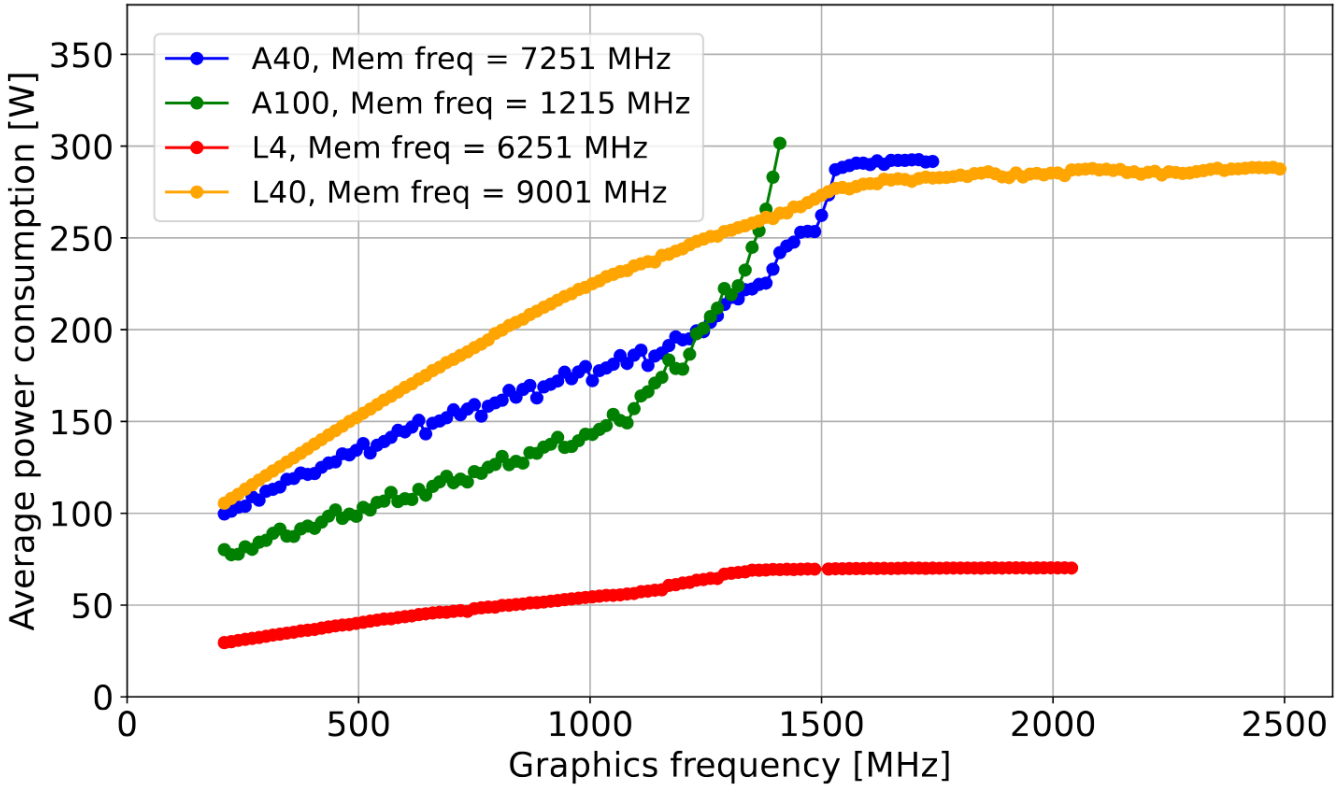
- Sandy Bridge (2012): 20% baseline power

Afzal et al., DOI 10.1145/3624062.3624197

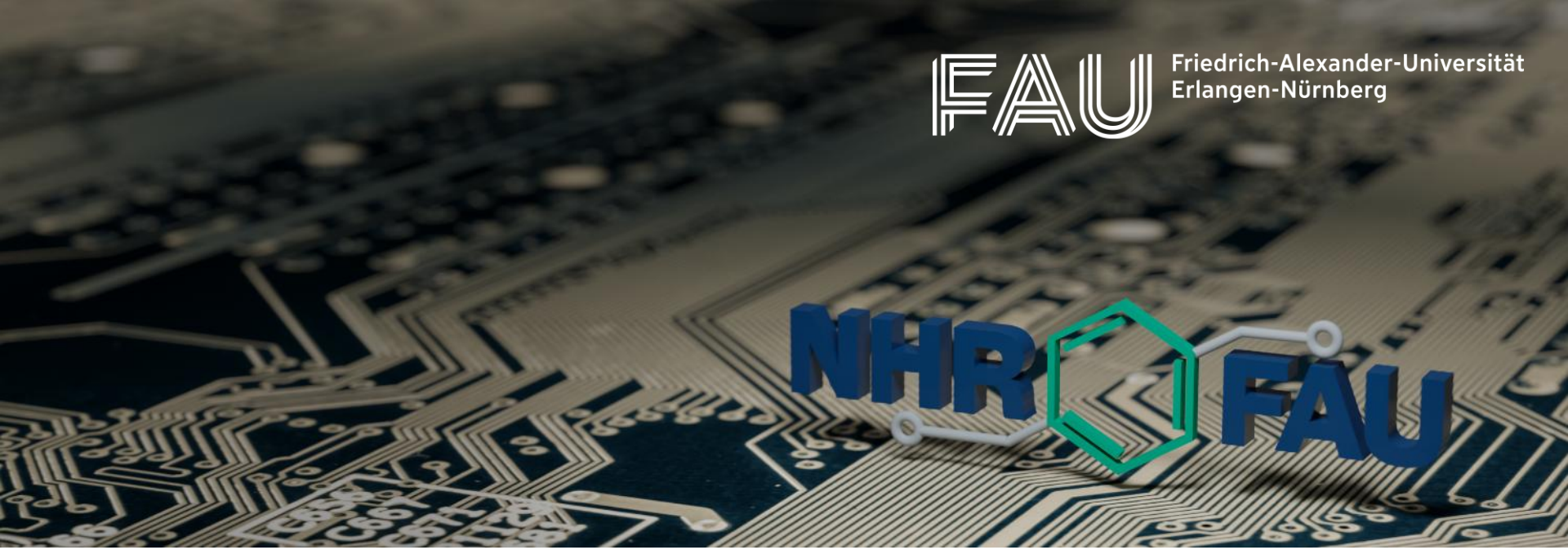


Power-frequency behavior with modern GPUs

Yuck.



Data by A. Afzal


Friedrich-Alexander-Universität
Erlangen-Nürnberg
NHR
FAU

Thank You.