NHR FAU

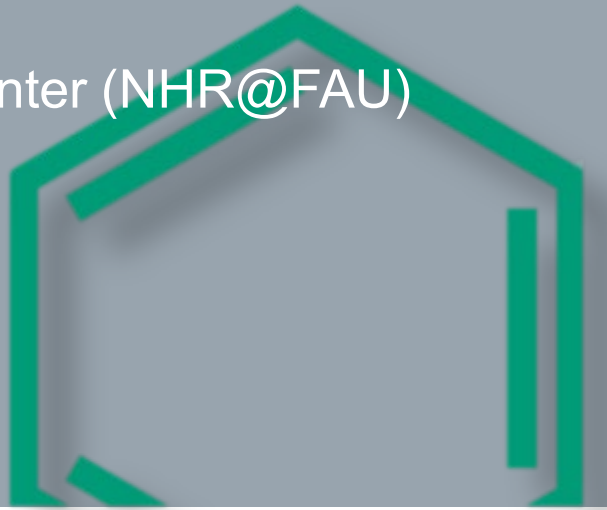
Friedrich-Alexander-Universität Erlangen-Nürnberg

Computer Architecture 101 for Scientists

Georg Hager

Erlangen National High Performance Computing Center (NHR@FAU)

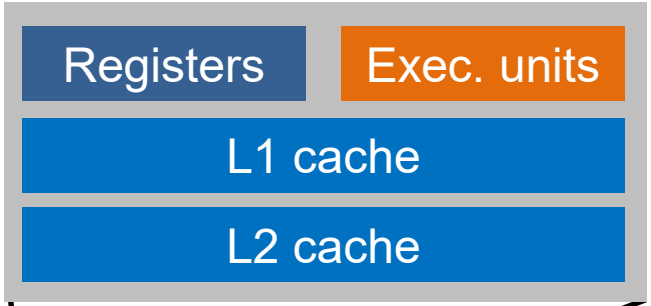
NHR@FAU HPC Café, 2023-06-13



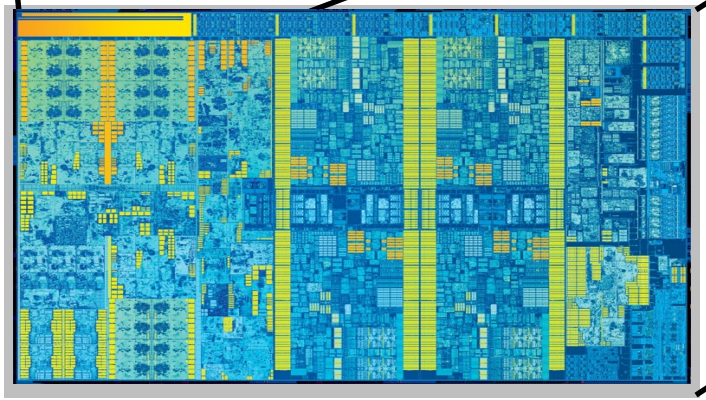
(Super)computer Architecture 101 Overview

- Hardware architecture
- Parallel programs
- Performance
- Limitations of parallelism
- Some best practices

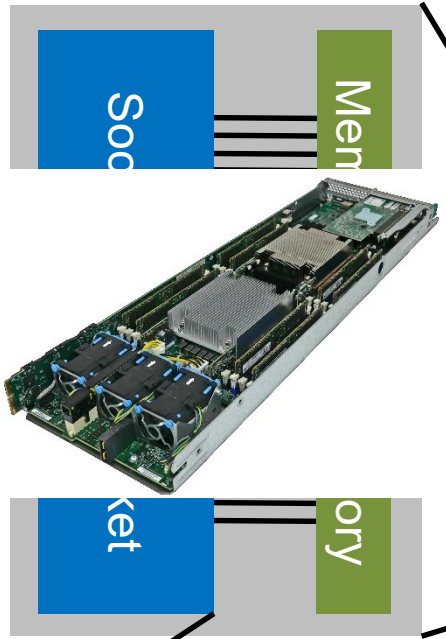
Anatomy of a (CPU) compute cluster



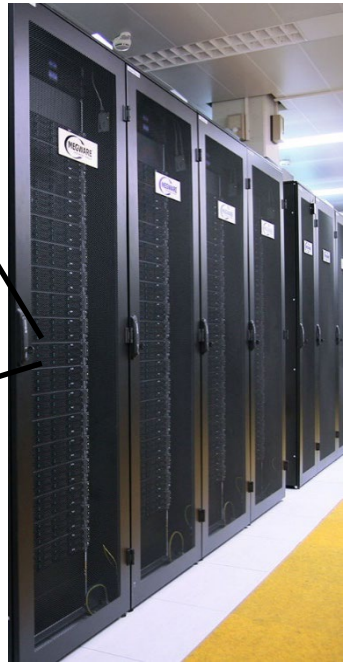
Core



Chip (up to 64 Cores)

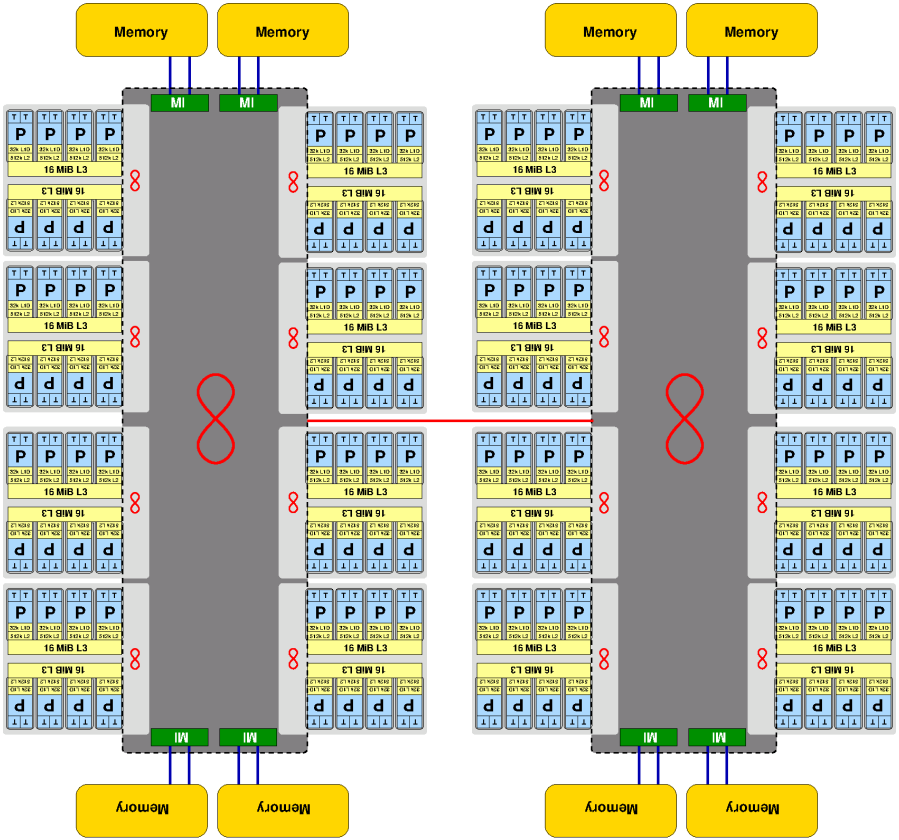


Node (2 sockets + memory + I/O, possibly multiple chips per socket)

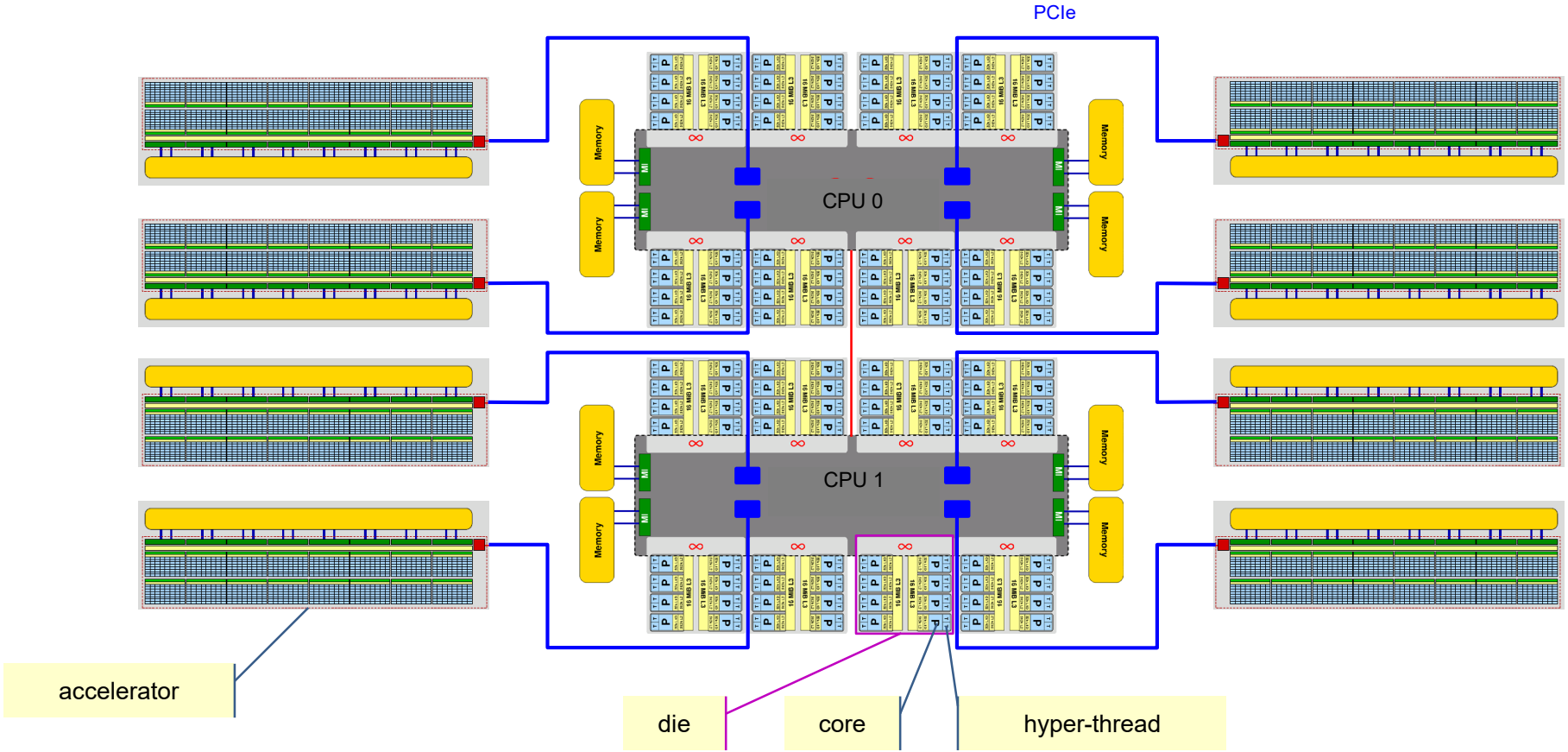


Supercomputer (many nodes, high-performance network, storage)

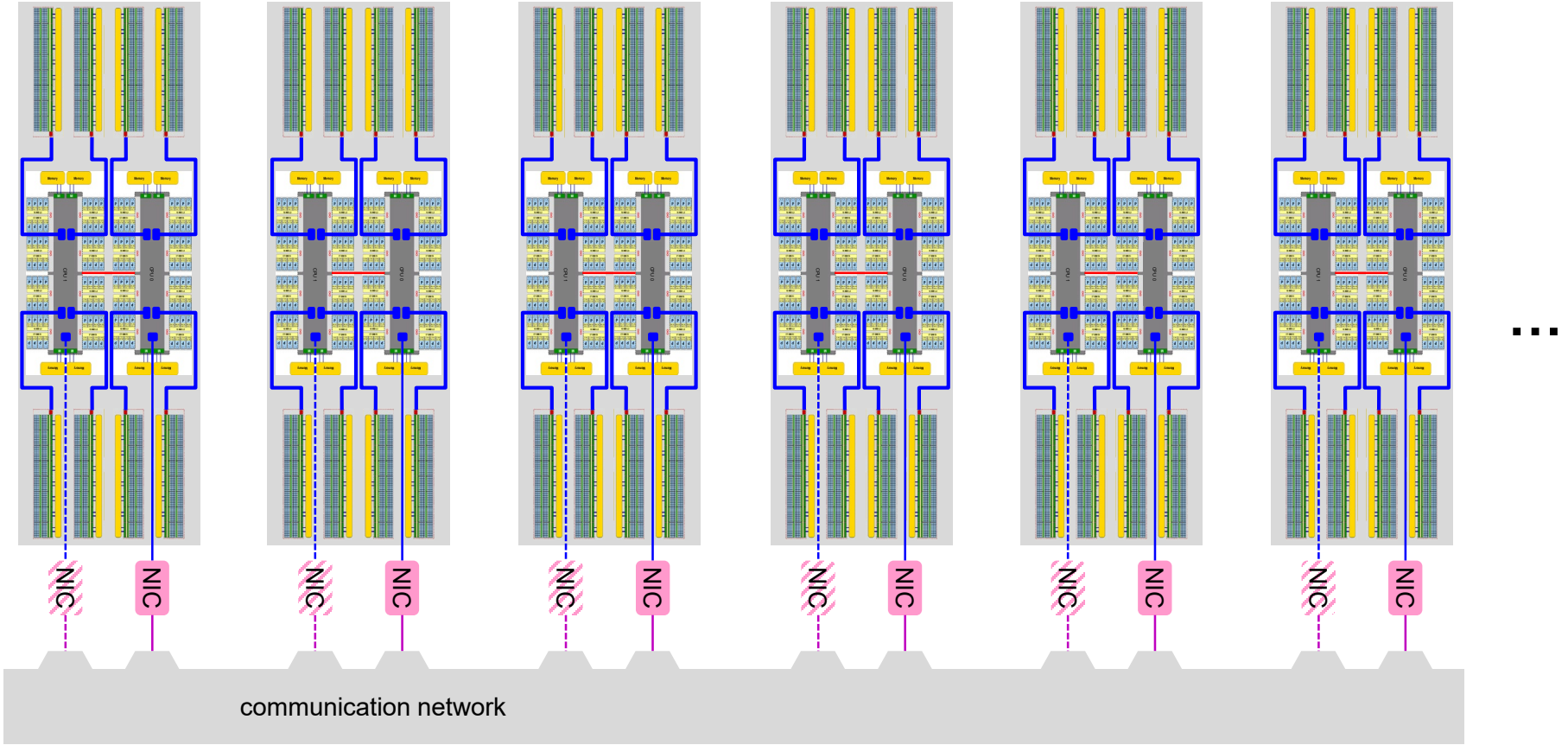
A modern CPU compute node (AMD Zen2 “Rome”)



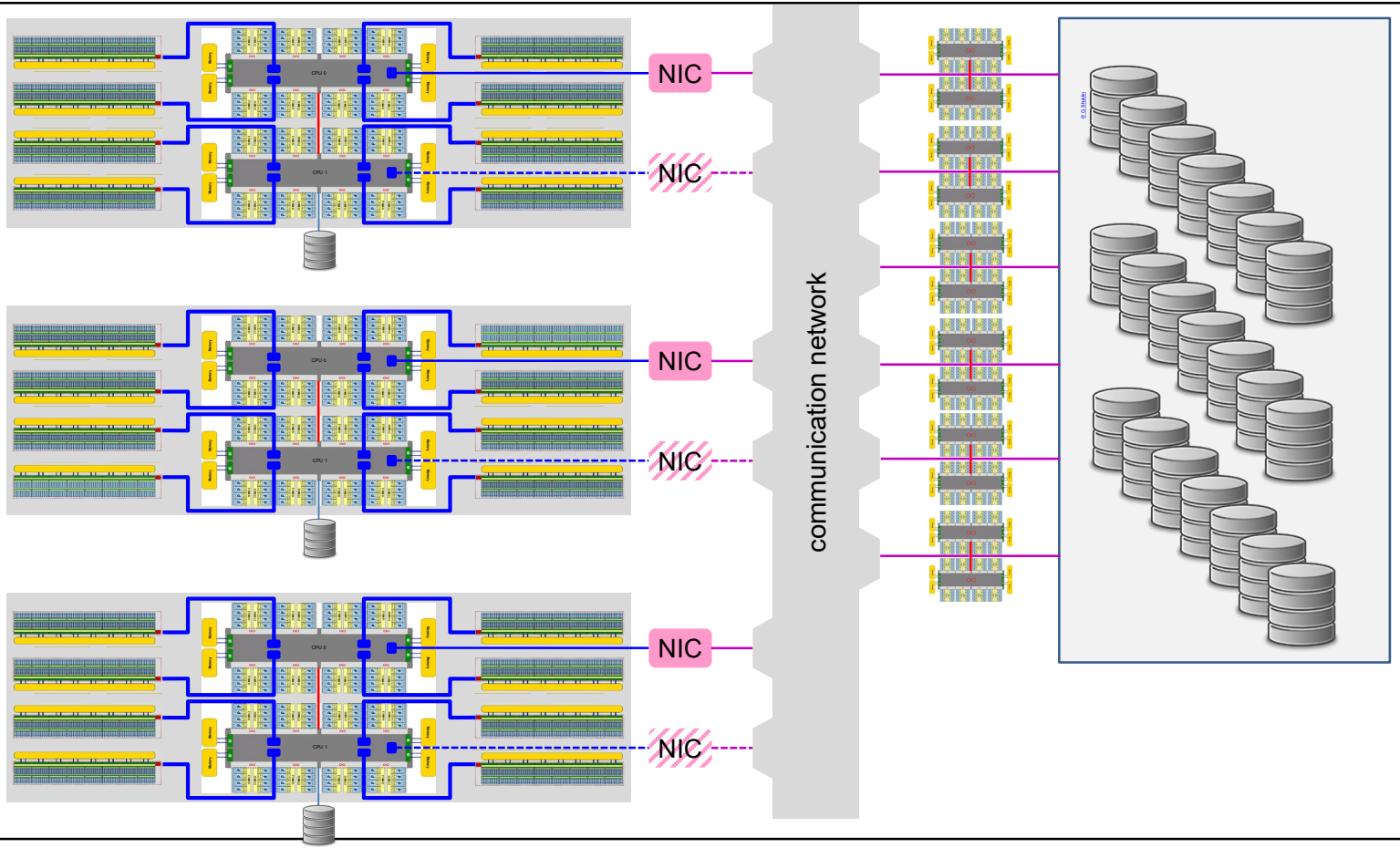
Adding accelerators to the node



Turning it into a cluster



Adding permanent storage

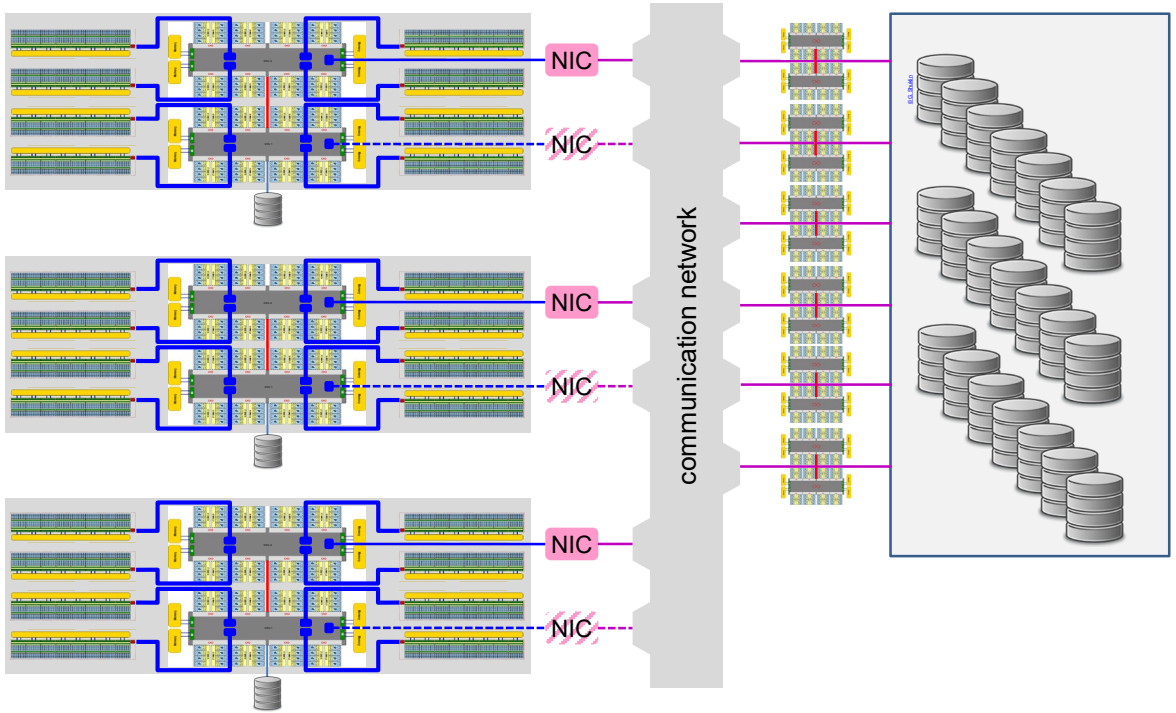


The crucial questions

Questions

- What are the hardware components that limit the performance of my code?
- What software properties limit the performance of my code?

- How should I know?
- What can I do about it?



Parallel computing

Task: Map a numerical algorithm to the hardware of a parallel computer

$$v_i = \sum_{j=1}^{n} A_{ij} b_j$$

Goal: Execute the task as fast and effectively as possible
→ “high performance”

What is “performance”?

Performance metric:

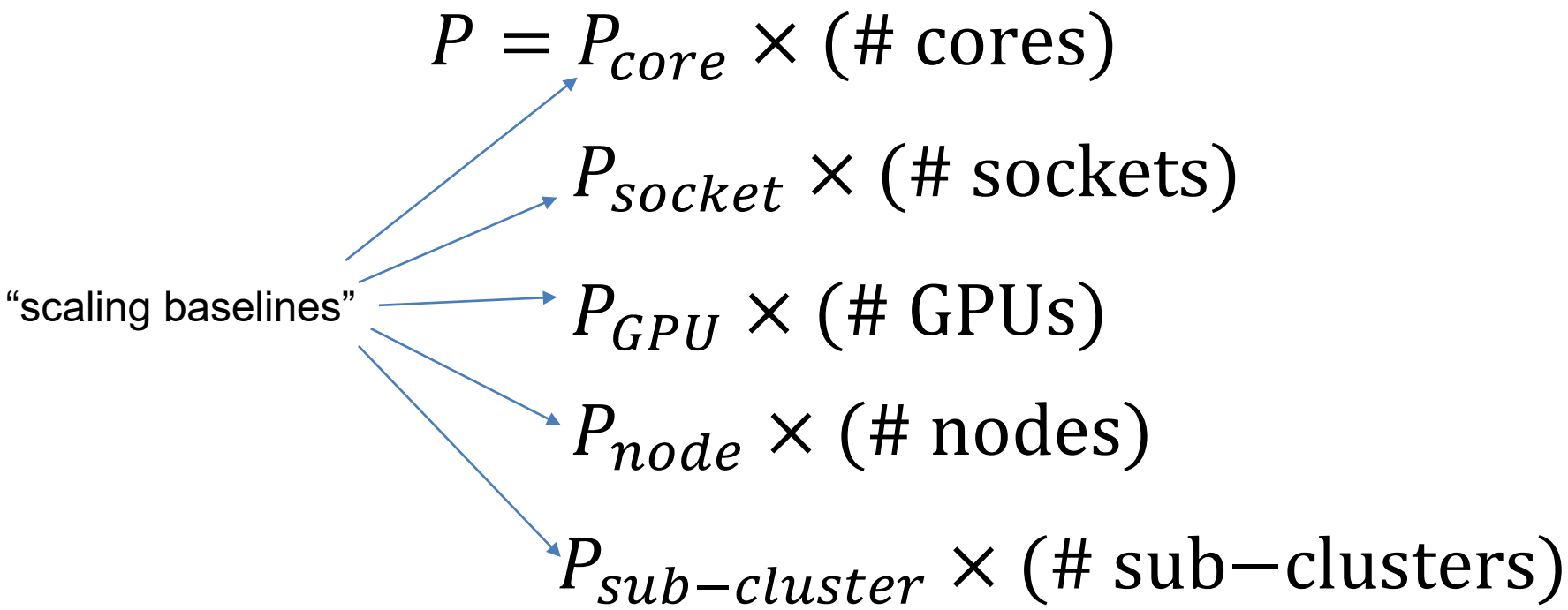
$$P = \frac{\text{Work}}{\text{Time}}$$

- # of **flops** (+ - * /)
- # of lattice site updates
- # of images processed
- ns of simulated time
- # of iterations
- “Solving the problem”...

“Wall-clock time”

Parallel performance

Performance is generated by parallelism!



Speedup

"How much faster can I compute with n times as much resources?"

cores GPUs nodes sockets …

$$S(n) = \frac{P(n)}{P(1)}$$

Best case (sort of): $S(n) = n$

Usual case: $S(n) < n$

Worst-case scenario: $S(n) < 1$

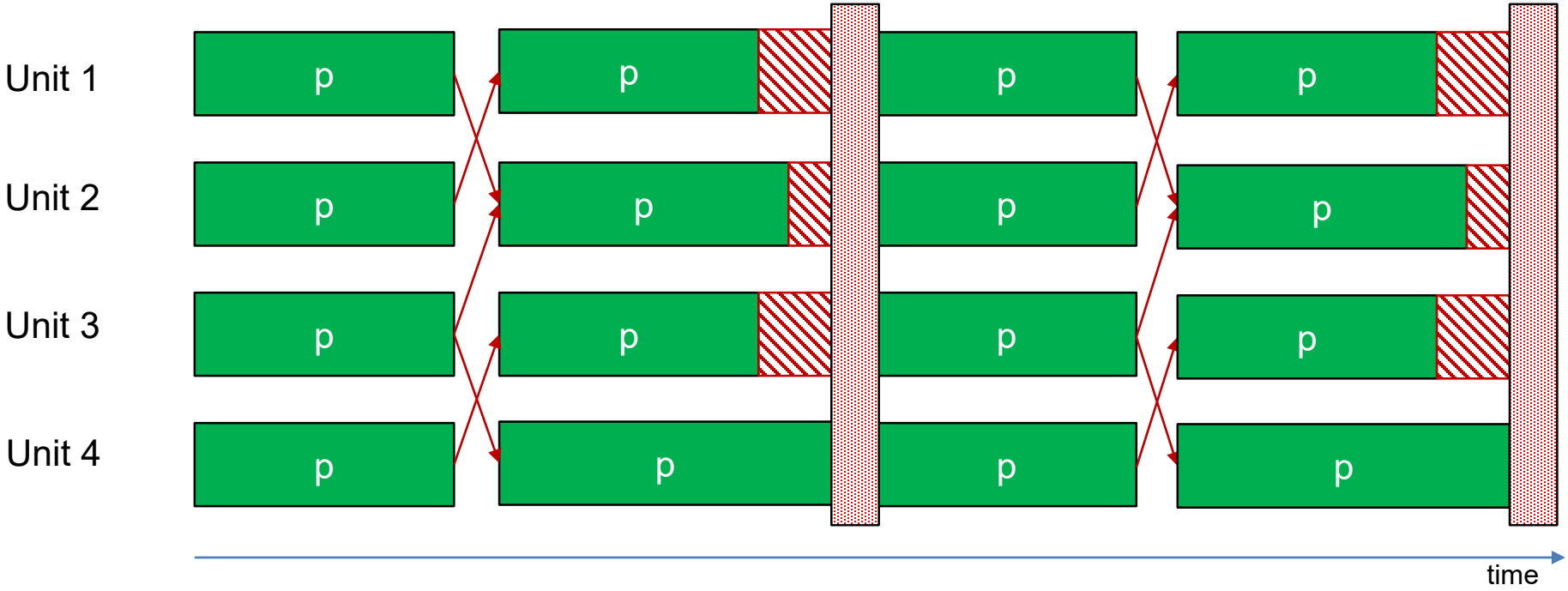
Parallel efficiency:

$$\varepsilon(n) = \frac{S(n)}{n}$$

Limitations on parallelism?

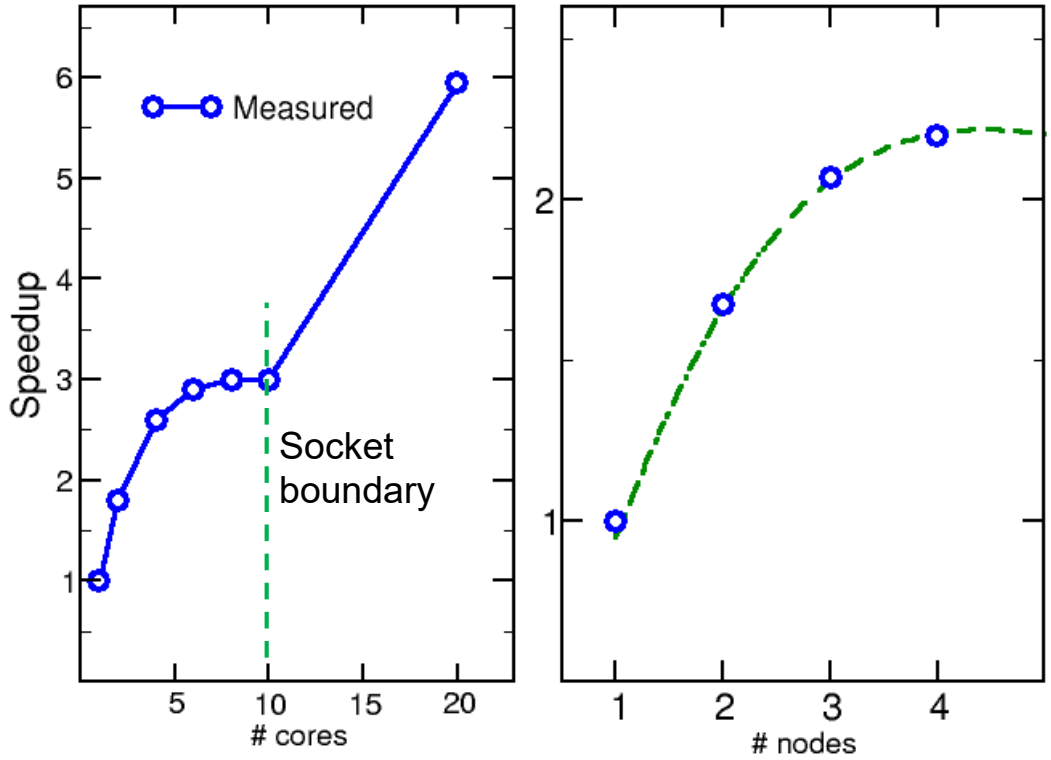
Yes. Lots, actually.

→ Communication, synchronization, work imbalance



Scaling baselines: Some resources do not scale

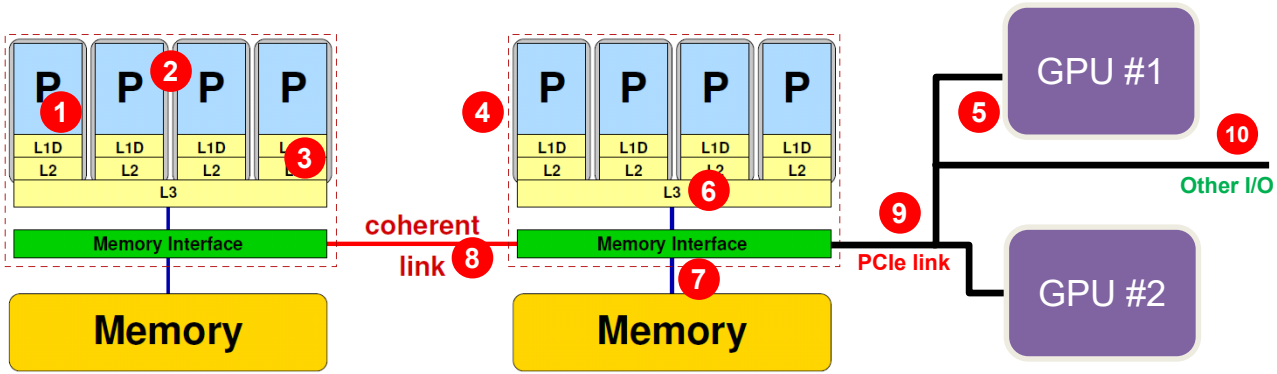
Scaling across cores, sockets, nodes



Does this code “scale”?

Scalablility of hardware components

Parallel and shared resources within a shared-memory node



Parallel resources:

- Execution units 1
- Cores 2
- Inner cache levels 3
- Sockets / memory domains 4
- Multiple accelerators 5

Shared resources:

- Outer cache levels 6
- Memory bus per socket 7
- Intersocket link 8
- PCIe bus(es) 9
- Other I/O resources 10

How does your application react to all of those details?

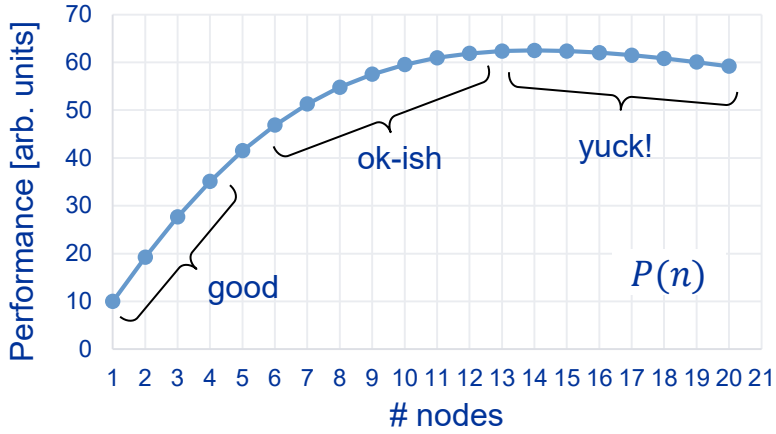
So what should I do?

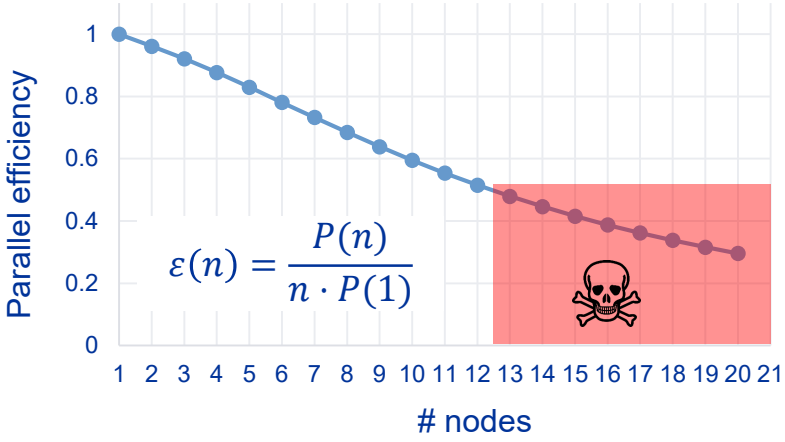
Assess the scaling properties of your code by benchmarking

- Scaling baseline: Basic allocation unit (node, GPU) first, then others
- Less than 50% efficiency is a blatant waste of resources
- Example: https://www.youtube.com/watch?v=j6bqq3dw0lQ

If you change the input (geometry, model, data set size), scaling will probably change, too

- Repeat scaling runs after significant changes to setup





What about performance (vs. scaling)?

- “Good” scaling does not mean that your code is fast
- It may still be that it makes bad use of the available main resources
  - Computational performance
  - Memory bandwidth

- Clustercockpit monitoring to the rescue
  - https://monitoring.nhr.fau.de
  - HPC Café (January 2023) on ClusterCockpit and the HPC Portal: https://www.fau.tv/clip/id/46327

ClusterCockpit example

Other things to consider

- Be considerate. Clusters are valuable shared resources that have been paid by the taxpayer
- Check your jobs regularly
  - Are the results OK?
  - Does the job actually use the allocated nodes in the intended way? Does it run with the expected performance?
  - Memory consumption? Disk quota exceeded?
- File storage is shared by all users!
  - https://www.fau.tv/clip/id/40199

Why should I care?

Minimum total cost (hardware, power, infrastructure, people) for one node-hour on modern HPC cluster: 0.5 €

1 node for 1 year: 4000 €

300 nodes for 1 day: 3600 €

This is money. Money that you burn for nothing cannot be used by others.

Nor by you.

NHR@FAU upcoming courses

- *Node-Level Performance Engineering*. Four-day online tutorial at HLRS Stuttgart, June 27-30 (in collaboration with ZIH Dresden)
- *Introduction to the LIKWID performance tool suite*. Full-day online tutorial, July 24
- *Python for HPC*. Three-day online course organized by MPCDF, July 25-27
- *Fundamentals of Accelerated Computing with CUDA C/C++*. Full-day on-site course at NHR@FAU, July 28
- *Fundamentals of Accelerated Computing with CUDA Python*. Full-day on-site course at NHR@FAU, September 18
- *Introduction to Parallel Programming with OpenMP, Part 1*. Full-day online course, September 20
- *Introduction to Parallel Programming with OpenMP, Part 2*. Full-day online course, September 27
- *Node-Level Performance Engineering*. Three-day on-site tutorial at NHR@FAU, October 4-6
- *Performance analysis on GPUs with NVIDIA tools*. Half-day online course, October 10
- *C++ for Beginners*. Six-day online course, September 14/15, 21/22, and 28/29
- *Modern C++ Software Design*. Three-day online course, October 11-13
- *Node-Level Performance Engineering*. Three-day online tutorial at LRZ Garching, December 4-6

See also: https://hpc.fau.de/teaching/tutorials-and-courses/