How to parallelize
with CPUs & GPUs

15.6.2023

Lewin Stein

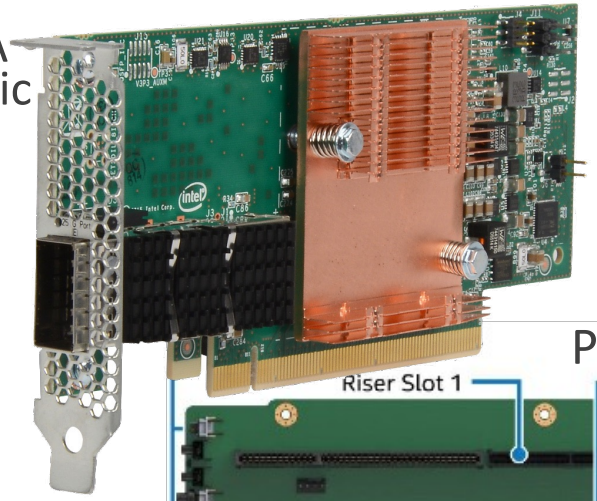
Structure

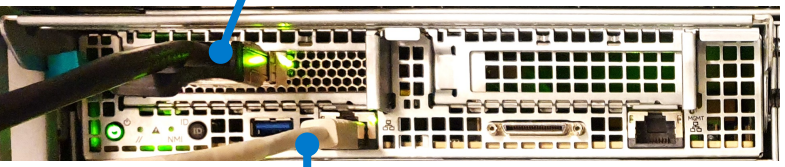
1. Hierarchy of the hardware architecture
2. Parallelization approaches and their implementation
3. Parallelization in CFD: spatial decomposition
4. How to evaluate your parallelization

Intel Xeon Cascade Lake (CXL) node

QSFP28

100G OPA
Host Fabric
Interface

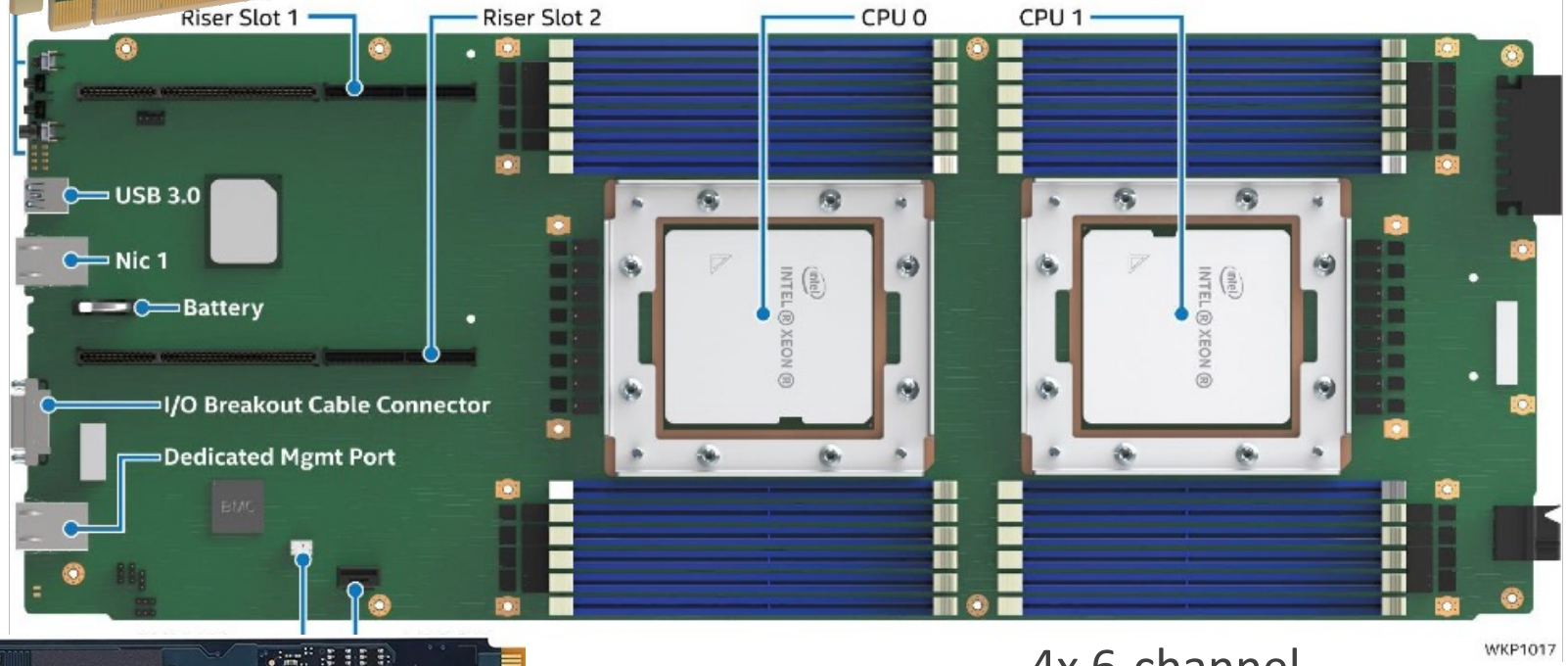




ethernet

384-1522GB total RAM

PCIe v3.0 x16



power
connec-
tors

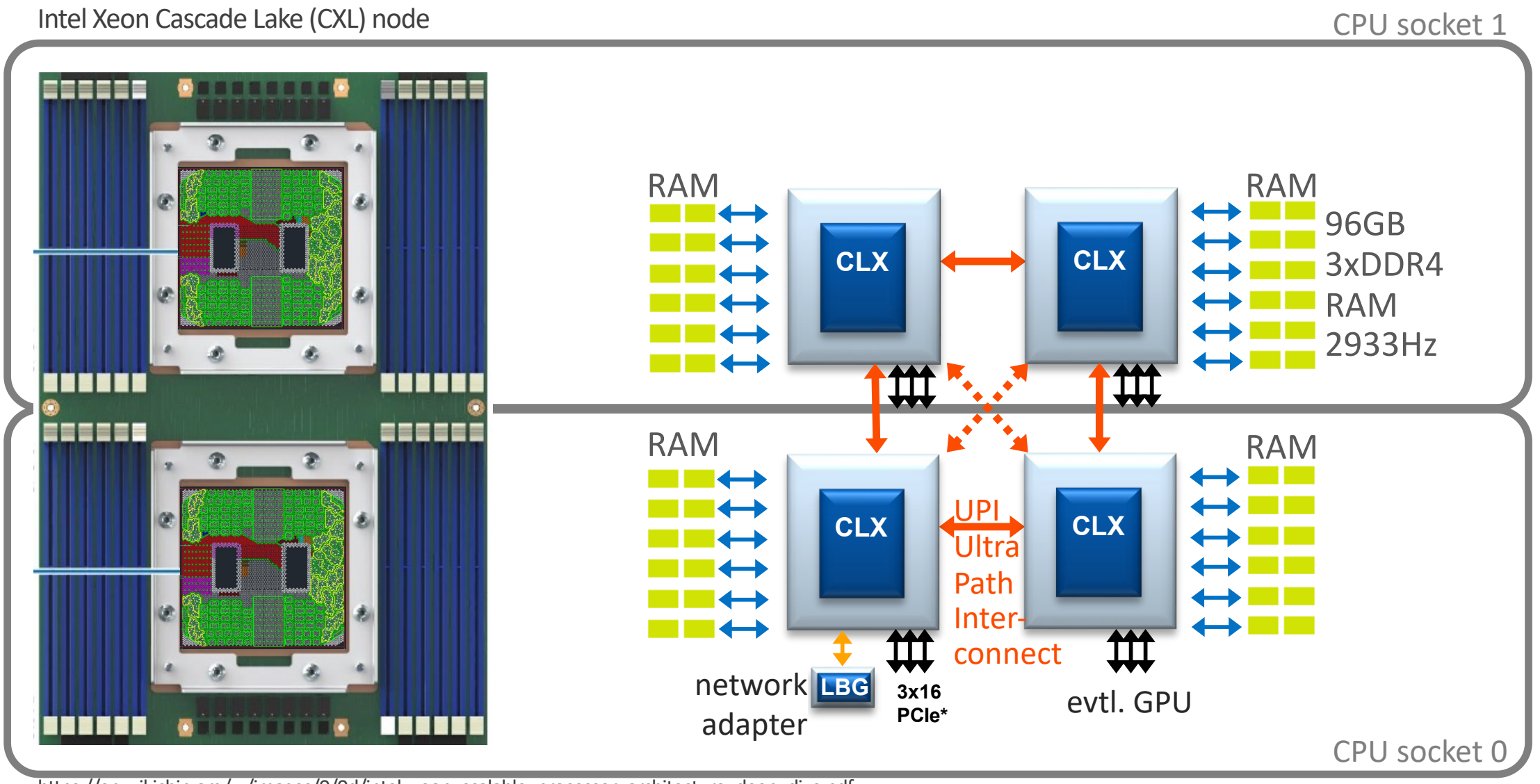
NVMe SSD up to 2TB

M.2

4x 6-channel
DDR4-2933

2 sockets & 4 NUMA domains/dies per node

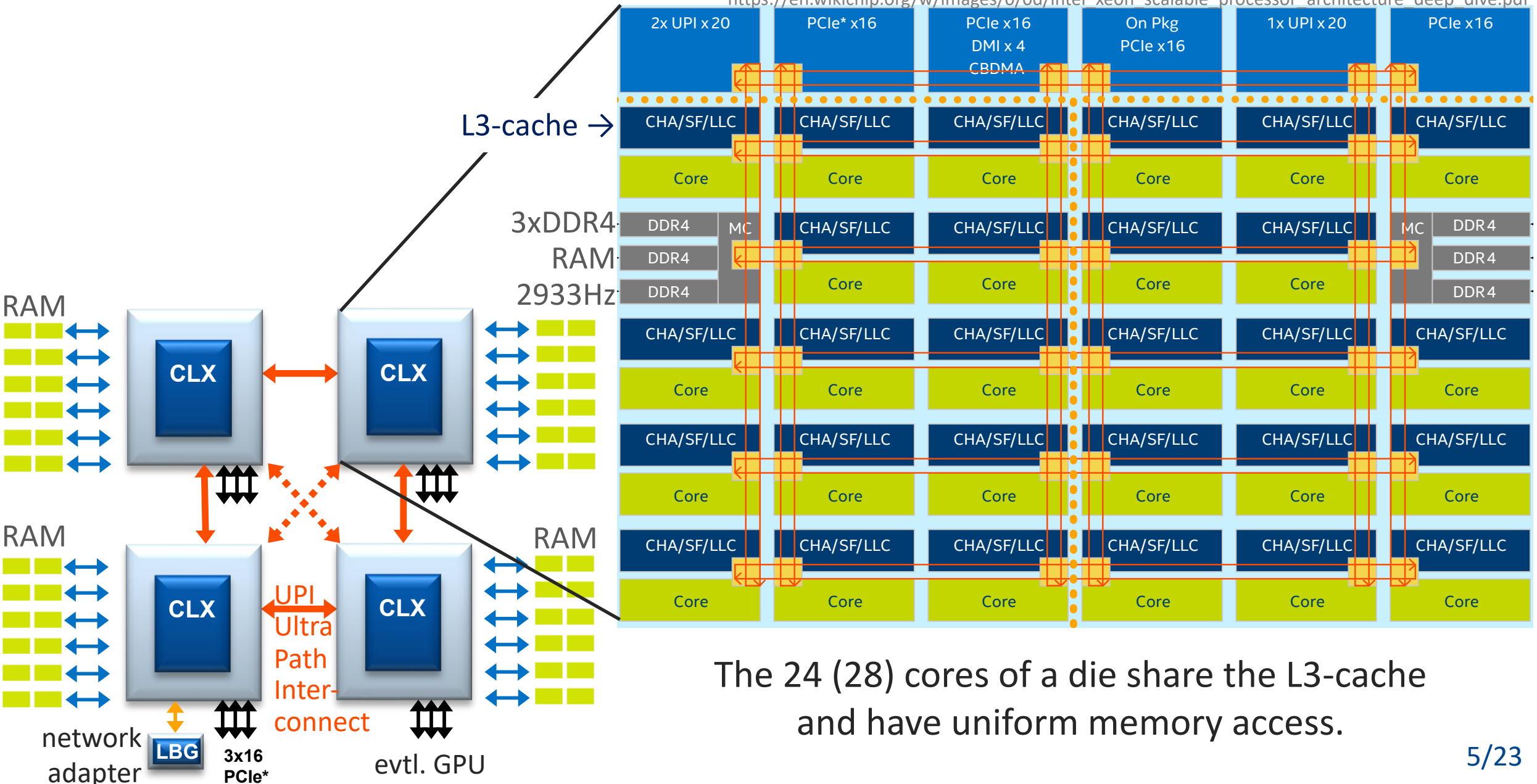
Intel Xeon Cascade Lake (CXL) node



https://en.wikichip.org/w/images/0/0d/intel_xeon_scalable_processor_architecture_deep_dive.pdf

24 cores per NUMA domains/die

https://en.wikichip.org/w/images/0/0d/intel_xeon_scalable_processor_architecture_deep_dive.pdf



The 24 (28) cores of a die share the L3-cache and have uniform memory access.

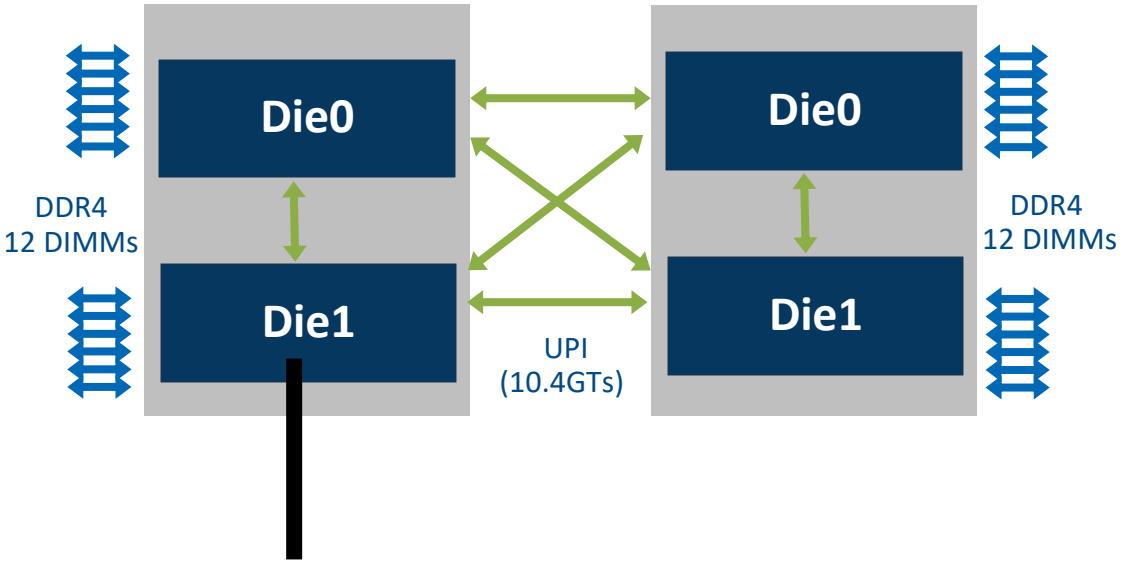
Hardware scales/hierarchy of a CPU node

optimization chances
hlrn.de/doc/display/PUB/Special+Filesystems

	unit	children of unit	accessible storage/memory	
shared	**switch network**	24 nodes per switch	HDD “work”	SSD “work”
exclusive	**node**	48x2 cores (2 sockets)	local SSD @PCIe	
	socket	24x2 cores (2 dies)	NUMA	
	die	24 cores	UMA, L3-cache	
	core		L1,L2-cache	

natural scaling / locality steps

Cascade Lake node



NVIDIA A100 GPU (80GB)

https://images.nvidia.com/aem-dam/en-zz/Solutions/data-center/nvidia-ampere-architecture-whitepaper.pdf

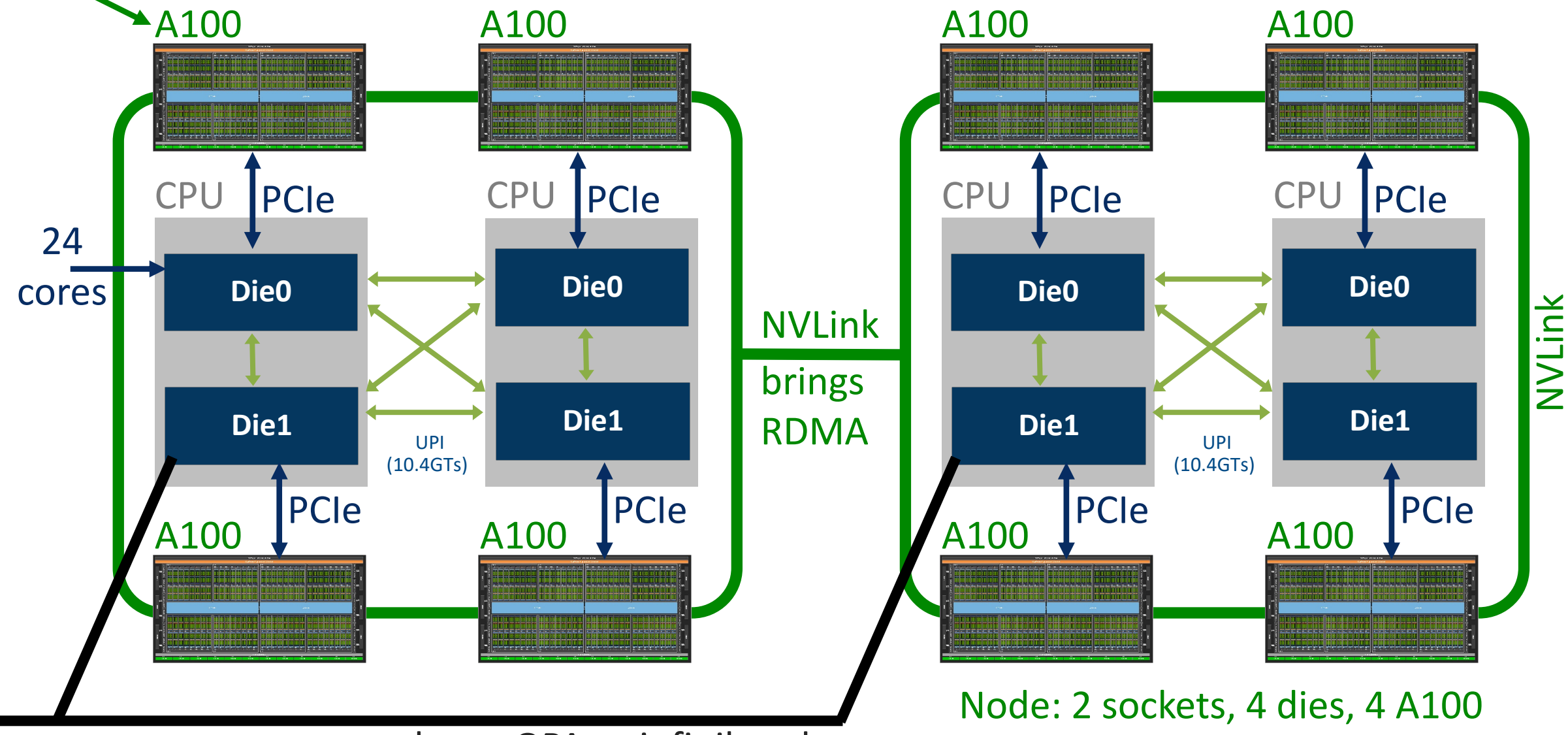
Streaming Multiprocessor (SM)

an SM has: shared L1-cache
4 Tensor Cores (FP16-32)
64 FP32 CUDA Cores
64 INT32 CUDA Cores
32 FP64 CUDA Cores

The 108 (128) SMs of a A100 die share the L2-cache and have uniform memory access.

Links between two CPU-GPU nodes

Fixed roles: CPU is host, GPU accelerator/offload device



How to parallelize at all scales

	typical range	memory	class / style	API	name of virtual unit	size of unit
high	inter node local network	distributed (RDMA)	**M**ultiple **I**nstruction **M**ultiple **D**ata (typically 1 process per core)	MPI	processes / ranks	memory per core
mid	intra node	shared (N)UMA	**M**ulti**T**hreading (multiple threads for multiple cores)	OpenMP, (MPI)	threads	< L3-cache per core
low	intra die	shared UMA (L3-cache)	**S**ingle **I**nstruction **M**ultiple **D**ata (on one thread and one core)	SYCL OpenCL/-MP	processing elements	register size (CLX: 512 bit)
	intra GPU (accelerator)	shared within GPU	**S**ingle **I**nstruction **M**ultiple **T**hreads (using multiple threads and cores)	CUDA, SYCL OpenCL/-MP	threads	memory per GPU SM

⟶ hierarchy provides program structure

CPU versus GPU

one big core	**S**treaming **M**ultiprocessor made of many small cores
general HPC applications	massively data parallel applications
high and low level parallelization	low level parallelization e.g. vectorization or SIMT
minimal memory latency	optimized for data throughput / memory bandwidth (best for large problem sizes)
can handle random memory access	most efficient for sequential/linear memory access

Which API to choose?

Application Programming Interface	Programming language			CPU			GPU		
	C/C++	Fortran	Python	Intel	AMD	ARM	Intel	AMD	Nividia
MPI	✓	✓	(✓)	✓	✓	✓	~	~	~
OpenMP (directives)	✓	✓	(✓)	✓	✓	✓	✓	✓	✓
OpenCL	✓	~	(✓)	✓	✓	✓	✓	✓	~
SYCL (DPC++, hipSYCL, triSYCL, …)	✓			(✓)	(✓)	(✓)	(✓)	(✓)	~
ROCm / HIP (AMD)	✓	(✓)	(✓)	~	~	~		✓	✓
CUDA (Nividia)	✓	(✓)	(✓)						✓

Legend	
✓	directly
(✓)	via extension
~	limited functionality

Ordering matters at all scales

high

- Network

  an edge switch holds odd or even nodes (edge[1]⇄bcn[1,3,5,7,…,47]) → slurm places jobs “edge aware”

  (and ignors leafs/directors ☹)

mid/ high

- Pinning/Affinity “Bind process to core” (min. cache misses), MPI flags:

  `scatter` (intel) / `rank-by L3cache:span` (gnu) → max. memory bandwidth for partial node use

  `compact` (intel) / `map-by core` (gnu) → faster copy between cores (closer memory)

low

- Data Structures/Arrays

  Array **o**f **S**tructs (people[i].name: Bob,5; Eve,7) → intuitive/readable (AoSoA needed if SoA to big)

  Struct **o**f **A**rrays (people.name[j]: Bob,Eve; 5,7) → item of same type more close (SIMD/T)

low

- Nested loops (linear access of cache lines, memory, storage)

  Colum major order (A[i,j,k]): outer-k, mid-j, inner-i loop) → Fortran

  Row major order (A[i,j,k]): outer-i, mid-j, inner-k loop) → C/C++, Python

Local physics scales best

space/time local:

- transport processes (hyperbolic), diffusion (parabolic)
- example: compressible fluids
- information spreads at finite speed

algorithms suggested by locality:

- explicit (time) integration
- differentiation operator with a few side diagonals

parallelization suggested by locality:

- partitioning in “local” blocks exchanging boundary information only (MPI_Send & Receive)

space/time global:

- steady state (elliptic PDEs)
- example: incompressible fluids, stiff systems
- information spreads at infinite speed

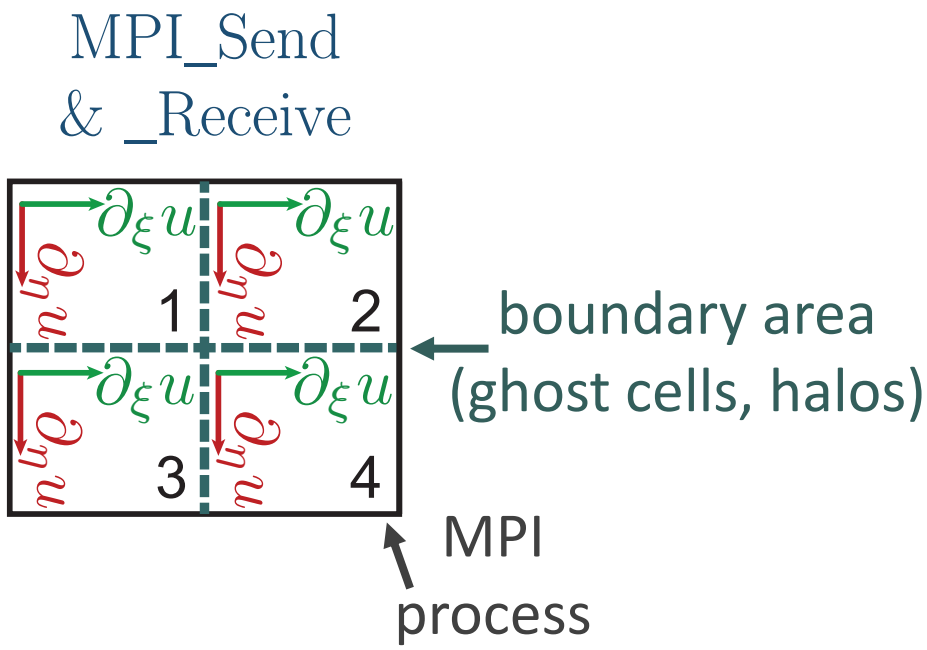
algorithms accompanying globality:

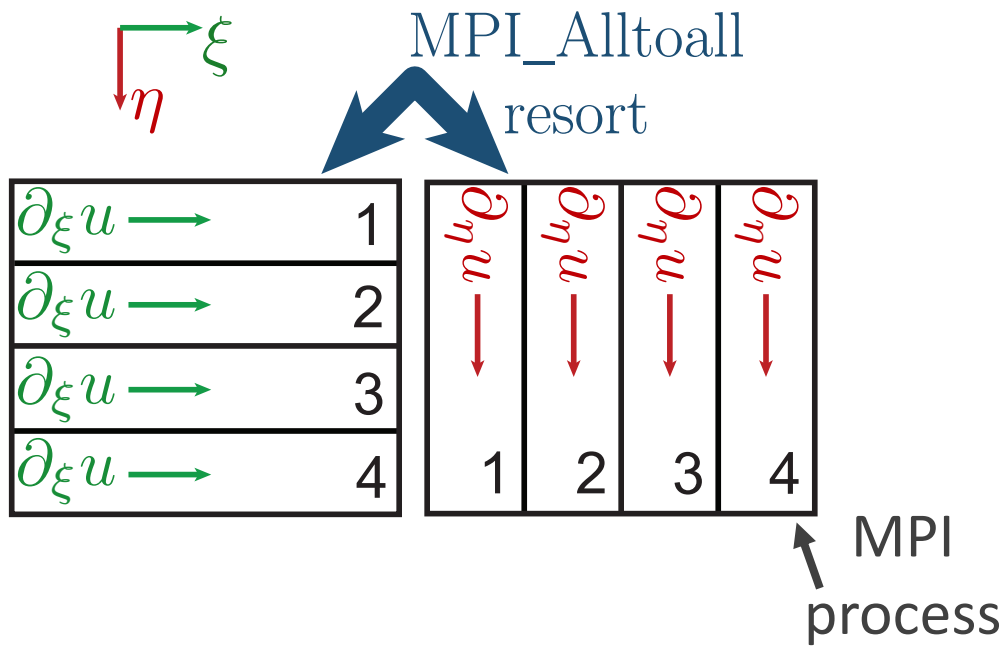
- implicit/iterative (time) integration
- compact/spectral operators (full matrix)

parallelization suggested by globality:

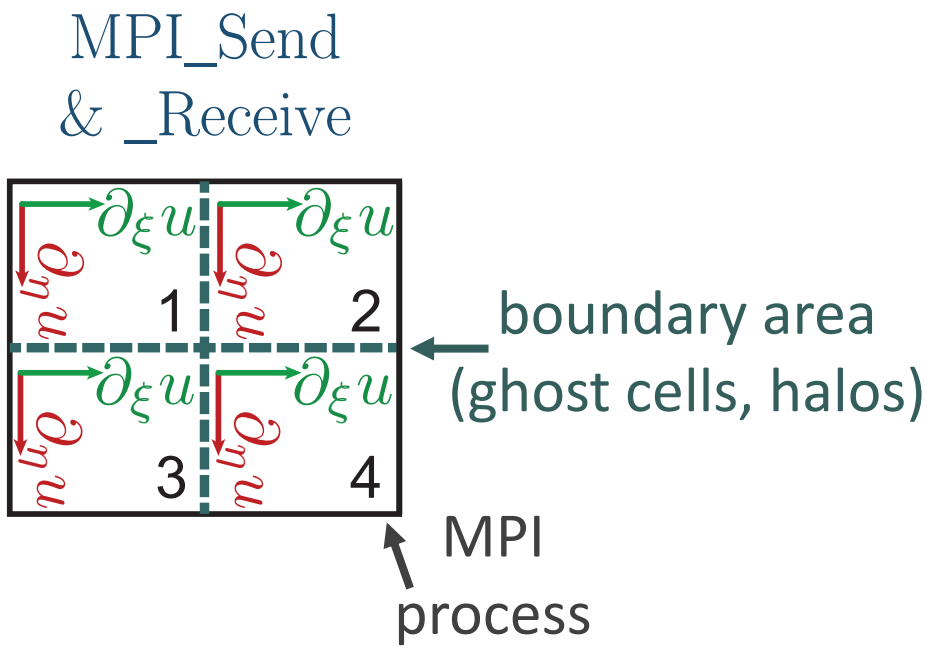
- partition-switching providing all info. of at least one dimension at once (MPI_Alltoall)

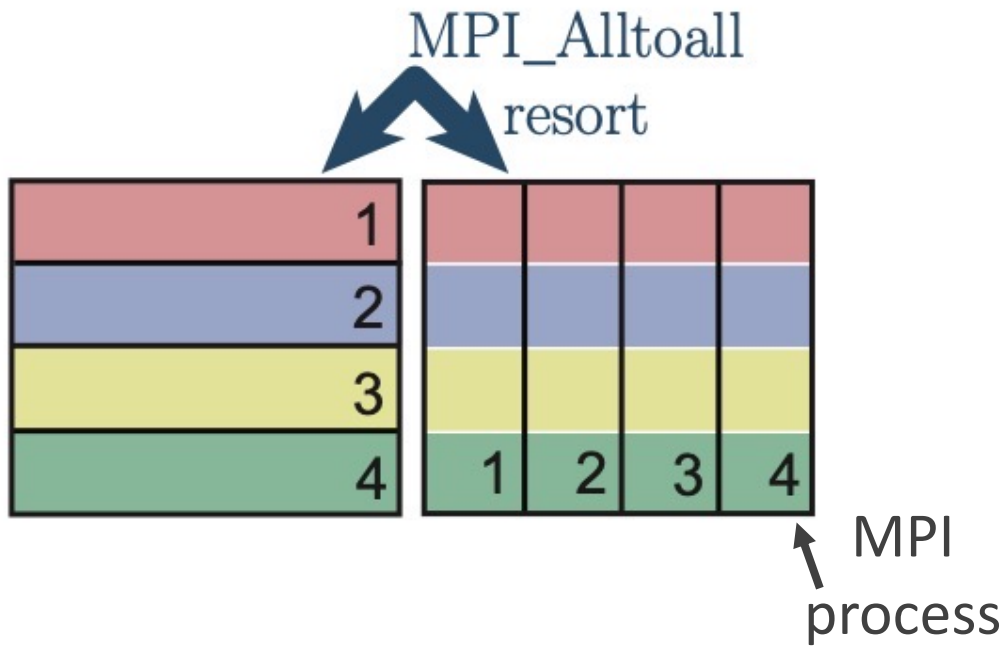
Multi-Block vs. “FFTW” partitioning





Multi-Block vs. “FFTW” partitioning



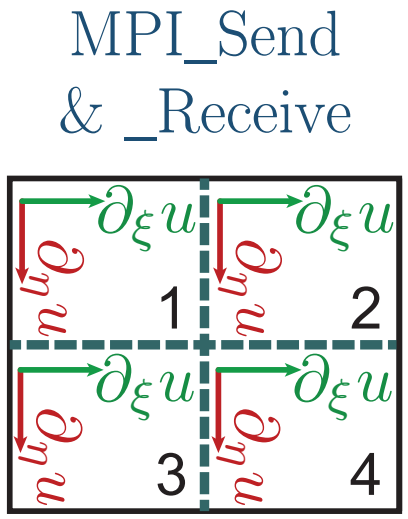


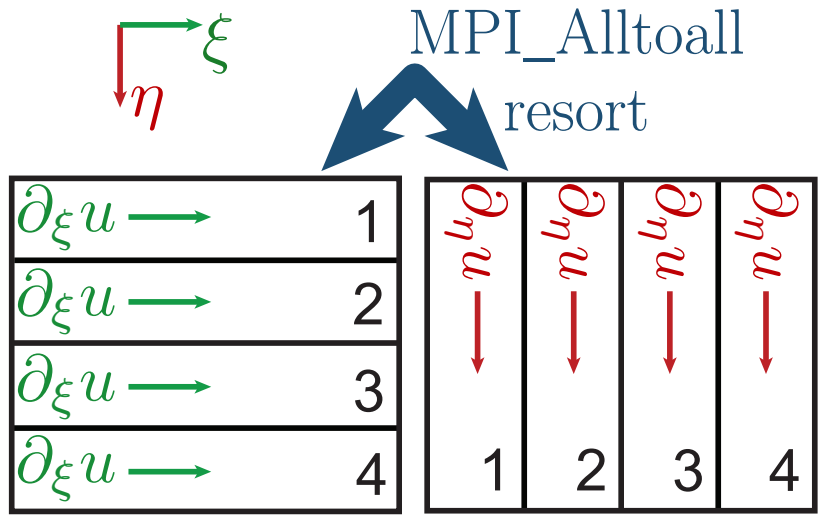
weak-scaling overhead of communication
$= \text{boundary area}/\text{total area}$

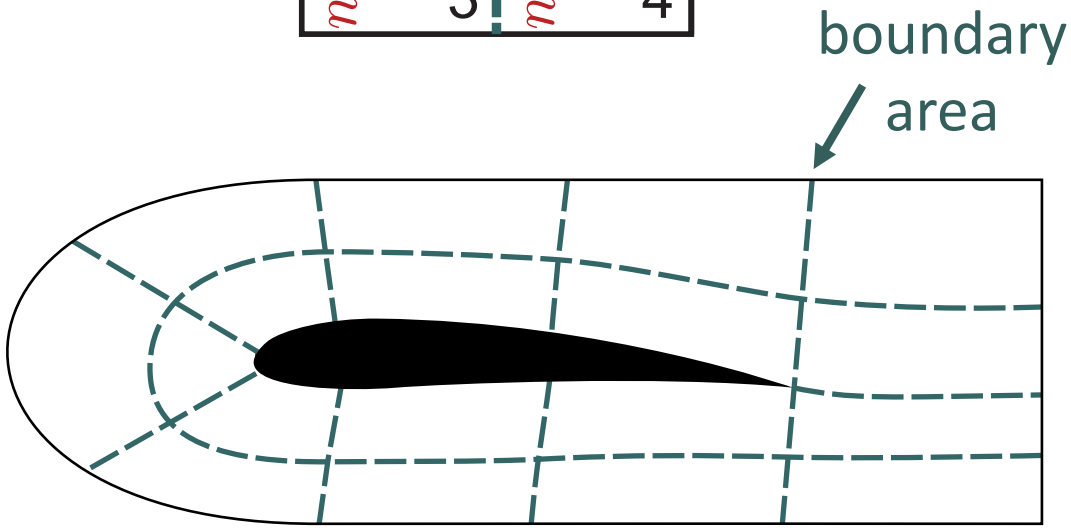
weak-scaling overhead of communication
$\propto 1-1/np$

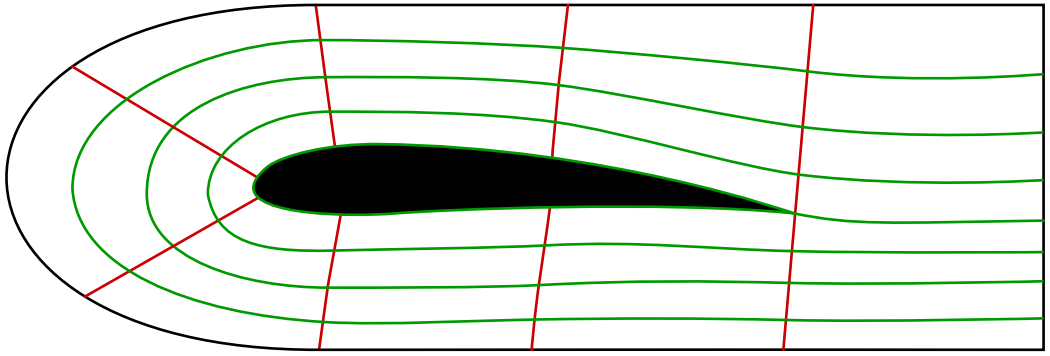
number
of processes

Multi-Block vs. “FFTW” partitioning

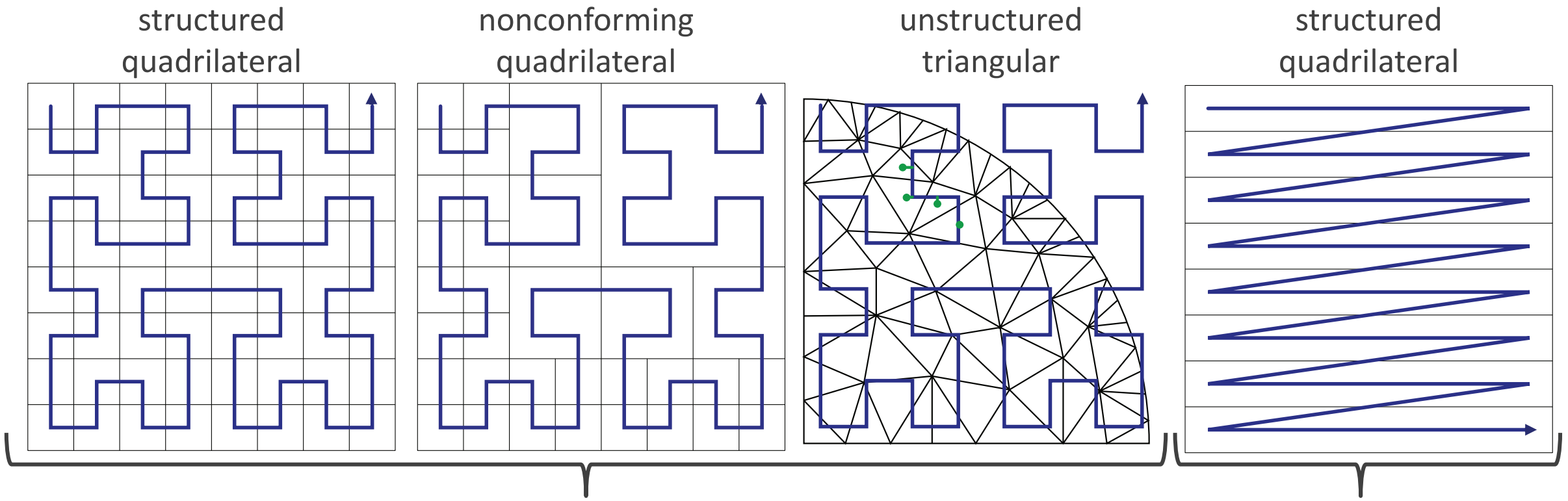




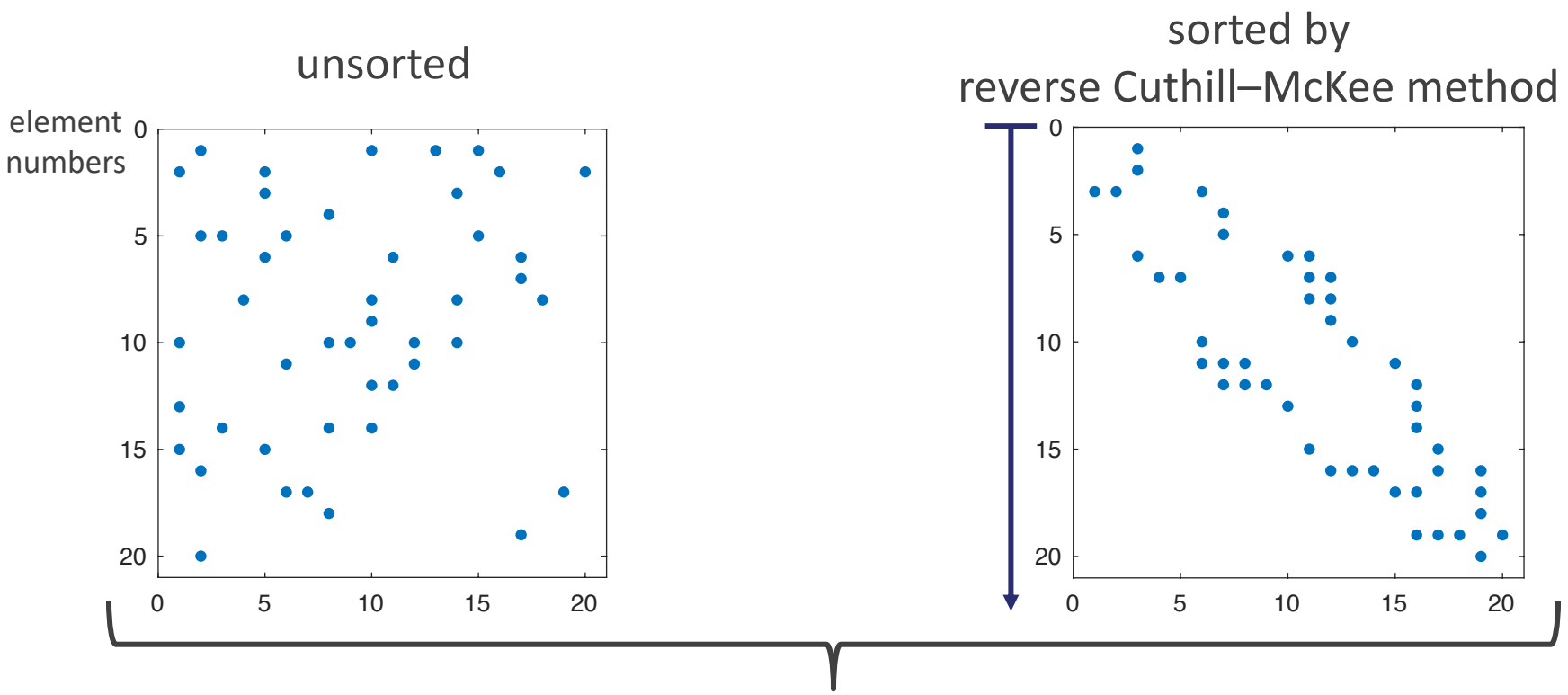




Ordering a mesh most locally



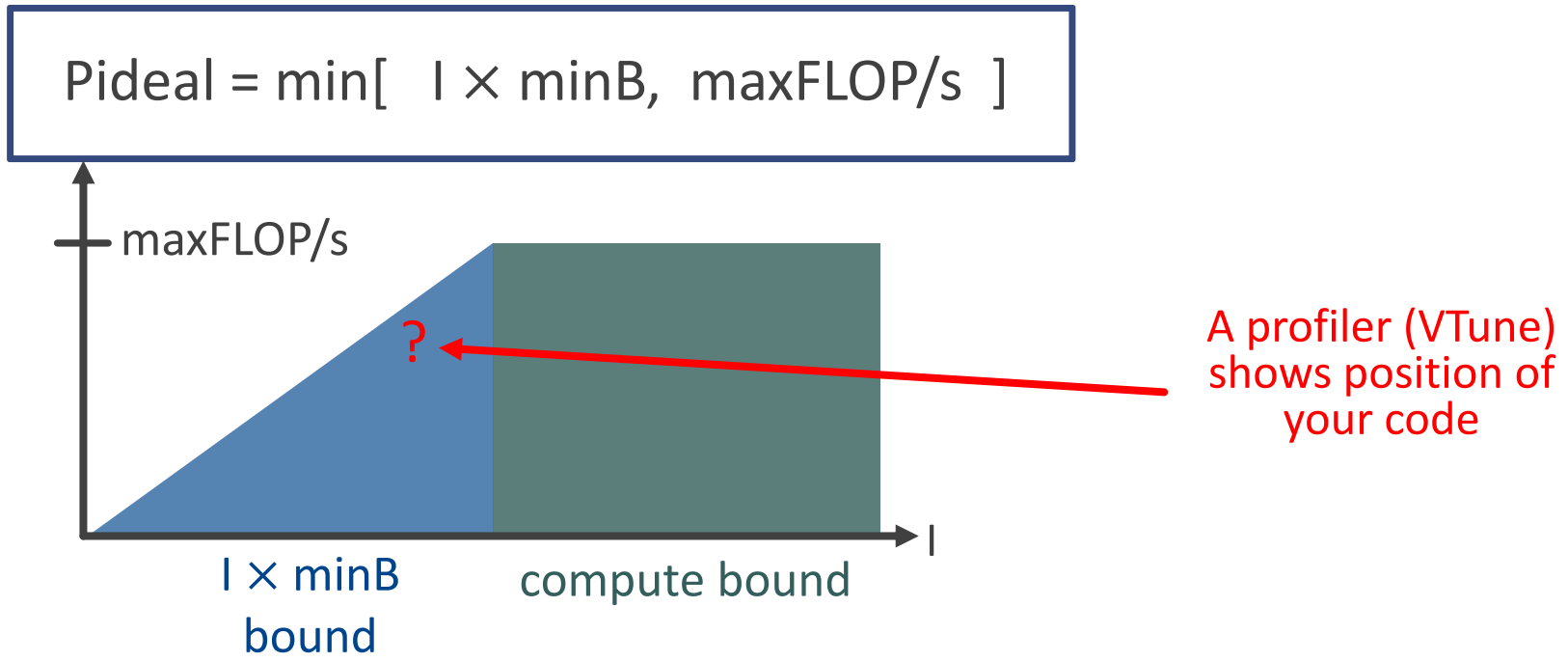
Ordering a mesh most locally



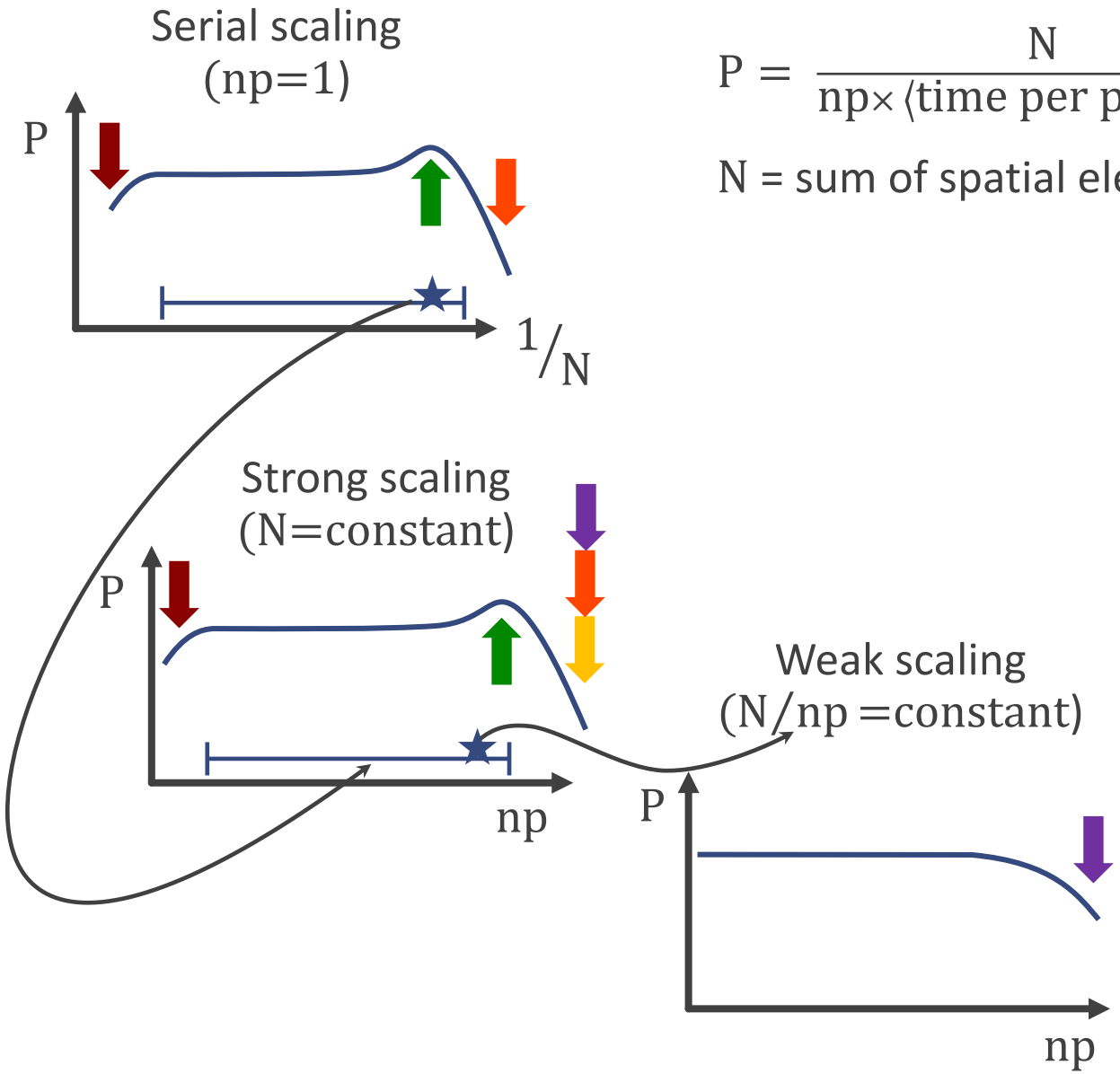
Roofline model → optimization idea

The max. theoretical performance "**P**ideal" is limited by

- maxFLOP/s = maximal **FL**oating point **OP**erations per second conducted by a certain **c**om**p**utation **u**nit (cpu)
- minB (**B**andwidth) = minimal byte/s = transfer rate of slowest data path (bandwidth of main memory or cache)
- I (arithmetic **I**ntensity) = FLOP/byte = FLOP/(size of instruction needed to conduct the operation)



Enemys of ideal scaling & sweet spot ★



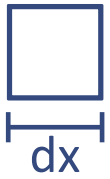
$$P = \frac{N}{np \times \langle \text{time per process} \rangle}$$

N = sum of spatial elements

Possible effects:

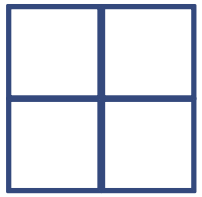
- cache match
- memory swapping [`free -m`]
- communication complexity (inter core)
- low arithmetic intensity (instructions>N)
- serial fraction dominates a.k.a. Amdahl's law (global operations, initialization overhead)

Practical advices for CFD-solver scaling



weak scaling prodecure

recommended



doubled spatial domain with constant dx

alternative

constant domain with doubled elements
(potential adaption of time stepper!)

cores per node?

full

recommended if N per core is tunable (sweet spot run)

partial

more memory (bandwidth/size) per core
(process pinning matters!)

Scaling literature

Book: “Big CPU, Big Data” (Ch.9: Strong Scaling, Ch.10: Weak Scaling), Alan Kaminsky

Article: “Amdahl's Law, Gustafson's Trend, and the Performance Limits of Parallel Applications”, Matt Gillespie

Article: “Parallel Application Scaling, Performance, and Efficiency”, David Skinner & Katie Antypas

Book: “Using HPC for Computational Fluid Dynamics” (Ch. 3), Shamoon Jamshed

Book: “Introduction to High Performance Computing for Scientists and Engineers”, Georg Hager & Gerhard Wellein

https://www.hlrn.de/doc/display/PUB/Workshop+2020+Material

https://www.d.umn.edu/~tkwon/course/5315/HW/MultiprocessorLaws.pdf

http://www.cs.columbia.edu/~martha/courses/4130/au12/scaling-theory.pdf

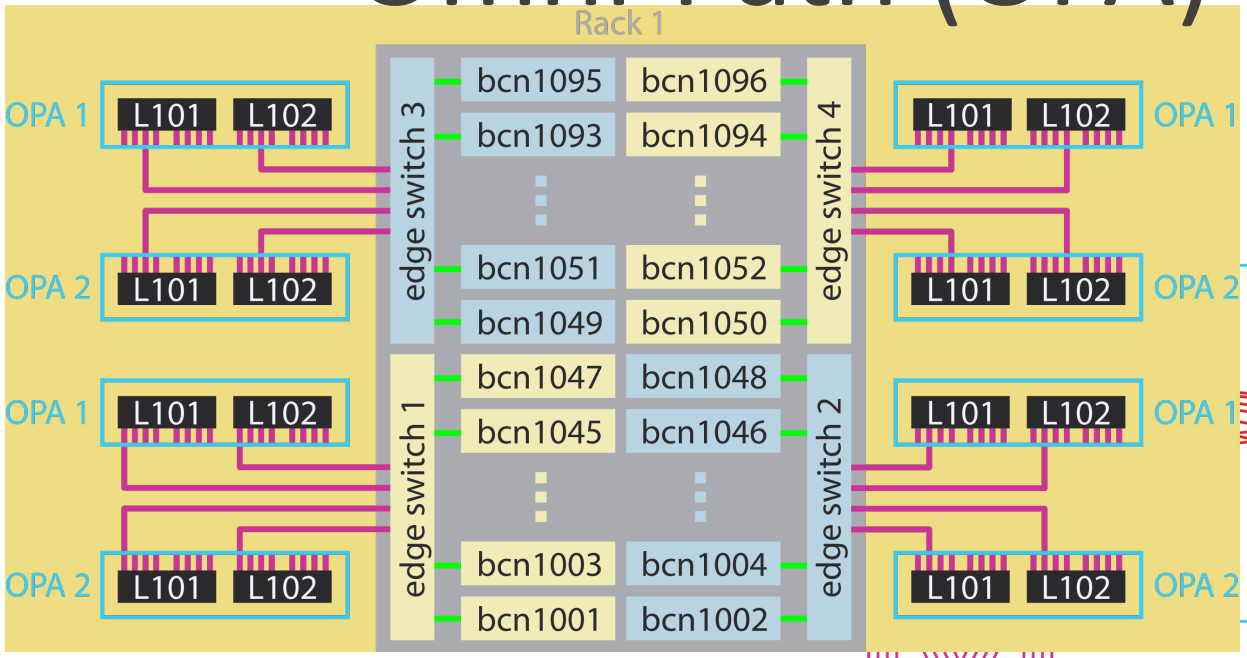
https://www.hlrs.de/about-us/media-publications/teaching-training-material

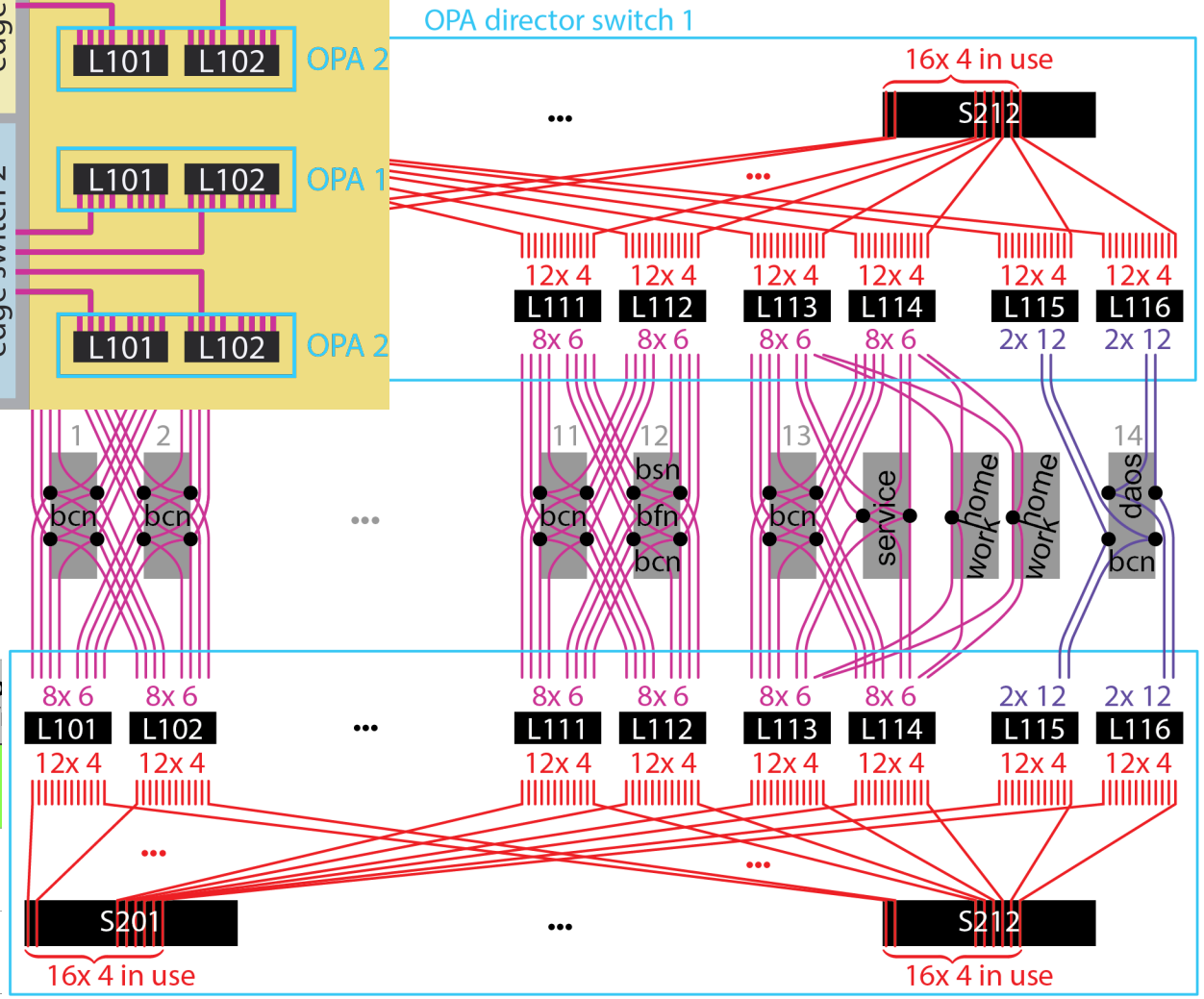
https://geb.sts.nt.uni-siegen.de/hpcfd/pages/materialien.html

That’s it.

Any questions?

Omni-Path (OPA) network of Lise





2x48 port switch either S for spine or L for leaf

single OPA lane (100Gbps)

4 OPA lanes (each 100Gbps)

6 OPA lanes (each 100Gbps)

12 OPA lanes (each 100Gbps)

2x24 port edge switch

rack (96 nodes max)

unit	connected nodes/cores	link: ↑↓ or ↔ units	length of link [m]	accessib storage/
director -switch	all nodes	OPA (5 hops)	10	HDD “work”
leaf-switch	24x8 nodes (2 racks)	OPA (3 hops)	10	
edge-switch	24 nodes	OPA (1 hop)	1	