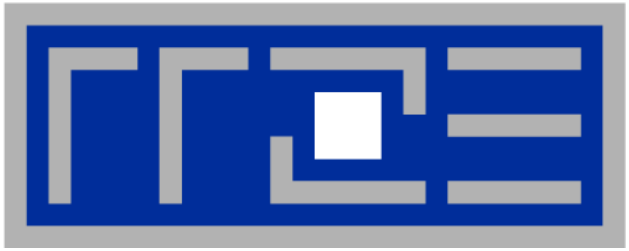


Programming Techniques for Supercomputers:

Parallel Computers (2*)

Distributed-memory computers / Hybrid systems
Communication Networks

Parallelism vs. communication

Prof. Dr. G. Wellein[a,b] , Dr. G. Hager[a]
[a]HPC Services – Regionales Rechenzentrum Erlangen
[b]Department für Informatik

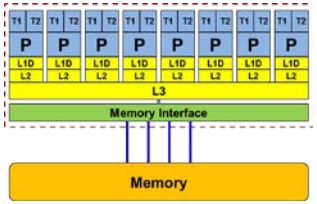
University Erlangen-Nürnberg, Sommersemester 2019

*see lecture 7 for first part

Classification according to address space organization

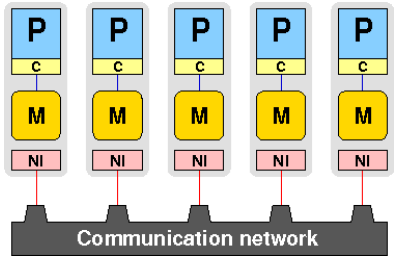
- Shared-Memory Architectures:

  Cache-Coherent Single Address Space

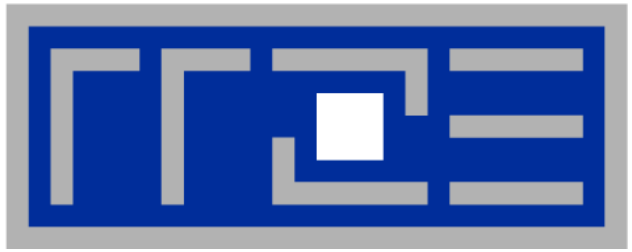


- **Distributed-Memory Architectures**

  No Cache-Coherent Single Address Space



Hybrid architectures containing both concepts are state-of-the art



Distributed-memory computers & hybrid systems

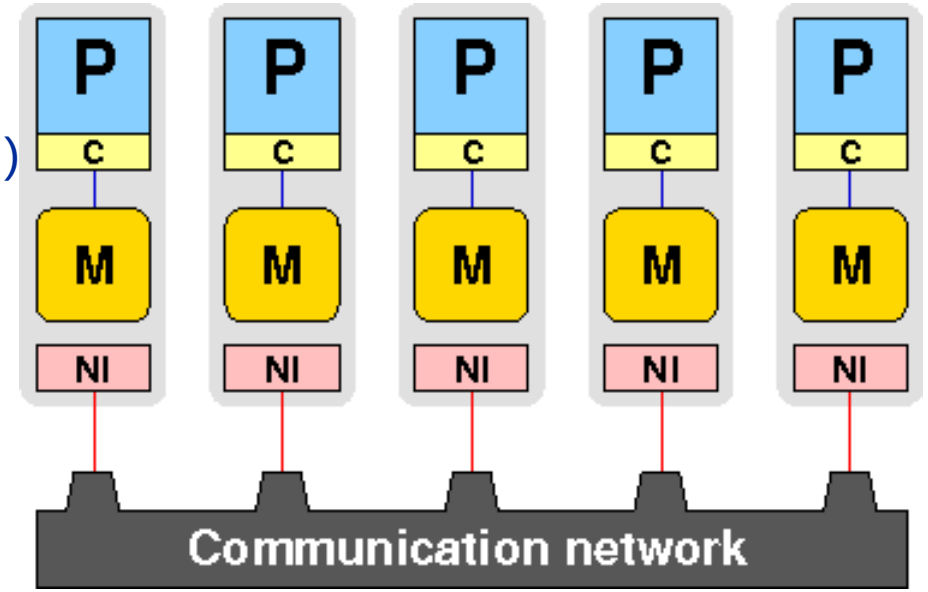
Parallel distributed-memory computers: Basics

Distributed-memory parallel computer:

• Each processor P is connected to exclusive local memory (M) and a network interface (NI) → "Node"

• A (dedicated) communication network connects all nodes

•Data exchange between nodes: Passing messages via network ("Message Passing")



Variants:

•No global (shared) address space
→ **N**o **R**emote **M**emory **A**ccess (**NORMA**)

•NON-COHENRENT shared address space (**NUMA**), e.g. CRAY
→ PGAS languages (CoArray Fortran, UPC)

Prototype of first PC clusters:

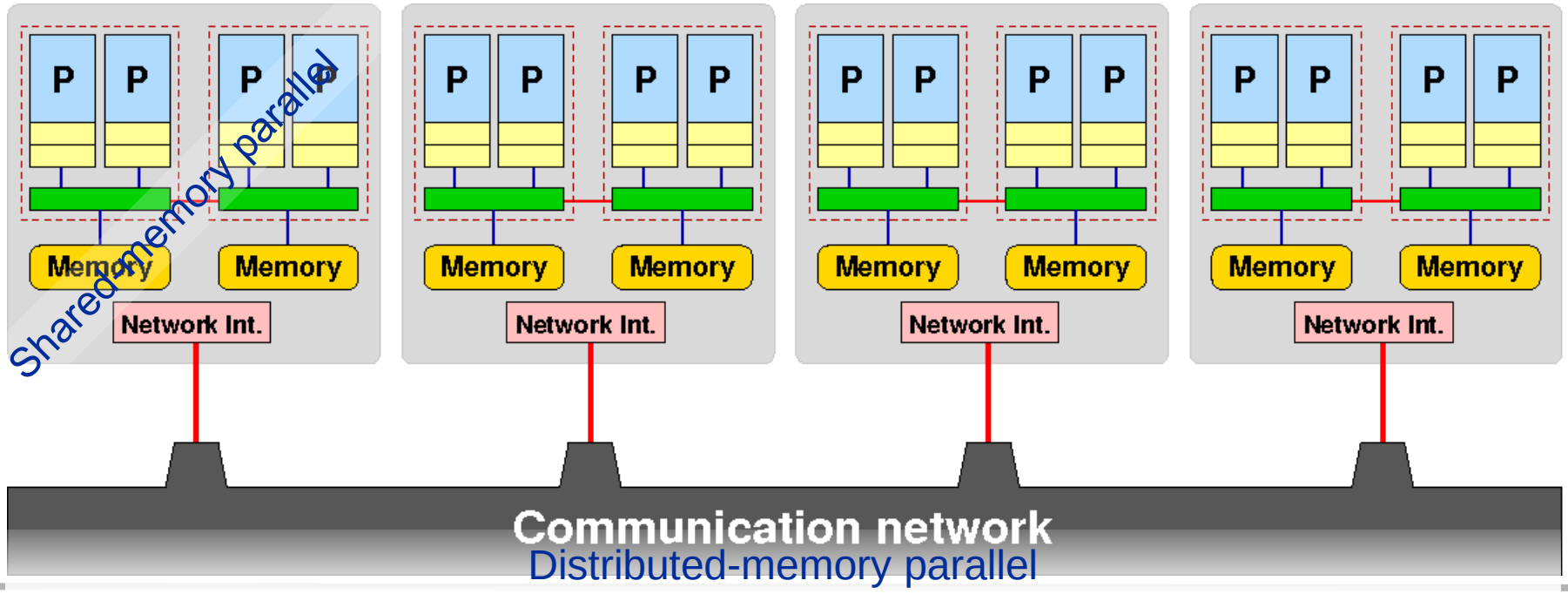
• Node: Single CPU → PC
Network: Ethernet

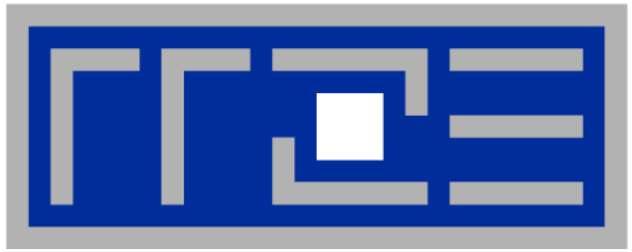
First Massively Parallel Processing (MPP) architectures:
CRAY T3D/E, Intel Paragon

Parallel distributed-memory computers: Hybrid system

Standard concept of most modern large parallel computers: **Hybrid**/hierarchical

- Compute node is a **2- or 4-socket shared memory compute** nodes with a NI.
- **Communication network** (GBit, Infiniband) connects the nodes
- Parallel Programming? Pure Message Passing is standard!
  Hybrid programming (MPI + OpenMP)?
- Today: GPUs / Accelerators are added to the nodes to further increase complexity





Networks

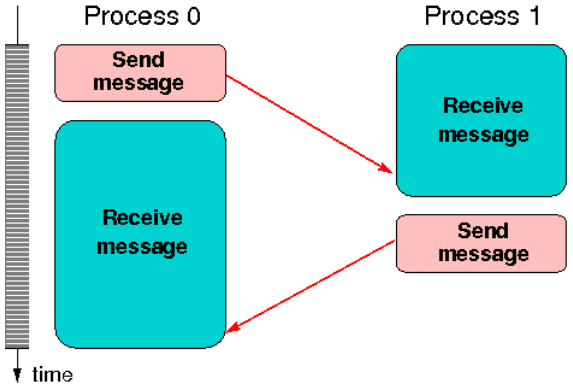
What are the basic ideas and performance characteristics of modern networks…

Networks – Basic performance characteristics

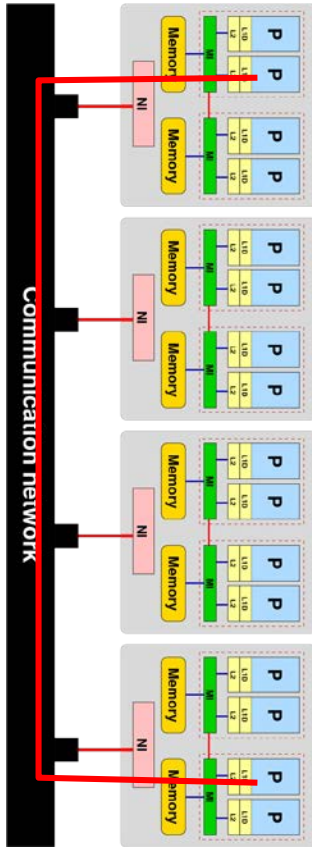
- **Evaluate the network capabilities to transfer data**
- **Use the same idea as for main memory access:**
  - Total transfer time for a message of V Bytes is:

$$T_{comm} = \lambda + \frac{V}{b_{network}}$$

  - λ is the latency (transfer setup time [sec]) and $b_{network}$ is asymptotic (N → oo) network bandwidth [MBytes/sec]
- **Consider simplest case ("Ping Pong")**
  - Two processors in different nodes communicate via network ("Point-to-point")
  - A single message of N Bytes is sent forward and backward



Overall data transfer is 2 N Bytes!

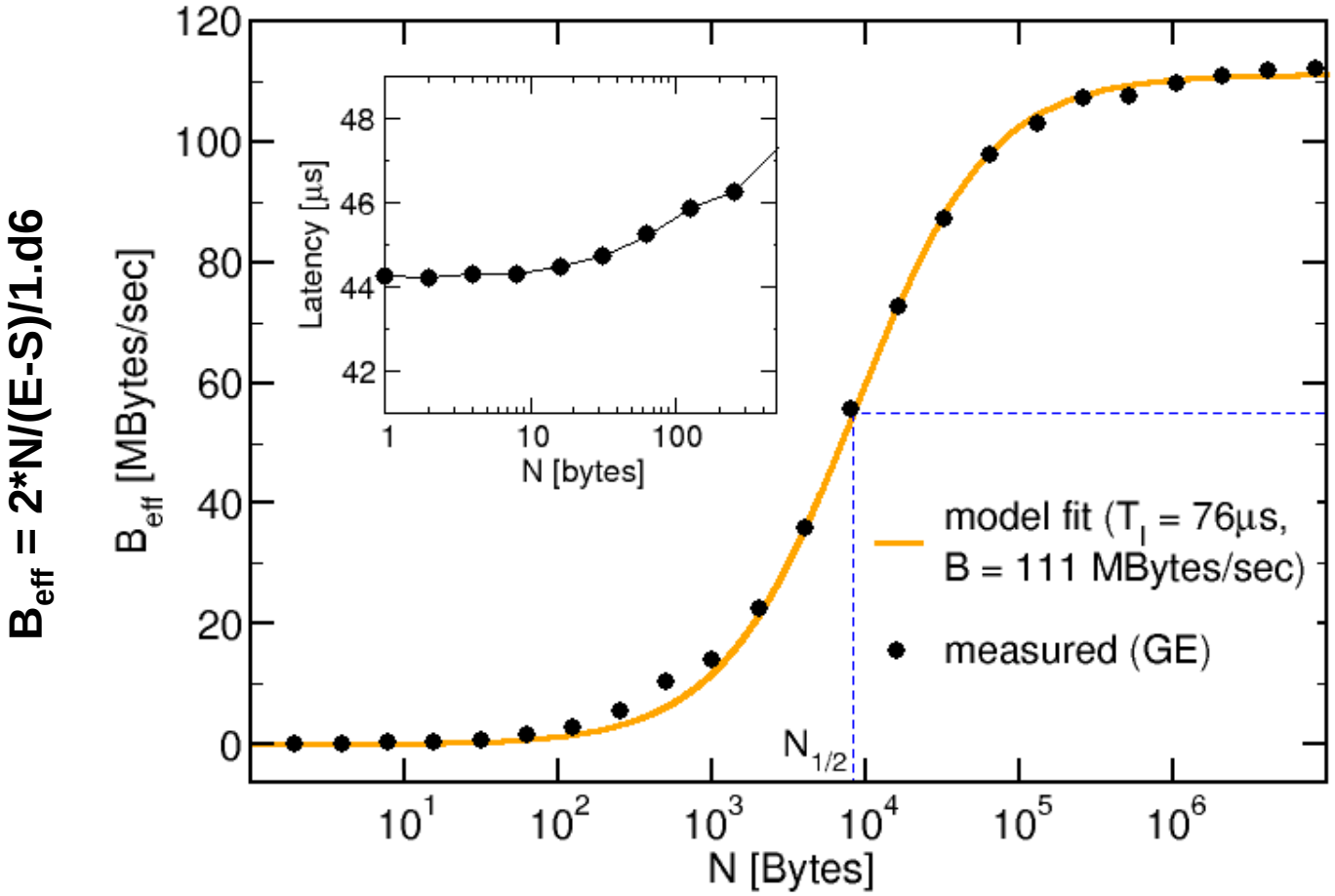


Ping-Pong benchmark (schematic view)

```
1 myID = get_process_ID()
2 if(myID.eq.0) then
3    targetID = 1
4    S = get_walltime()
5    call Send_message(buffer,N,targetID)
6    call Receive_message(buffer,N,targetID)
7    E = get_walltime()
8    MBYTES = 2*N/(E-S)/1.d6 ! MBytes/sec rate
9    TIME = (E-S)/(2*1.d6) ! transfer time in microsecs
10 ! for single message
11 else
12   targetID = 0
13   call Receive_message(buffer,N,targetID)
14   call Send_message(buffer,N,targetID)
15 endif
```

Ping-Pong benchmark for GBit-Ethernet (GigE) network

$N_{1/2}$: Message size where 50% of peak bandwidth is achieved



Asymptotic bandwidth $b_{network}$=111 MB/s ←→ 0.888 GBit/s

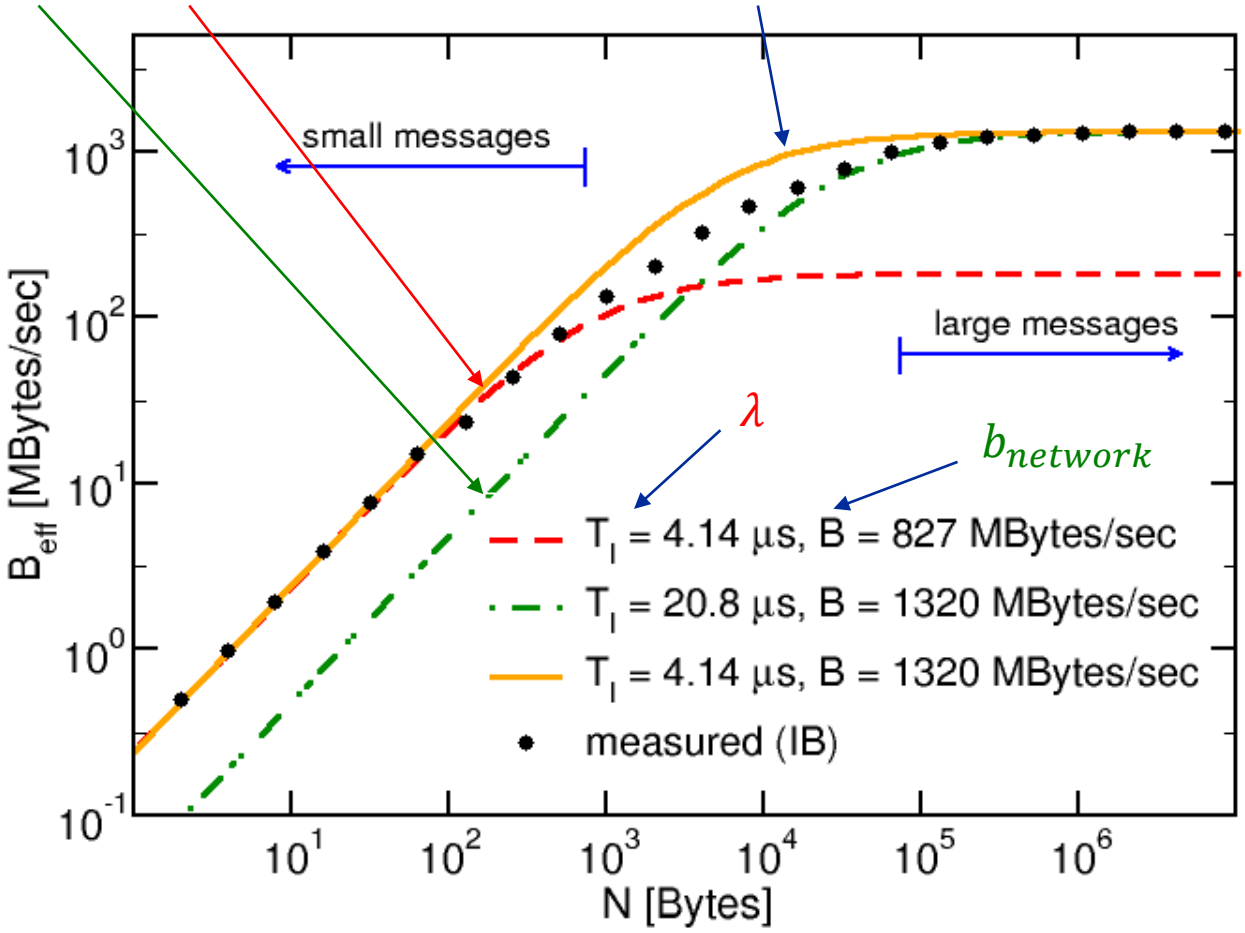
Latency (N → 0):

Only qualitative agreement:

44 µs vs. 76 µs

Ping-Pong benchmark for DDR Infiniband (DDR-IB) network

Determine $b_{network}$ and λ independently and combine them



Networks – Basic performance characteristics

- "First Principles" modeling of $B_{eff}(V)$ provides good qualitative results but quantitative description in particular of latency dominated region (V small) may fail because
  - Overhead for transmission protocols, e.g. message headers
  - Minimum frame size for message transmission, e.g. TCP/IP over Ethernet does always transfer frames with V>1
  - Message setup/initialization involves multiple software layers and protocols; each software layer adds to latency; hardware only latency is often small
  - As the message size increases the software may switch to different protocol, e.g. from "eager" to "rendezvous"

- Typical message sizes in applications are neither small nor large
  → $V_{1/2}$ value is also important: $V_{1/2} = \lambda * b_{network}$

- Network balance: Relate network bandwidth ($b_{network}$ or $B_{eff}(V_{1/2})$) to compute power (or memory bandwidth) of the nodes (cf. code balance)

Networks: Topologies & Bisection bandwidth

- **Network bisection bandwidth B_b is a general metric for the data transfer "capability" of a system:**

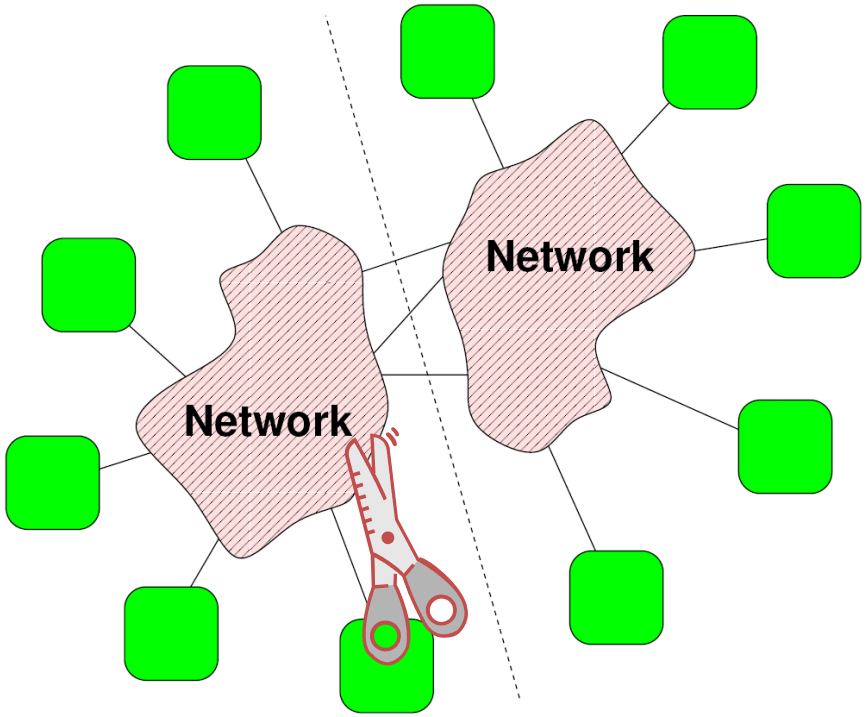
> **Minimum sum of the bandwidths of all connections cut when splitting the system into two equal parts**

- **More meaningful metric in terms of system scalability:**

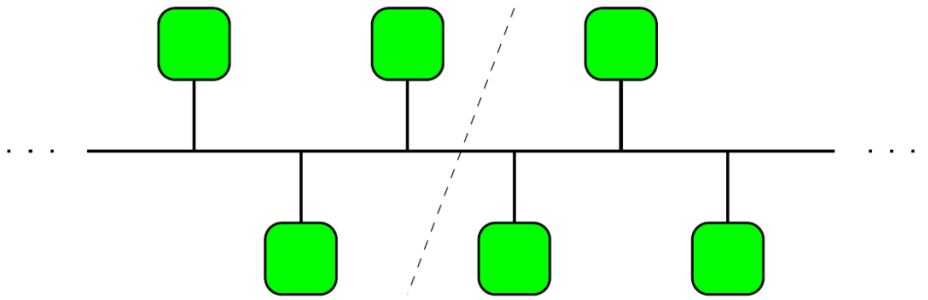
> **Bisection BW per node: B_b/N_{nodes}**

- **Bisection BW depends on**
  - Bandwidth per link
  - Network topology

- **Uni- or Bi-directional bandwidth?!**



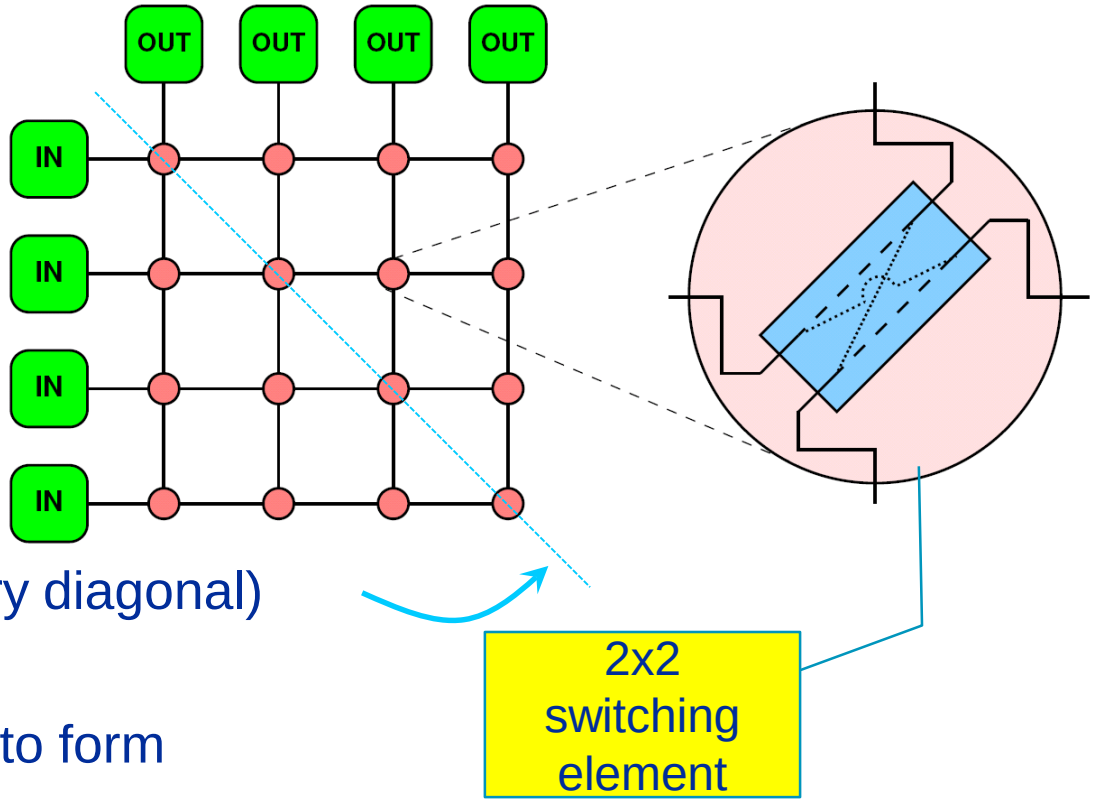
Network topologies: Bus



- Bus can be used by one connection at a time
- Bandwidth is shared among all devices
- Bisection BW is constant → $B_b/N_{nodes} \sim 1/N_{nodes}$
- Examples: PCI bus, diagnostic buses
- Advantages
  - Low latency
  - Easy to implement
- Disadvantages
  - Shared bandwidth, not scalable
  - Problems with failure resiliency (one defective agent may block bus)
  - Fast buses for large N require large signal power

Non-blocking crossbar

- A non-blocking crossbar can mediate a number of connections between a group of input and a group of output elements
- This can be used as a 4-port non-blocking switch (fold at the secondary diagonal)



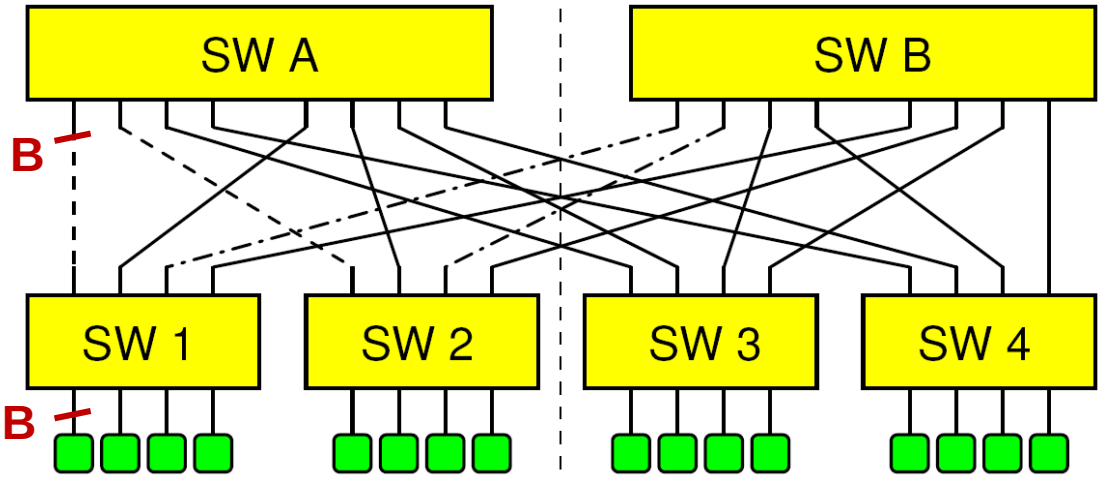
- Switches can be cascaded to form hierarchies (common case)
- Allows scalable communication at high hardware/energy costs
- Crossbars are rarely used as interconnects for large scale computers
  - NEC SX9 vector system (“IXS”)

Network topologies: Switches and Fat-Trees

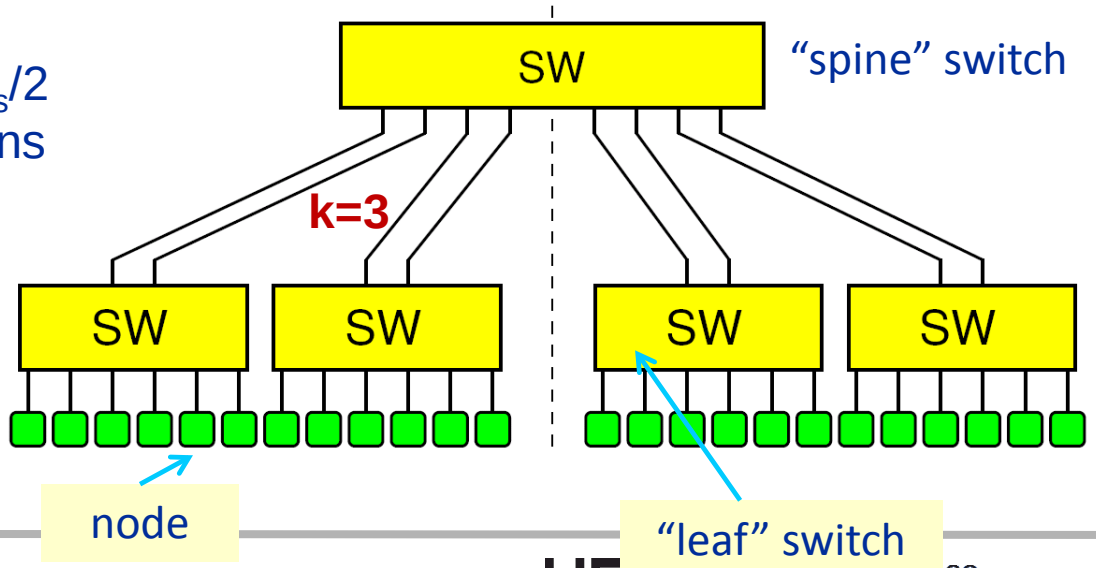
- Standard clusters are built with switched networks
- Compute nodes (“devices”) are split up in groups – each group is connected to single (non-blocking crossbar-)switch (“leaf switches”)
- Leaf switches are connected with each other using an additional switch hierarchy (“spine switches”) or directly (for small configs.)
- Switched networks: “Distance” between any two devices is heterogeneous (number of “hops” in switch hierarchy)
  - → Diameter of network: The maximum number of hops required to connect two arbitrary devices, e.g. diameter of bus=1
- “Perfect” world: “Fully non-blocking”, i.e. any choice of $N_{nodes}/2$ disjoint node (device) pairs can communicate at full speed

“Fat tree” switch hierarchies

- **“Fully non-blocking”**
  - $N_{nodes}/2$ end-to-end connections with full bandwidth → $B_b = B * N_{nodes}/2$
  - **B_b/N_{nodes} = const. = B/2**
  - Sounds good, but see next slide

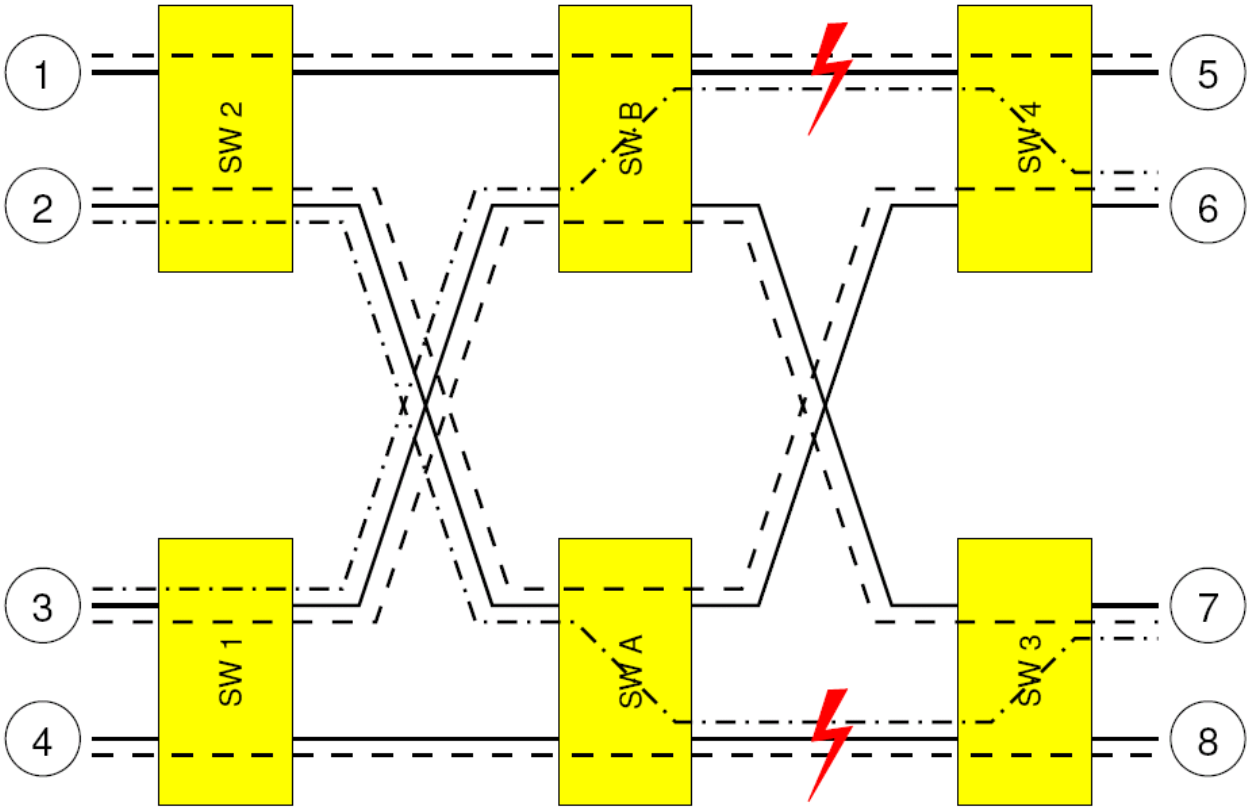


- **“Oversubscribed”**
  - Spine does not support $N_{nodes}/2$ full BW end-to-end connections
  - **B_b/N_{nodes} = const. = B/(2k)**, with k oversubscription factor
  - Resource management (job placement) is crucial

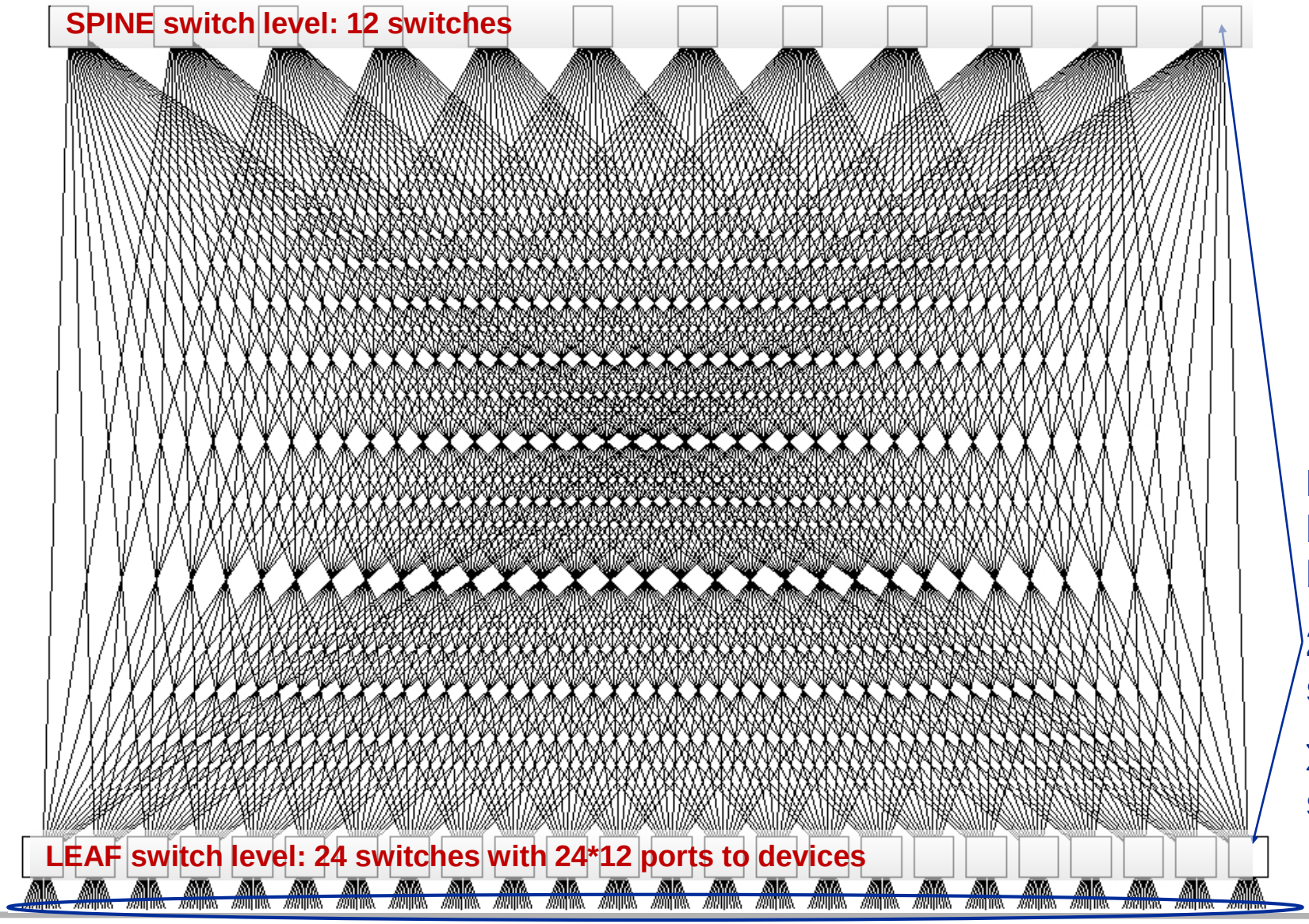


Fat trees and static routing

- If all end-to-end data paths are preconfigured ("static routing"), not all possible combinations of N agents will get full bandwidth
  - Example: – – – – is a collision-free pattern here (1→5, 2→6,3→7, 4→8)
  - Change 2→6, 3→7 to 2→7, 3→6: – · – · – has collisions if no other connections are re-routed at the same time
- Static routing: Quasi-standard in "commodity" inter-connects
- However, things are starting to improve slowly



Full fat-tree: „Single“ 288-port IB DDR-Switch



Basic building blocks:

24-port switches

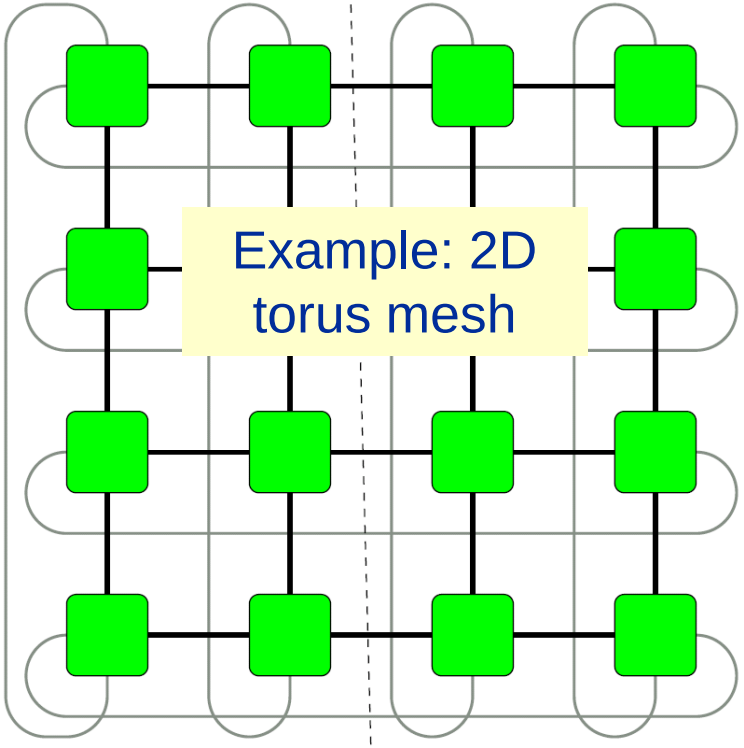
Σ = 12+24 switches

Fat tree networks – Examples

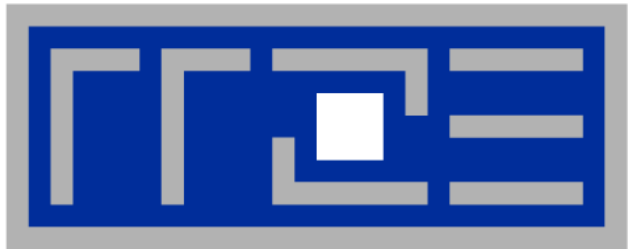
- **Ethernet**
  - 1 Gbit/s &10 &100 Gbit/s variants
- **InfiniBand**
  - Dominant high-performance “commodity” interconnect
    - DDR: 20 Gbit/s per link and direction (Building blocks: 24-port switches)
    - QDR: 40 Gbit/s per link and direction
      - QDR IB is used in the RRZE’s LiMa and Emmy clusters
      - Building blocks: 36 port switches → “Large” 36*18=648-port switches
    - FDR-10 / FDR: 40/56 Gbit/s per link and direction
    - EDR: 100 Gbit/s per link and direction
- **Intel OmniPath**
  - Up to 100 Gbit/s per link & 48-port baseline switches
  - RRZE Meggie cluster
- **Expensive & complex to scale to very high node counts**

Meshes

- Fat trees can become prohibitively expensive in large systems
- Compromise: Meshes
  - n-dimensional Hypercubes
  - Toruses (2D / 3D)
  - Many others (including hybrids)
- Each node is a “router”
- Direct connections only between direct neighbors
- This is not a non-blocking corossbar!
  - Intelligent resource management and routing algorithms are essential



- Toruses at very large systems: Cray XE/XK series, IBM Blue Gene
  - $B_b \sim N_{nodes}^{(d-1)/d}$ → $B_b/N_{nodes} \to 0$ for large N_{nodes}
  - Sounds bad, but those machines show good scaling for many codes
  - Well-defined and predictable bandwidth behavior!

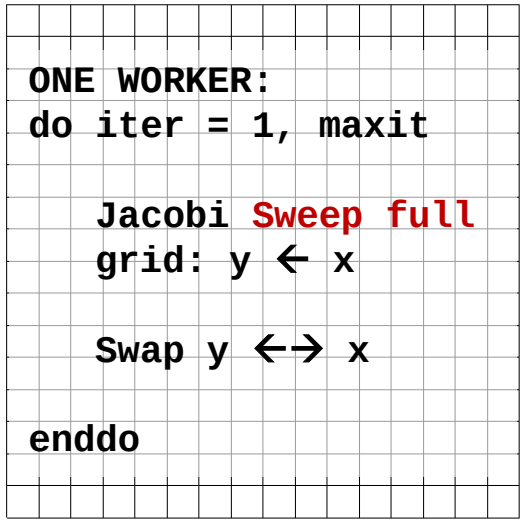


Parallelism vs. Communication

Modelling a 3D parallel Jacobi solver

Parallelism vs. Communication
Parallelization by Domain Decomposition: Jacobi 2D5pt

- **Assume 2D regular grid with $L \times L$ grid points: $t_{update} = L^2/P$**




```
ON ALL N WORKERS:
do iter = 1, maxit

   Jacobi Sweep local
   grid: y ← x
   Exchange BOUNDARIES
   Swap y ↔ x
enddo
```

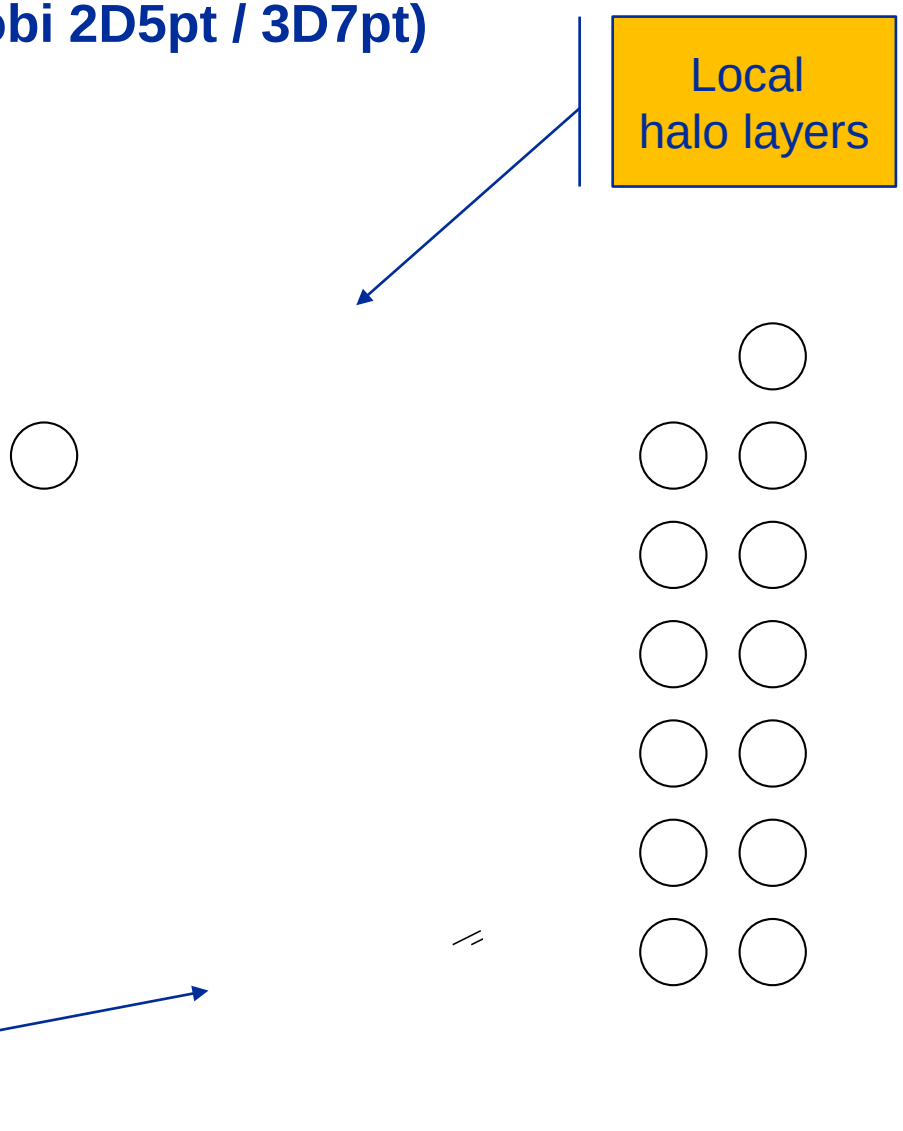
"Slices" - #boundary cells $const \times L$

"Squares" – #boundary cells: $const \times \frac{L}{\sqrt{N}}$

Balanced workload distribution: $t_{update}(N) = L^2/(N * P)$

Parallelism vs. Communication
Domain Decomposition: Boundary Exchange

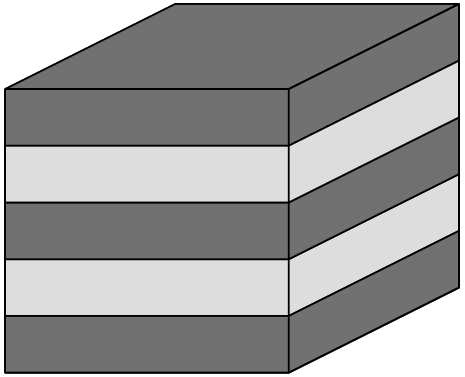
- **For a distance-1 stencil (e.g. Jacobi 2D5pt / 3D7pt)**
- **For every domain/worker:**
  - Update local domain
  - Send local boundary layers to neighboring domain/worker
  - Put data from neighboring domain/worker into local halo layer
- **Communication volume per domain/worker: Surface of local domain**



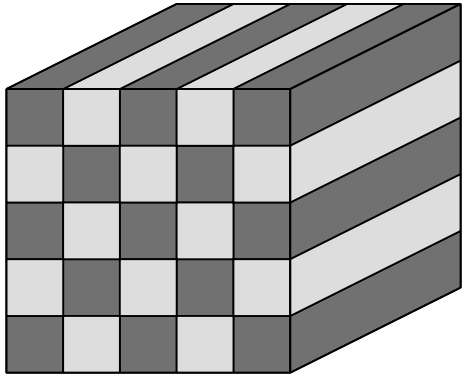
Parallelism vs. Communication
Domain Decomposition: Basic 3D decompositions

Approaches to decompose a 3D cubic domain (L^3 Lattice Sites)

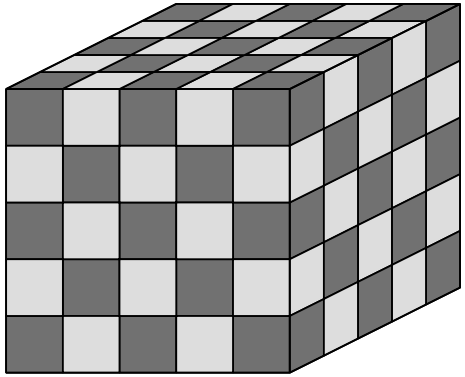
1D decomposition
(e.g. N=5 domains/worker)



2D decomposition
(e.g. N=25 domains/worker)



3D decomposition
(e.g. N=125 domains/worker)



Communication volume of boundary layers [B] per worker/domain

$$V_{1d}(L,N) = 2\,w\,L^2$$

Size [B] for a single site

$$V_{2d}(L,N) = 4\,w\,\frac{L^2}{\sqrt{N}}$$

Max. number of neighbors

$$V_{3d}(L,N) = 6\,w\,\frac{L^2}{N^{\frac{2}{3}}}$$

#sites in each boundary layer

Parallelism vs. Communication
Domain Decomposition: Scalability

- Assume full non-blocking network, i.e. N/2 pairs of neighbors communicate in parallel (see slide 7) →

$$T_{comm}(N) = \lambda + \frac{V(L,N)}{b_{network}}$$

- Use extended Amdahl's law to model scalability properties of the 3 different decomposition approaches

- Serial (one worker) runtime: $T(1) = L^3/P$

For $N > 1$ we have

- Parallel (N workers) runtime: $T(N) = s + (L^3/(N * P)) + T_{comm}(N)$
- Network latency is constant: $T(N) = s + \frac{L^3}{N*P} + \lambda + \frac{V(L,N)}{b_{network}}$

Network latency
←→
serial code

Parallelism vs. Communication
Domain Decomposition: Scalability

- Speed-Up: $S = \frac{T(1)}{T(N)} = \frac{1}{\frac{T(N)}{T(1)}} = \frac{1}{\bar{s}+\frac{1}{N}+\bar{\lambda}+\frac{V(L,N)}{T(1)\,b_{network}}}$

  with $\bar{s} = \frac{s}{T(1)}$ and $\bar{\lambda} = \frac{\lambda}{T(1)}$

- Using $\bar{c} = \frac{2w}{L}\,\frac{P}{b_{network}}$ we get

  Powerful devices/nodes
  →powerful networks

  - 1D decomposition: $S_{1d} = \frac{1}{\bar{s}+\frac{1}{N}+\bar{\lambda}+\bar{c}}$

    Constant ommunication time
    → serial fraction → $S_{1d} \leq \frac{1}{\bar{\lambda}+\bar{c}}$

  - 2D decomposition: $S_{2d} = \frac{1}{\bar{s}+\frac{1}{N}+\bar{\lambda}+\frac{2\bar{c}}{\sqrt{N}}}$

  - 3D decomposition: $S_{3d} = \frac{1}{\bar{s}+\frac{1}{N}+\bar{\lambda}+\frac{3\bar{c}}{N^{2/3}}}$

    Communication time
    → serial fraction → $S_{3d} \leq \frac{1}{\bar{\lambda}+3\bar{c}/N^{2/3}}$

→ Use 3D decomposition if possible

Parallelism vs. Communication
Domain Decomposition: Scalability

- Estimating the relevant parameters for real world problems:
  - Domains size for Jacobi 3D stencil: $L = 5000 \rightarrow 125 * 10^9 \; Lattice \; Sites$
  - Double precision computation: $w = 8B$
  - Node performance: $P = 4 \; GLUP/s$ (assuming $I = \frac{1LUP}{24B}$ and $b_S = 96 \frac{GB}{s}$)

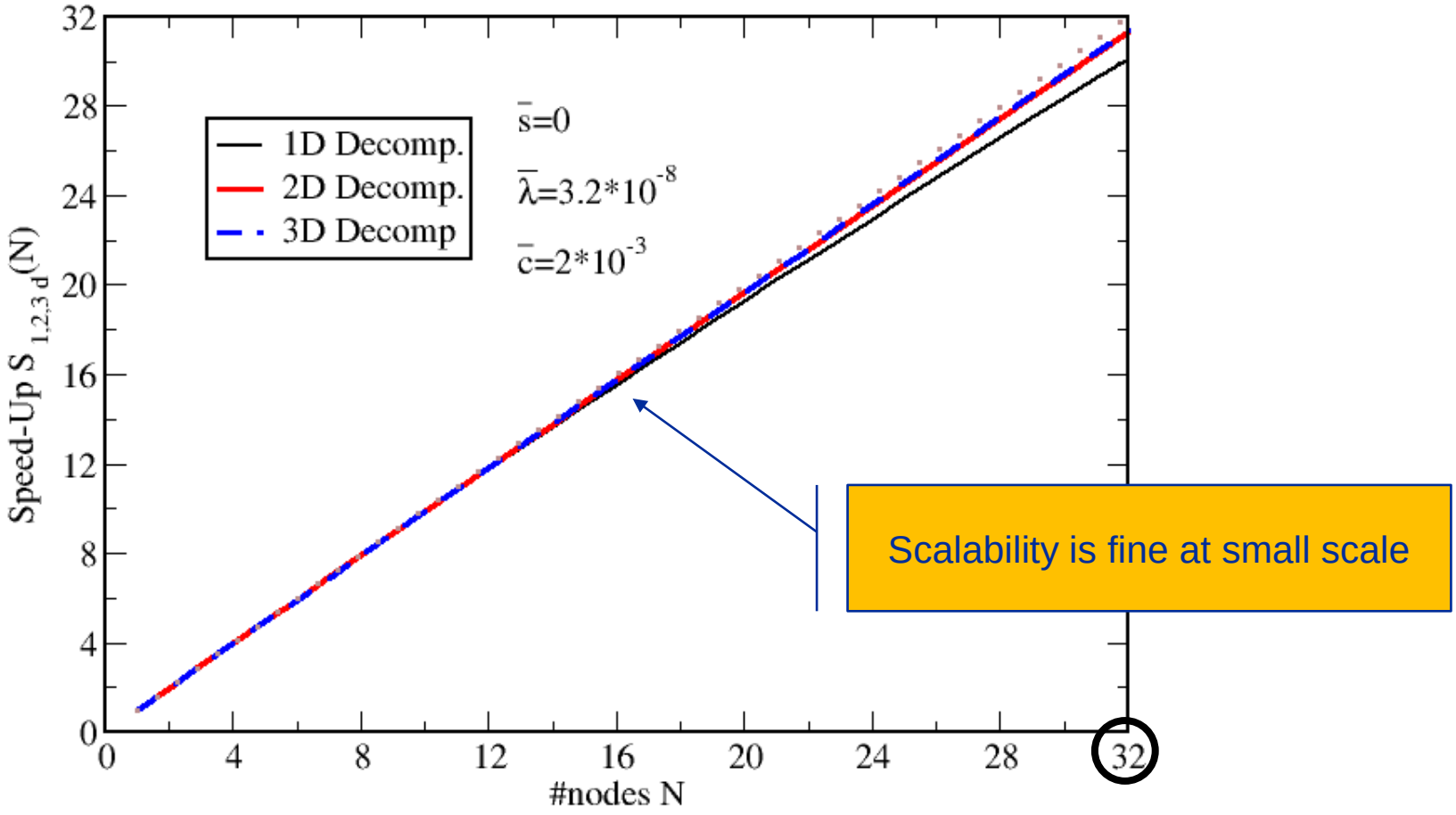
    → Single node runtime $T(1) = \frac{L^3}{P} = 31{,}25 \; s$
  - Network performance (approx. QDR IB): $b_{network} = \frac{3GB}{s}$; $\lambda = 10^{-6} s$

→ $\bar{\lambda} = \frac{\lambda}{T(1)} = 3.2 * 10^{-8}$

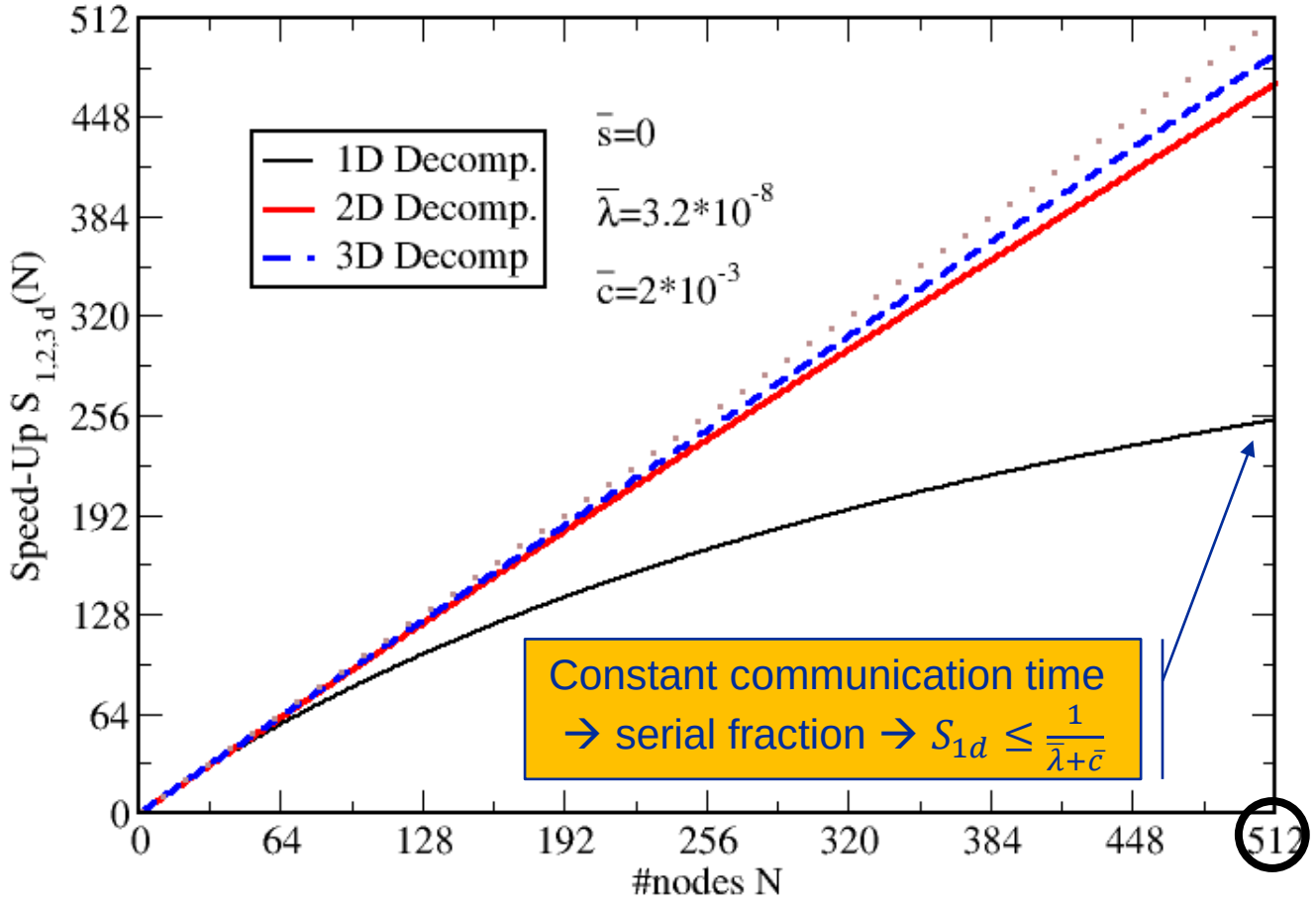
→ $\bar{c} = \frac{2w}{L} \frac{P}{b_{network}} = 2 * 10^{-3}$

→ $\bar{s} = s = 0$ (assumption)

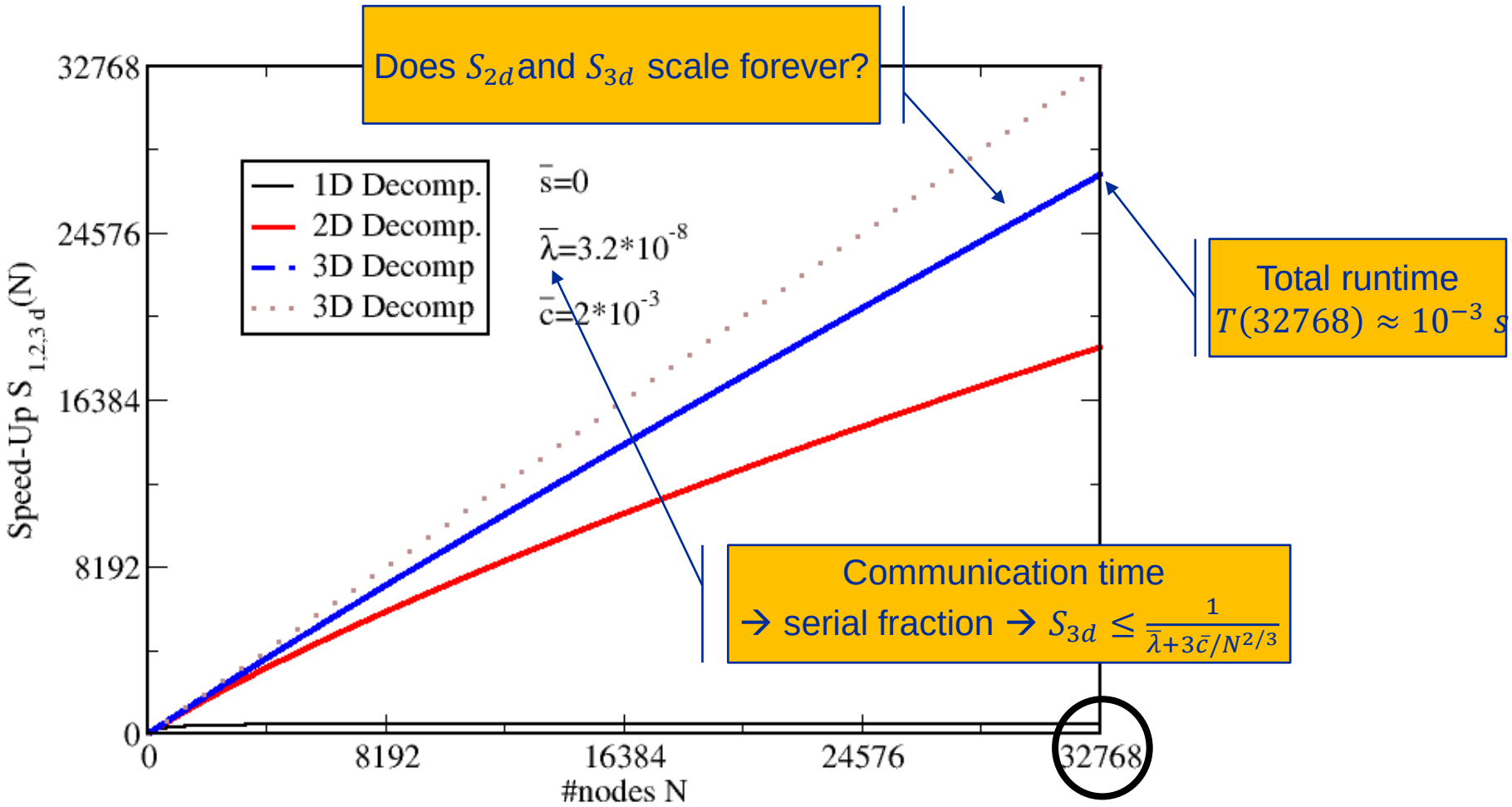
Parallelism vs. Communication
Domain Decomposition: Scalability



Parallelism vs. Communication
Domain Decomposition: Scalability



Parallelism vs. Communication
Domain Decomposition: Scalability



Parallelism vs. Communication
Domain Decomposition: Scalability

- Communication and data distribution always matters on large scale computations – often impact is not visible at small scales
- Finite communication latency introduces serial fraction!
- Impact of communication depends on subtle interplay of communication parameters (bandwidth, latency), node performance and problem size.
- Not considered so far: Impact of network topology,multiple communications per node,...