Programming Techniques for Supercomputers: Parallel Computers (2)

Distributed-memory computers / Hybrid systems

Communication Networks

Parallelism vs. communication



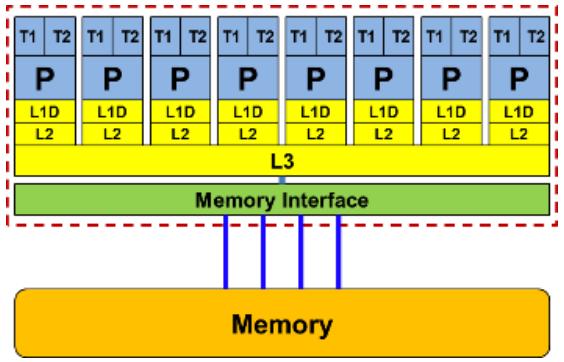
Prof. Dr. G. Wellein[a,b] , Dr. G. Hager[a], A. Afzal[b], C.L. Alappat[b], D. Ernst[b]
[a] Erlangen National High Performance Computing Center (NHR@FAU)
[b] Department für Informatik
Friedrich-Alexander-Universität Erlangen-Nürnberg, Sommersemester 2022

Parallel Computers (2) - Introduction

Classification according to address space organization

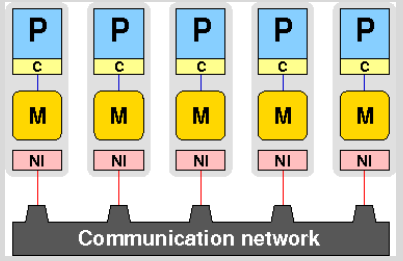
- Shared-Memory Architectures:

  Cache-Coherent Single Address Space



- Distributed-Memory Architectures

  No Cache-Coherent Single Address Space



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

NHR FAU

FAU Friedrich-Alexander-Universität Erlangen-Nürnberg

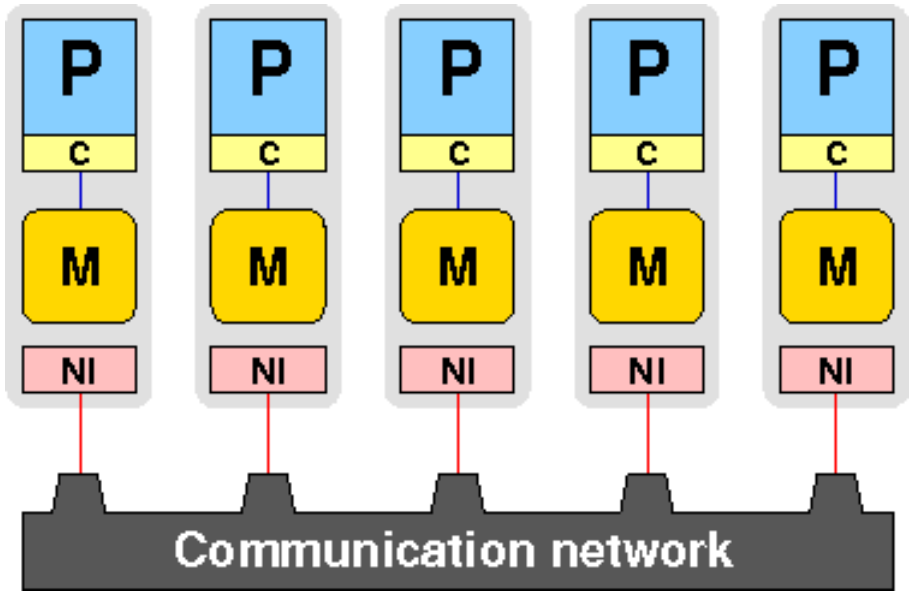
Distributed-memory computers & hybrid systems



Parallel distributed-memory computers: Basics

"Pure" 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
  → No Remote Memory Access (NORMA)
- Non-cache-coherent shared address space (NUMA) (e.g., older Cray systems)
  → PGAS languages (CoArray Fortran, UPC)

First "PC clusters" back in the day:

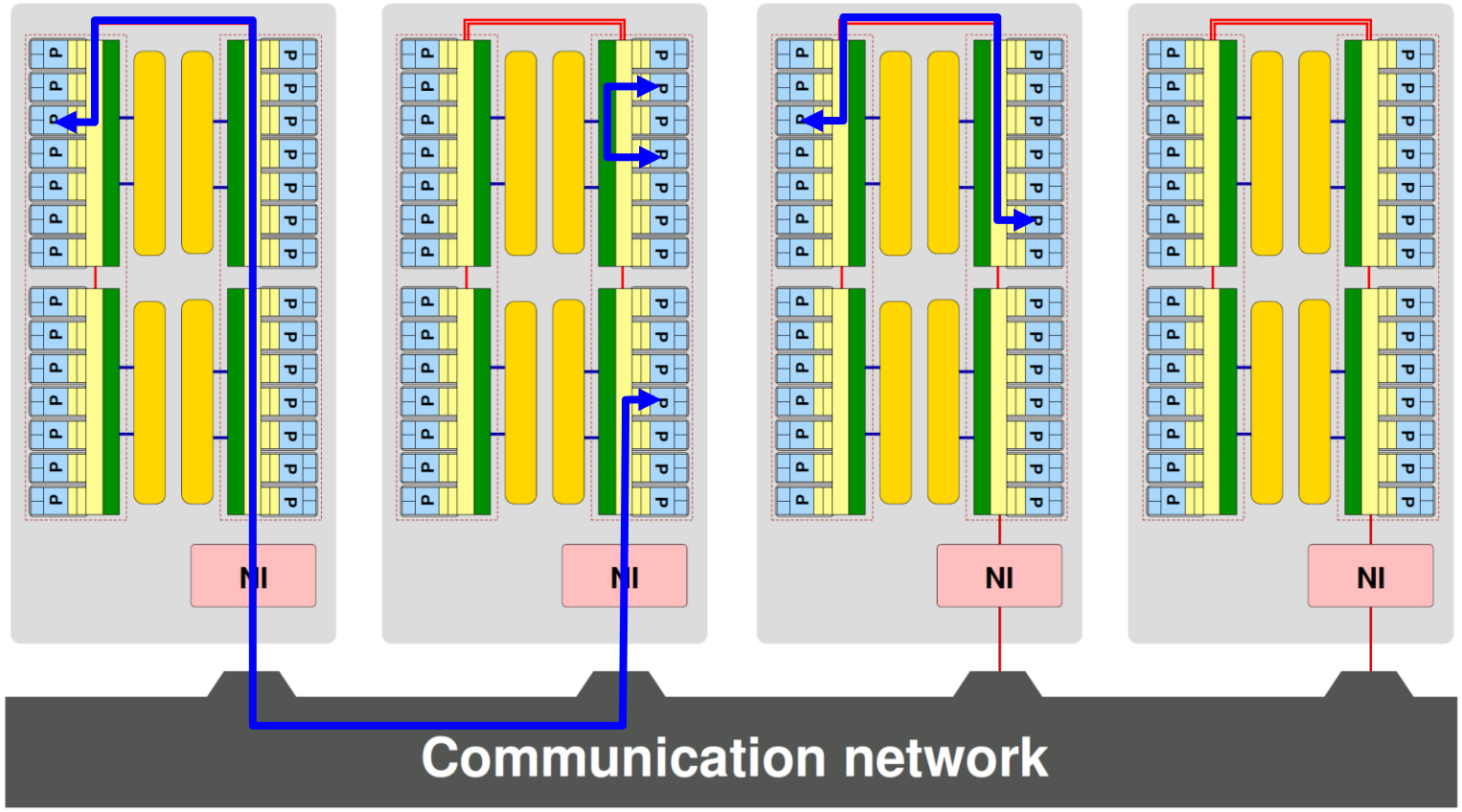
Node: Single CPU → PC
Network: Ethernet

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 network interface
  - ccNUMA architecture
- Communication network (GBit, Infiniband) connects the nodes
- Parallel Programming?
  - Pure Message Passing
  - Hybrid programming (MPI + OpenMP)
- Expect strong topology effects in communication performance
- Today:
  GPUs / Accelerators are added to the nodes and further increase complexity



NHR FAU

FAU Friedrich-Alexander-Universität Erlangen-Nürnberg

Networks

Basic ideas and performance characteristics of modern networks

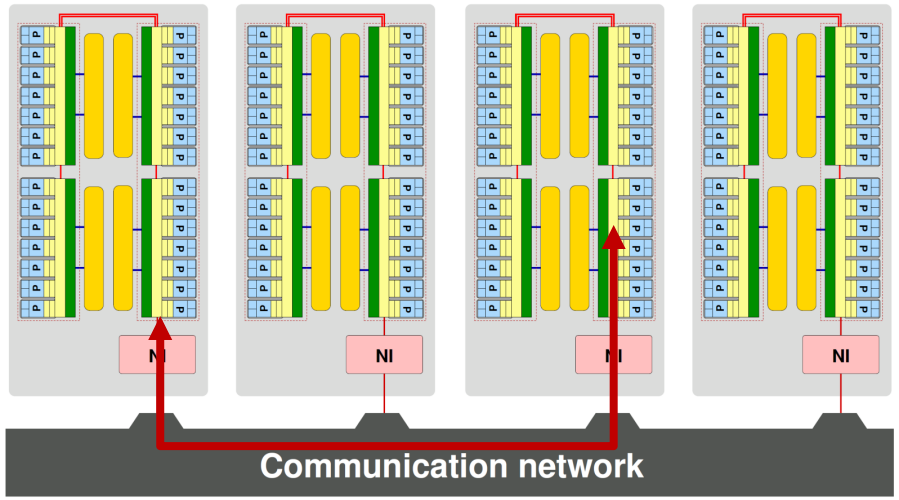


Networks – Basic performance characteristics

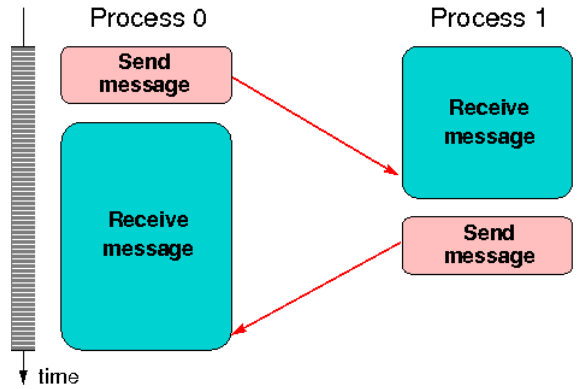
- What are the basic capabilities of a network connection?
- Use the same idea as for main memory access:
  - Total transfer time for a message of N bytes:

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

  - λ is the latency (transfer setup time [sec]) and $b_{network}$ is asymptotic ($V \to \infty$) network bandwidth [MBytes/sec]



- Simple benchmark: "Ping-Pong"
  - Two processes in different nodes communicate via network ("Point-to-point")
  - A single message of N Bytes is sent forward and backward

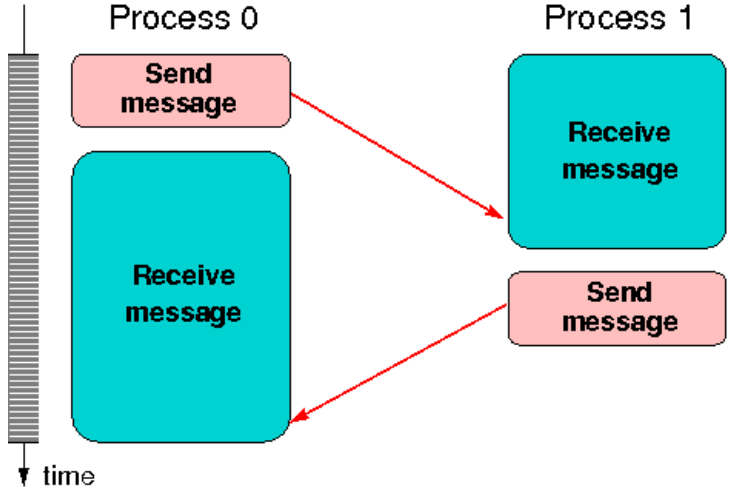


Overall data transfer is $2N$

Networks – Basic performance characteristics

Ping-Pong benchmark

```
 myID = get_process_ID()
 if(myID.eq.0) then
  targetID = 1
  S = get_walltime()
  call Send_message(buffer,N,targetID)
  call Receive_message(buffer,N,targetID)
  E = get_walltime()
  TIME = (E-S)/(2*1.d6) ! transfer time in microsecs
  Beff = 2*N/TIME       ! MBytes/sec rate
else
  targetID = 0
  call Receive_message(buffer,N,targetID)
  call Send_message(buffer,N,targetID)
 endif
```



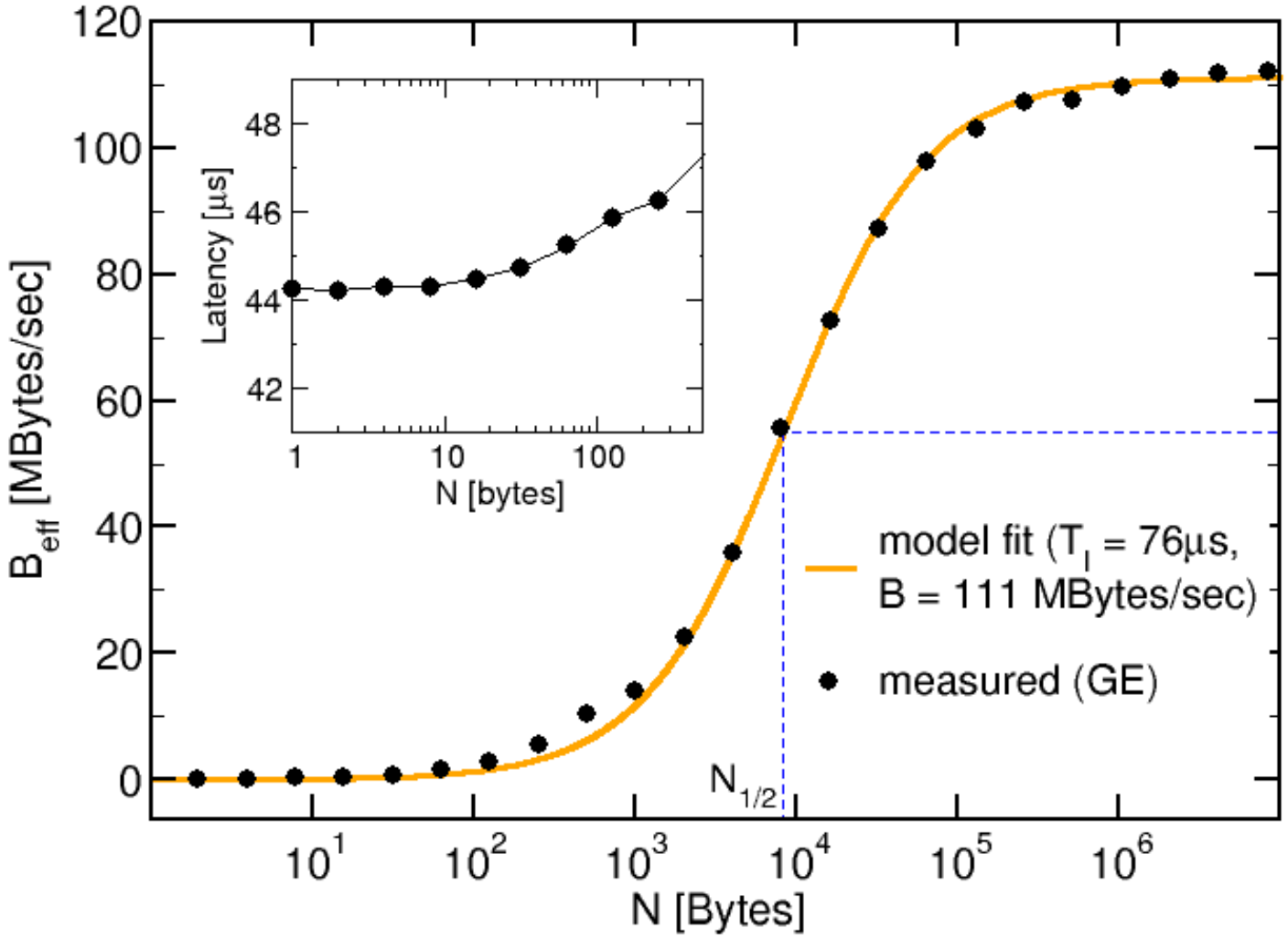
Effective transfer bandwidth:

$$B_{eff} = \frac{N}{T_{comm}} = \frac{N}{\lambda + \frac{N}{b_{network}}}$$

Networks – Basic performance characteristics

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 ($N \to 0$):

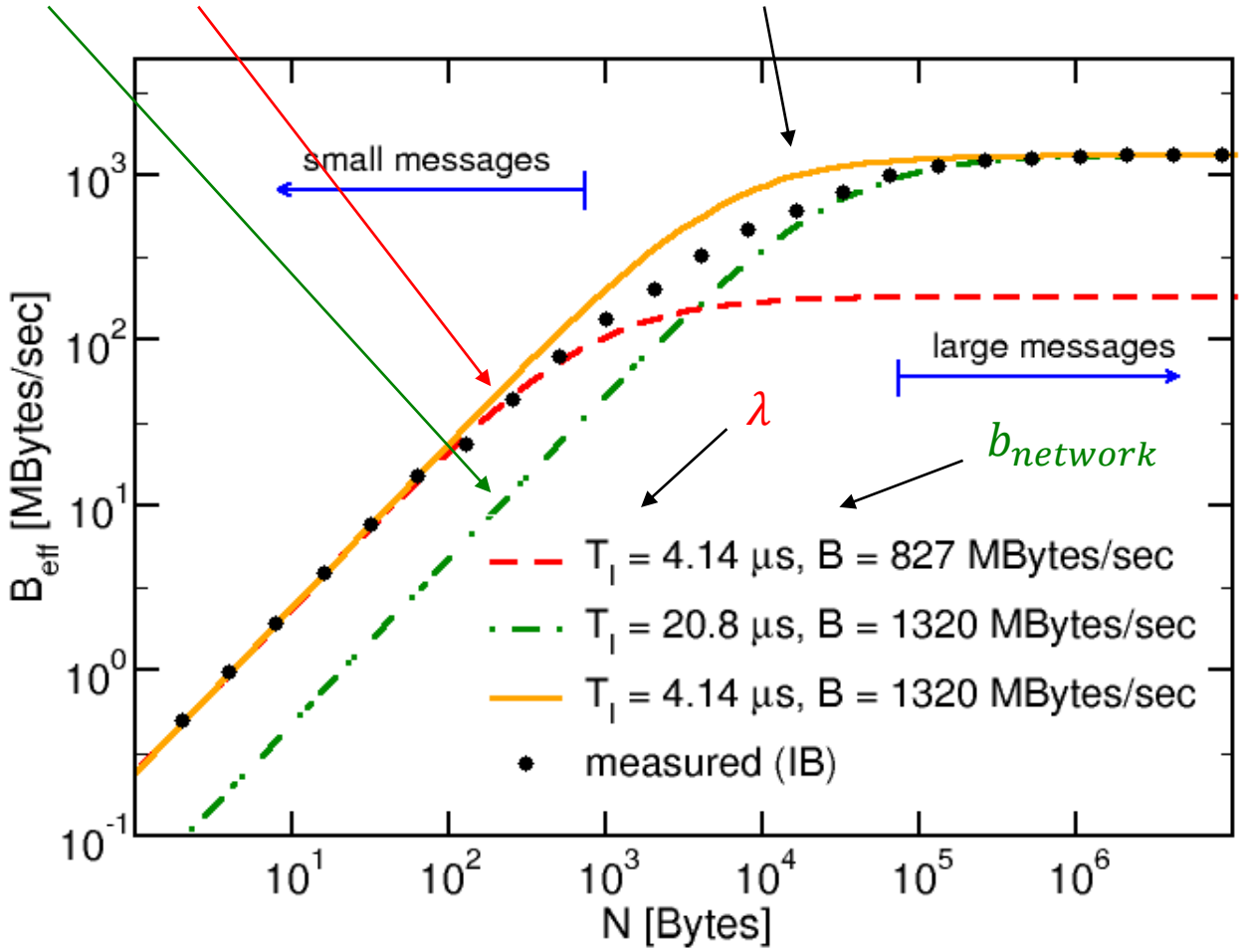
Only qualitative agreement:

44 µs vs. 76 µs

Networks – Basic performance characteristics

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 (small V) may fail because
  - Overhead for transmission protocols, e.g. message headers
  - Minimum frame size for message transmission (e.g., TCP/IP)
  - 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
  → $N_{1/2}$ is also important: $N_{1/2} = \lambda \times b_{network}$

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

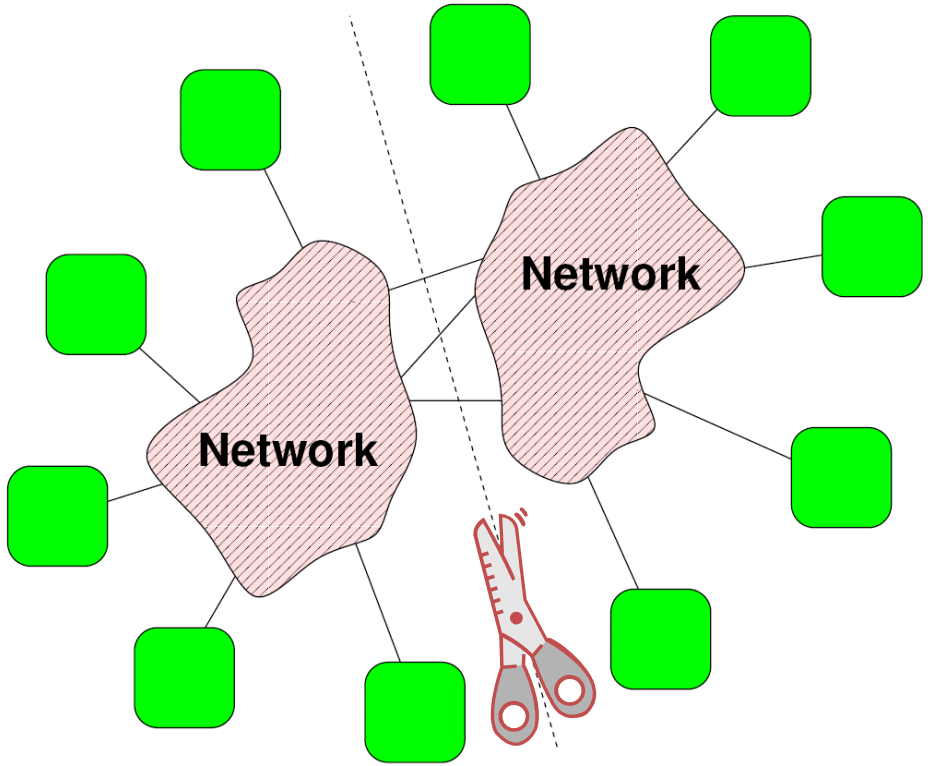
Networks: Topologies & bisection bandwidth

- 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}

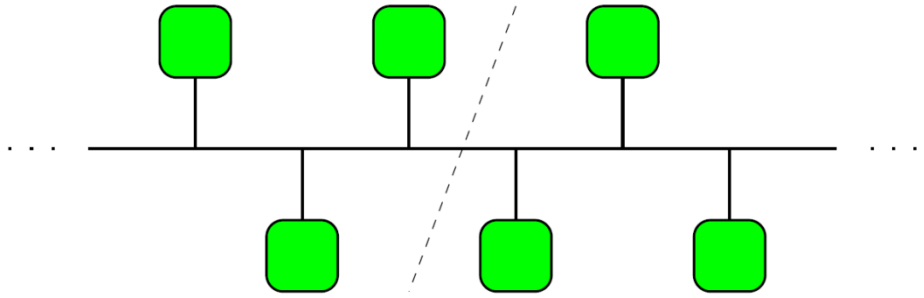


- Bisection BW depends on
  - Bandwidth per link
  - Network topology

- Uni- or Bi-directional bandwidth?!

Network topologies: Bus

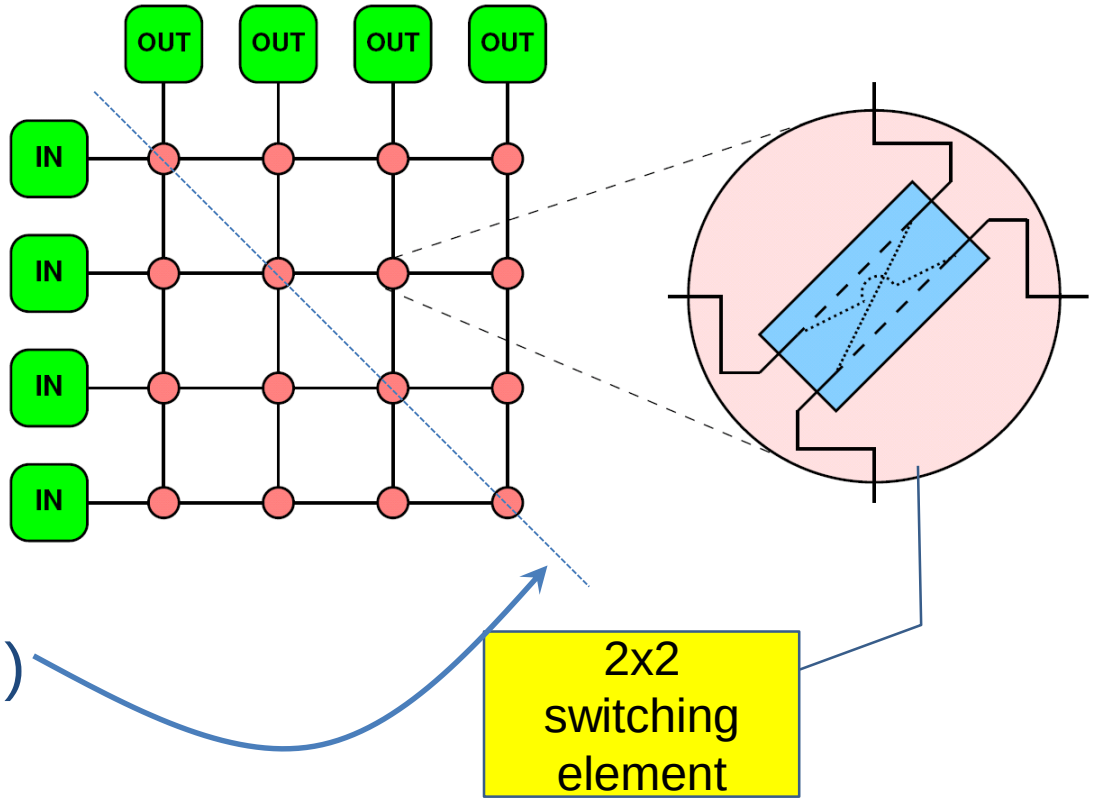
- Bus can be used by one connection at a time
- Bandwidth is shared among all devices
- Bisection BW is constant → $B_b/Nnode_s \sim 1/N_{nodes}$
- Examples: PCI bus lane, diagnostic buses
- Advantages
  - Low latency
  - Easy to implement



- 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

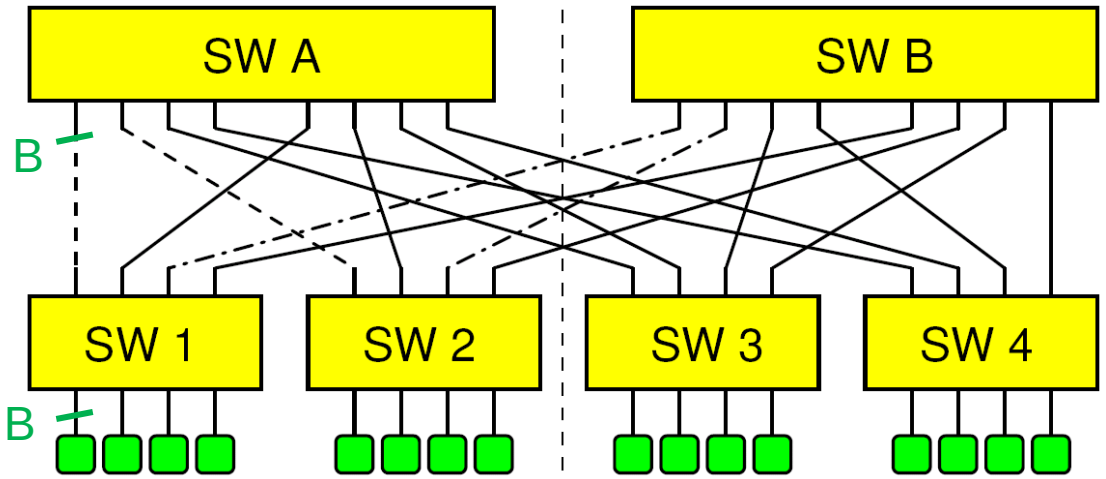
- Standard clusters are built with switched networks
- Compute nodes (“devices”) are split up in groups – each group is connected to single (non-blocking crossbar-)switch (“leaf switches”)
- Leaf switches are connected with each other using an additional switch hierarchy (“spine switches”) or directly (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

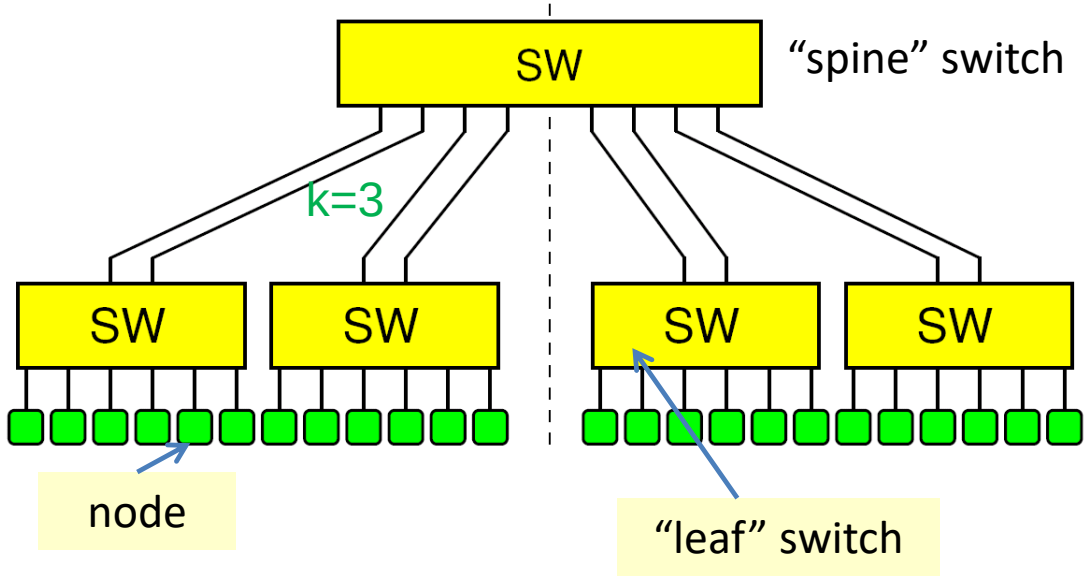
- “Perfect” world: “Fully non-blocking”, i.e. any choice of $N_{nodes}/2$ disjoint node (device) pairs can communicate at full speed

“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

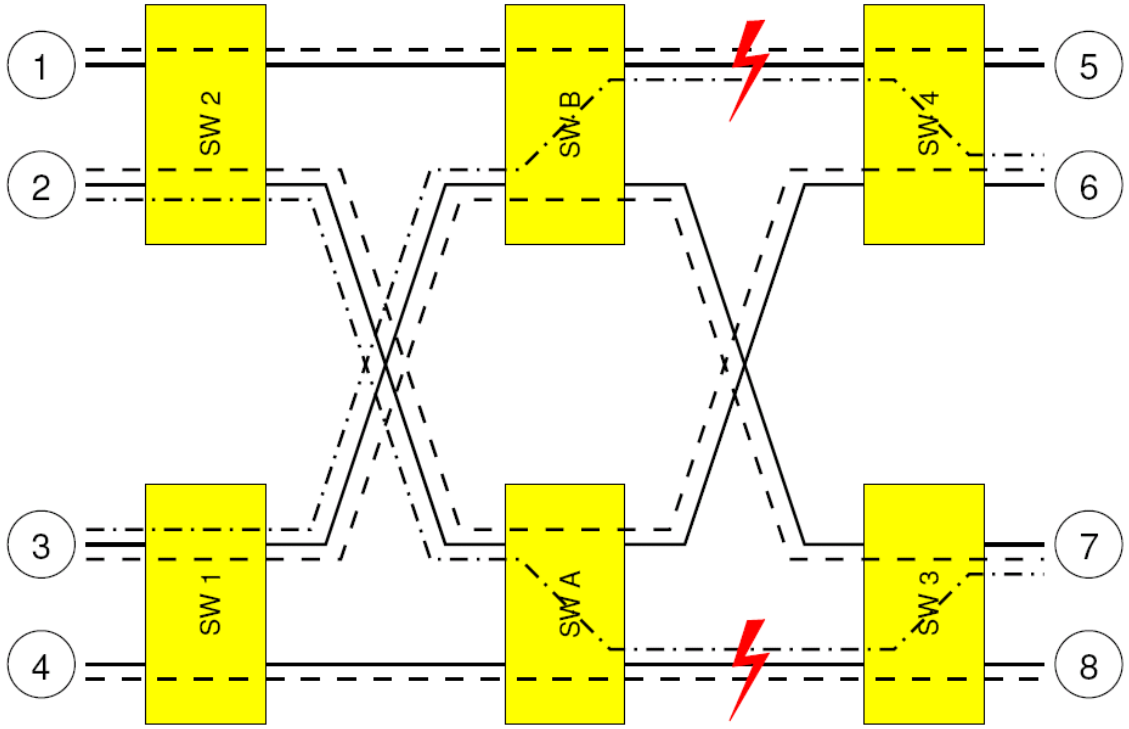


- “Oversubscribed”
  - Spine does not support $N_{nodes}/2$ full BW end-to-end connections
  - B_b/N_{nodes} = const. = B/(2k), with k oversubscription factor
  - Resource management (job placement) is crucial

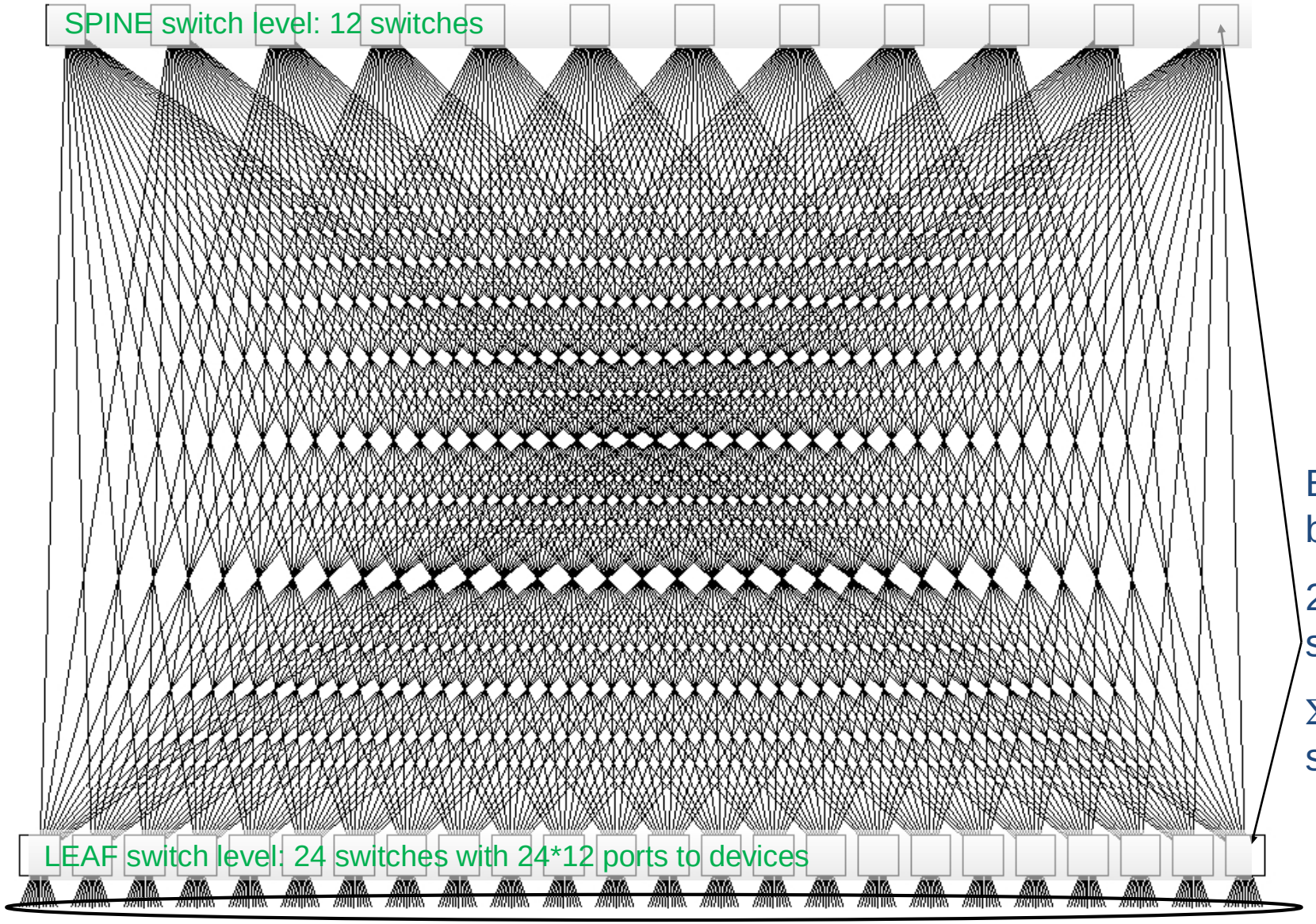


Fat trees and static routing

- 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

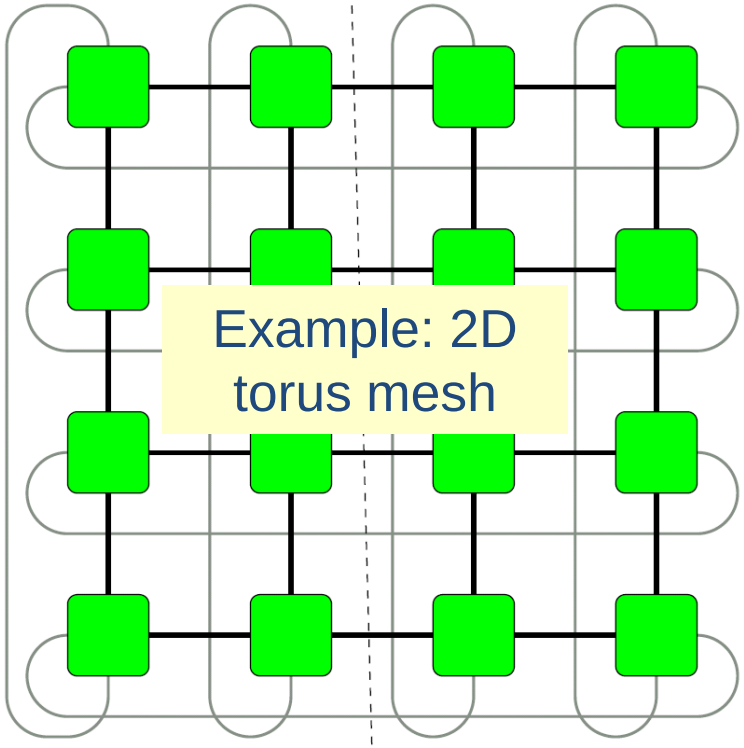
Σ = 12+24 switches

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 (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
    - HDR: 200 Gbit/s per link and direction (often used as 2xHDR100)
- “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!

NHR FAU

FAU Friedrich-Alexander-Universität Erlangen-Nürnberg

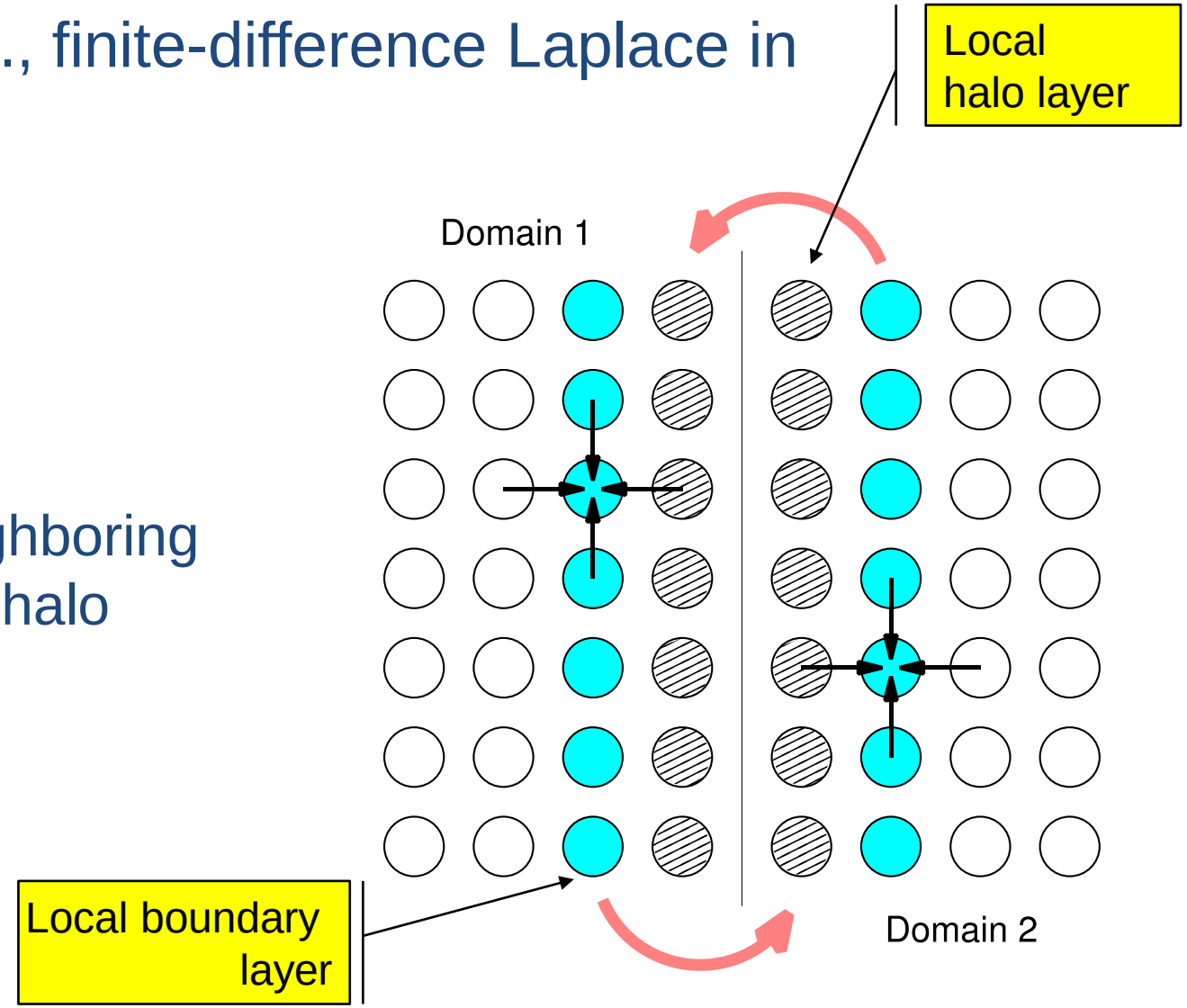
Parallelism vs. Communication

Modelling a 3D parallel Jacobi solver

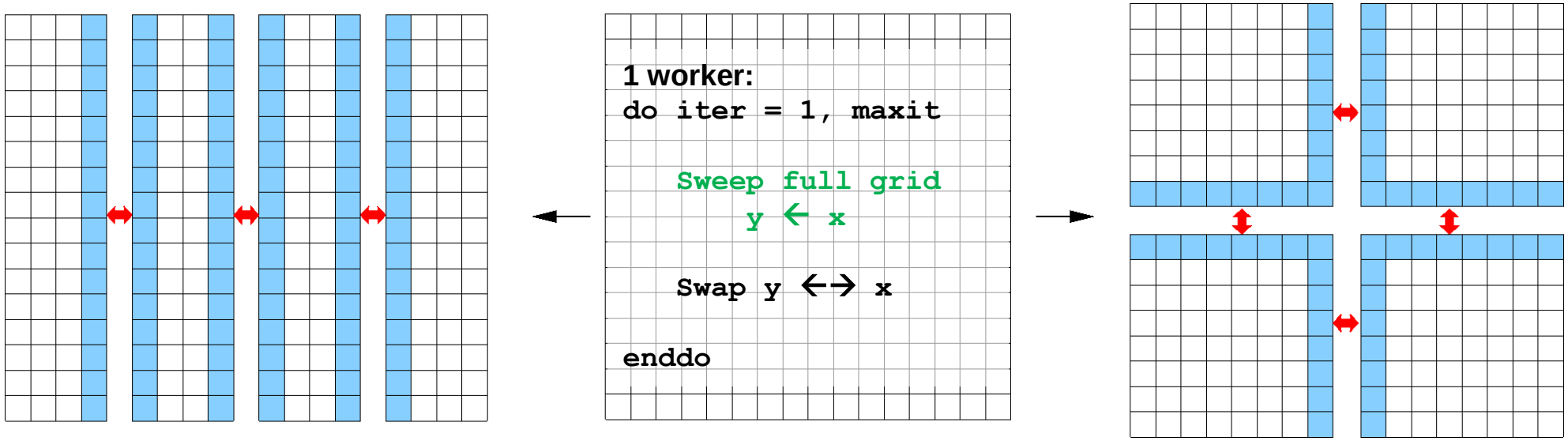


Domain decomposition: boundary exchange

- Example: distance-1 stencil (e.g., finite-difference Laplace in Jacobi smoother)
- Every domain/worker:
  - Update local domain
  - Send local boundary layers to neighboring domain/worker
  - Receive boundary data from neighboring domain/worker and copy to local halo
- Communication volume per domain/worker: Surface of local domain



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



```
1 worker:
do iter = 1, maxit

   Sweep full grid
      y ← x

   Swap y ↔ x

enddo
```

```
All N workers:
do iter = 1, maxit

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

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

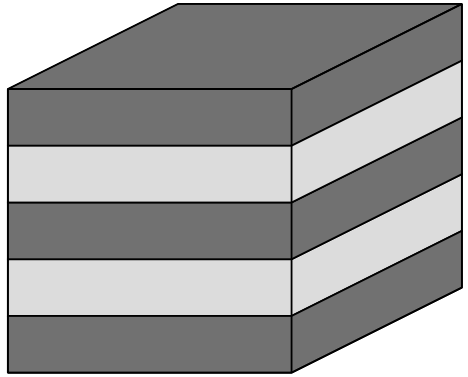
"Squares"
#boundary cells:
$const \times \frac{L}{\sqrt{N}}$

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

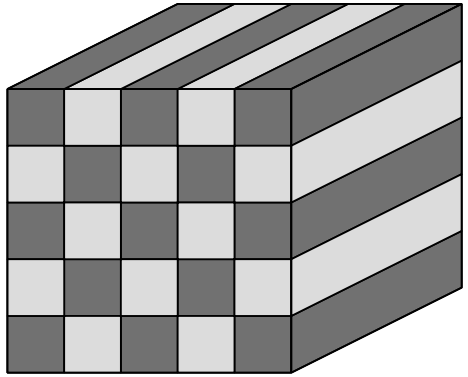
Parallelism vs. communication
Domain decomposition: basic 3D decompositions

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

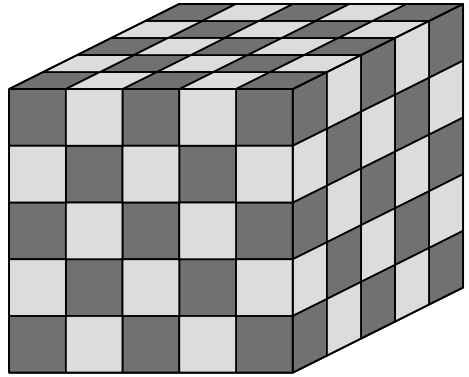
1D decomposition
(N=5 workers)



2D decomposition
(N=25 workers)



3D decomposition
(N=125 workers)



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

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

#sites in each boundary layer

Parallelism vs. communication
Domain decomposition: scalability

- Assume fully non-blocking network, i.e. $N/2$ pairs of neighbors communicate in parallel:

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

- 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: $T(N) = T_s + (L^3/(NP)) + T_{comm}(N)$
- Network latency is constant: $T(N) = T_s + \frac{L^3}{NP} + \lambda + \frac{V(L,N)}{b_{network}}$

Network latency enters as "serial fraction"

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

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

Powerful workers require fast networks!

- Using $\bar{c} = \frac{2w}{L}\,\frac{P}{b_{network}}$ we get
  - 1D decomposition: $S_{1d} = \frac{1}{\bar{s}+\frac{1}{N}+\bar{\lambda}+\bar{c}}$
  - 2D decomposition: $S_{2d} = \frac{1}{\bar{s}+\frac{1}{N}+\bar{\lambda}+\frac{2\bar{c}}{\sqrt{N}}}$
  - 3D decomposition: $S_{3d} = \frac{1}{\bar{s}+\frac{1}{N}+\bar{\lambda}+\frac{3\bar{c}}{N^{2/3}}}$

Constant communication time is like serial fraction → $S_{1d} \xrightarrow{N\to\infty} \frac{1}{\bar{s}+\bar{\lambda}+\bar{c}}$

Communication time goes down with N → $S_{3d} \xrightarrow{N\gg1} \frac{1}{\bar{s}+\bar{\lambda}+3\bar{c}/N^{2/3}}$

→ Advice: Use 3D decomposition if possible!

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

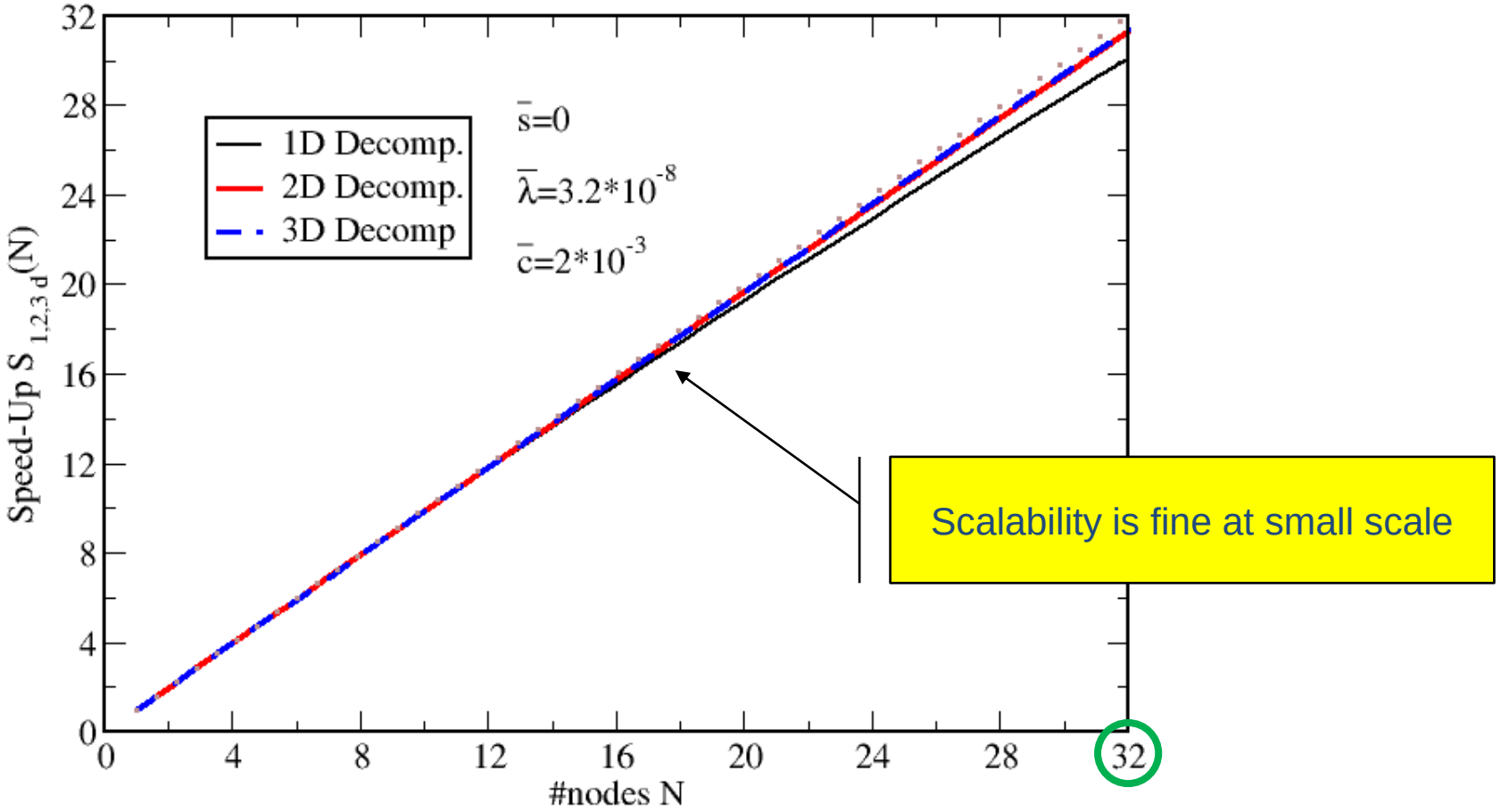
    → Single-node runtime $T(1) = \frac{L^3}{P} = 31.25\,s$
  - Network performance (~ QDR IB): $b_{network} = \frac{3GB}{s}$; $\lambda = 10^{-6}s$

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

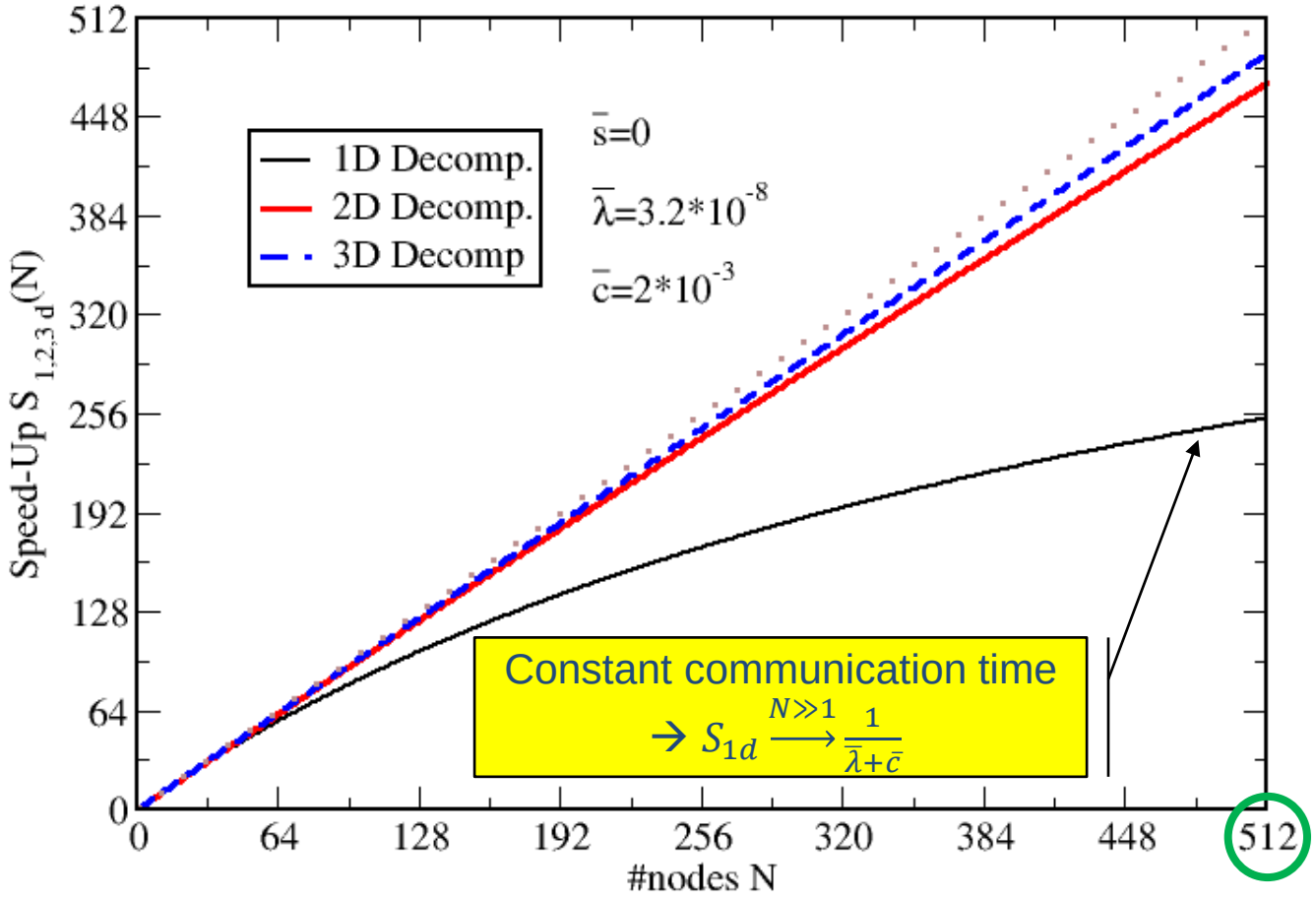
→ $\bar{c} = \frac{2w}{L}\frac{P}{b_{network}} = 2 \times 10^{-3}$

→ $\bar{s} = 0$ (assumption)

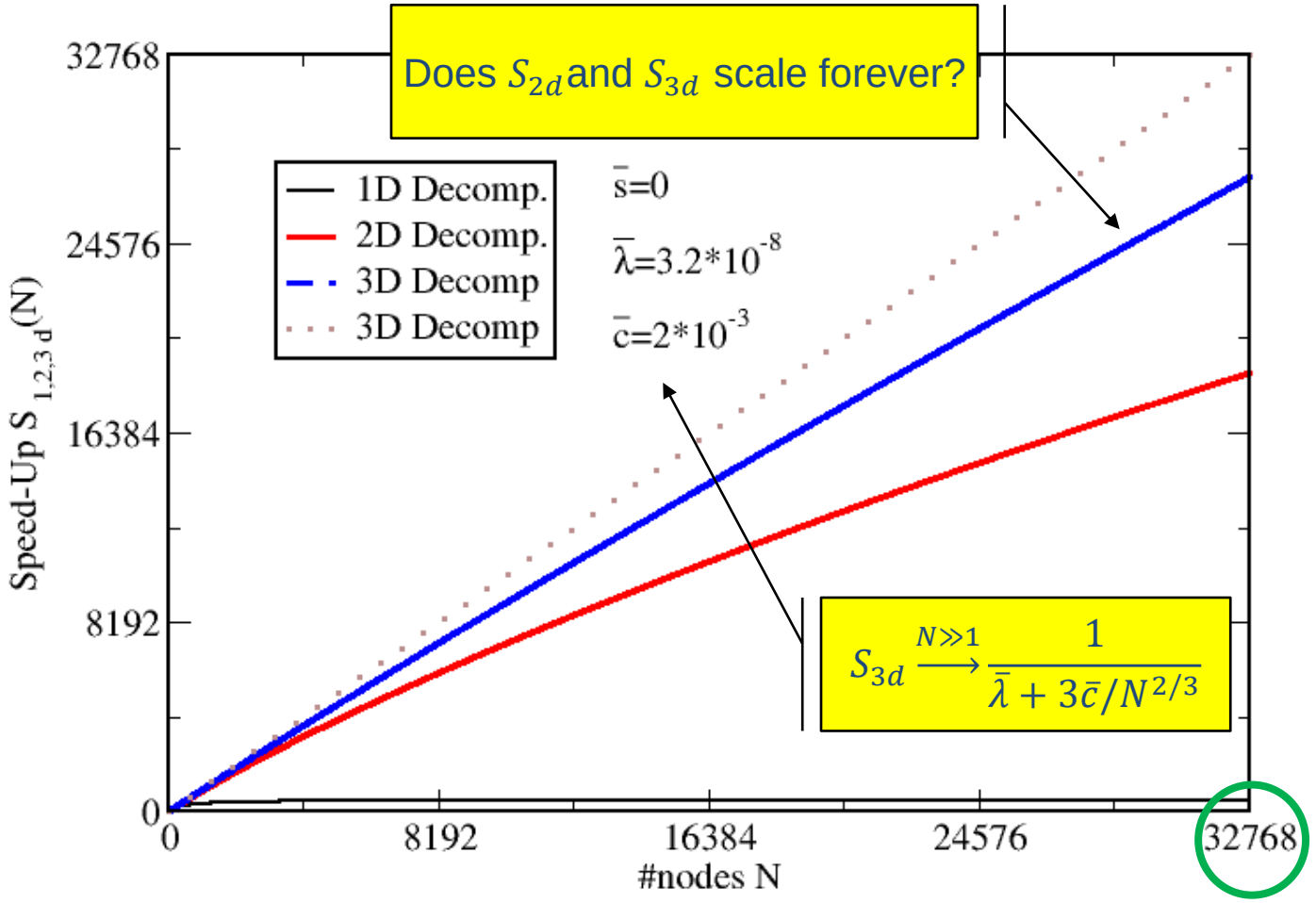
Parallelism vs. communication
Domain decomposition: scalability (small scale)



Parallelism vs. communication
Domain decomposition: scalability (medium scale)



Parallelism vs. communication
Domain decomposition: scalability (large scale)



Conclusions on scalability models with domain decomposition

- Communication and data distribution always matters on large scale – impact may not be visible at small scale
- Finite communication latency is like a 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, …