Foundations of High Performance Computing

Lecture 8: Performance Evaluation of MPI applications

2020-2021 Stefano Cozzini

“Foundation of HPC” course

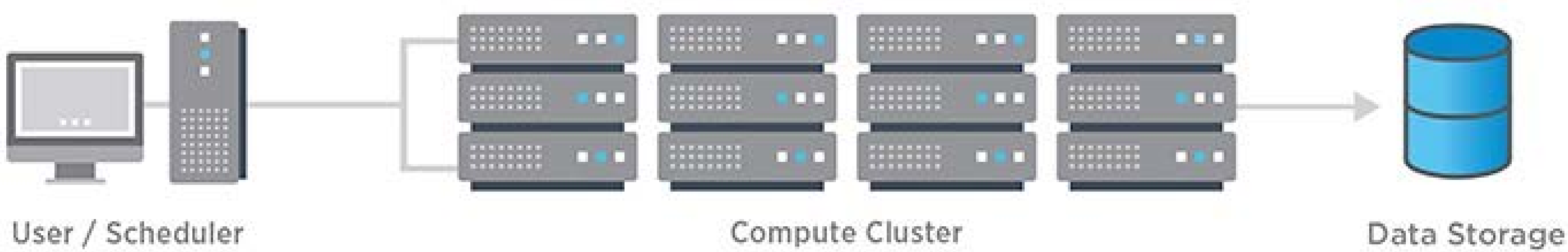
DATA SCIENCE & SCIENTIFIC COMPUTING

Agenda

- Network basic for parallel architecture
- Network basic performance characteristics
- Discussion of the Jacoby 3D solver

Recap on HPC architecture

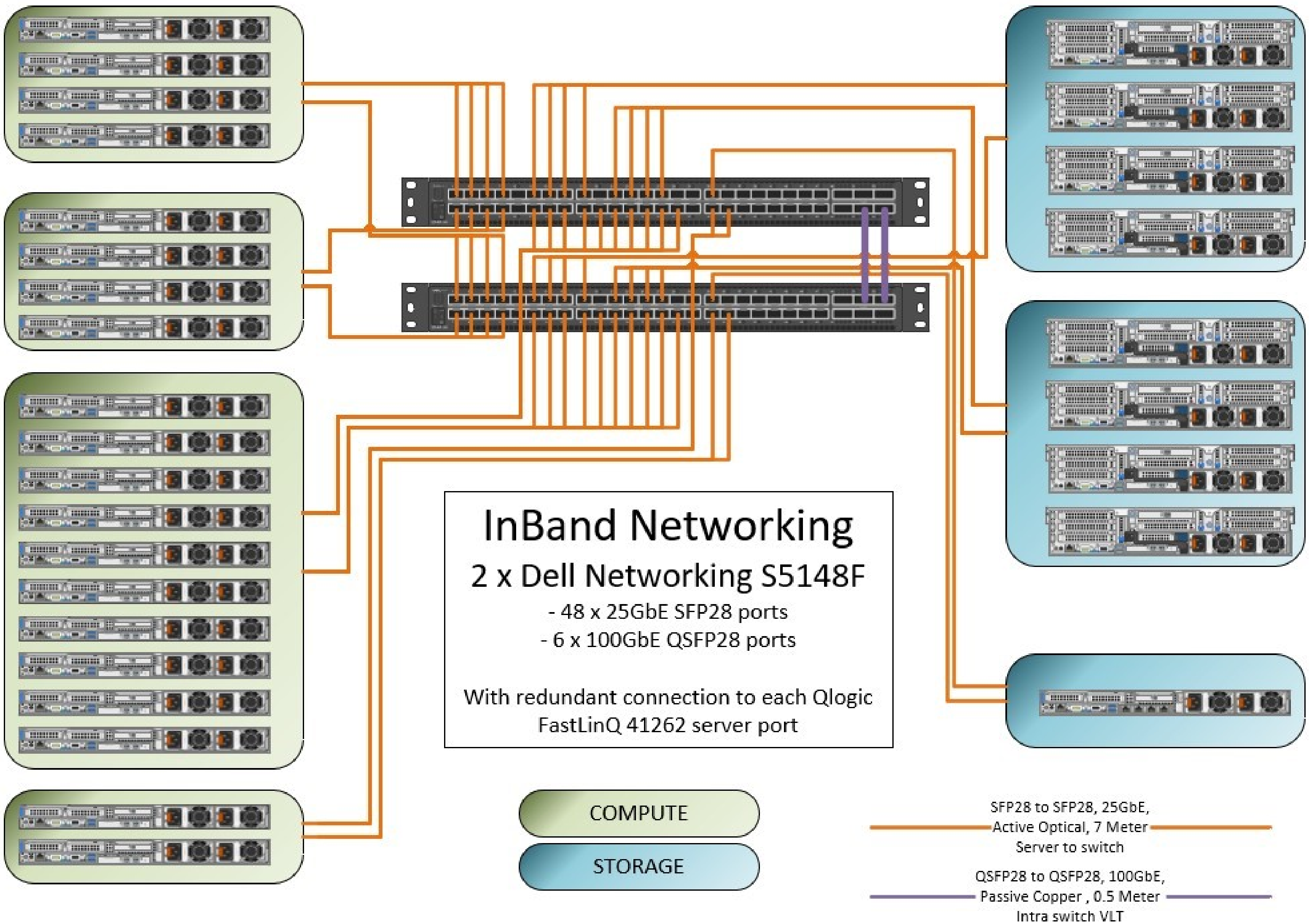
- Several computers (nodes) often in special cases for easy mounting in a rack
- One or more networks (interconnects) to hook the nodes together
- MP application’ performance rely on the characteristics of the networks.



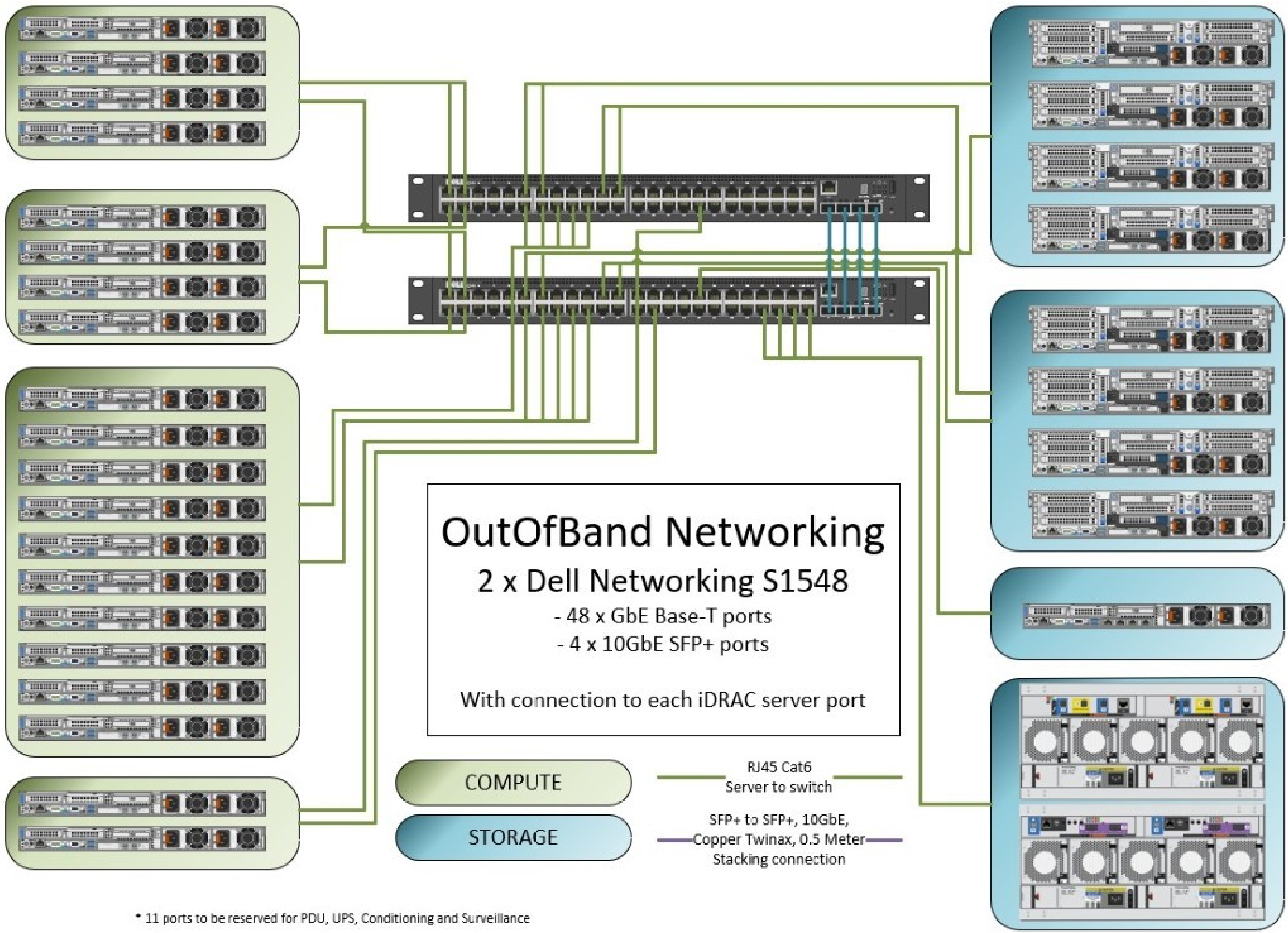
Network cluster classification

- HIGH SPEED NETWORK
  - parallel computation
  - low latency /high bandwidth
  - Usual choices: Infiniband...
- I/O NETWORK
  - I/O requests (NFS and/or parallel FS)
  - latency not fundamental/ good bandwidth
  - GIGABIT could be ok /10Gb and/or Infiniband better
- In band Management network
  - management traffic of all services (LRMS/NFS/software etc..)
- Out of band Management network:
  - Remote control of nodes and any other device

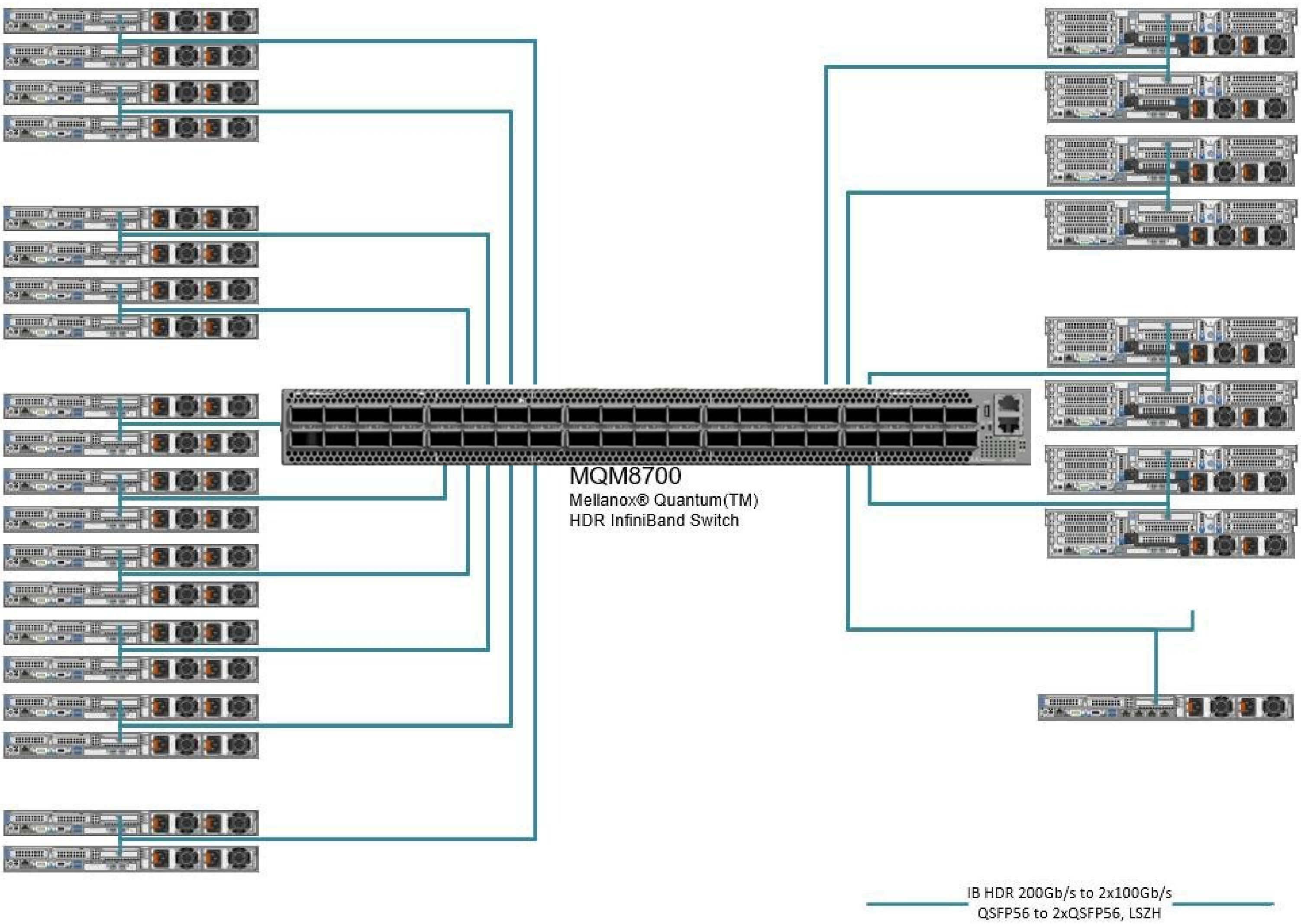
Orfeo in band management network: 25 Gbit ethernet



Orfeo out of band management network: 10 Gbit ethernet



Orfeo High Speed network: 100 Gbit Infiniband



Orfeo network classification

- HIGH SPEED NETWORK
- I/O NETWORK

100 Gbit HDR Infiniband

- In band Management network

25Gbit Ethernet

- Out of band Management network:

10Gbit Ethernet

how to model network performance ?

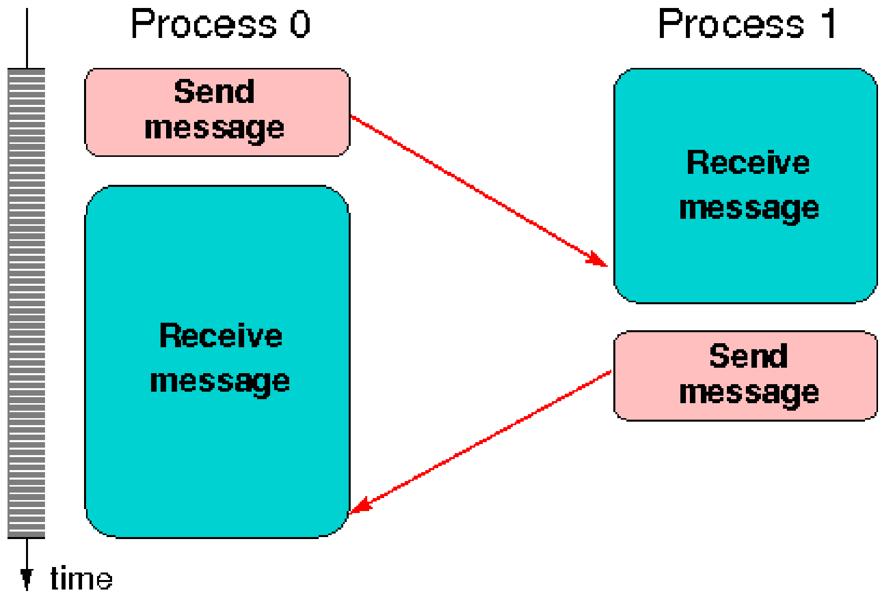
- Network capacity to tranfer data
- Very simple model :
  - Total transfer time of a message

$$T_{comm} = \lambda + (Size\ of\ message) / b_{network}$$

  - λ is the latency of the network : i.e. the time to setup the communication channel
  - $b_{network}$ is the asymptotic network bandwith measured in Mb/sec.

How can we estimate/measure latency and bandwidth ?

- Using a simple “Ping-Pong” program :
  - Two processes on the network exchange point-to-point message.
  - A single message of N Bytes is sent forward and backward: data transfer is 2N



Ping-Pong algorithm

```
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()
    MBYTES = 2*N/(E-S)/1.d6 ! MBytes/sec rate
    TIME = (E-S)/(2*1.d6) ! transfer time in microsecs
                          ! for single message
 else
   targetID= 0
    call Receive_message(buffer,N,targetID)
    call Send_message(buffer,N,targetID)
 endif
```

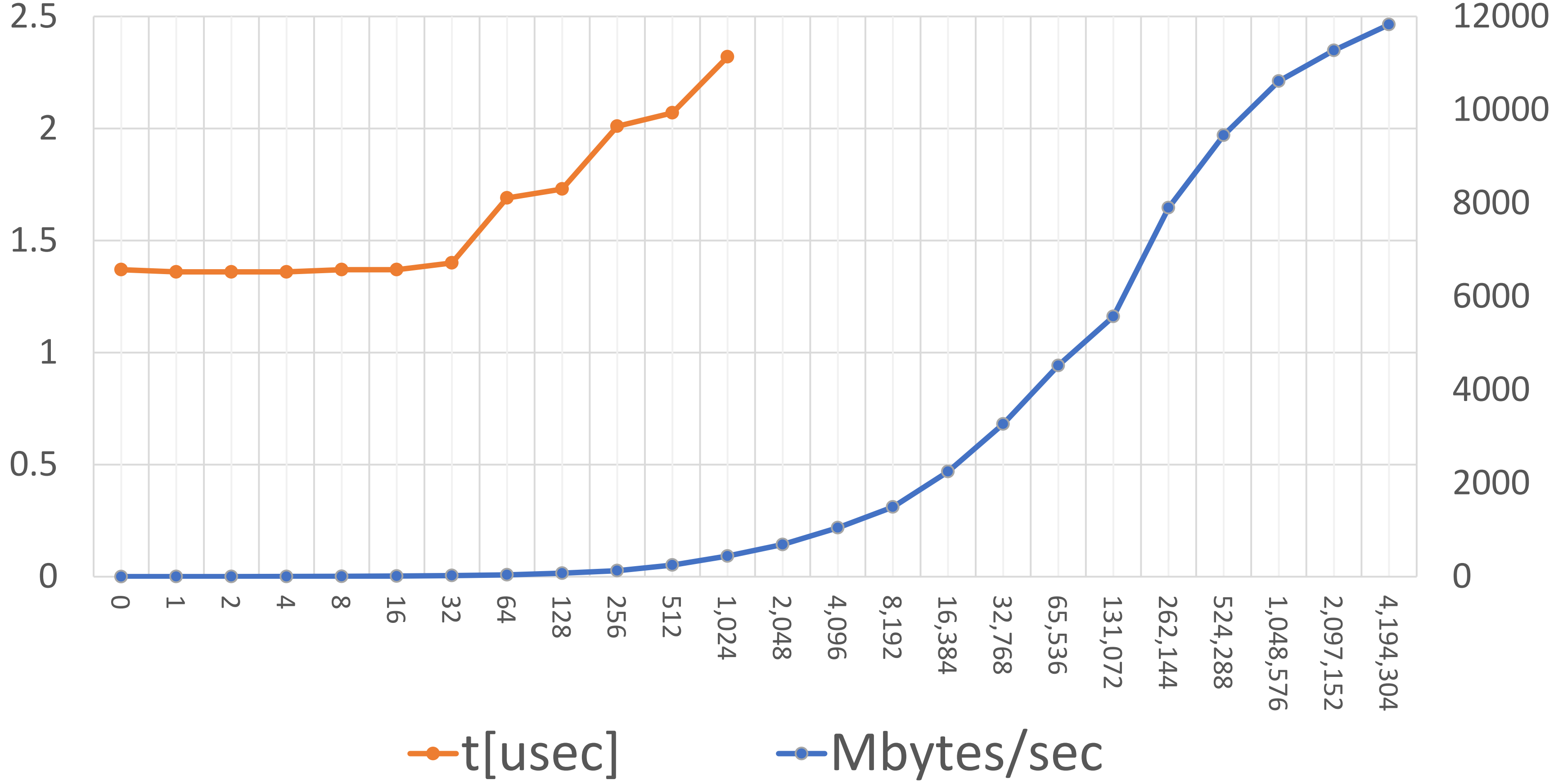
Ping Pong implementations

- Available on the most common benchmark suite:
- IMB: Intel MPI benchmark
  - intel/mpi-benchmarks (github.com)
- OSU microbenchmarks
  - MVAPICH :: Benchmarks (ohio-state.edu)

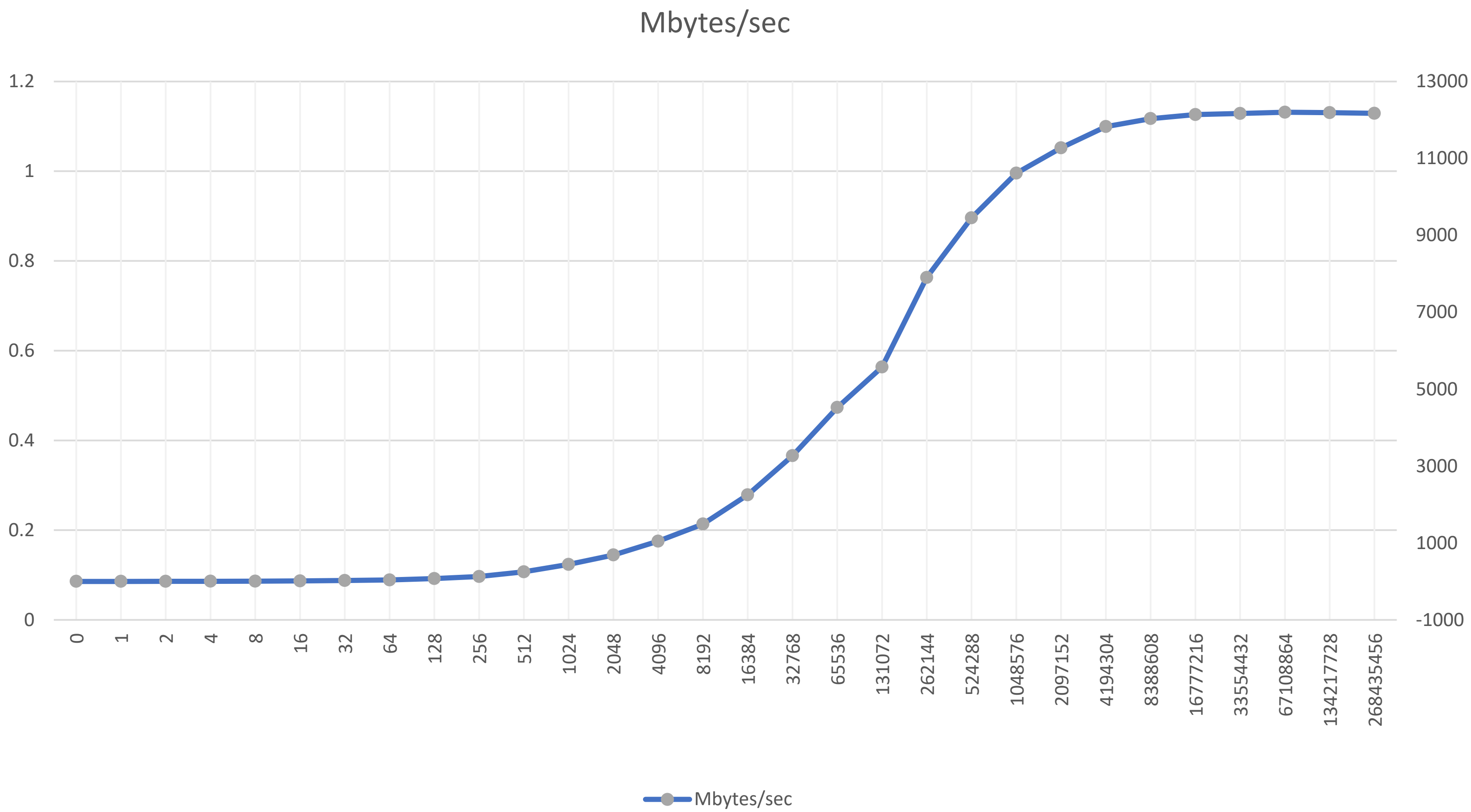
Measuring MPI point-to-point performance on Orfeo

- Download Intel MPI benchmark
- Compile it
- Run it
- Get results and interpret it
- See README file in day16 directory..

Measuring MPI point-to-point performance on Orfeo



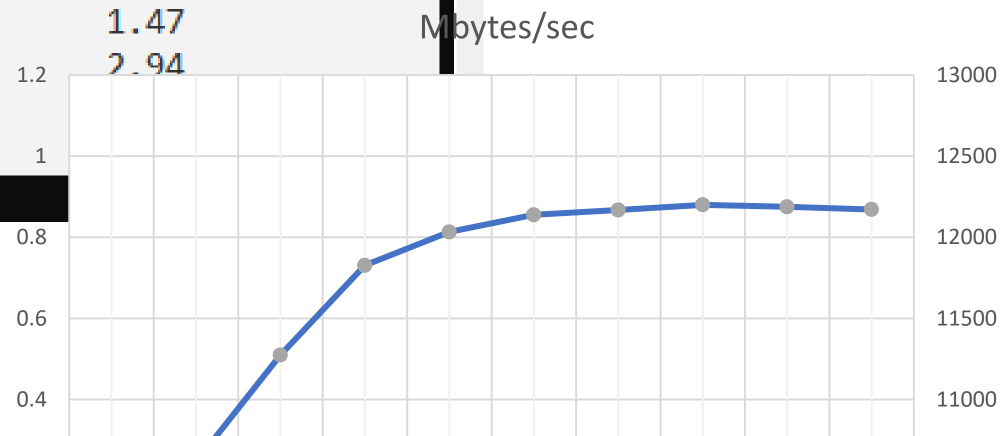
Measuring MPI performance on Orfeo



Extrapolating values for internode communication

```
#-----------------------------------------------------
# Benchmarking PingPong
# #processes = 2
#-----------------------------------------------------
       #bytes #repetitions      t[usec]   Mbytes/sec
            0         1000         1.36         0.00
            1         1000         1.36         0.74
            2         1000         1.36         1.47
            4         1000         1.36         2.94
            8         1000         1.36
           16         1000         1.36
           32         1000         1.40
```

λ

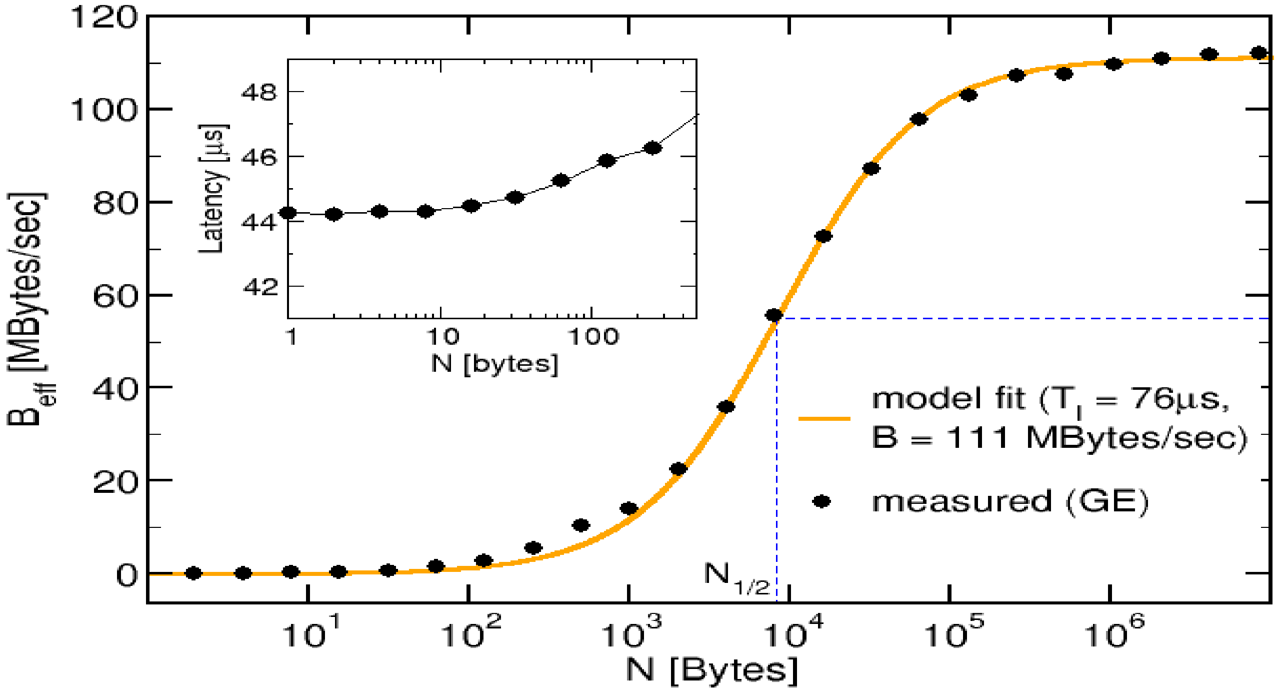


$b_{network}$

```
      4194304           10       355.03     11813.78
      8388608            5       697.14     12032.92
     16777216            2      1382.15     12138.48
     33554432            1      2757.62     12167.91
     67108864            1      5500.71     12200.04
    134217728            1     11012.36     12187.92
    268435456            1     22054.87     12171.25
```

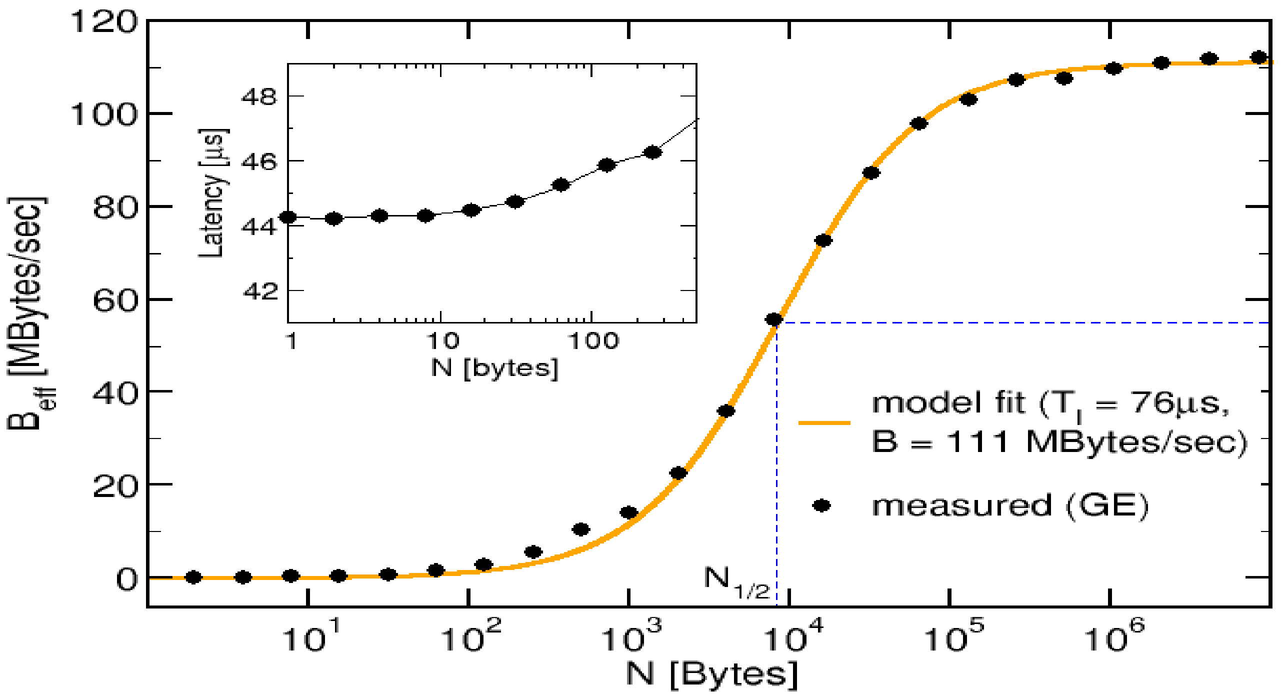
Gbit ethernet performance

- Latency: ~44 microsecond
- BW: ~ 111 Mb/second (theoretical one: 1000/8=125)



Gbit ethernet performance model

- Latency: 76 microsecond >> 44 (qualitative agreement)
- BW: good agreement



Network performance modeling

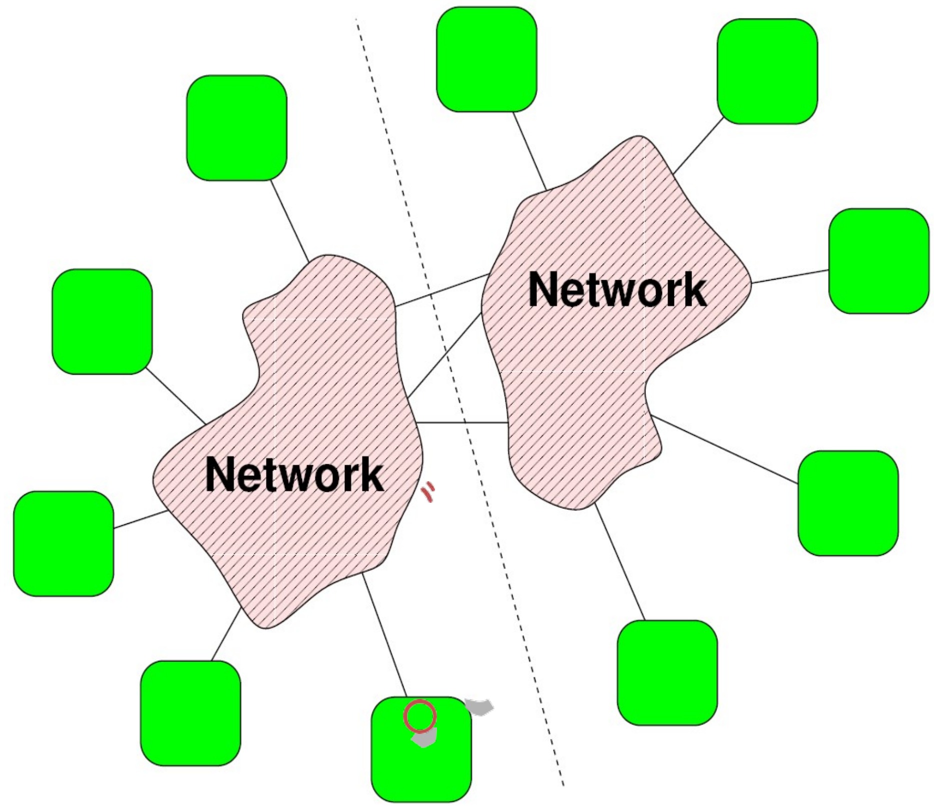
- “First Principles” modeling of Beff(V) provides good qualitative results but quantitative description in particular of latency dominated region (V small) may fail because
  - Overhead for transmission protocols, e.g. message headers
  - Minimum frame size for message transmission, e.g. TCP/IP over 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”

Network topology

- How the components are connected.
- Important properties
  - Diameter: maximum distance between any two nodes in the network (hop count, or # of links).
  - Nodal degree: how many links connect to each node.
  - Bisection bandwidth: The smallest bandwidth between half of the nodes to another half of the nodes.
- A good topology: small diameter, small nodal degree, large bisection bandwidth

Bisection bandwidth: B_b

- Split N nodes into two groups of N/2 nodes such that the bandwidth between these two groups is minimum
- general metric for the data transfer "capability" of a system
- More meaningful metric in terms of system scalability: B_b /Nodes

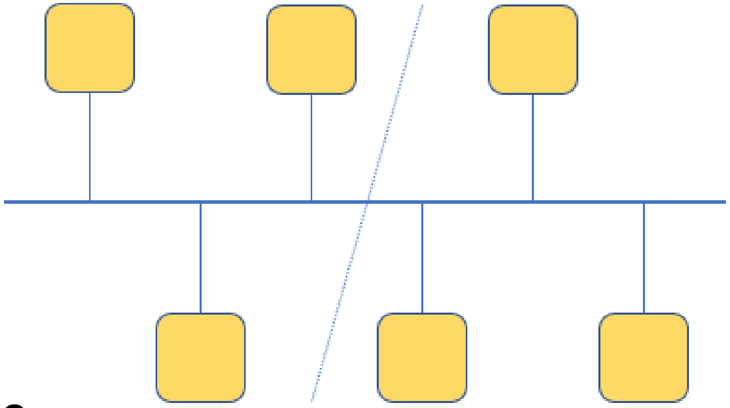


Common Topologies in HPC

- Bus
- Crossbar switches
- Fat tree
- CBB (Constant Bi-sectional Bandwidth)
- Mesh
  - 3D torus

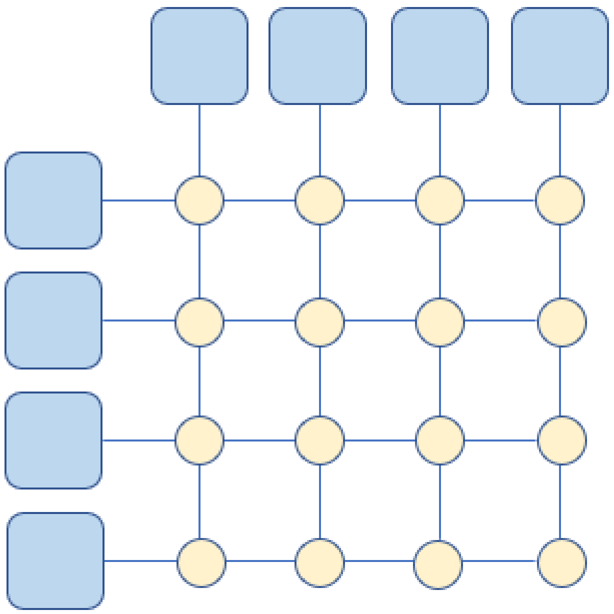
Bus topology

- Bus can be used by one connection at a time
- Bandwidth is shared among all devices
- Bisection BW is constant: Bb/Nnodes ~ 1/Nnodes
- Examples: PCI bus
- Advantages
  - Low latency
  - Easy to implement
- Disadvantages
  - Shared bandwidth, not scalable
  - Problems with failure resiliency (one defective agent may block bus)



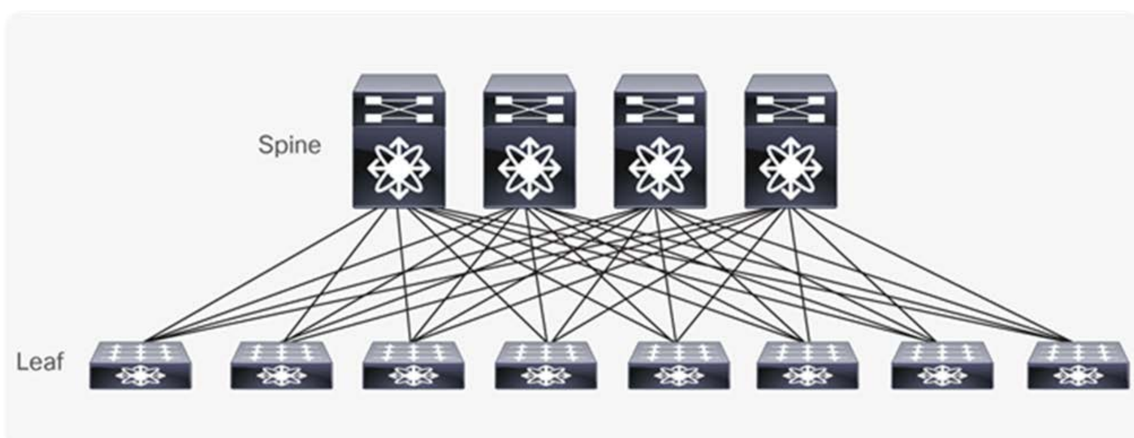
Non blocking crossbar switch

- 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
- Switches can be cascaded to form hierarchies (common case)
- Allows scalable communication at high hardware/energy costs
- Not feasible for large HPC installations



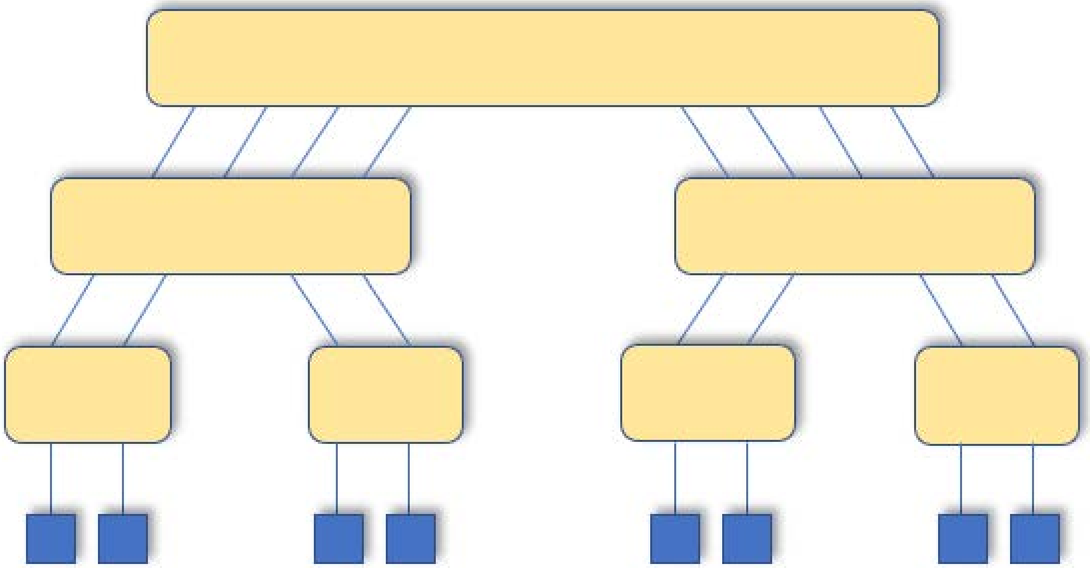
Switches and Fat-Trees

- HPC 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/edge switches")
- Leaf switches are connected with each other using an additional switch hierarchy ("spine switches") or directly (for small configs.)
- "Perfect" world: fat- trees



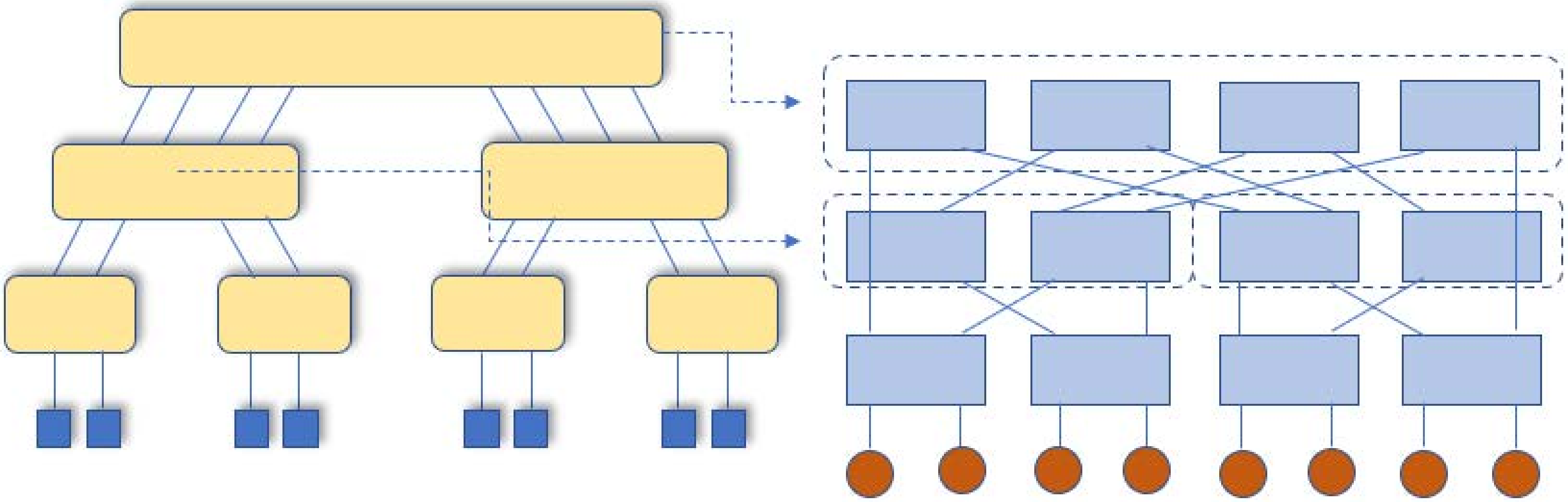
Fat-trees switch hierarchy..

- Fully non-blocking:
  - Each level double the number of link of the switches
  - Not practical. Root is NXN switch



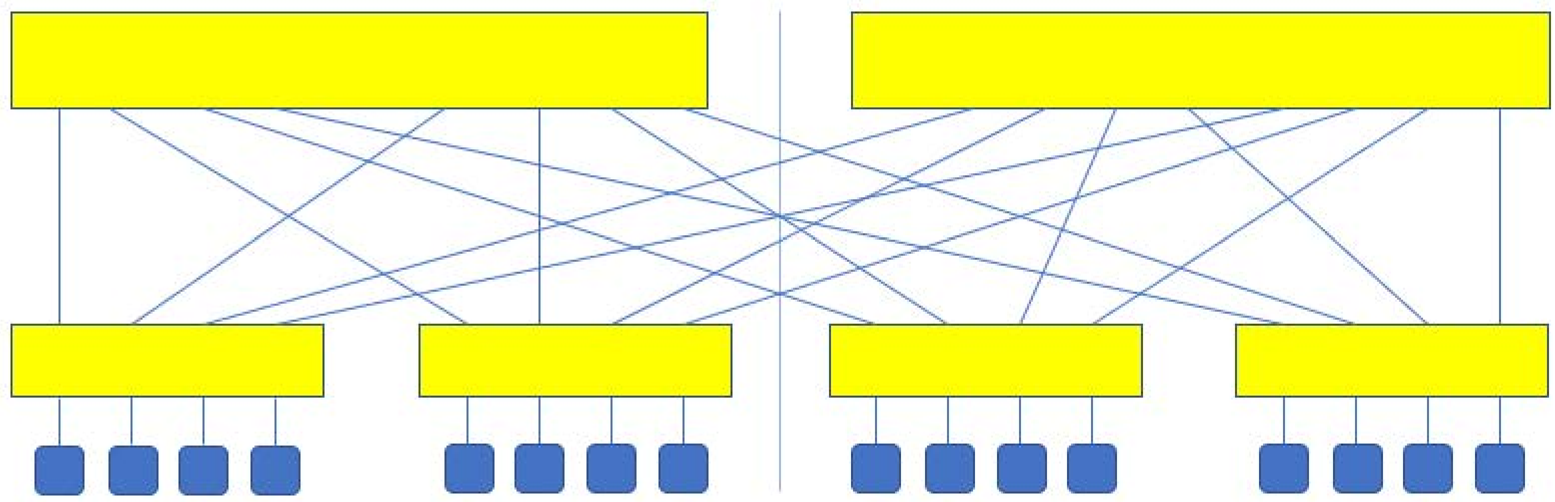
Practical fat-tree implementation

- Use smaller switches to approximate large switches.
- Most commodity large clusters use this topology.
- Also call constant bisection bandwidth network (CBB)
  - $N_{nodes}/2$ end-to-end connections with full bandwidth
  - $B_b = B * N_{nodes}/2$
  - B_b / N_{nodes} = const. = $B/2$



Two level CBB example

- $N_{nodes}/2$ end-to-end connections with full bandwidth: 8
- $B_b = B * N_{nodes}/2 = 8B$
- $B_b / N_{nodes} = const. = B/2$



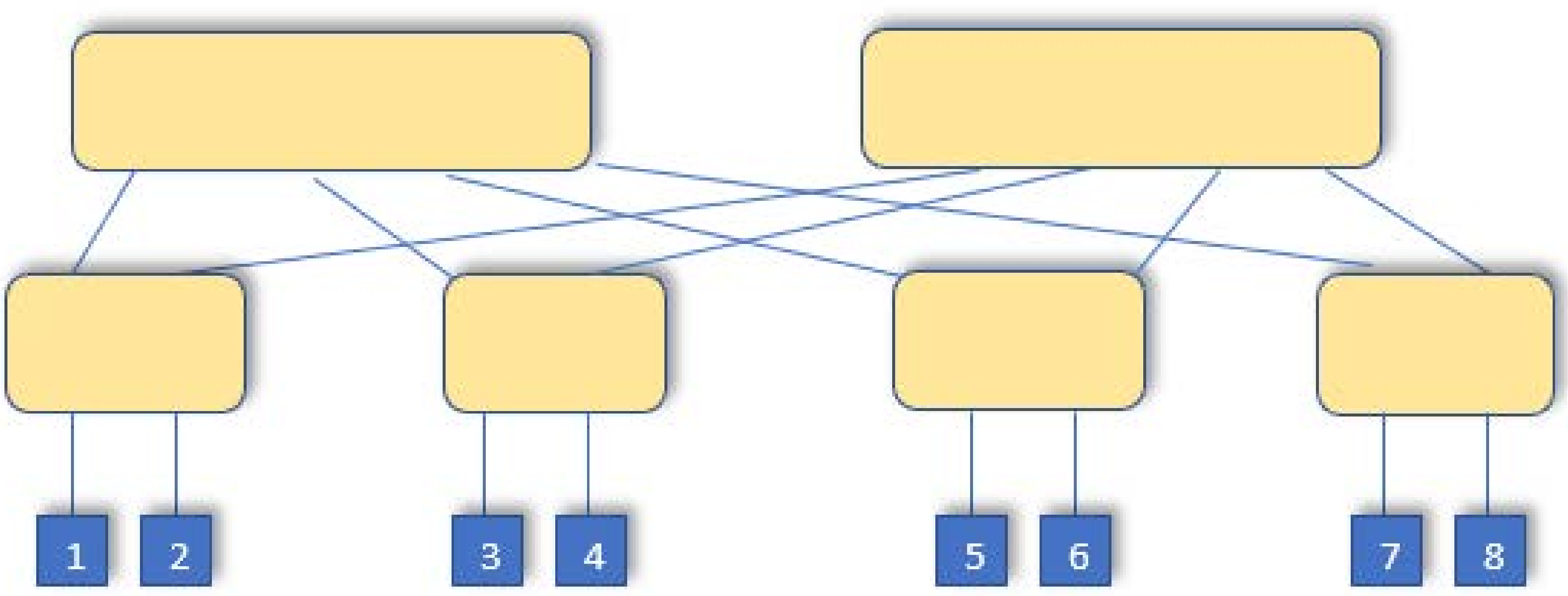
Fat tree and static routing

- Generally, CBB are using static routing algorithm.
- path taken between any two node pairs is statically computed
- Full bandwidth is not always seen in practice.
- The number of potential routes *R* for a total node count of *N*: $R=N(N-1)=N^2-N$.
- Number of routes o(N^2)
- Number of Intermediate Spine link is o (N)

➔ There are scenarios where certain host communications will use the same Intermediate Spine link

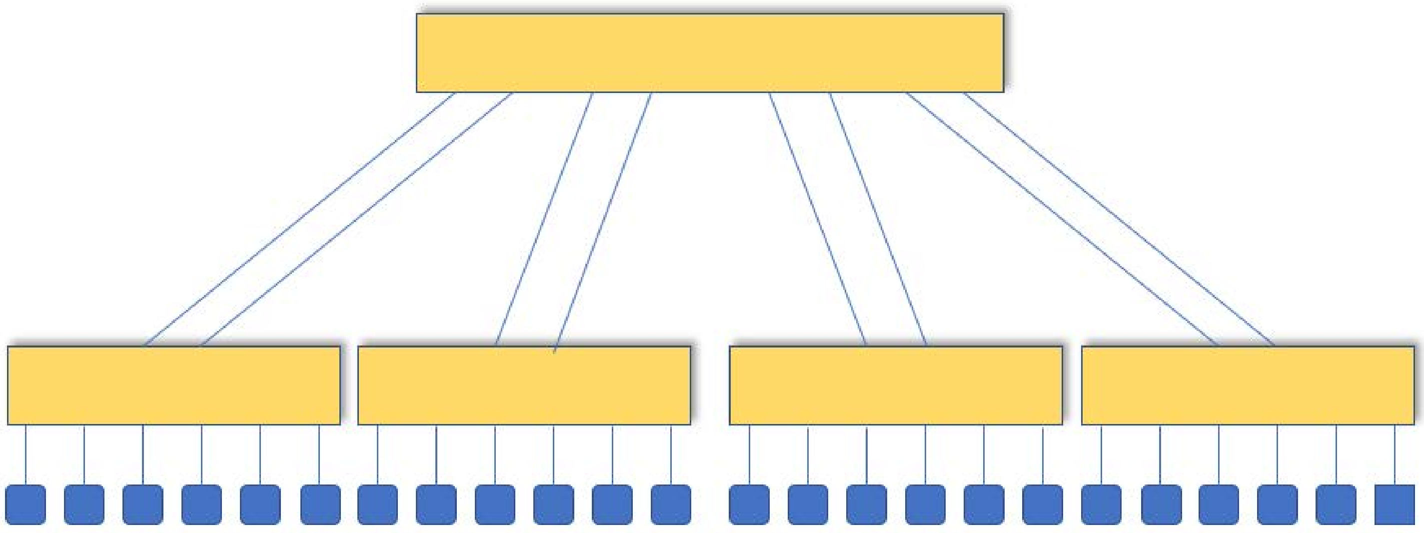
Example

- For 1->5,2->6,3->7,4->8 is ok
- For 1->5,2->7,3->6,4->8 is no longer fine if there is static routing



Oversubscription

- Spine does not support Nnodes/2 full BW end-to-end connections
- B_b/Nnodes = const. = B/(2k), with k oversubscription factor (k=3 for the example)
- Resource management (job placement) is crucial

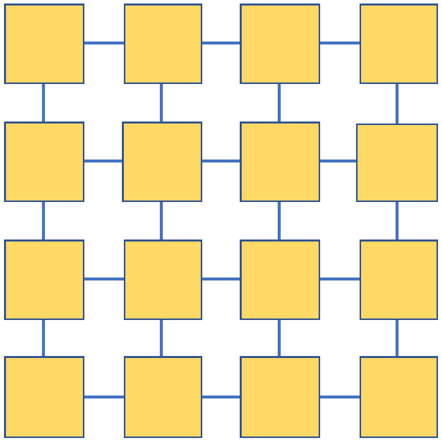


High speed networks

- Infiniband
  - The de-facto standard
  - 27% of ToP500 are based on infiniband
- Omni Path
  - started by Intel in 2015
  - one of the youngest HPC interconnects
  - 8.6% of Top4500 are Omni-Path systems
- Both are used behind a MPI implementation..

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
- Different from a crossbar!
- Intelligent resource management and routing algorithms are essential



Infiniband speed: physical layer..

- InfiniBand uses serial stream of bits for data transfer
- Linkwidth
  - 1x – One differential pair per Tx/Rx
  - 4x – Four differential pairs per Tx/Rx
  - 12x - Twelve differential pairs per Tx and per Rx
- LinkSpeed
  - Single Data Rate (SDR) - 2.5Gb/s per lane (10Gb/s for 4x)
  - Double Data Rate (DDR) - 5Gb/s per lane (20Gb/s for 4x)
  - Quad Data Rate (QDR) - 10Gb/s per lane (40Gb/s for 4x)
  - Fourteen Data Rate (FDR) - 14Gb/s per lane (56Gb/s for 4x)
  - Enhanced Data rate (EDR) - 25Gb/s per lane (100Gb/s for 4x)
- Linkrate
  - Multiplication of the link width and link speed
  - Most common shipping today is 4x ports DFR/EDR

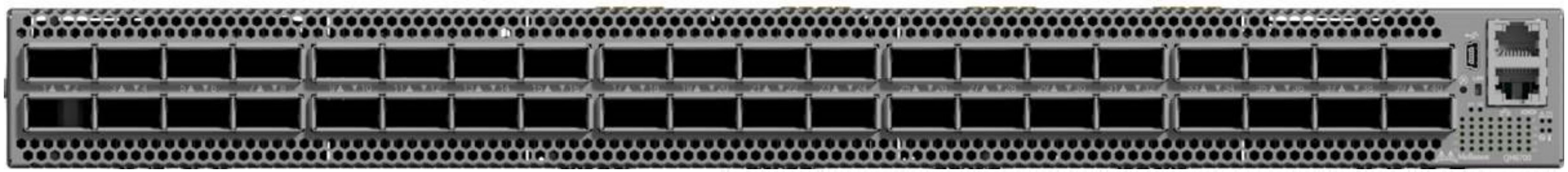
Infiniband speed: data encoding

- For SDR, DDR and QDR, links use 8b/10b encoding:
  - every 10 bits sent carry 8bits of data
- Thus single, double, and quad data rates carry 2, 4, or 8 Gbit/s useful data, respectively.
- For FDR and EDR, links use 64b/66b encoding
  - every 66 bits sent carry 64 bits of data.

InfiniBand performance

	SDR	DDR	QDR	FDR	EDR	HDR
Signaling rate (Gbit/s)	2.5	5	10	14.0625	25.78125	50
Encoding (bits)	8/10	8/10	8/10	64/66	64/66	64/66
Theoretical throughput 1x (Gbit/s)	2	4	8	13.64	25	50
Theoretical throughput 4x (Gbit/s)	8	16	32	54.54	100	200
Theoretical throughput 12x (Gbit/s)	24	48	96	163.64	300	600

ORFEO IB network



Performance

• 40 x HDR 200Gb/s ports in a 1U switch

• 80 x HDR100 100Gb/s ports (using splitter cables)

• 16Tb/s aggregate switch throughput

• Sub-130ns switch latency

Our network: ORFEO ones..

- We can assume full non-blocking network:

  P/2 pair of nodes communicate in parallel at full speed

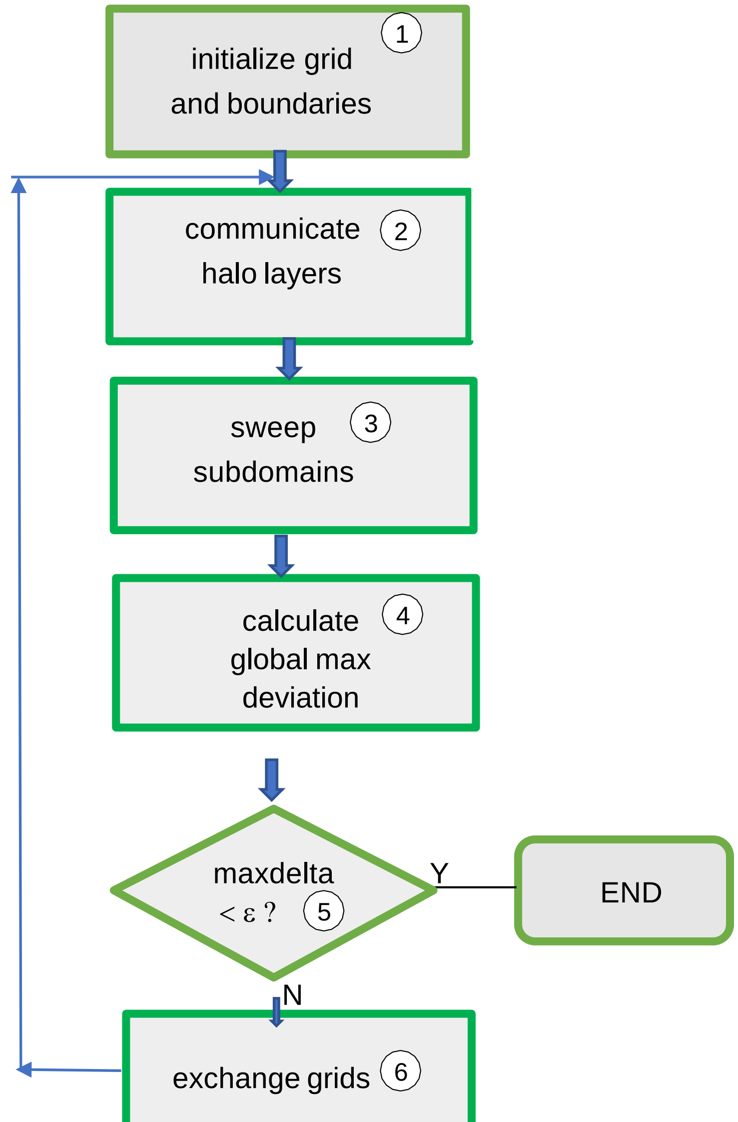
$$T_{comm} = \lambda + \text{message-size}/b_{network}$$

- Where

  $\lambda = 1.35$ microsecond

  $b_{network} = 12\text{Gb/second}$

Jacoby solver parallel algorithm



Analysis of the Jacobi Solver…

- Performance depends on L grid point over a $N=N_xN_yN_z$ processors
- Where:
  - Ts(L) sweep partl: sequential time
  - Tc(L,N) communication time

$$P(L,\vec{N}) = \frac{L^3N}{T_\mathrm{s}(L)+T_\mathrm{c}(L,\vec{N})} ,$$

Jacoby: computing time

- Cost of sweep domain:
  - The subdomain size (L^3) is the same regardless of the number of processes, so the raw compute time T_s for all cell updates in a Jacobi sweep is also constant

→ Ts= 22.07 seconds (for L=1200) on GPU node on ORFEO

Jacoby: communication time: halo exchange

- Each process sends each of the 6 of point to point communication at the same time.
- we need to consider bandwidth number for full-duplex data transfer over a single link.
- Communication time T_c depends on the number and size of domain cuts that lead to internode communication.
- Assumption:
  - copying to/from intermediate buffers and communication of a process with itself come at no cost.

Communication time:

$$T_c(L,\vec{N}) = \frac{c(L,\vec{N})}{B} + kT_\ell \ .$$

- c(L,N): amount of data volume transferred over a node's network link
- B: bidirectional bandwidth of the network link
- T_l = latency of the network
- k the largest number (over all domains) of coordinate directions in which the number of processes is greater than one.

→ $c(L,N)=L^2 *k *2 *8$ (Mb)

Our model prediction:

N	Nx	Ny	Nz	k	C(L,N)	Tc(L,N)	P(L,N)	P(1)*N/P(L,N)
1	1	1	1	0	0	0.00	78.30	1.00
2	2	1	1	2	46080	1.92	144.06	1.09
3	3	1	1	2	46080	1.92	216.09	1.09
4	2	2	1	4	92160	3.84	266.77	1.17
6	3	2	1	4	92160	3.84	400.15	1.17
8	2	2	2	6	138240	5.76	496.73	1.26
12	3	3	3	6	138240	5.76	745.10	1.26
16	4	2	2	6	138240	5.76	993.46	1.26

Exercises

- Verify if the model discussed is working correctly…