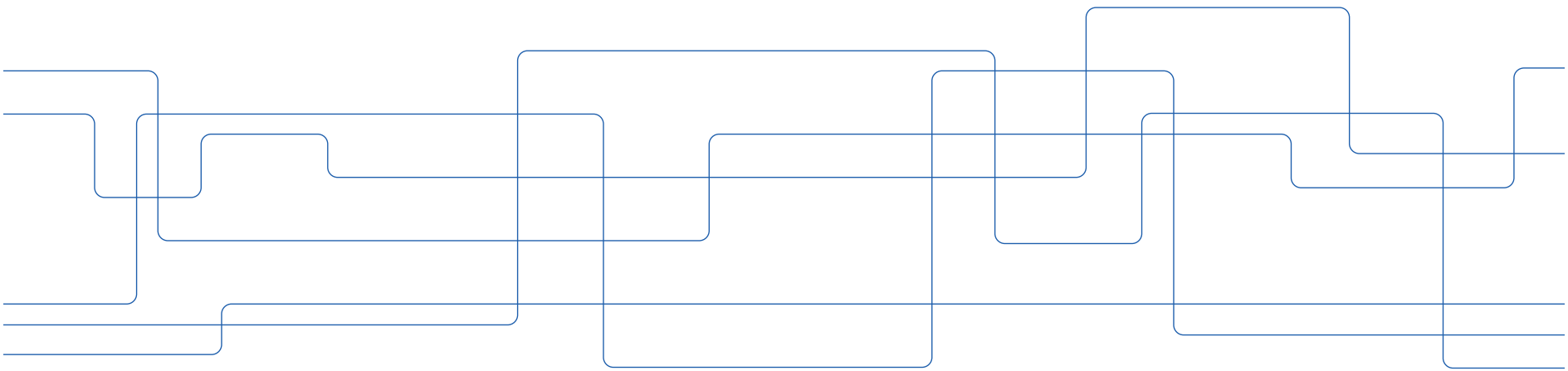
HPC Computer Architectures: Part IV

AQTIVATE Training Workshop I

Dirk Pleiter

CST | EECS | KTH

December 2023



Overview

Parallel Architectures

Network Architecture

Energy Efficiency

Exascale Challenges and Machines

Content

Parallel Architectures

Network Architecture

Energy Efficiency

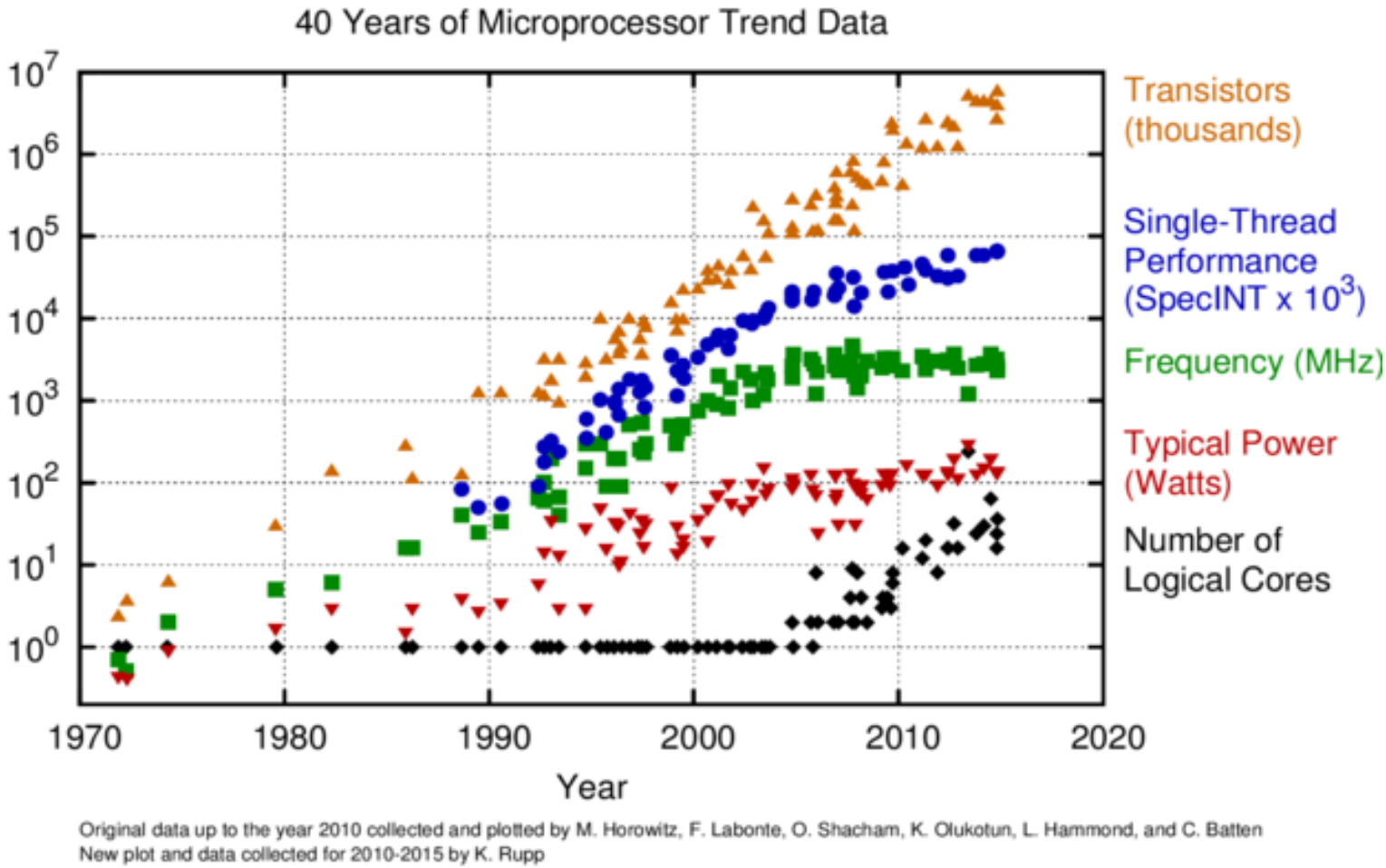
Exascale Challenges and Machines

Need for Massively-Parallel Compute Architectures

- Increasing computing demands
  - Solve large problems faster
- Single processor performance reaches limits
  - Power envelope limits clock frequency
    - More instruction pipelines increase instruction throughput
  - Limited performance gain from instruction level parallism
    - Moderate increase of number of execution pipelines
    - Improved resource utilization through Simultaneous Multi-Threading (SMT)
- Solution: Increase parallelism at multiple levels
  - Examples: more cores, more nodes, ...

Uni-Processor Performance Limits

[Karl Rupp, 2015]



Performance Metric

- **Parallel speedup** using P processes

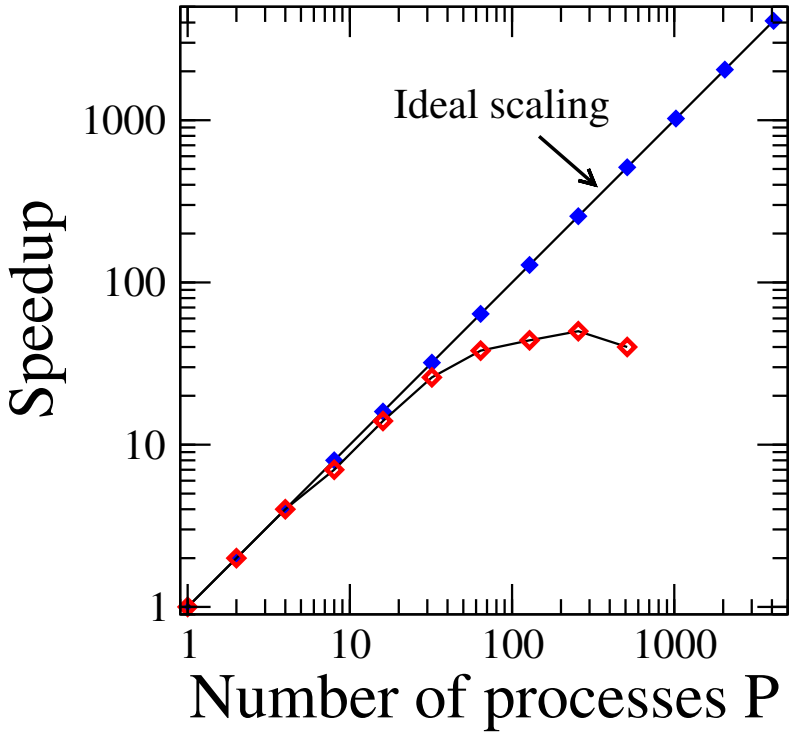
$$S(P) = \frac{\Delta t_{\text{sequential}}}{\Delta t_{\text{parallel}}(P)}$$

- **Parallel efficiency**

$$\epsilon = \frac{S(P)}{P}$$

Example: $S = 26$ using $P = 32$ processing elements
$\Rightarrow \epsilon = 81\%$

- **Super-linear speedup**: $S > P$



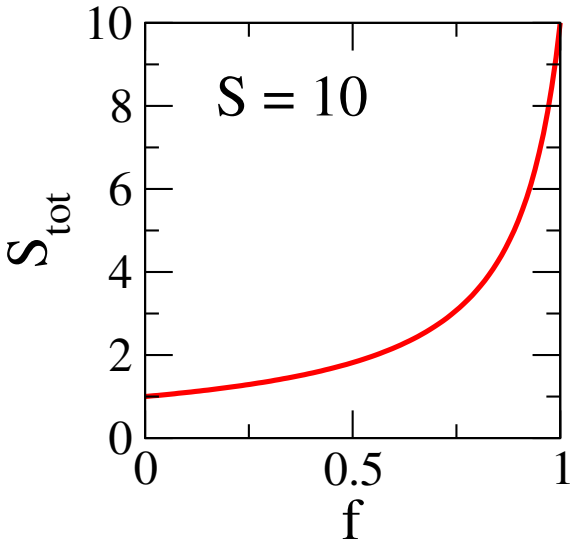
Amdahl's Law

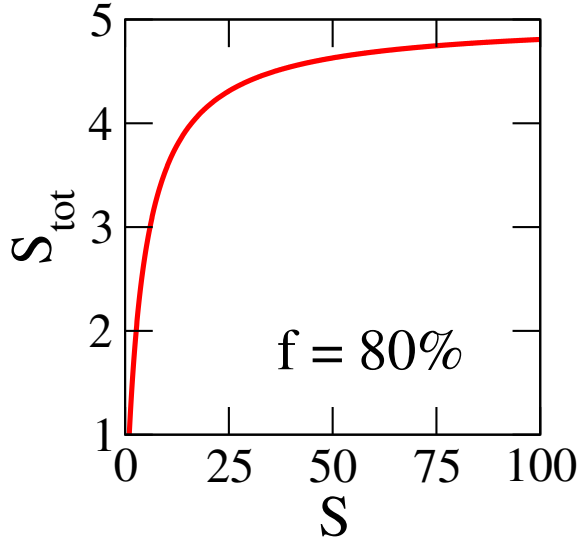
- Consider the following case:
  - Fraction f of application can be parallelized resulting in speed-up by factor S
  - The remaining fraction $1 - f$ remains serial
- Execution time after acceleration:

$$\Delta t' = (1 - f)\Delta t + \frac{f\,\Delta t}{S}$$

- Total speed-up

$$S_{\text{tot}} = \frac{\Delta t}{\Delta t'} = \frac{1}{(1 - f) + f/S}$$





Gustafson's Law

- Amdahl's law: Only considers scaling with fixed workload
  ☞ **strong scaling**
- Different ansatz by Gustafson:
  - Increase workload proportional to number of processing elements P ☞ **weak scaling**
  - Assume non-parallelized code part to be independent of workload
- Execution times
  - Serial execution
    - Execution time for original problem: Δt
    - Execution time for scaled problem size: $(1-f)\Delta t + f\,P\,\Delta t$
  - Parallel execution
    - Execution time for scaled problem size: $(1-f)\Delta t + f\,\Delta t$
- Speed-up:

$$S = \frac{(1-f)\Delta t + f\,P\,\Delta t}{(1-f)\Delta t + f\,\Delta t} = (1-f) + f\,P$$

Parallel Efficiency: Amdahl vs. Gustafson

- Parallel efficiency according to Amdahl:

$$\epsilon_{\text{Amdahl}}(P) = \frac{1}{P} \cdot \frac{1}{(1-f) + f/P} = \frac{1}{P\,(1-f) + f}$$

- Parallel efficiency according to Gustafson:

$$\epsilon_{\text{Gustafson}}(P) = \frac{(1-f) + f\,P}{P} = \frac{1-f}{P} + f$$

- Limit of large number of processing elements P:

$$\begin{aligned} \lim_{P\to\infty} \epsilon_{\text{Amdahl}}(P) &= 0 \\ \lim_{P\to\infty} \epsilon_{\text{Gustafson}}(P) &= f \end{aligned}$$

Content

Parallel Architectures

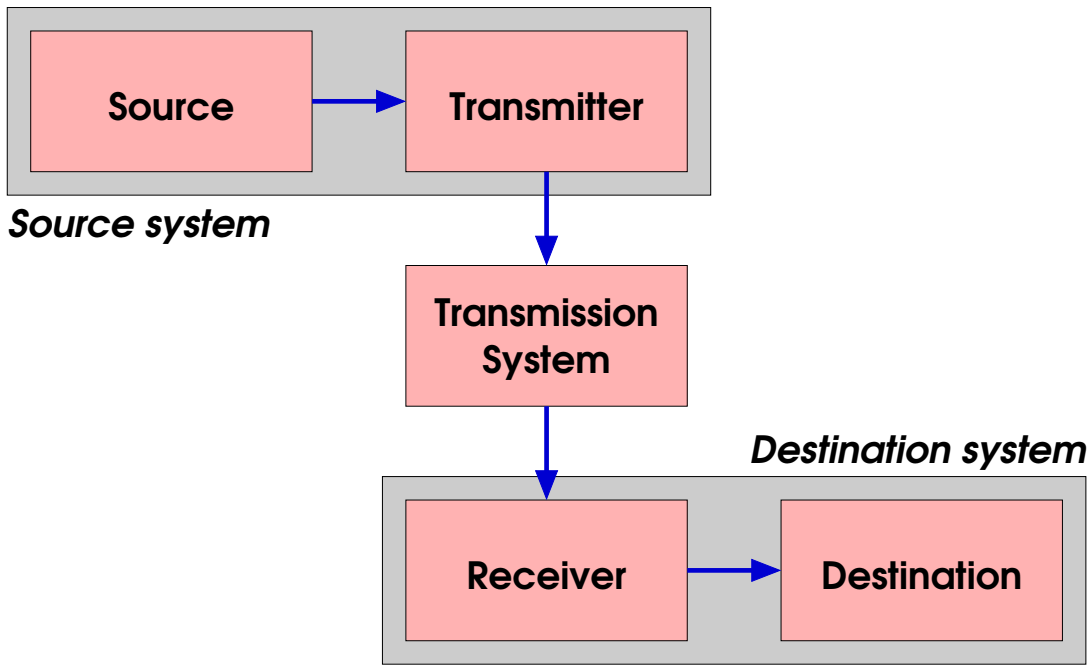
Network Architecture

Energy Efficiency

Exascale Challenges and Machines

Computer Network

- **Computer network** = interconnection between computing nodes
- Simplistic view:

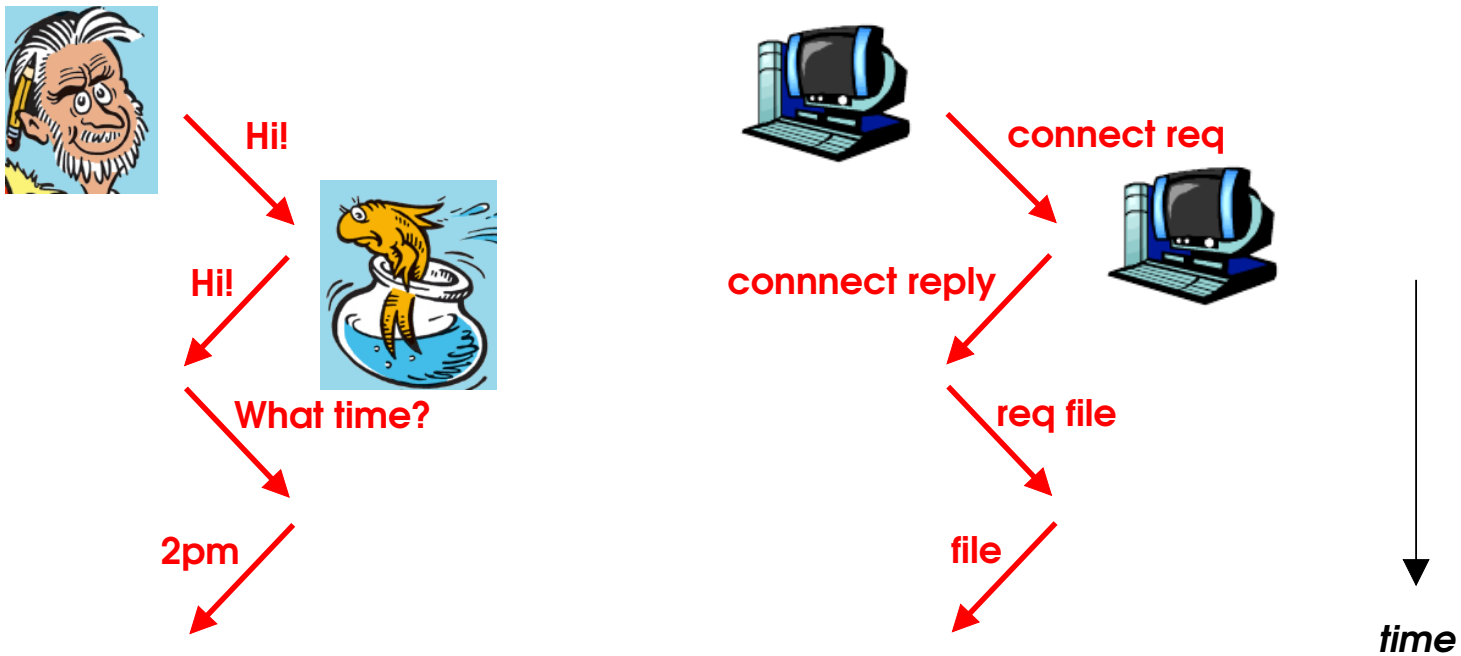


Protocol Architecture

- **Protocol** = Agreement about communication between two or more entities
- Key elements
  - Syntax
    - Data formats
    - Signal levels
  - Semantics
    - Control information
    - Error handling
  - Timing
    - Speed matching
    - Sequencing

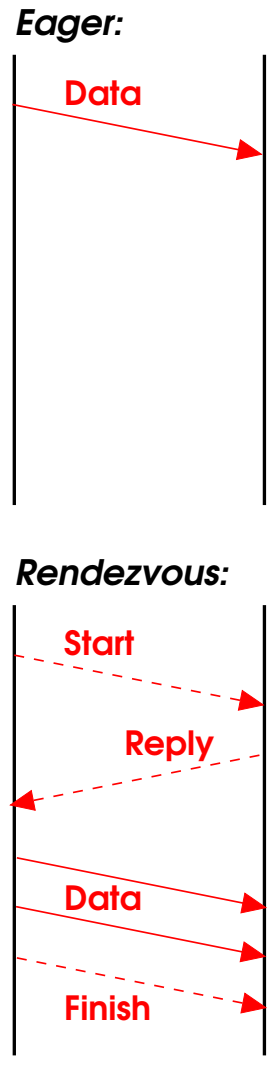
Protocol Architecture

Human protocol vs. computer network protocol



Message Communication Protocols

- **Eager protocol**:
  - Sender pushes message regardless of receiver state
  - Low overhead
  - Useful for small messages
- **Rendezvous protocol**:
  - Handshake happens between the sender and the receiver before sending data
  - High overhead due handshake involving control messages
  - Useful for large messages

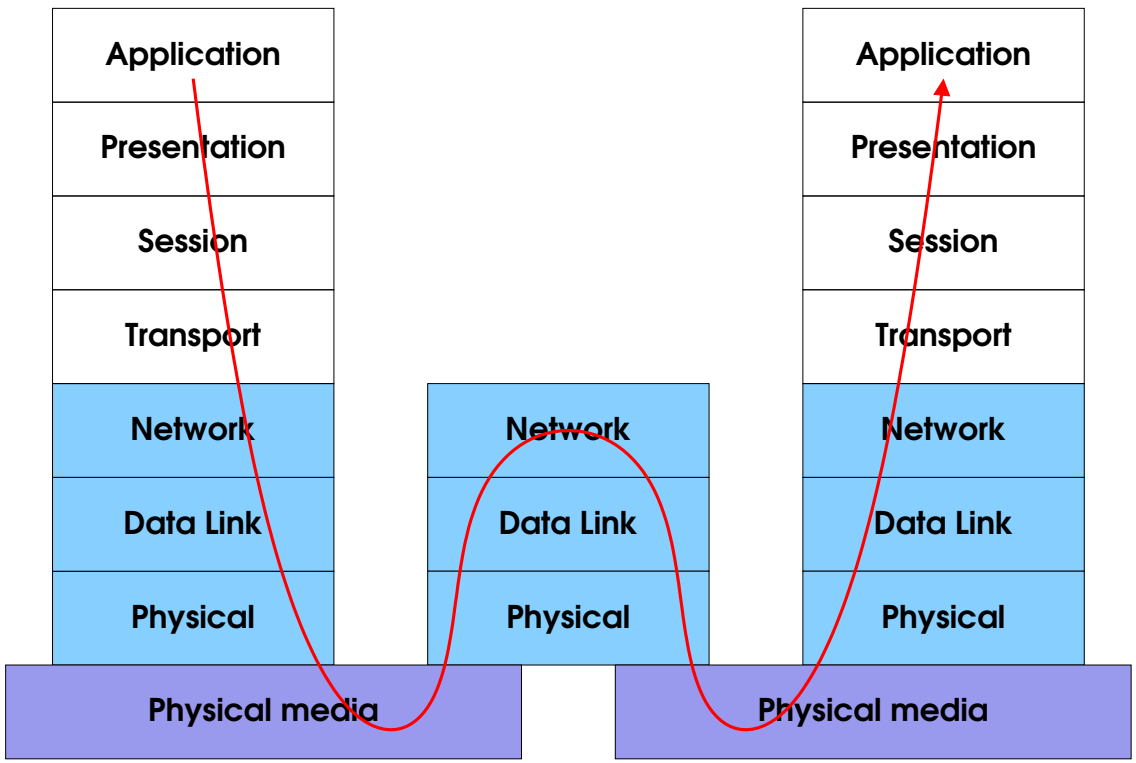


Layering of Networks

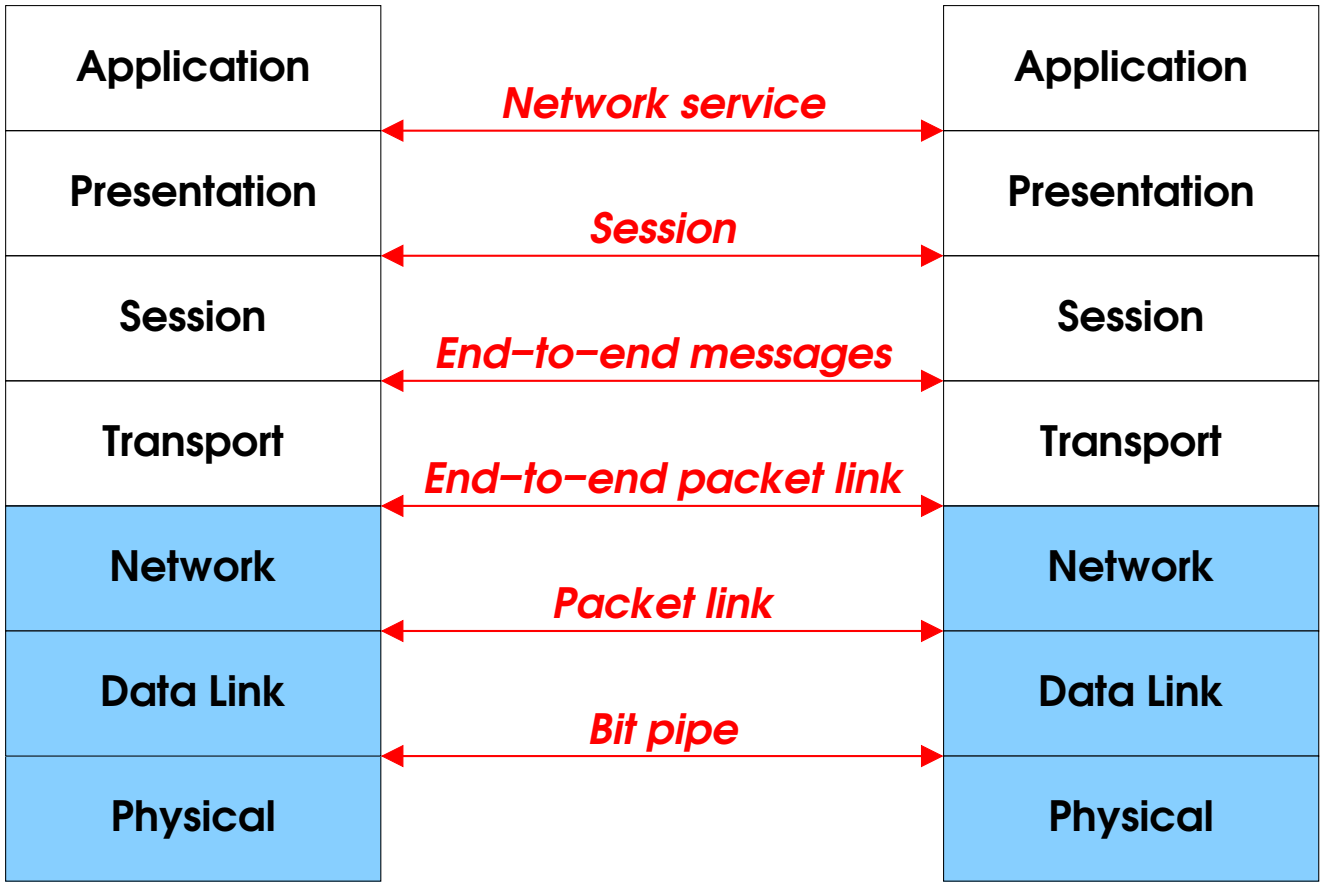
- Most networks are organised as a series of layers
  - Physical communication takes place at lowest layer
  - At higher layers: virtual communication
  - Lower levels are implemented in hardware, while top levels are implemented in software
- API between layers are tightly defined
- Advantages of layering:
  - Design becomes less complex as problem is broken down
  - Changes typically affect only one layer
    - Example: new physical layer, new protocols
  - For different layers solutions from different providers can be used

OSI Network Layer Model

- **OSI** = Open Systems Interconnection
  - Standard defined by the International Organization for Standardization (ISO)
  - 2nd edition published in 1994



OSI Model: Virtual View



Network Topology

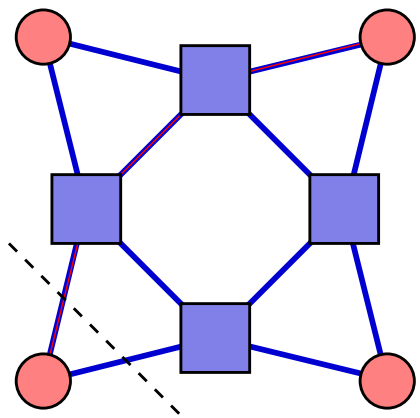
- **Network topology** refers to the way in which the network of computing nodes is connected
- Physical topology = topology of the physical network
- Logical topology = logical view on the way network is connected
  - Not explored in this lecture
- Simplest topology: point-to-point topology
  - Permanent link between two **endpoints**

Network Evaluation Metric

- Diameter
  - Maximum minimal distance between 2 nodes
  - Smaller is better
- Connectivity
  - The minimum number of arcs that must be removed to break it into two disconnected networks
  - Larger is better
- Bi-section width
  - The minimum number of arcs that must be removed to partition the network into two equal halves
  - Larger is better
- Bi-section bandwidth
  - Aggregate bandwidth of all links connecting two equal halves
  - Larger is better
- Costs
  - Smaller is better

Network Evaluation Metric: Diameter and Connectivity

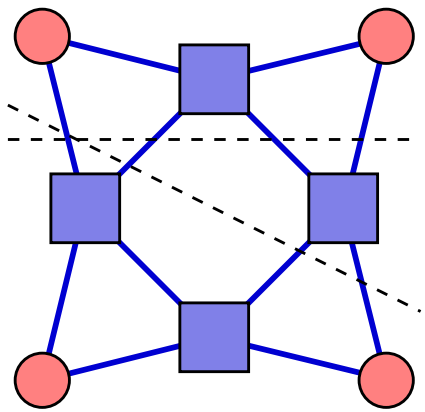
- Diameter is the maximum minimal distance between 2 nodes
- Connectivity gives the minimum number of edges to be removed for breaking network in disconnected pieces
- Example:



  - Diameter $= 3$
  - Connectivity $= 2$

Network Evaluation Metric: Bi-section (Band-)Width

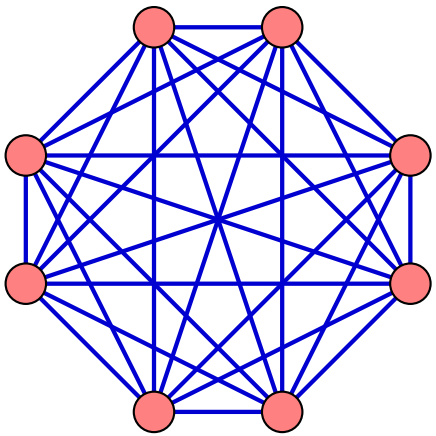
- Procedure to determine bi-section width and bandwidth:
  1. Determine all possible ways to create 2 equi-partitions
  2. Select the partitioning with minimal number of edges removed
  3. Compute aggregate bandwidth of removed links
- Example:



  - Bi-section width $= 4$

All-connected and Star Topology

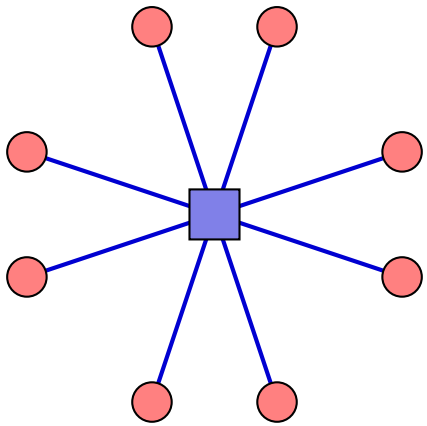
All-connected network



Evaluation:

- Diameter: 1
- Connectivity: $P - 1$
- Bi-section width: $(P/2)^2$
- Costs: $O(P^2)$

Star topology



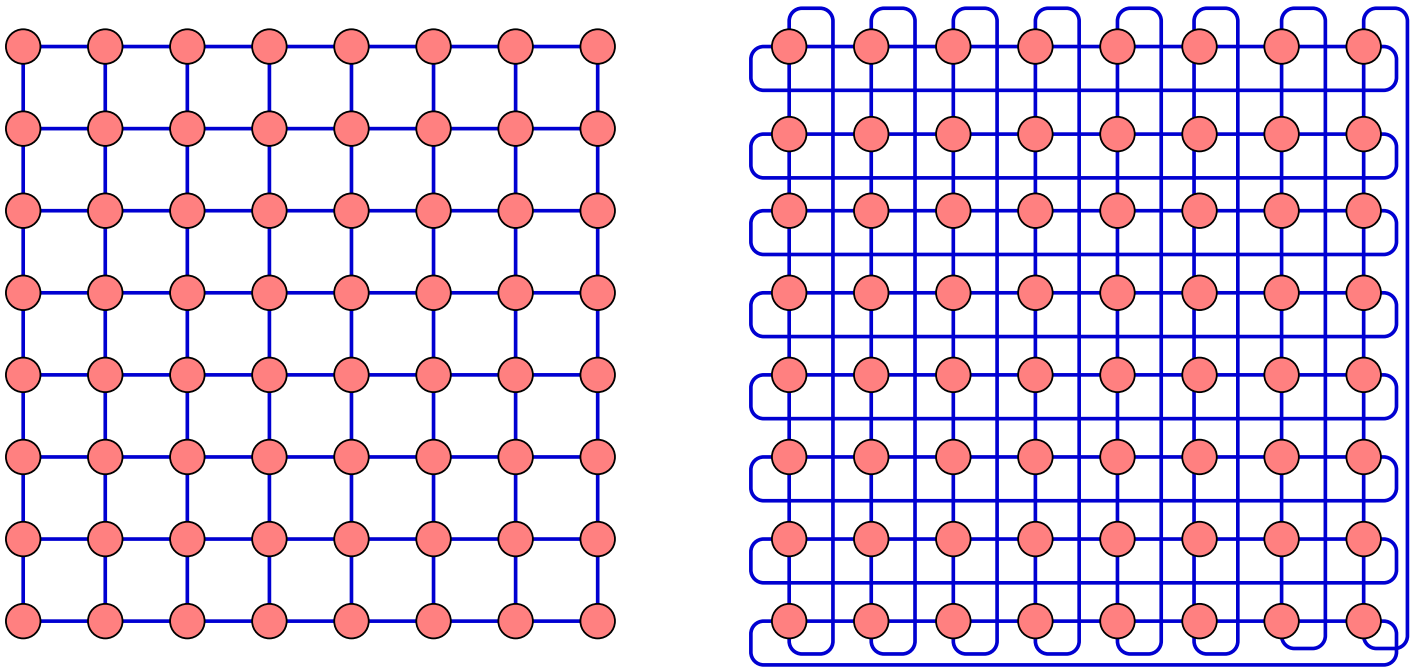
Evaluation:

- Diameter: 2
- Connectivity: 1
- Bi-section width: —
- Costs: $O(P)$

Cartesian Network Topologies

- 1-dimensional linear array and ring:

- 2-dimensional mesh and torus:



Cartesian Network Topologies (2)

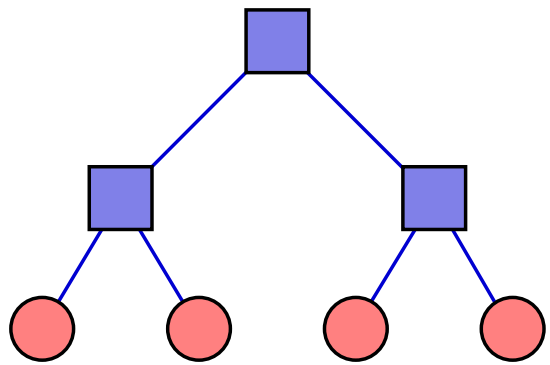
- Evaluation:

	Mesh	Torus
Diameter	$d(P^{1/d} - 1)$	$d(P^{1/d}/2)$
Connectivity	d	$2d$
Bi-section width	$P^{(d-1)/d}$	$2P^{(d-1)/d}$

Network Topology: Tree

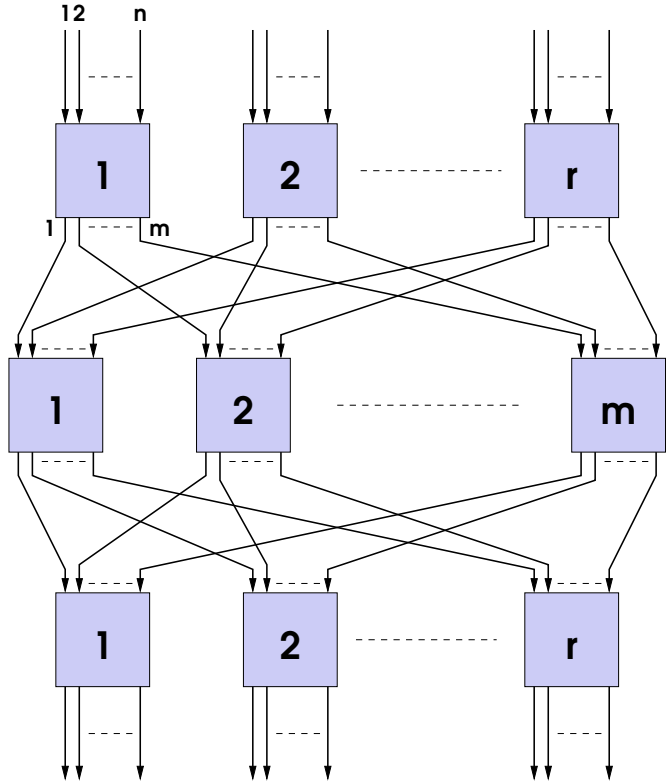
- Switch-based network
- Evaluation binary tree:
  - Diameter: $O(\log(P))$
  - Connectivity: 1
  - Bi-section width: 1
  - Costs: $O(P)$
- Number of links between upper levels smaller

  ☞ **Blocking network topology**



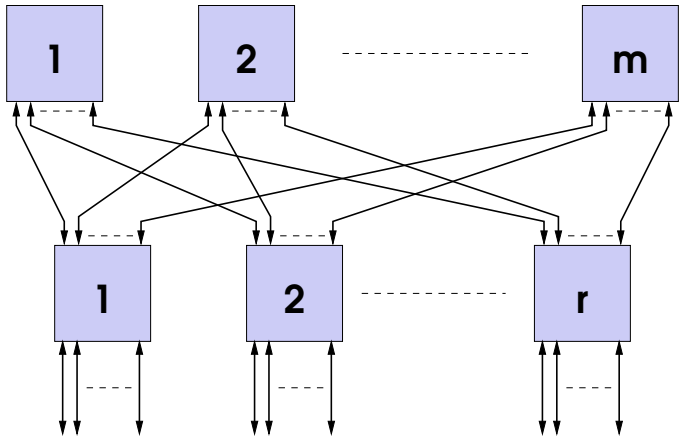
Network Topology: Clos Network

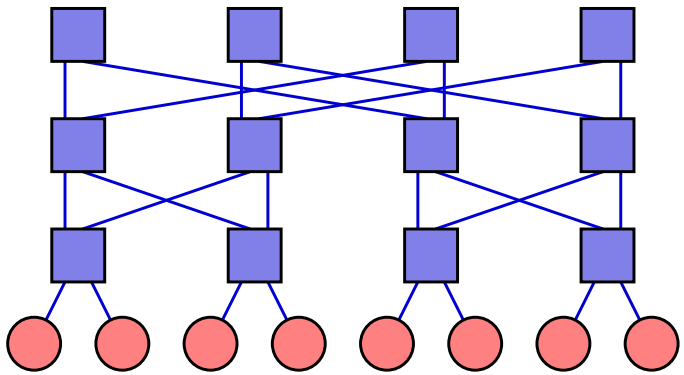
- N-stage switch-based network
  - Here: $N = 3$
- Network is characterised by
  - r: number of ingress stage switches
  - n: number of ingress ports at ingress stage
  - m: number of egress ports at ingress stage
- Evaluation:
  - Diameter: $N + 1$
  - Connectivity: m (at switch-level)
  - Bi-section width: $r\frac{m}{2}$
- A Clos network can be made **non-blocking**
  - Non-blocking for $m \geq n$ if one allows for rearrangements
  - Strictly non-blocking if $m \geq 2n - 1$



Network Topology: Folded Clos Network

- Topology obtained by folding ingress and egress stage
  - Example: folded 3-stage Clos
- Evaluation of Clos and Folded Clos network:
  - Diameter $= N + 1$
  - Connectivity $= m$
  - Bi-section width $= r\frac{m}{2}$
- Closely related to **fat-tree** topology
  - Here: similar to folded 5-stage Clos

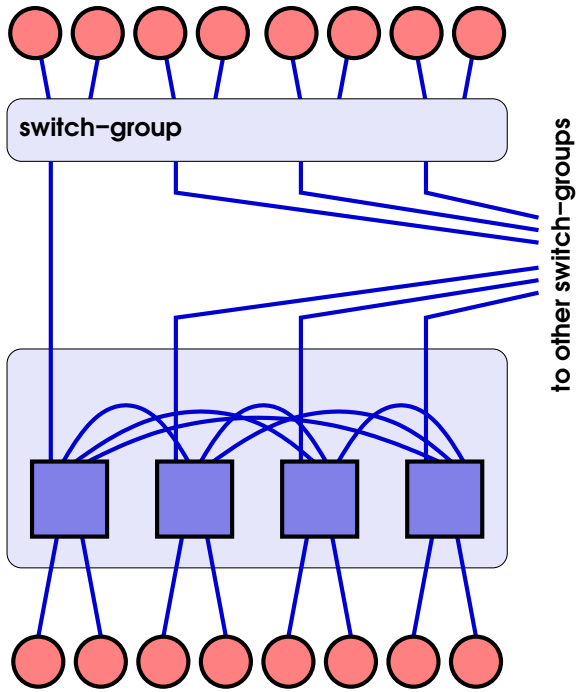




Network Topology: Dragonfly

- Multi-layer network
  - Switch layer
  - Switch-group layer
  - System layer
- All-to-all connectivity within layer
- Network parameters
  - p: number of nodes per switch
  - a: number of switches per group
  - h: number of links per switch to other groups
- Evaluation:
  - Diameter: 5
  - Connectivity: $ah - b$

    (at system layer; b = bundel width)
  - Bi-section width: $\left(\frac{a\,h-1}{2}\right)^2$



Content

Parallel Architectures

Network Architecture

Energy Efficiency

Exascale Challenges and Machines

Strategies for Reducing Energy-to-Solution

Selected strategies for improving energy-efficiency:

- **Use of power-efficient components**
  - Focus on compute accelerators
- **Improve data centre efficiency**
  - Avoid consuming energy except for executing instructions and data transport
  - Improve efficiency of cooling and power conversion
- **Use of energy-efficient algorithms**

Costs of Data Transport

[W. Keckler et al., 2011]

Comparison of

- NVIDIA Fermi GPU as of 2010
- Projection NVIDIA GPU as of 2017

	2010	2017
Process technology	40 nm	10 nm
Voltage (V_{DD})	0.9 V	0.65 V
Frequency	1.6 GHz	2 GHz
DP-FMA energy	50 pJ	6.5 pJ
Wire energy (256 bit, 10 mm)	310 pJ	150 pJ

☞ **Data movement costs will in future dominate even more**

Energy Efficiency

- **Energy efficient** = The amount of work W which can be performed for a given amount of energy E is maximized
- **Energy efficiency** ϵ_E:

$$\epsilon_E = \frac{W}{E}$$

- Important to reduce operational costs
- Consumed energy

$$E = \int_t^{t+\Delta t} P(\tau)\, d\tau \leq \max(P)\Delta t$$

Δt	...	**time-to-solution**
E	...	**energy-to-solution**
$P(\tau)$	...	power consumed at time τ

Power Efficiency

- **Power efficient** = Best performance dW/dt for given power consumption P
- **Power efficiency** ϵ_P:

$$\epsilon_P = \frac{dW/dt}{P}$$

- Example: Maximize floating-point performance over power consumption (Flop/s/Watt)
- Important if available power is limited
  - Examples: limitations in power distribution, costs for cooling

Energy vs. Power Efficiency

- Assume constant power consumption: $E = P\Delta t$
- Assume constant throughput $b = dW/dt = W/\Delta t$:
- Relation between energy and power efficiency under these assumptions:

$$\epsilon_E = \frac{W}{E} = \frac{b\,\Delta t}{P\,\Delta t} = \frac{dW/dt}{P} = \epsilon_P$$

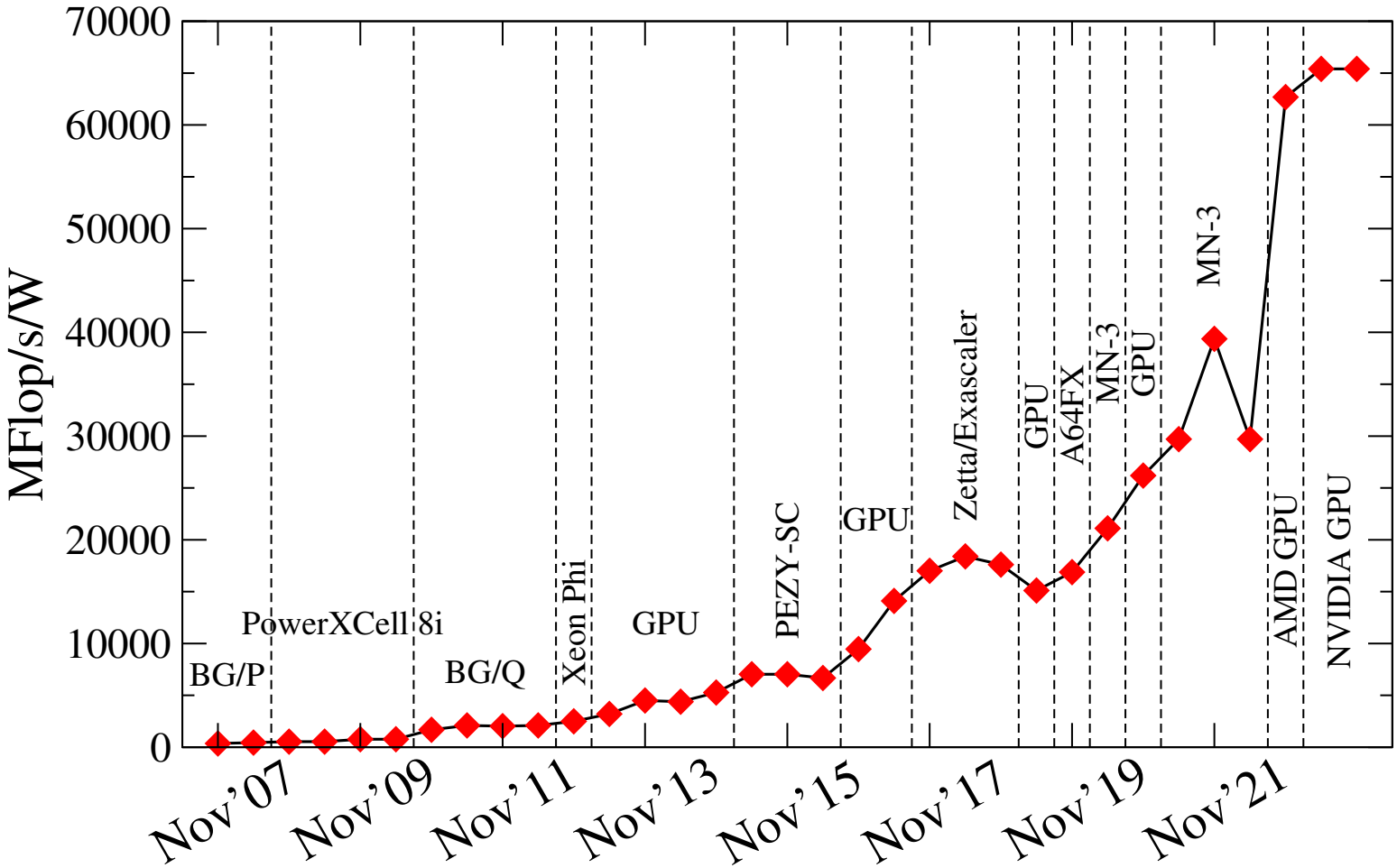
  - ☞ Special case of similar energy and power efficiency
  - ☞ Maximizing $\epsilon_P \neq$ minimizing E
    - E.g., W may change for different algorithms
- Popular power efficiency metric: floating-point throughput vs. power consumption

$$\epsilon_P = \frac{b_{\mathrm{fp}}}{P}$$

Power Efficiency: The Green500 List

- Green500: Rank supercomputers according to power efficiency
  - Metric = floating-point performance vs. power consumption
  - Supercomputer = system listed in TOP500
  - Performance = HPL performance (like for TOP500)
- Current number #1 (Nov'23): 65.4 GFlop/s/W (or 15.3 pJ/Flop)
- Exascale goal: keep below 20 MW (or 20 pJ/Flop)
- Exascale system Frontier: 52.6 GFlop/s/W (or 19pJ/Flop)
- Criticism
  - The High-Performance LINPACK (HPL) benchmark load is not representative
  - Green500 does not cover full system

The Green500 List (cont.)



Content

Parallel Architectures

Network Architecture

Energy Efficiency

Exascale Challenges and Machines

What is Exascale Computing?

- Answer 1: Running HPL at 1 EFlop/s
  - HPL = High-Performance LINPACK
  - HPL is the load used for Top500
- Answer 2: Future generation supercomputers enabling new science
  - Exascale application challenges:
    - Weather, climatology and solid earth sciences
    - Astrophysics, high-energy physics and plasma physics
    - Materials science, chemistry and nano-science
    - Engineering sciences and industrial applications
    - Life sciences and medicine
- Answer 3: Exascale computer = $100 - 1000\times$ of today's petascale supercomputers
  - Flop/s metric will become less relevant

Canon of Exascale Challenges

- Reducing power requirements
  - Need energy-to-solution reduction by about 100×
- Exploiting massive parallelism
  - More computational performance → more parallelism
  - Scalability challenge
- Maintaining a balanced system
  - Floating-point operations are cheap, but data transport is expensive
- Coping with run-time errors
  - More components → higher risk of system failures
  - Number of components in future systems is likely to increase

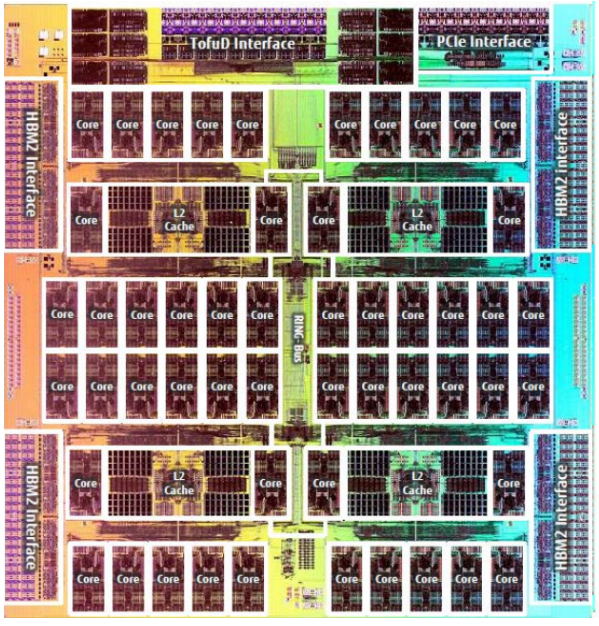
Top500 #2: Fugaku

- Machine installed at RIKEN research lab (Japan)
- System parameters

Number of cores	7,630,848
Peak FP64 performance	537 PFlop/s
Memory capacity	4.8 PiByte
Power consumption	30 MW (Linpack)

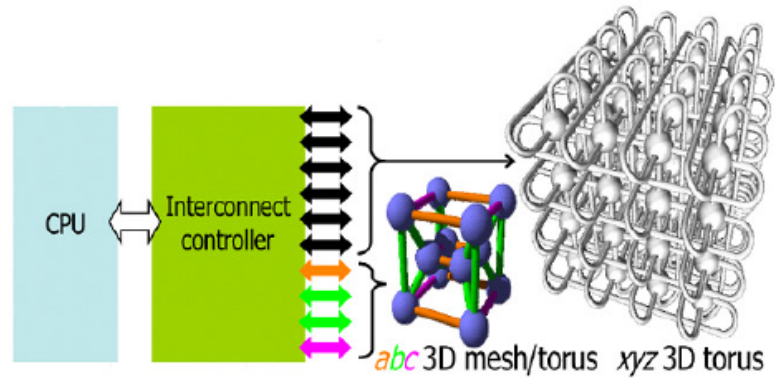
Fugaku: A64FX Processor

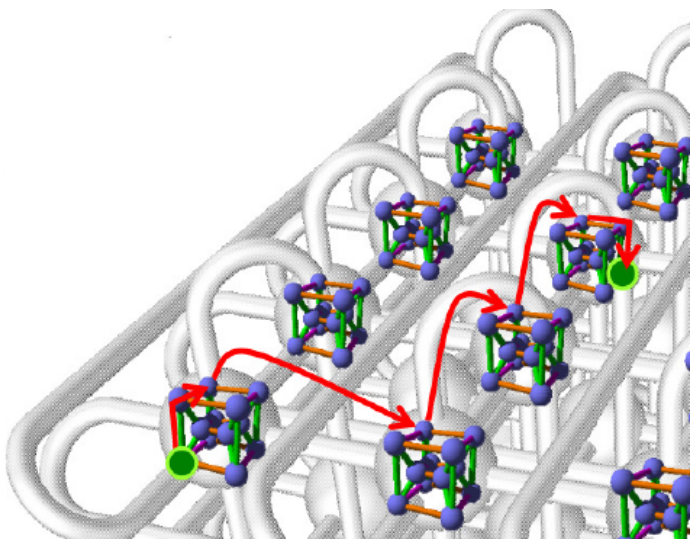
- 48+4 cores
- 12 × 8 MByte shared L2 cache
- 3072 GFlop/s (DP) peak performance at 2 GHz
  - 2× 512-bit SIMD units per core (SVE ISA)
- Nominal memory bandwidth: 1 TByte/s
  - 4× HBM2 stacks



Fugaku: Network

- Tofu D (Tofu = Torus Fusion)
  - Units of 12 nodes interconnected within 3-dimensional torus (6 links)
  - Nodes within unit interconnected as 3-dimensional torus/mesh (4 links)
- High connectivity ☞ multiple pathes connecting each pair of nodes
- Performance
  - Link bandwidth: 56 GBit/s





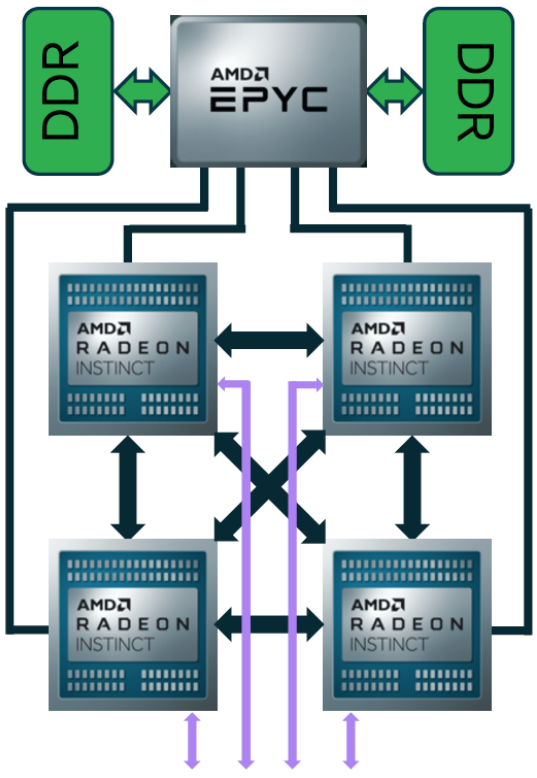
Top500 #1: Frontier

- Machine installed at Oak Ridge National Lab (ORNL, US)
- Size of system as for Top500 in June 2022:

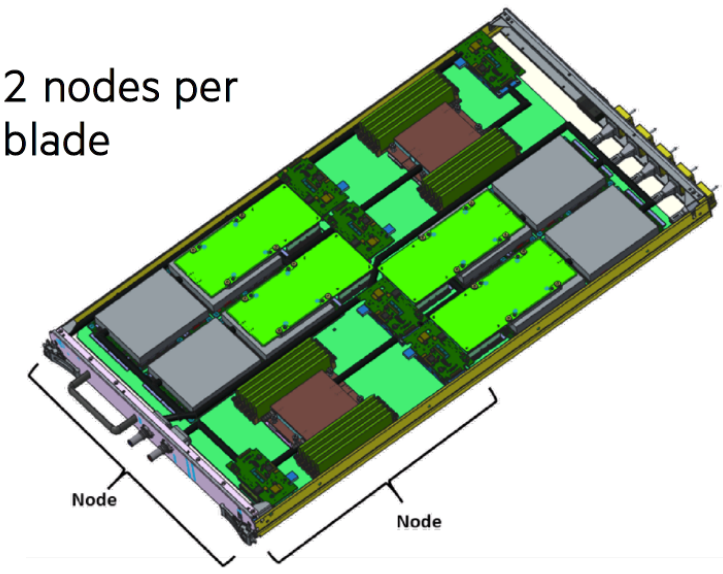
Number of nodes	9,248
Number of CPU cores	591,872
Number of GPU CUs	8,138,240
Peak FP64 performance	1,686 PFlop/s
Memory capacity	9.0 PiByte
Power consumption	21 MW (Linpack)

Frontier: Node Architecture

AMD GPU (ORNL)



2 nodes per blade

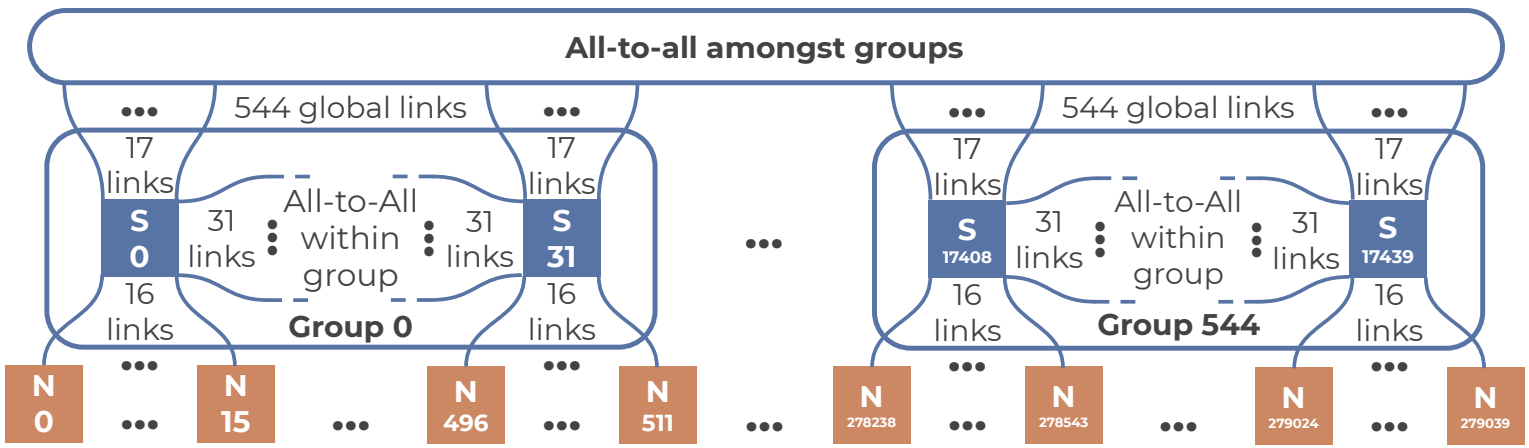


COPYRIGHT 2020 HPE

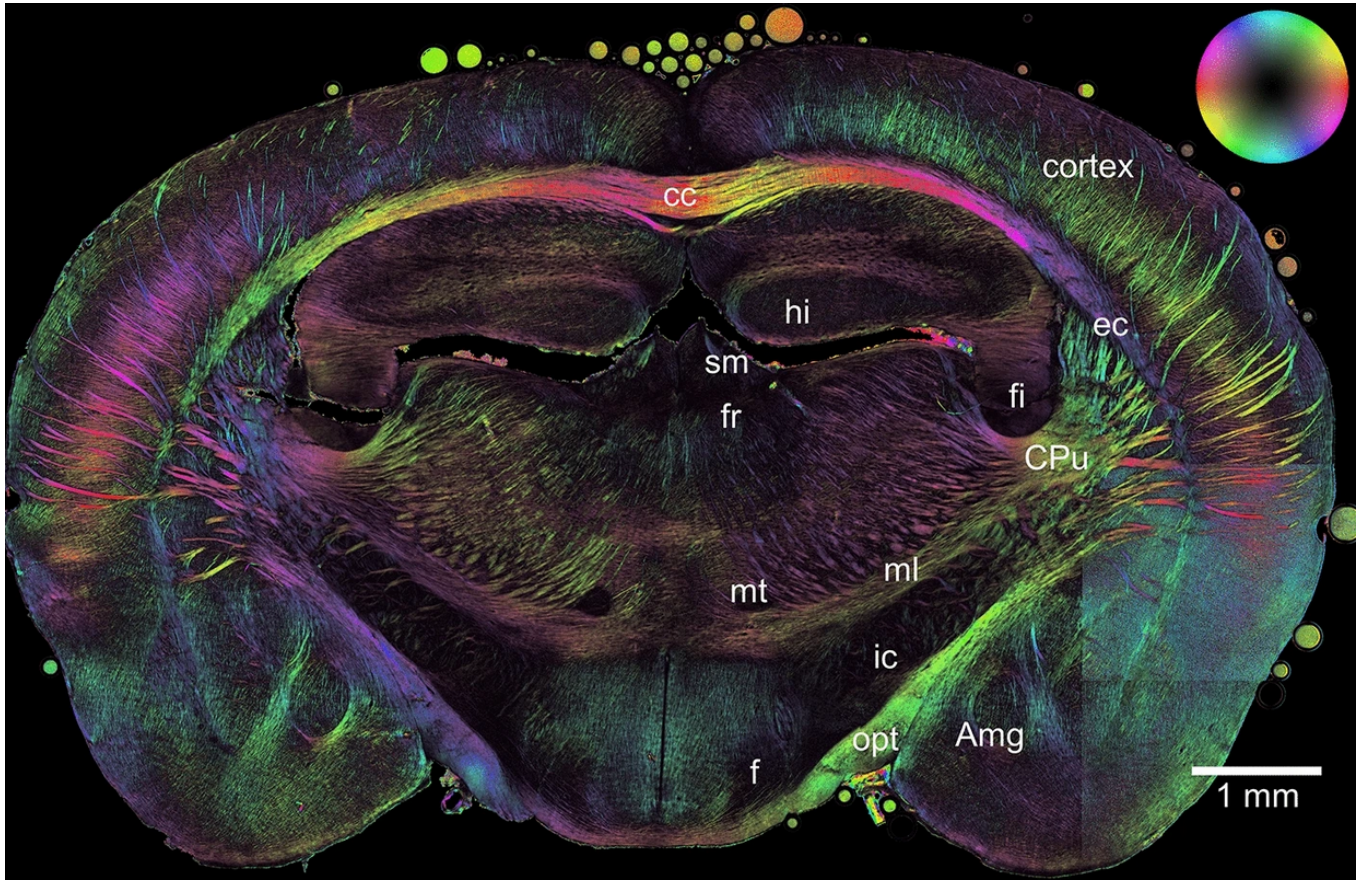
Frontier: Network Topology

- Dragonfly topology based on Slingshot technology
- Rosetta switches with 64 ports
- Maximum number of end-points: 279,040
  - Frontier compute nodes: 37,632 ports

[De Sensi et al., 2020]



Finish: Network of a Mouse Brain



[Maiti et al., 2021]