Parallel and distributed computing

N. Kälin

March 6, 2021

Contents

1 Architectures

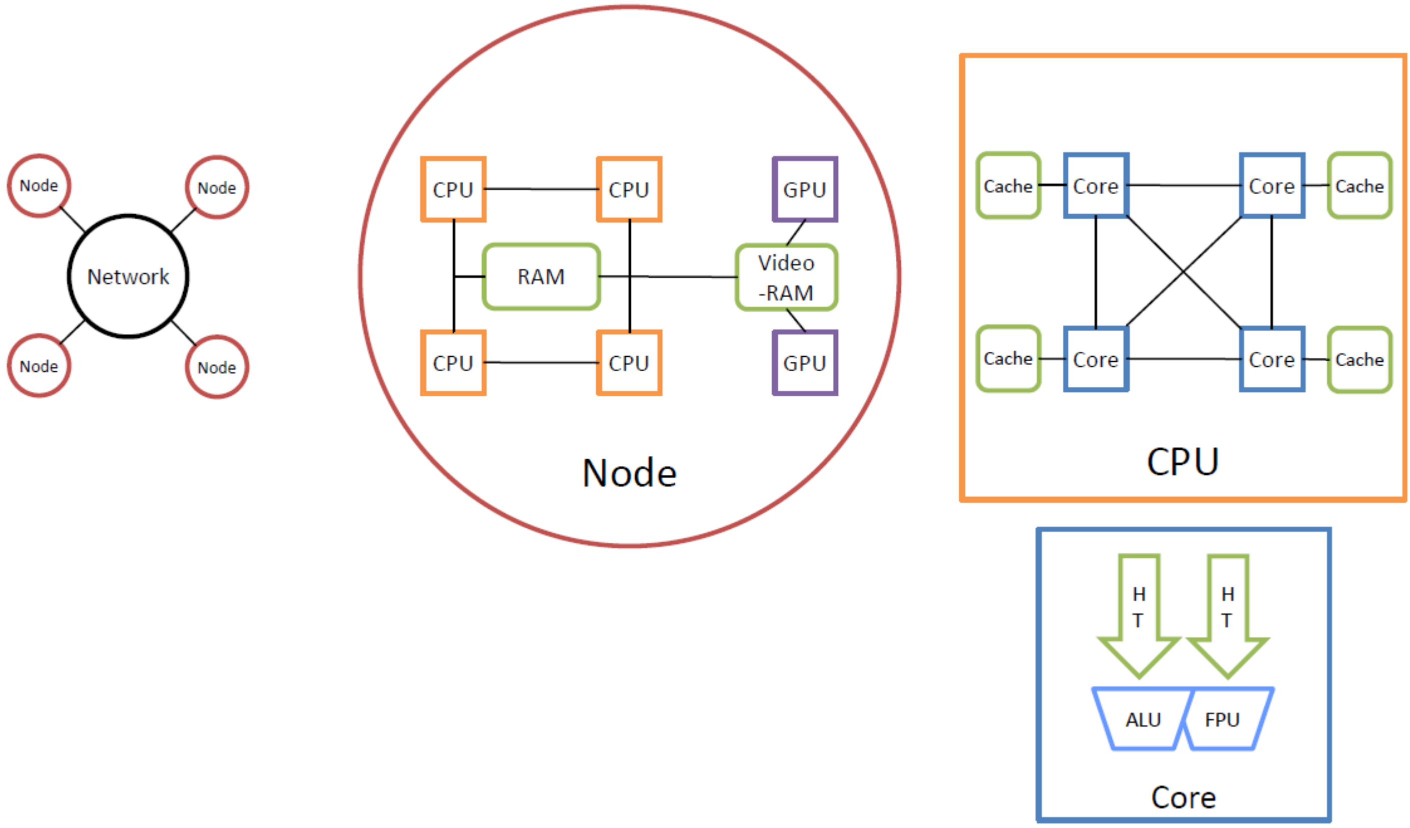


Figure 1: Parallel machine model (cluster)

1.1 Implicit vs. explicit parallelism

- Implicit Parallelism
  - processors have multiple functional units and execute multiple instructions in the same cycle
  - the precise way these instructions are selected and executed provides impressive diversity in Architectures
    - **pipelining**
    - **superscalar execution**
    - **very long instruction word processors**
- Explicit Parallelism
  - an explicitly parallel program must specify concurrency (**control structure**) and interaction (**communication model**) between concurrent subtasks

1.2 Parallel programming models

1.2.1 Overview of Programming models

- Programming models
  - provide support for expressing concurrency and synchronization
- Process based models
  - assume that all data associated with a process is private, by default, unless otherwise specified
- Lightweight processes and Threads

- assume that all memory is global (bounded by process boundaries)
- memory protection between threads of the same process is not necessary
- support much faster memory access than processes with explicitly allocated shared memory

- Parallel programming language with syntax to specify parallelism
  - Examples: Ada, SR, Occam (no longer common)
- Directive based programming models
  - extend the threaded model by facilitating creation and synchronization of threads
  - Examples: Open MP, Linda, POP-C++

1.2.2 Parallel Machine Model

- PRAM
  - a natural extension of the Random Access Machine (RAM) serial architecture
  - consists of p processors and a global memory of unbounded size that is uniformly accessible to all processors
  - procesors share a common clock but may execute different instructions in each cycle
- Handling of simultaneous memory accesses
  - Exclusive-read, exclusive-write (EREW)
  - Concurrecnt-read, exclusive-write (CREW)
  - Exclusive-read, concurrent-write (ERCW)
  - Concurrent-read, concurrent-write (CRCW)

What does concurrent write mean?

Common: write only if all values are identical.

Arbitrary: write the data from a randomly selected processor.

Priority: follow a predetermined priority order.

Sum: write the sum of all data items.

1.3 Different grains of parallelism

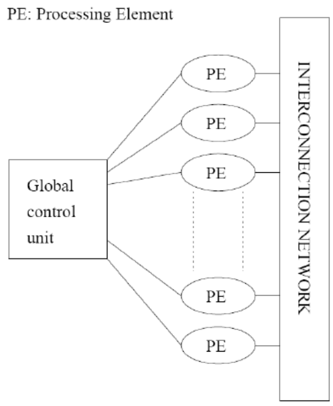
- Granularity: the ratio of computation to communication
  - periods of computation are separated from periods of communication by synchronization events
  - constrained by the inherent characteristics of the used algorithms
  - the parallel programmer must select the right granularity to benefit from the underlying platform
- Chunking
  - determining the amount of data to assign to each task (chunk or grain size)
- Which Granularity will lead to best performance?
  - depends on the algorithm and the used hardware environment
  - general rule: increase grain size if the communication overhead is too large

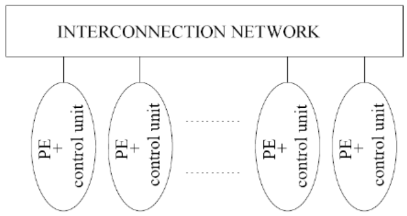
1.3.1 Trade-offs associated with chunk size

- Fine-grained parallelism
  - low arithmetic intensity
  - may not have enough work to hide long-duration asynchronous communication
  - facilitates load balancing by providing a larger number of more manageable (i.e. smaller) work units
  - too fine granularity can produce slower parallel implementation than the serial execution (too much overhead required for communication)
- Coarse-grained parallelism
  - high arithmetic intensity
  - complete applications can serve as the grain of parallelism
  - more difficult to load balance efficiently

1.4 Control structure of parallel platforms

- SIMD: Single Instruction stream, Multiple Data stream
  - there is a single control unit that dispatches the same instruction to various processors (that work on different data)
  - data parallelization
- MIMD: Multiple Instruction stream, Multiple Data stream
  - each processor has its own control unit
  - each processor can execute different instructions on different data items





1.4.1 SIMD Computers

- Hardware requirements
  - SIMD computers require less HW than MIMD computers (only one control unit)
  - SIMD computers require less memory (only one copy of the program is stored)
- Current implementations
  - Graphics Processing Units (GPUs)
  - Digital Signal Processors (DSPs) are widely used in cameras and sound equipments
  - Co-processing units in Intel CPUs: SSEx, AVX-512
- Software requirements
  - SIMD relies on the regular structure of computations (such as those in image and video processing or in deep learning)
  - it is often necessary to selectively turn off operations on certain data items

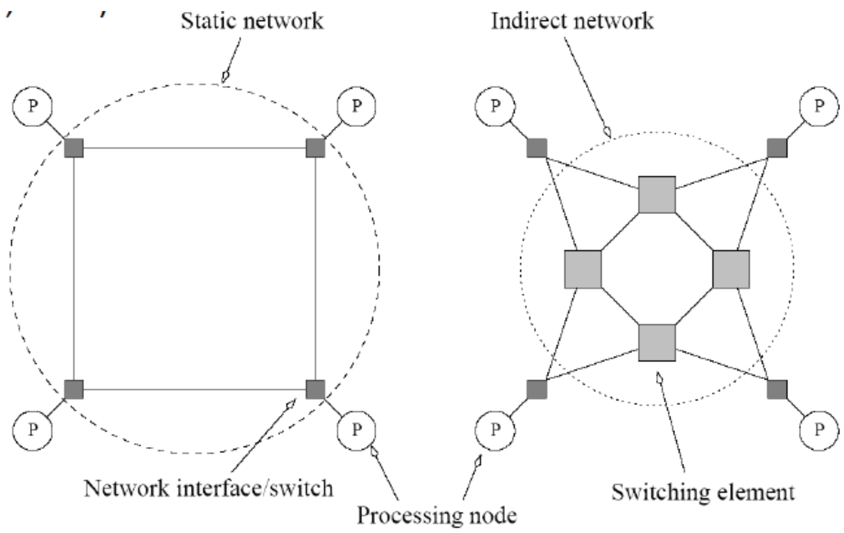
1.4.2 MIMD Computers

- Single Program Multiple Data (SPMD)
  - a simple variant of MIMD executes the same program on different processors
  - SPMD and MIMD are closely related in terms of programming flexibility and underlying architectural support
  - a single program consisting of several programs in a large switch block with conditions specified by the task identifiers is equivalent to the MIMD model
- Current MIMD implementations
  - SPARC servers, multiprocessor PCs, NASA Beowulf inspired workstation clusters
- Key advantages of workstation clusters
  - high performance workstations and PCs available at low cost
  - latest processors can easily be incorporated into the system as they become available
  - existing software can be used or modified

1.5 Communication models of parallel platforms

- Shared-Address-Space Platforms (Multiprocessors)
  - part (or all) of the memory is accessible to all processors
  - processors interact by modifying data objects stored in this shared-address-space
  - uniform or non-uniform memory access time (UMA vs. NUMA)
- Message Passing Platforms (Multicomputers)
  - comprise of a set of processors and their own (exclusive) memory
  - instances come naturally from clustered workstations (distributed systems) and non-shared-address-space multi-computers
  - are programmed using sending messages (variants of send and receive primitives)
  - libraries such as MPI (1990's) provide such primitives

1.6 Interconnection networks



- Interconnection Networks for Parallel Computers
  - carry data between processors and to memory
  - are made of switches and links (wires, fiber)
  - are classified as static or dynamic
    - static (direct) networks consist of point-to-point communication links among processing nodes
    - dynamic (indirect) networks are built using switches and communication links
- Network Topologies
  - a variety of network topologies have been proposed and implemented
  - tradeoff performance for cost
  - two basic categories: physical and logical topologies
  - commercial machines often implement hybrids of multiple topologies

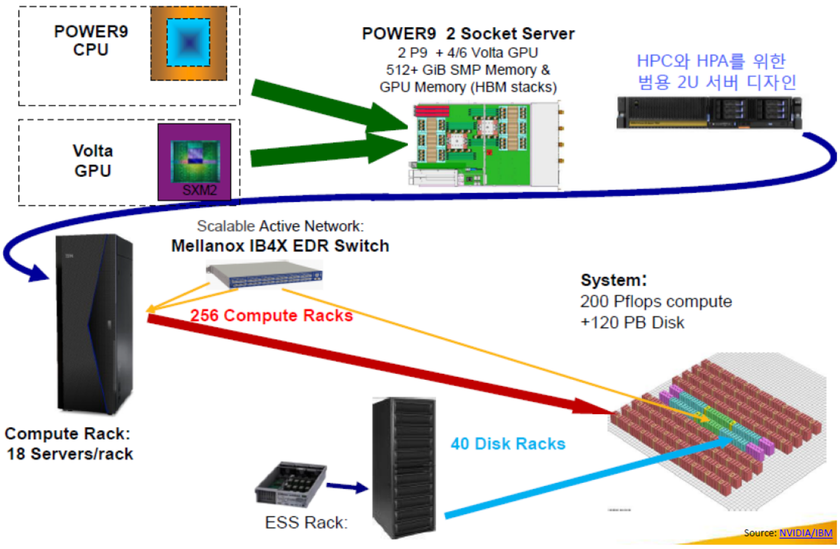
1.6.1 Interconnection Network for HPC

- Infiniband
  - a computer-networking communications standard used in HPC that features very high throughput and very low latency
  - it is used for data interconnect both among and within computers
  - it is also utilized as either a direct, or switched interconnect between servers and storage systems, as well as an interonnect between storage systems
  - it is designed to be scalable and uses a switched fabric network topology

Year	FDR 2011	EDR 2014	HDR 2017	NDR 2020	XDR 2023
Throughput, per 1x [Gbit/s]	13.64	25	50	100	250
Speed for 4x links [Gbit/s]	54.54	100	200	400	1000
Speed for 12x links [Gbit/s]	163.64	300	600	1200	3000
Latency [μs]	0.7	0.5	0.5	tbd	tbd

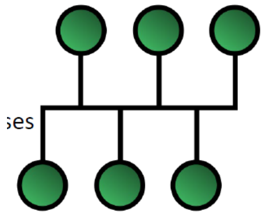
- PCI Express Version 4
  - a high-speed serial computer expansion bus standard
  - has numerous improvements over the older standards
    - higher maximum system bus throughput
    - lower I/O pin count and smaller physical footprint
  - has been drafted with final specifications expected in 2017
  - throughput:
    - x1: 1.969 GByte/s
    - x16: 31.508 GByte/s
  - external cabling: Thunderbolt
- NVIDIA NVLink
  - is a high-bandwidth, energy-efficient interconnect
  - enables ultra-fast communication between the CPU and GPU, and between GPU
  - throughput:
    - version 1 (used in NVIDIA Pascal): x1: 20 GByte/s, x4: 80 GByte/s
    - version 2 (used in IBM Power9 chip, NVIDIA Volta GPUs): x1: 25 GByte/s, x8: 200 GByte/s

1.6.2 Data-Centric IT Environments

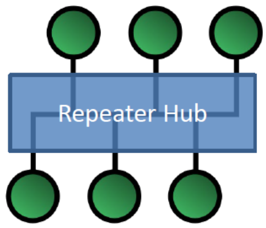


1.7 Network topologies

1.7.1 Network Topologies: Bus



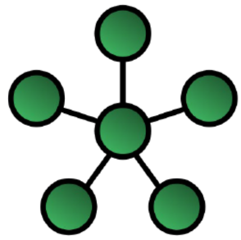
physical bus



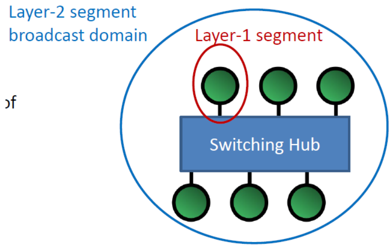
logical bus, physical star topology

- Principle and Properties
  - some of the simplest and earliest parallel machines used buses
  - all processors access a common bus for exchanging data
  - the distance between any two nodes is $O(1)$ in a bus
- Bottleneck
  - the bandwidth of the shared bus is a major bottleneck
  - typical bus-based machines are limited to dozens of nodes
- Examples
  - WLAN zone (logical bus topology)
  - PCI bus (physical bus topology)

1.7.2 Network Topologies: Star



physical star

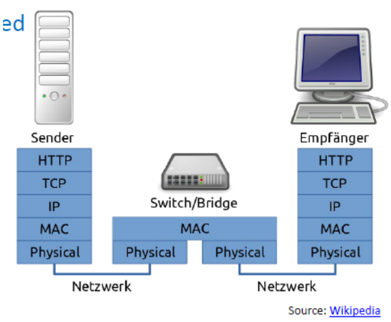


logical and physical star topology

- Principle and Properties
  - every node is connected only to a common node at the center
  - distance between any pair of node is $O(1)$
- Bottleneck
  - the central node
- Example
  - today's Ethernet based LANs with bridging hub (Bridge) or switching hub (Switch) as the center of the start topology

Network Infrastructure: Switchung Hub

- Principle and Properties
  - frame forwarding depends on learned physical device-addresses (MAC) per port (Layer-2 switching)
  - non-blocking: several input-output connections can be used in parallel without blocking
  - store-and-forward
    * the switch buffers and verifies each frame before forwarding it
    * a frame is received in its entirety before it is forwarded
    * error checking can be done before forwarding
  - cut-through
    * the switch starts forwarding after the frame's destination address is received
    * when the outgoing port is busy at the time, the switch falls back to store-and-forward operation
    * there is no error checking with this method

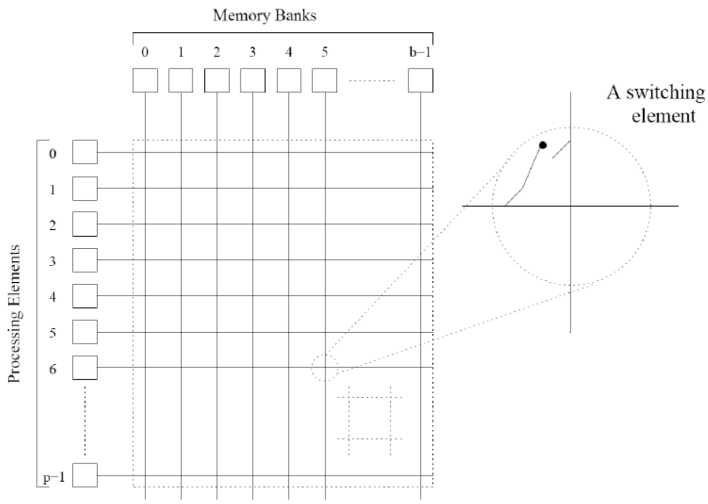


Switching Hub: Advanced Features

- Spanning Tree Protocol → Shortest Path Bridging
  - classic bridges may also interconnect using a spanning tree protocol that disables links so that the resulting local area network is a tree without loops
  - in contrast to routers, spanning tree bridges must have topologies with only one active path between two points
  - IEEE 802.1aq allows all paths to be active with multiple equal cost paths
    - provides much larger layer 2 topologies (up to 16 million compared to the 4096 VLANs limit)
    - improves the use of the **mesh topologies** through increased bandwidth and redundancy between all devices by allowing traffic to load share across all paths of a mesh network
- IEEE 802
  - is a family of IEEE standards dealing with local area networks and metropolitan area networkds
  - services and protocols specified in IEEE 802 map to the lower two layers (Data Link and Physical) of the sevel-layer OSI networking reference model
  - small subset of the working groups
    - 802.1: higher layer LAN protocols (bridging)
    - 802.1D: Spanning Tree Protocol (forwarding stopped while the spanning tree re-converged)
    - 802.1s: Multiple Spanning Tree Protocol
    - 802.1w: Rapid Spanning Tree Protocol
    - 802.1aq: Shortest Path Bridging (SPB) (incorporate all the older spanning tree protocols)

1.7.3 Network Topologies: Crossbar

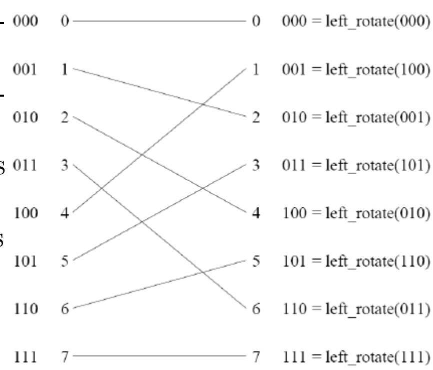
- Principle and Properties
  - a crossbar network uses an $p \cdot b$ grid of switches to connect p inputs to b outputs in a non-blocking manner
- Bottleneck
  - the cost of a crossbar of p processors grows as $O(p^2)$ → difficult to scale for large values of p
- Usage
  - in non-blocking switches
  - between L2- and L2-caches



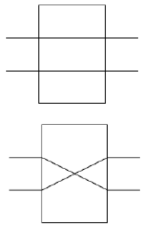
1.7.4 Network Topologies: Multistage Network

- Scalability
  - busses have excellent cost scalability, but poor performance scalability
  - crossbars have excellent performance scalability but poor cost scalability
  - multistage interconnects strike a compromise between these extremes
- Example: Omega Network
  - it consists of $log(p)$ stages, where p is the number of inputs/outputs
  - at each stage, input i is connected to output j if:

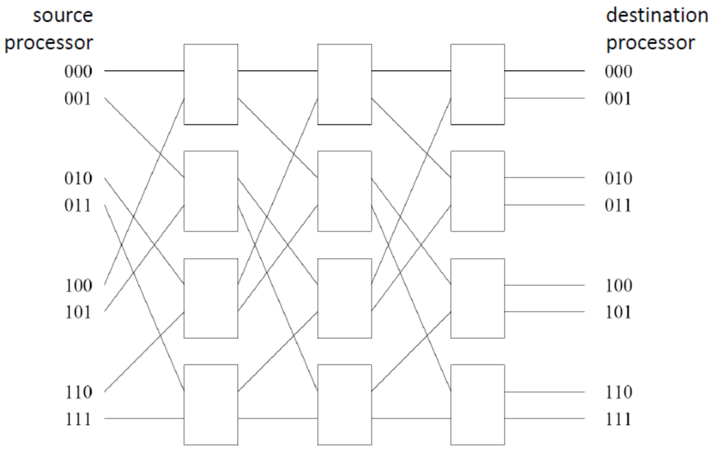
$$j = \begin{cases} 2i, & 0 \leq i \leq p/2 - 1 \\ 2i + 1 - p, & p/2 \leq i \leq p - 1 \end{cases}$$



Omega Network

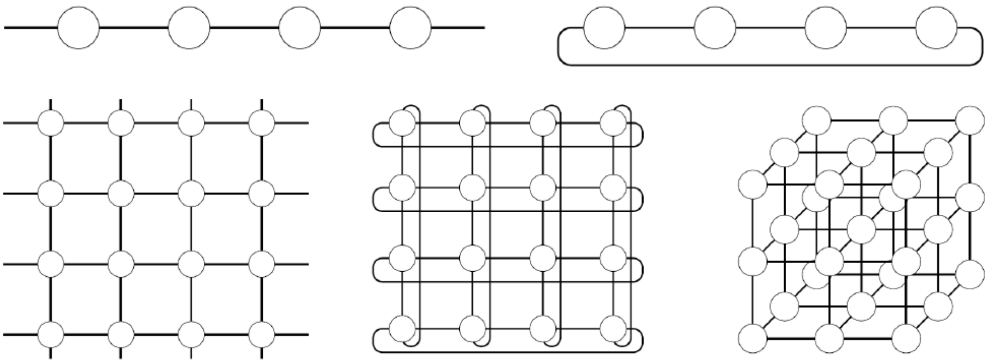


- Principle and Properties
  - the perfect shuffle patterns are connected using 2x2 switches
  - the switches operate in two modes: pass-through or cross-over



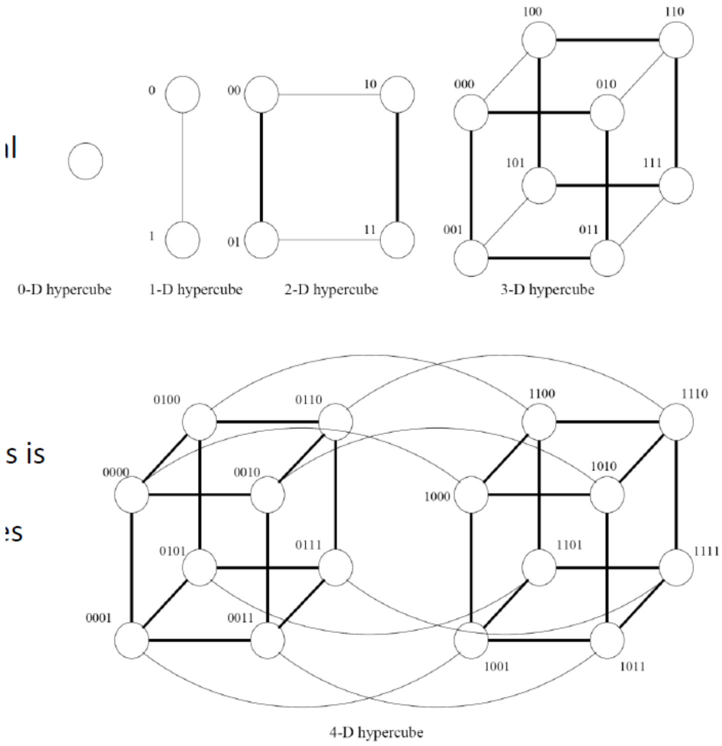
Linear Array, Mesh, and $k-d$ Mesh

- Principle and Properties
  - in a linear array, each node has two neighbors, one to its left and one to its right
  - if the nodes at either end are connected, we refer to it as a 1-D torus or a ring
  - a generalization to 2 dimensions has nodes with 4 neighbors, to the north, south, east, and west (toroidal mesh)
  - a further generalization to d dimensions has nodes with $2d$ neighbors

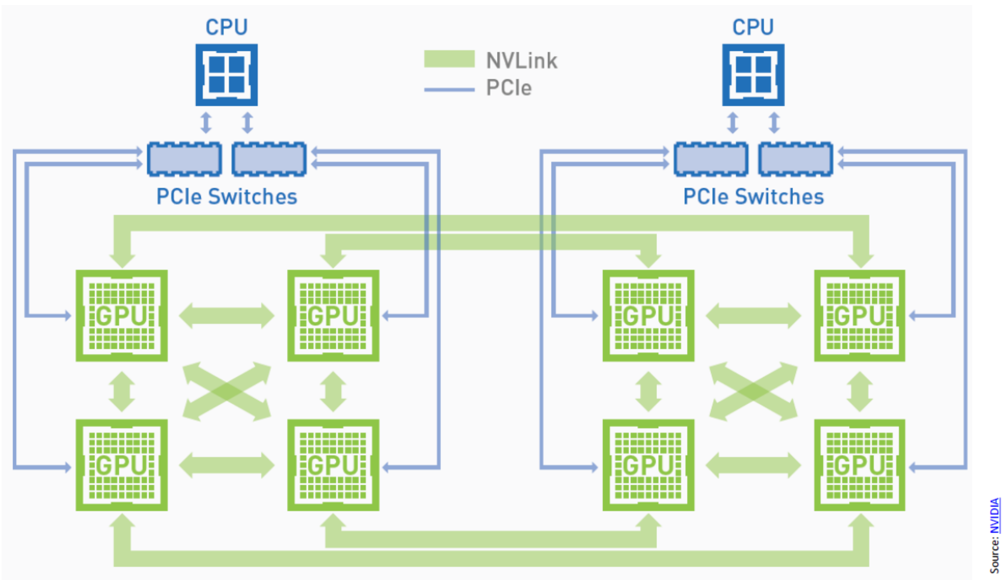


1.7.5 Network Topologies: Hypercube

- Principle and Properties
  - a special case of a d-dimensional mesh is a hypercube
  - $d = log(p)$, where p is the total number of nodes
  - the distance between any two nodes is at most $log(p)$
  - each node has $log(p)$ neighbors
  - the distance between two nodes is given by the number of bit positions at which the two nodes differ

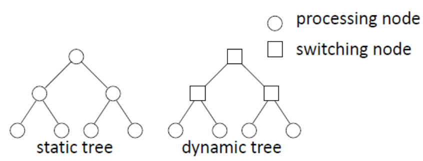


NVIDIA NVLink: Hypercube Mesh Hybrid

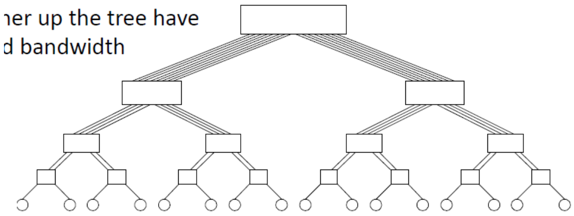


1.7.6 Tree-Based Networks

- Principle and Properties
  - one path between any pair of nodes
    - linear arrays and star-connected networks are special cases of tree networks
  - the distance between any two nodes is no more than $2log(p)$
  - links higher up the tree potentially carry more traffic than those at the lower levels
  - trees can be laid out in 2D with no wire crossings
- Fat-Tree



- links higher-up the tree have increased bandwidth



1.7.7 Evaluating Interconnection Networks

- Diameter
  - the distance between the farthest two nodes in the network
- Channel Bandwidth = channel width x channel rate
  - channel width: number of bits that can be communicated simultaneously over a link
  - channel rate: peak data transfer rate per link
- Cross-Section Bandwidth = bisection width x channel bandwidth
  - bisection width: the minimum number of wires one must cut to divide the network into two equal parts
- Cost
  - the number of links or switches (whichever is asymptotically higher) is a meaningful measure of the cost
  - the ability to layout the network
  - the length of wires
  - ...

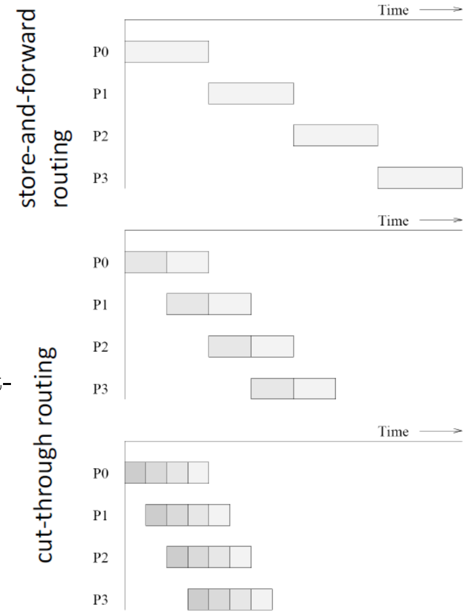
Network	Diameter	Bisection width	Arc connectivity	Cost (No. of links)
Completely-connected	1	$p^2/4$	$p-1$	$p(p-1)/2$
Star	2	*	1	$p-1$
Complete binary tree	$2 \cdot log((p+1)/2)$	1	1	$p-1$
Linear array	$p-1$	1	1	$p-1$
2D Mesh, no wraparound	$2(\sqrt{p}-1)$	$\sqrt{p}$	2	$2(p-\sqrt{p})$
2D wraparound Mesh	$2\lfloor\sqrt{p}/2\rfloor$	$2\sqrt{p}$	4	$2p$
Hypercube	$log(p)$	$p/2$	$log(p)$	$(p \cdot log(p))/2$
Wraparound k-ary d-cube	$d\lfloor k/2\rfloor$	$2k^{d-1}$	$2d$	dp

* depends on the node (switch) in the center, e.g. Crossbar or Omega Network

Network	Diameter	Bisection width	Arc connectivity	Cost (No. of links)
Crossbar	1	p	1	p^2
Omega Network	$log(p)$	$p/2$	2	$p/2$
Dynamic Tree	$2 \cdot log(p)$	1	2	$p-1$

1.8 Communication costs in parallel systems

- Overhead in parallel programs
  - idling
  - contention
  - communication
- Communication costs depend on
  - communication model
  - the network topology
  - data handling and routing (e.g. packet routing, cut-through routing)
  - associated software protocols
  - ...



1.8.1 Message Passing Costs

- Total time to transfer a message over a network comprises of the following:
  - *Startup time* (t_s): Time spent at sending and receiving nodes (executing the routing algorithm, programming routers, etc.)
  - *Per-hop time* (t_h): This time is a function of number of hops and includes factors such as switch latencies, network delays, etc.
  - *Per-word transfer time* (t_w): This time includes all overheads that are determined by the length of the message. This includes bandwidth of links, error checking and correction, etc.

1.8.2 Cost Model for Communicating Messages

- Communication Costs
  - the cost of communicating a message between two nodes/hops away using cut-through routing is given by
$$t_{\text{comm}} = t_s + l \cdot t_h + m \cdot t_w$$
  - t_h is typically smaller than t_s and t_w, so the second term does not show, when m is large
  - furthermore, it is often not possible to control routing and placement of tasks
- Simplified Cost Model
$$t_{\text{comm}} = t_s + m \cdot t_w$$
- Remarks
  - it is important to note that the original expression for communication time is valid for only uncongested networks
  - if a link takes multiple messages, the corresponding t_w term must be scaled up by the number of messages
  - different communication patterns congest different networks to varying extents

1.8.3 Cost Model for Shared Memory Systems

- Simplified Cost Model (still practical, but accurate cost modeling is more difficult)
  - memory layout is typically determined by the system
  - finite cache sizes can result in cache thrashing

- overheads associated with invalidate and update operations are difficult to quantify
- spatial locality is difficult to model
- pre-fetching can play a role in reducing the overhead associated with data access
- false sharing and contention are difficult to model