CSE 891 - Section 1: Parallel Computing - Fundamentals and Applications

Fall 2014 - Lecture 3:

Parallel Computer Architectures (2)

H. Metin Aktulga

hma@cse.msu.edu

517-355-1646

Lecture 2 - Summary

- von Neumann Architecture
- Modifications to the von Neumann Architecture
  - Caches
    - Spatial & temporal locality, cache types and characteristics, examples
  - Seamless parallelism mechanisms
    - Bit-level parallelism
    - Instruction Level Parallelism (ILP)
      - Pipelining, multiple issue, speculation

Parallel Hardware

- Hardware multithreading
- SIMD systems
  - Vector processors
  - GPUs
- MIMD systems
  - Shared-memory systems
  - Distributed-memory systems

Threading

- **Process vs. thread?**

Processes have **their own**:

- executables
- unique process ids
- address spaces
- system resources
- environment vars
- priority
- 1+ thread of execution

Threads of a process **share the same**:

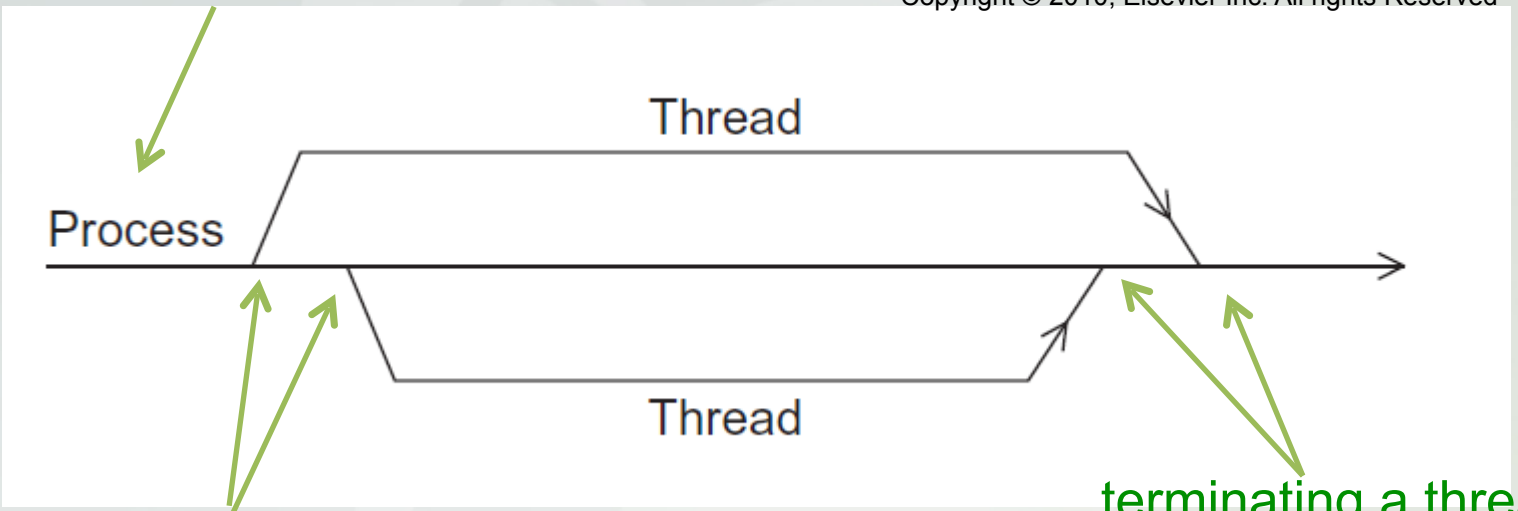
- executable
- address space
- system resource
- environment var
- priority

Threads have **different**:

- program counters
- call stacks
- register files

A process and two threads

the “master” thread



starting a thread
is called **forking**

terminating a thread
is called **joining**

Hardware multithreading

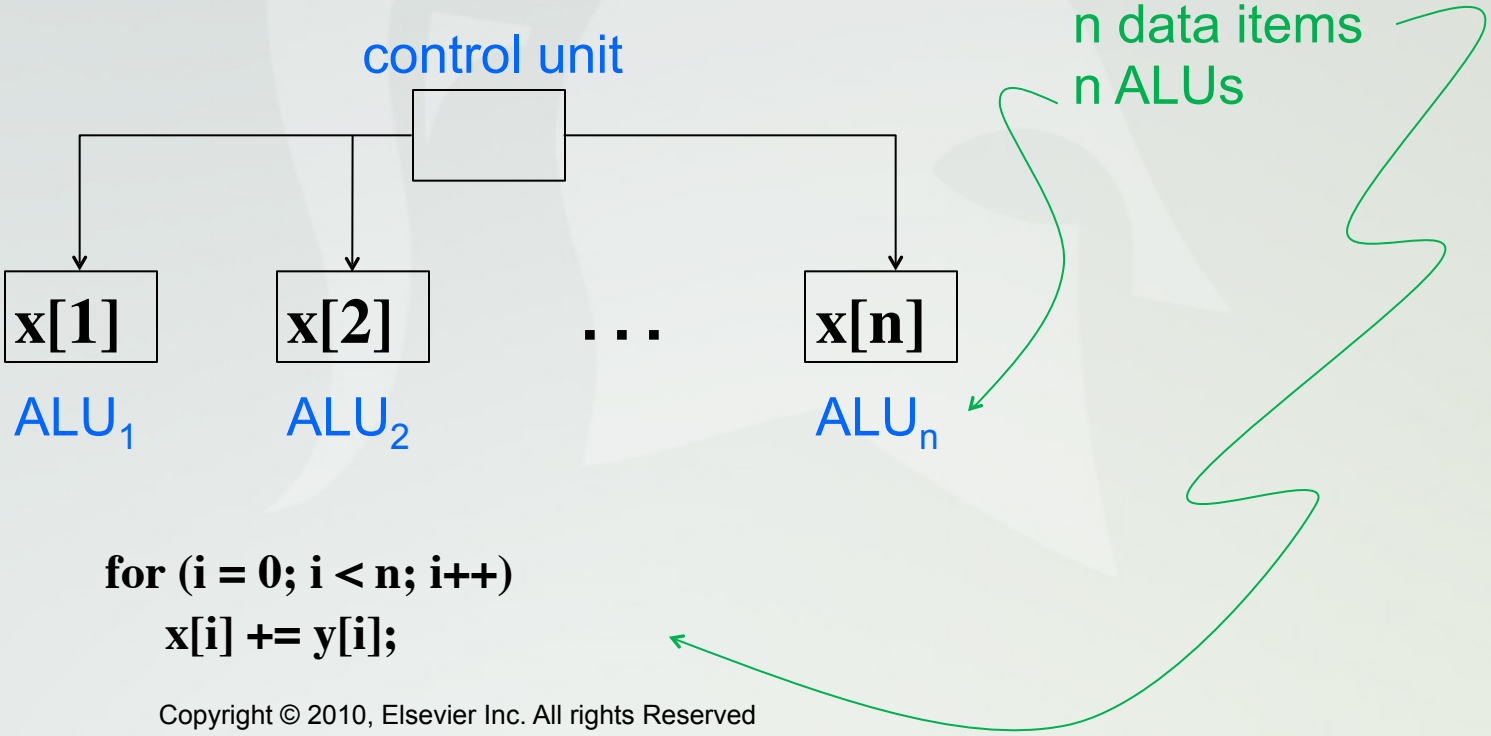
- ILP can be hard to exploit
- Programmers may expose explicit parallelism through threading
- Hardware support includes: thread private program counters, call stacks, register files
- **Coarse grained:** switch to another thread when the current execution thread stalls - one thread in the pipeline
- **Fine grained:** switch between all available threads after each instruction - skip the stalled ones - multiple threads in the pipeline at the same time
- **Simultaneous multithreading (SMT):** instructions from multiple threads are issued at each cycle (ex: Intel Hyper-Threading)

Flynn's Taxonomy

classic von Neumann SISD Single instruction stream Single data stream	(SIMD) Single instruction stream Multiple data stream
MISD Multiple instruction stream Single data stream very few examples: pipelining, task replication for fault tolerance	(MIMD) Multiple instruction stream Multiple data stream

SIMD architectures

- Apply the same instruction to multiple data items
- AKA data parallelism



SIMD architectures

- Typically, n >> #ALUs - so loops are processed in batches
  - Ex. m = 4 ALUs and n = 15 data items

Round	ALU	ALU	ALU	ALU
1	X[0]	X[1]	X[2]	X[3]
2	X[4]	X[5]	X[6]	X[7]
3	X[8]	X[9]	X[10]	X[11]
4	X[12]	X[13]	X[14]	

- ALUs must execute the same instructions synchronously
- Good for large data parallel tasks, but not for complex parallel problems

SIMD architecture examples

- Vector processors (Thinking Machines, Cray)
- GPUs (ATI, NVIDIA cards)
- CPUs with SIMD units (Xeon Phi: 512-bit intrs)

Vector processors

- Used to be very popular in 1990's (Thinking Machines, Cray systems)
- They are coming back in simpler forms (SIMD units in CPUs)
- Typical properties:
  - Vector registers (e.g. MMX, SSE, SSE2, AVX, AVX2…)
  - Vectorized functional units
  - Vector instructions
  - Strided memory access & hardware scatter/gather

Vector processors

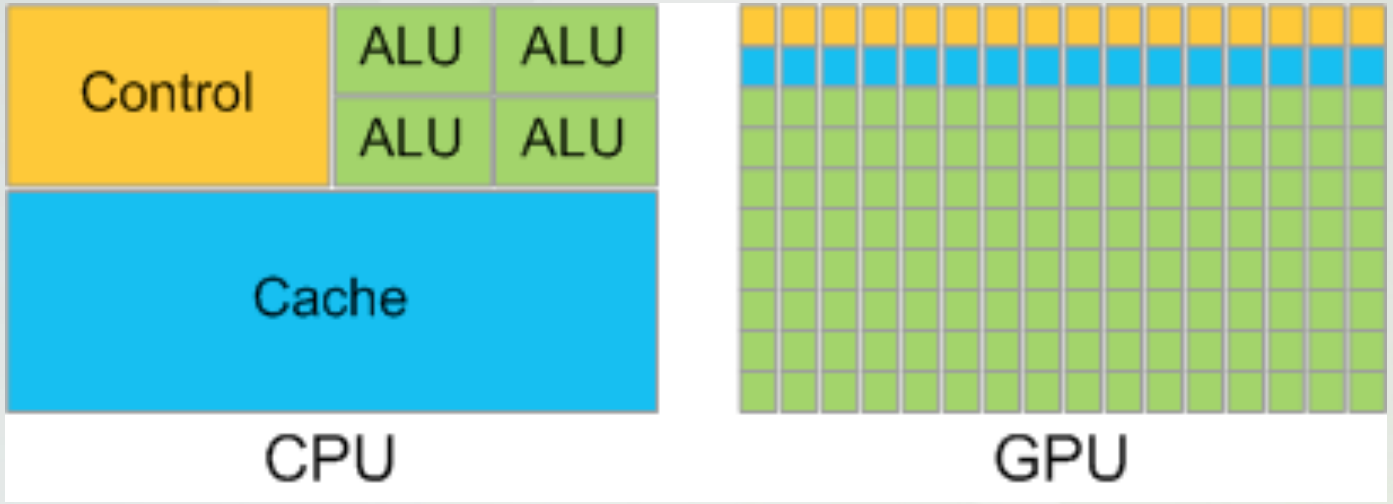
- **Pros**
  - Fast (saves on redundant instructions)
  - Easy to use.
  - Good compiler support
- **Cons**
  - Can not handle irregular data structures
  - Not so good scalability

GPUs

- Graphic Processing Units - used for rendering images on screens
- Lots of lines, rectangles = arrays, matrices = dense linear algebra as in LINPACK
- Lots of parallelism - each pixel can be rendered independently using the same instructions
  - SIMD parallelism

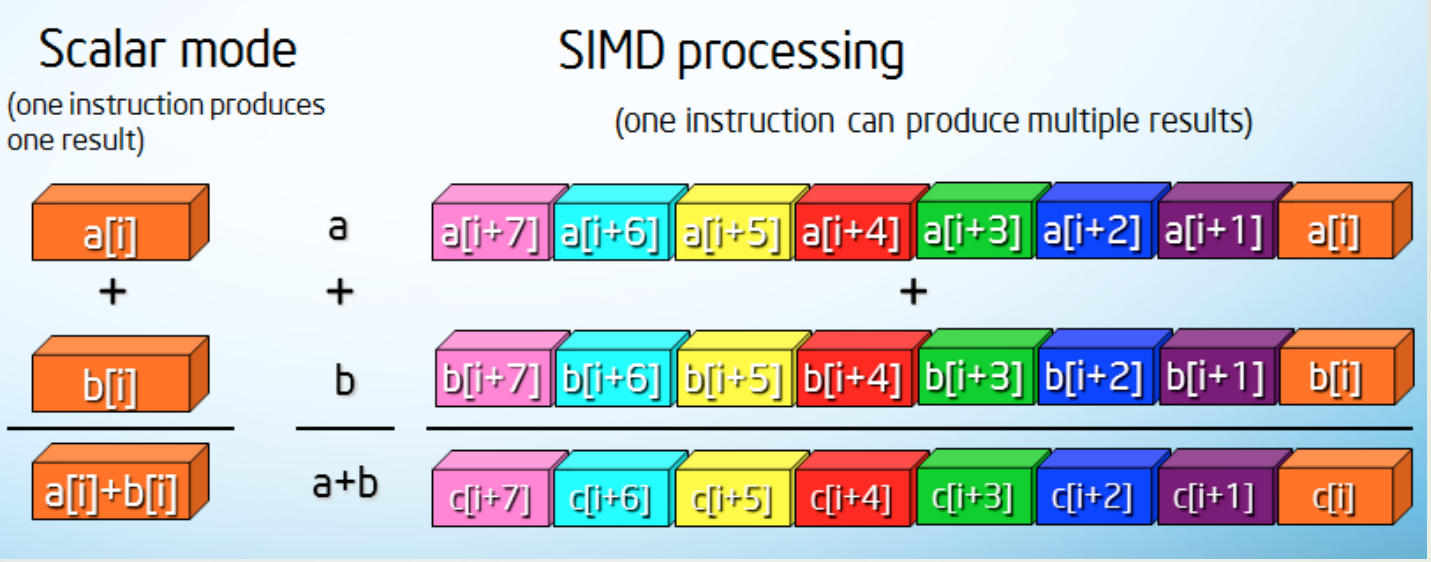
GPU vs CPU

- Much reduced control logic and cache space
- Great emphasis on processing logic



Xeon Phi

- Increased emphasis on SIMD processing
  - MMX - 64 bits
  - SSE - 128 bits
  - AVX - 256 bits
  - AVX2 - 512 bits



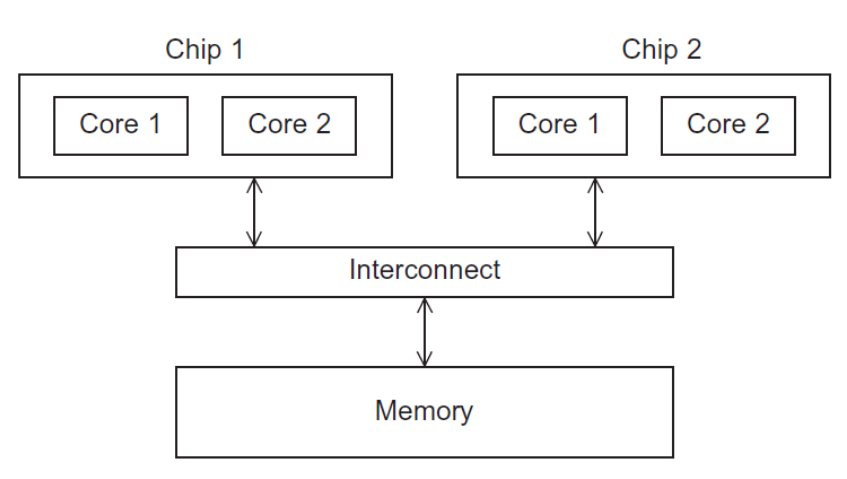
Multiple Instruction Multiple Data (MIMD) systems

- Support for multiple simultaneous instruction streams operating on multiple data streams.
- Typically consists of a collection of fully independent processing units or cores, each of which has its own control unit and its own ALU.
- Two types depending on memory:
  - Shared memory architectures
  - Distributed memory architectures

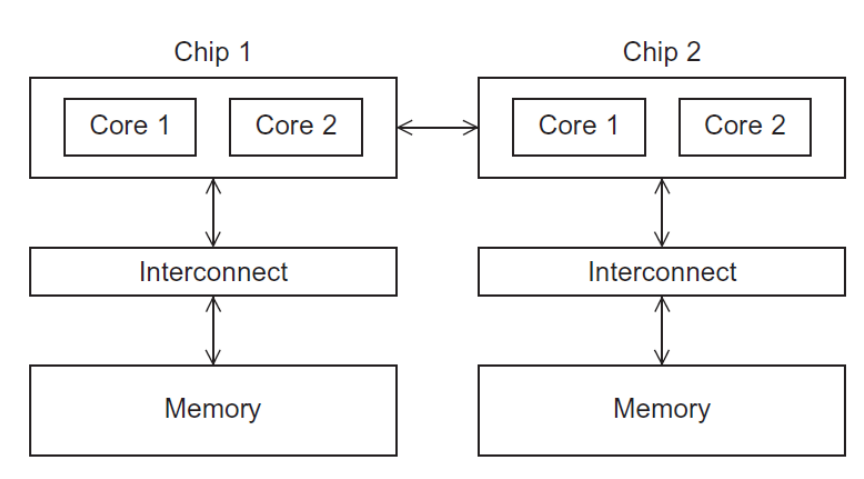
Shared memory architectures

- One or more (multi-core) processors
- Each with access to a single memory system

UMA



NUMA

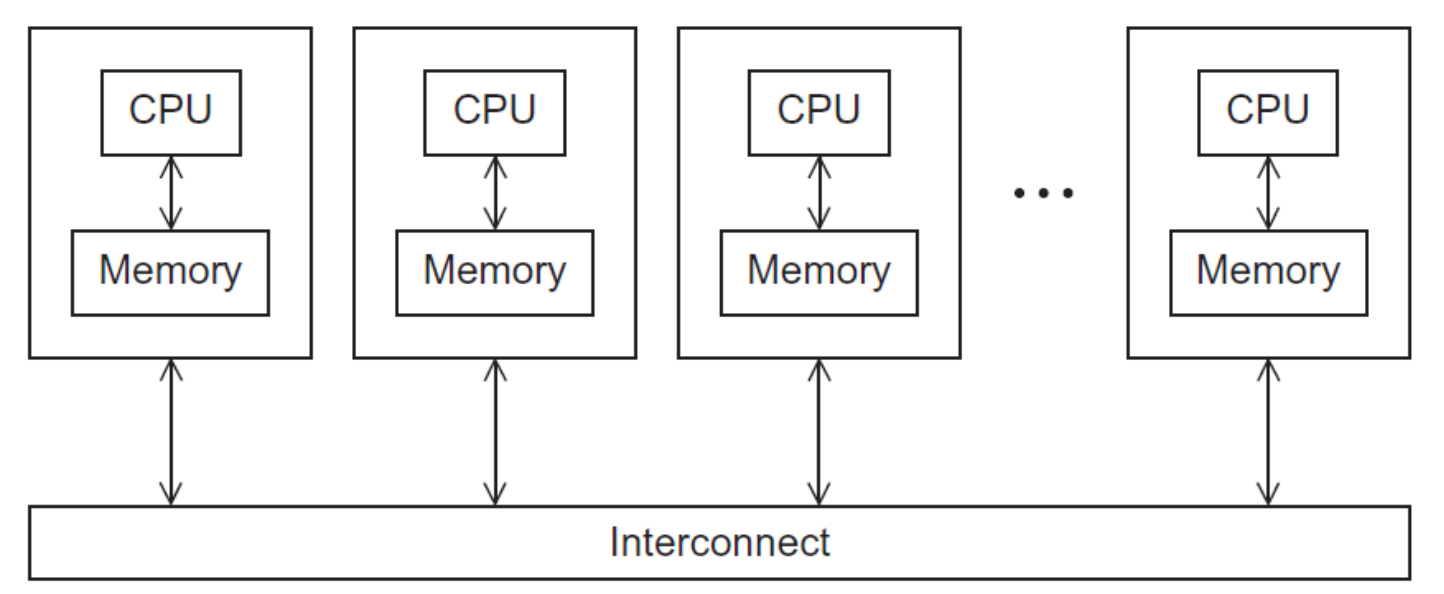


Uniform Memory Access

Non-Uniform Memory Access

Distributed memory architectures

- Clusters (commodity computers & interconnects)
- Supercomputers (high perf. nodes & interconnects)



Distributed memory architectures - Titan: A state-of-the-art system

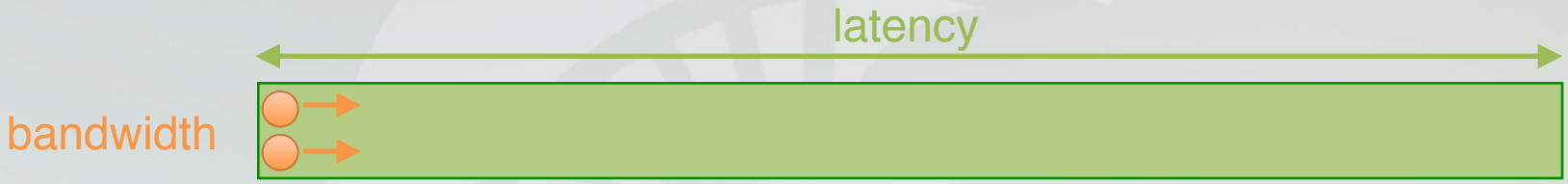
Interconnection networks

- The network hardware that connects different subsystems (processing units, memory banks, compute nodes) in a system.

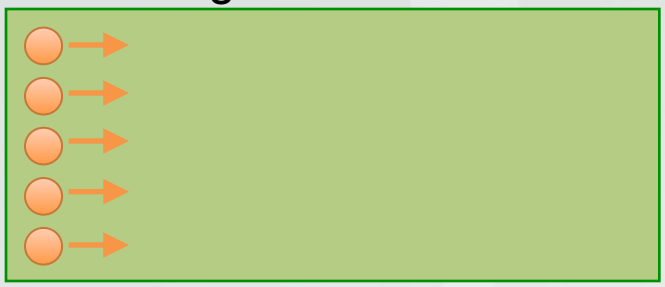
- **Latency (α):** Time elapsed between source & destination to transmit the first byte. Limited by the medium and length. Unit is seconds.

- **Bandwidth (β):** The rate at which data can be pumped between source and destination. Determined by the "width" of the medium. Unit is bytes/second.

Latency vs. Bandwidth



low latency
high bandwidth



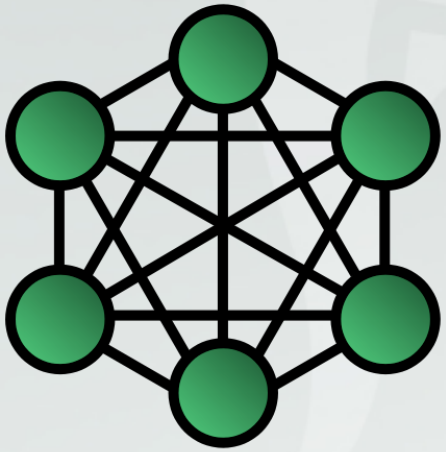
high latency, low bandwidth

data transmission time = $\alpha + D/\beta$

Interconnect topology

Ideal for performance:

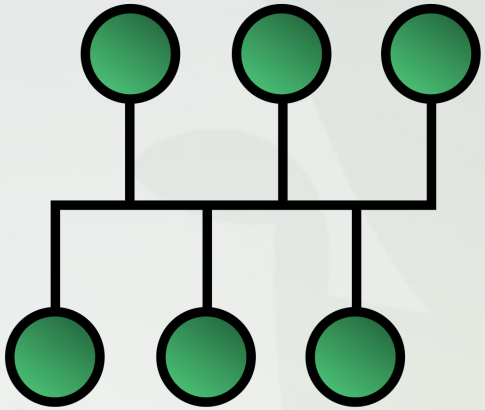
a fully connected network with high β, low α links



Ex: Internet backbone

Ideal for cost:

a bus with low β, low α



Ex: Memory bus,
Local area networks (LAN)

Need to make some trade-offs!

Interconnect terminology

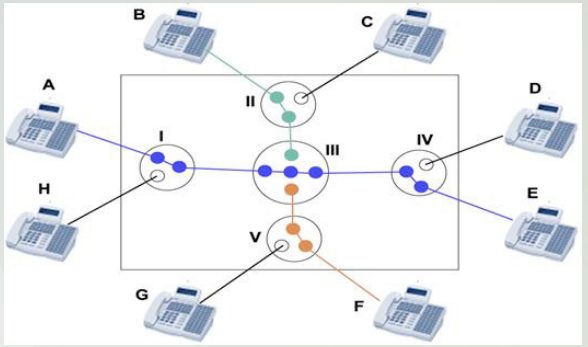
- **Network diameter:** maximum # of links between any two nodes (i.e. latency of the network)

- **Bisection bandwidth:** minimum bandwidth to cut the network into two equal parts (bandwidth of the network - # simultaneous communications, connectivity).

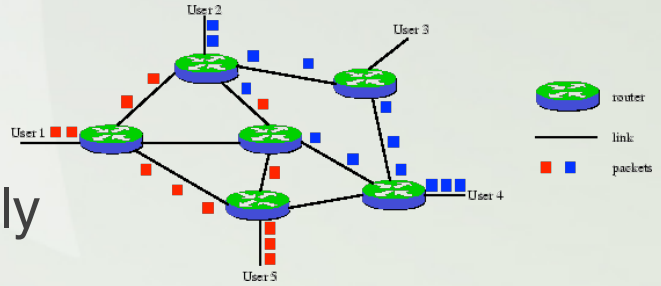
	Diameter	*Bisection bandwidth*
Full network	1	(p/2)
Bus	p-1	1

Types of interconnects (historical order, more or less)

- Circuit switching
  - Like telephone networks
  - Reliable communication between pairs
  - Limited number of simultaneous circuits

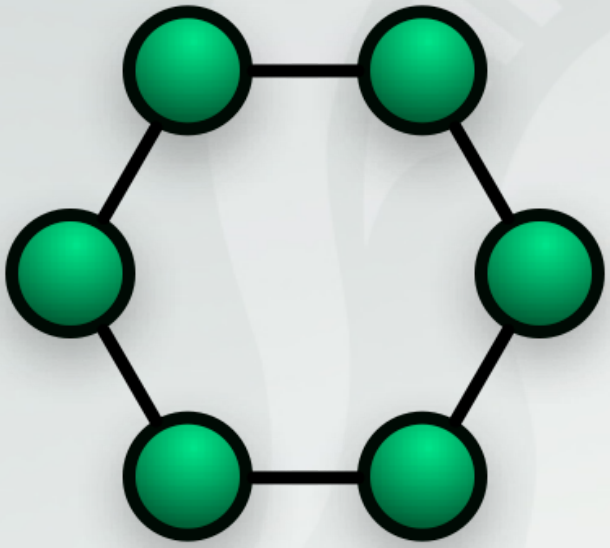


- Packet switching
  - Data is transmitted in small packages
  - Any available link can be used
  - All pairs can communicate simultaneously



Network topologies: Ring network

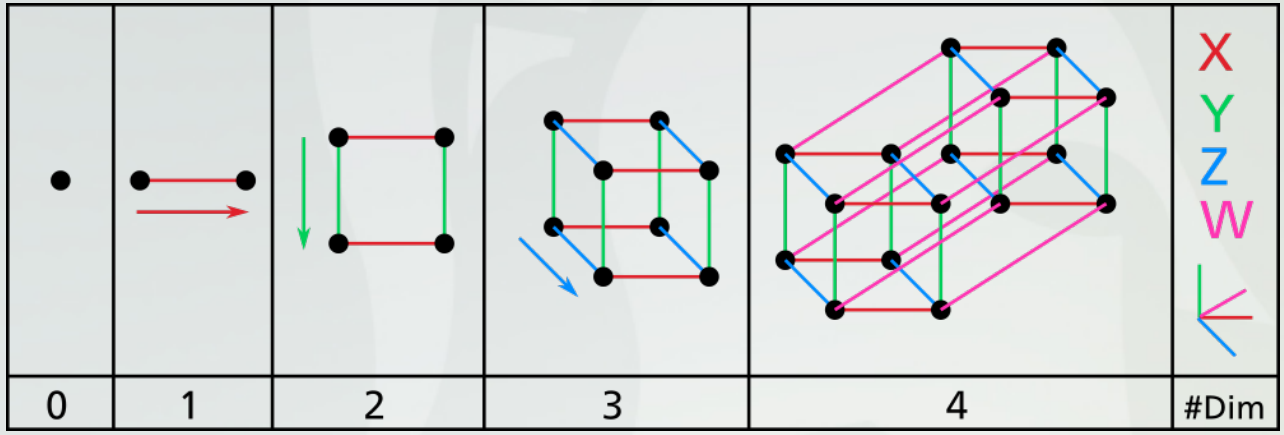
- A bus with a wrap-around link



- Diameter: $\lfloor p/2 \rfloor$
- Bisection: 2

Network topologies: Hypercube

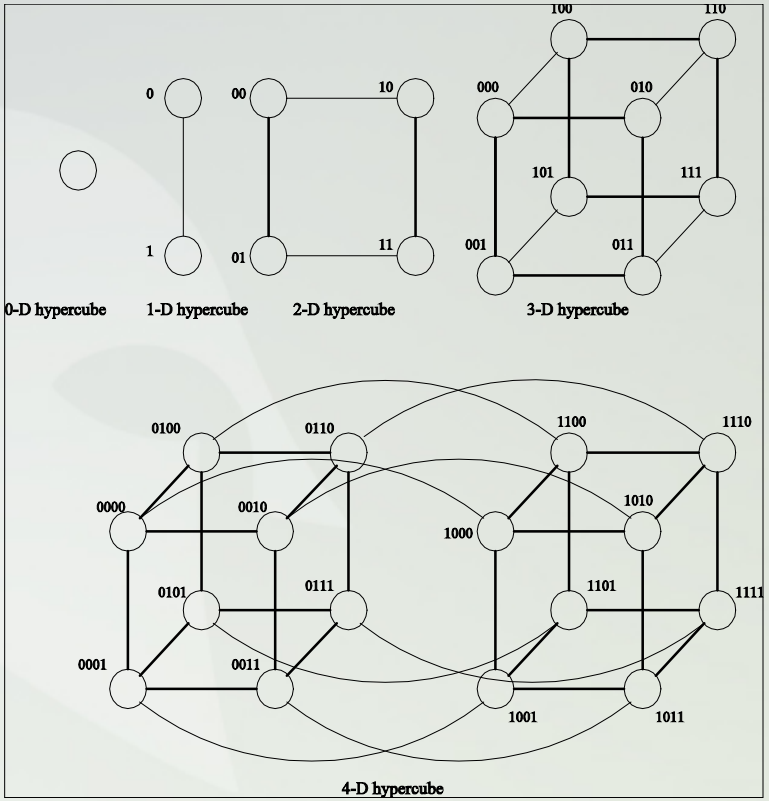
- Was very popular in the early years of parallel computing
- Built inductively:



0	1	2	3	4	#Dim	in general
1	2	4	8	16	vertices	2
0	1	4	12	32	edges	d2
0	1	2	3	4	diameter	d
0	1	2	4	8	bisection	2

Hypercube Properties

- The distance between any two nodes is at most *d*.
- Each node has *d* neighbors.
- The distance between two nodes is given by the number of bit positions at which the two nodes differ in their binary representations.
- Messages between two nodes can be routed by comparing their binary representations.

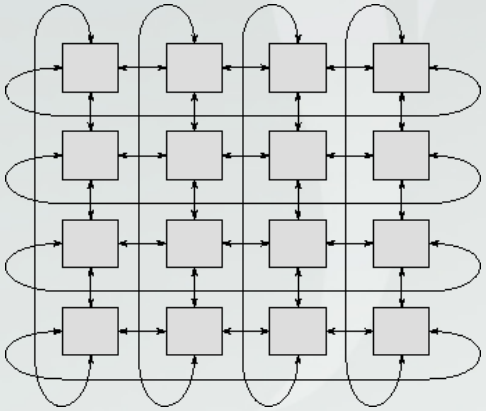


Hypercube Properties

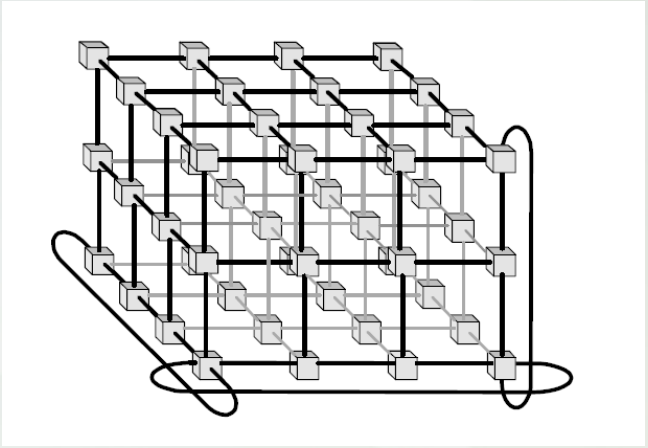
- Pros
  - Mathematically beautiful
  - Easy to route & develop algorithms for
  - Low dimension, high bisection bandwidth
- Cons
  - Expensive
  - Hard to scale
  - Special number of nodes only, 2^d

Network topologies: k-D p-ary torus

- A good trade-off between ring and hypercube
- Typically used as a 3D torus
- Japanese K-computer uses a custom 5D torus



2D Torus

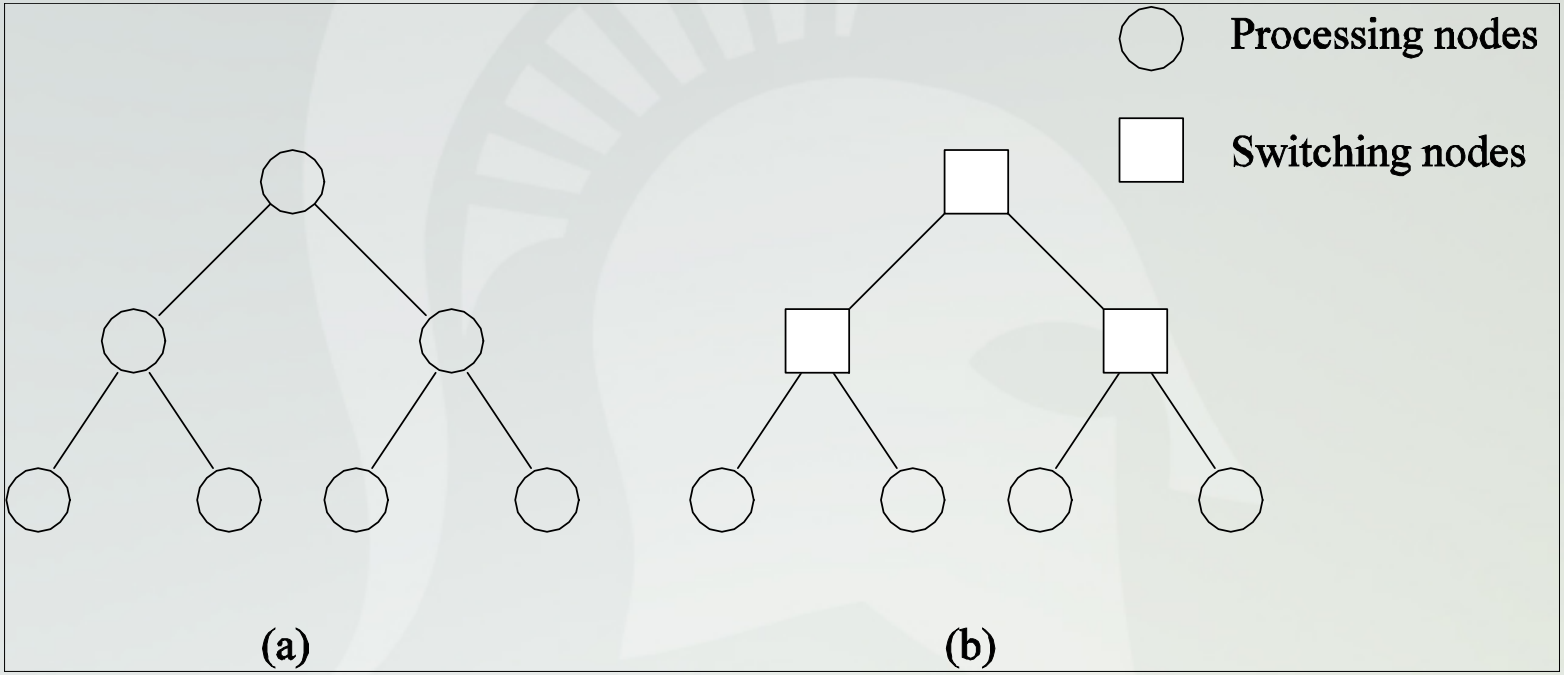


3D Torus

Network topologies: k-D p-ary torus properties

1	2	3		in general
p	p	p	vertices	p
p	2p	3p	edges	kp
$\lfloor p/2 \rfloor$	$2 \lfloor p/2 \rfloor$	$3 \lfloor p/2 \rfloor$	diameter	$k \lfloor p/2 \rfloor$
2	2p	2p	bisection	2p

Network topologies: Tree-based networks

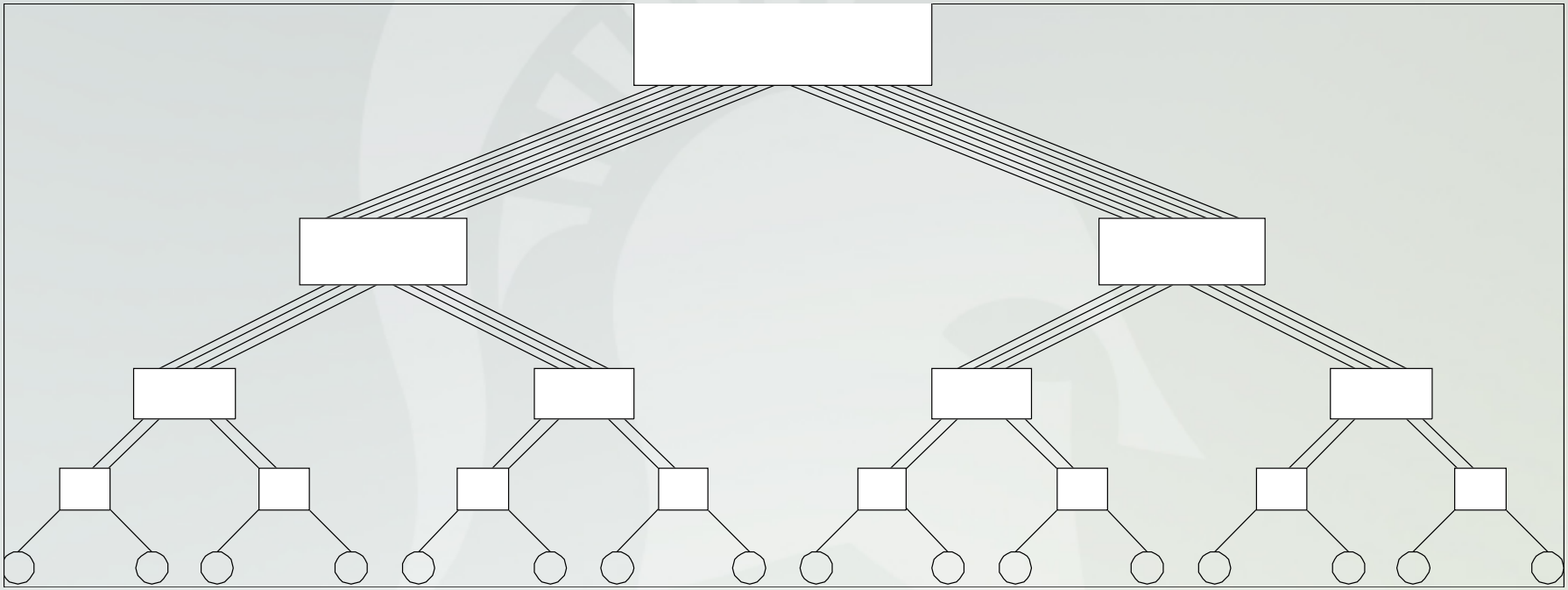


Complete binary tree networks: (a) a static tree network; and (b) a dynamic tree network.

Network topologies: Tree-based networks

- If there are p total nodes, the depth of the tree is $log(p)$.
- 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.
- For this reason, a variant called a fat-tree, fattens the links as we go up the tree.
- Trees can be laid out in 2D with no wire crossings. This is an attractive property of trees.

Network topologies: Fat trees



A fat tree network of 16 processing nodes.