CSCI-UA.0480-003

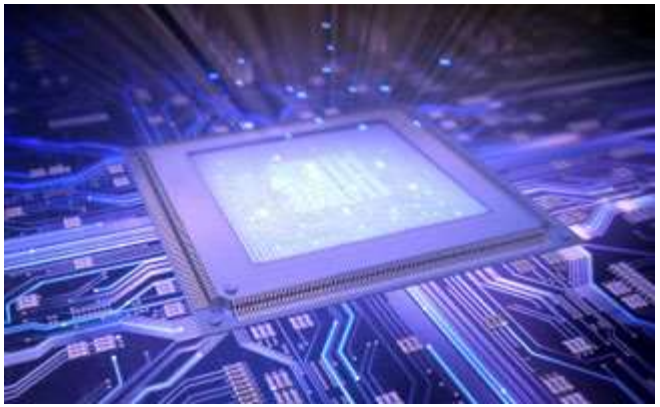
Parallel Computing

Lecture 3: Parallel Hardware: Advanced

Mohamed Zahran (aka Z)
mzahran@cs.nyu.edu
http://www.mzahran.com

Some slides are adopted from:

- G. Barlas book
- P. Pacheco book



Last lecture we looked at techniques to exploit ILP (Instruction Level Parallelism)

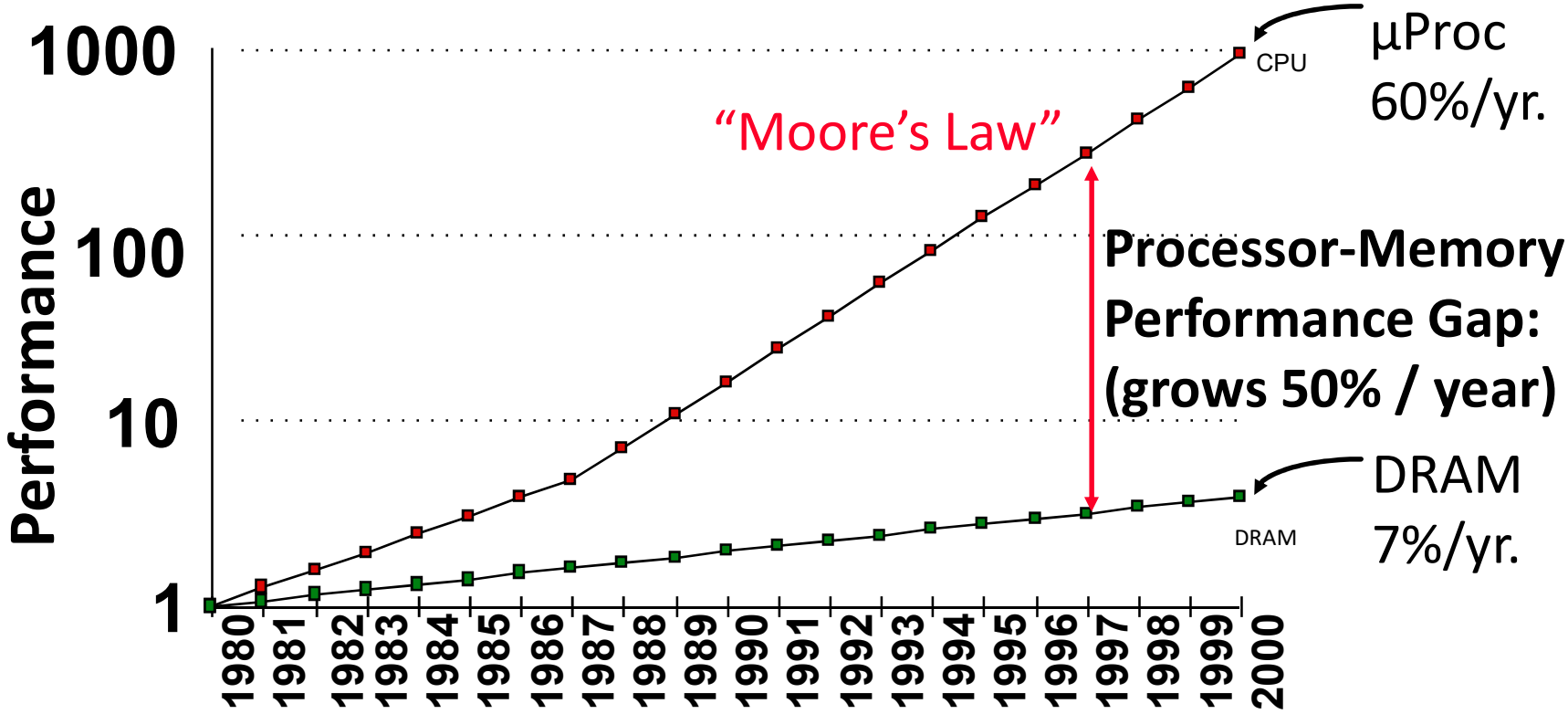
- Pipelining
- Superscalar
- Out-of-order execution
- Speculative execution
- Simultaneous Multithreading (aka Hyperthreading technology)

All the above require very little, if at all, work from the side of the programmer to make use of.

Computer Technology … Historically

- Memory
  - DRAM capacity: 2x / 2 years (since '96);
    64x size improvement in last decade.
- Processor
  - Speed 2x / 1.5 years (since '85); → BUT!!
    100X performance in last decade.
- Disk
  - Capacity: 2x / 1 year (since '97)
    250X size in last decade.

Memory Wall

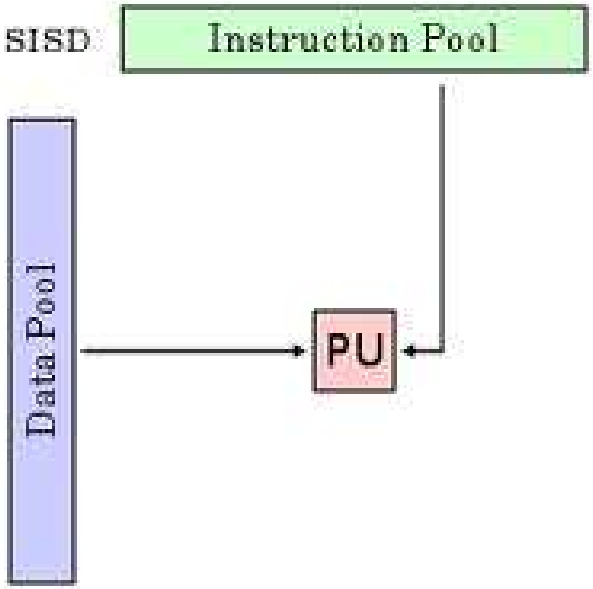


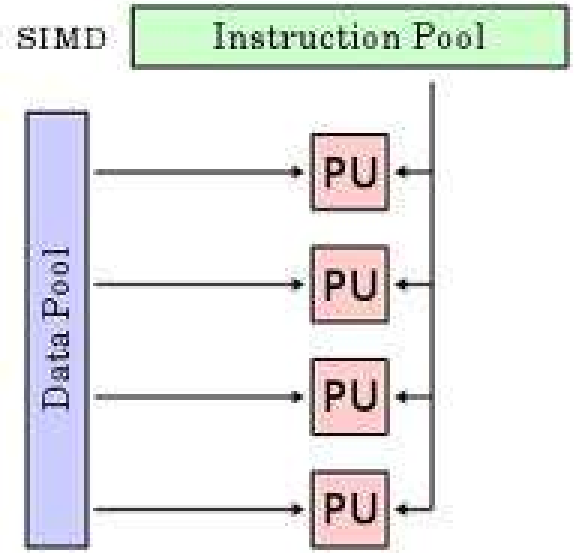
Most of the single core performance loss is on the memory system!

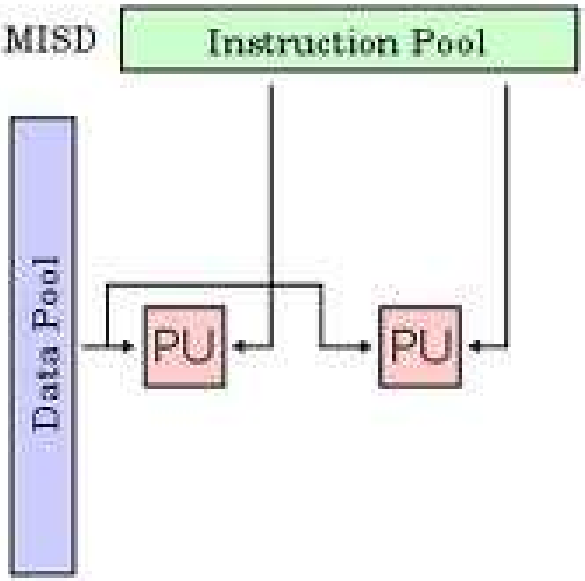
Flynn's Taxonomy

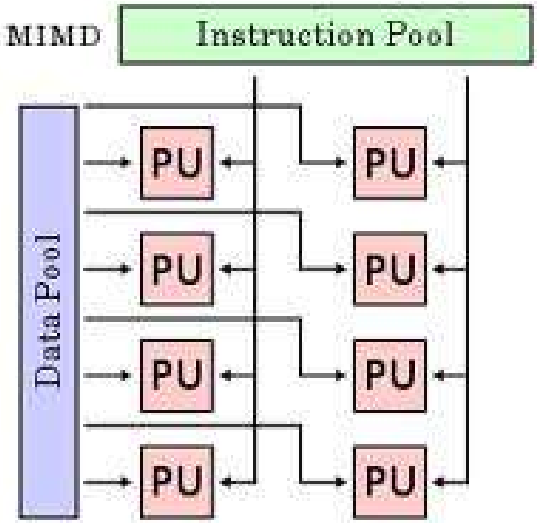
classic von Neumann

SISD Single instruction stream Single data stream	(SIMD) Single instruction stream Multiple data stream
MISD Multiple instruction stream Single data stream	(MIMD) Multiple instruction stream Multiple data stream


SISD
Instruction Pool
Data Pool
PU


SIMD
Instruction Pool
Data Pool
PU
PU
PU
PU

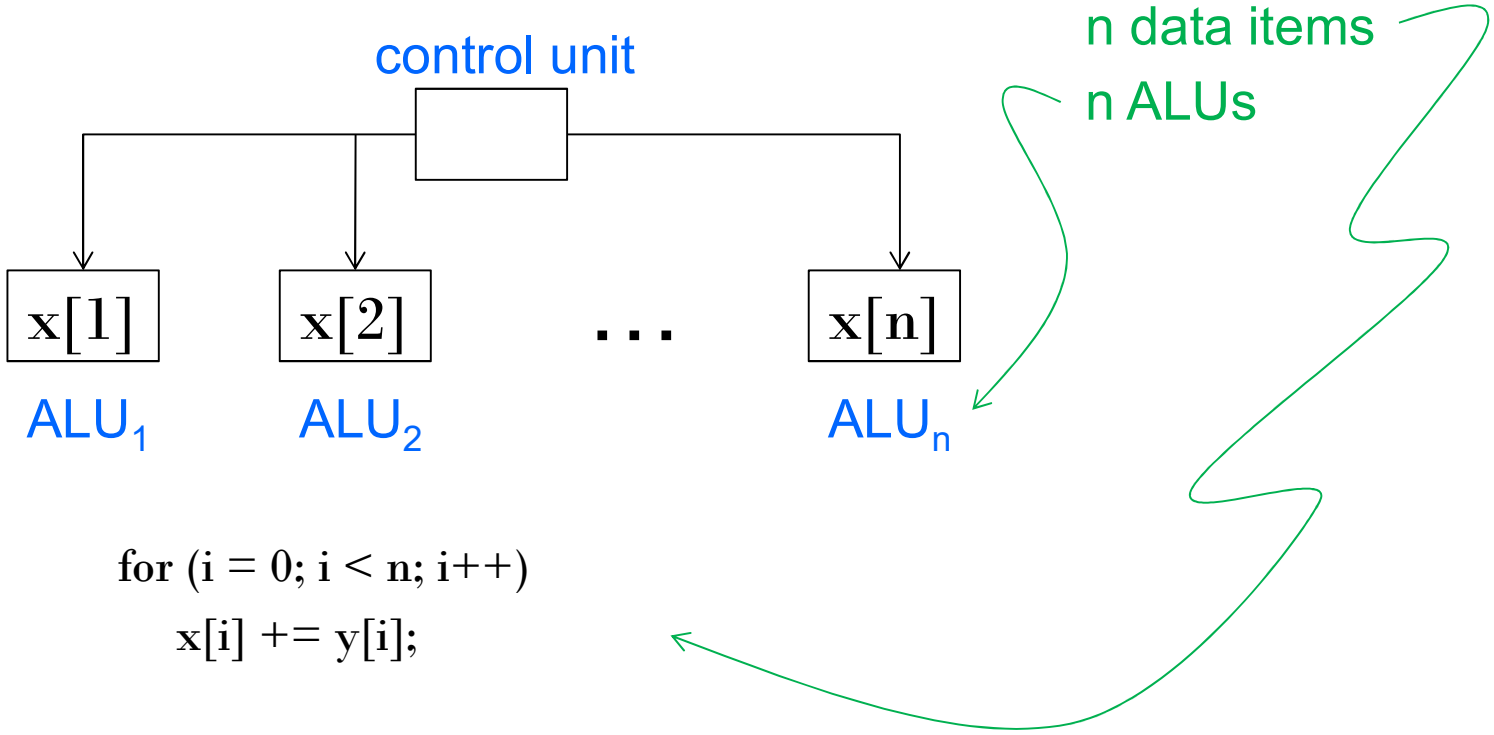

MISD
Instruction Pool
Data Pool
PU
PU


MIMD
Instruction Pool
Data Pool
PU
PU
PU
PU
PU
PU
PU
PU

SIMD

- Parallelism achieved by dividing data among the processors.
- Applies the same instruction (or group of instructions) to multiple data items.
- Called data parallelism.
- Example:
  - GPUs
  - vector processors

SIMD example




```
for (i = 0; i < n; i++)
    x[i] += y[i];
```

SIMD

- What if we don't have as many ALUs as data items?
- Divide the work and process iteratively.
- Example 4 ALUs and 15 data items.

Round3	ALU_1	ALU_2	ALU_3	ALU_4
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]	

SIMD drawbacks

- All ALUs are required to execute the same instruction(s), or remain idle.
- In classic design, they must also operate synchronously.
- The ALUs have no instruction storage.
- Efficient for large data parallel problems, but not other types of more complex parallel problems.

Vector processors

- Processors execute instructions where operands are vectors instead of individual data elements or scalars.
- This needs:
  - Vector registers
    - Capable of storing a vector of operands and operating simultaneously on their contents.
  - Vectorized functional units
    - The same operation is applied to each element in the vector (or pairs of elements)

Vector processors - Pros

- Fast
- Easy to use.
- Vectorizing compilers are good at identifying code to exploit.
- Compilers also can provide information about code that cannot be vectorized.
  - Helps the programmer re-evaluate code.
- High memory bandwidth
- Uses every item in a cache line.

Vector processors - Cons

- They don't handle irregular data structures.
- A very finite limit to their ability to handle ever larger problems. (scalability)

MIMD

- Supports multiple simultaneous instruction streams operating on multiple data streams.
- Typically consist of a collection of fully independent processing units or cores, each of which has its own control unit and its own ALU.
- Example: multicore processors, multiprocessor systems

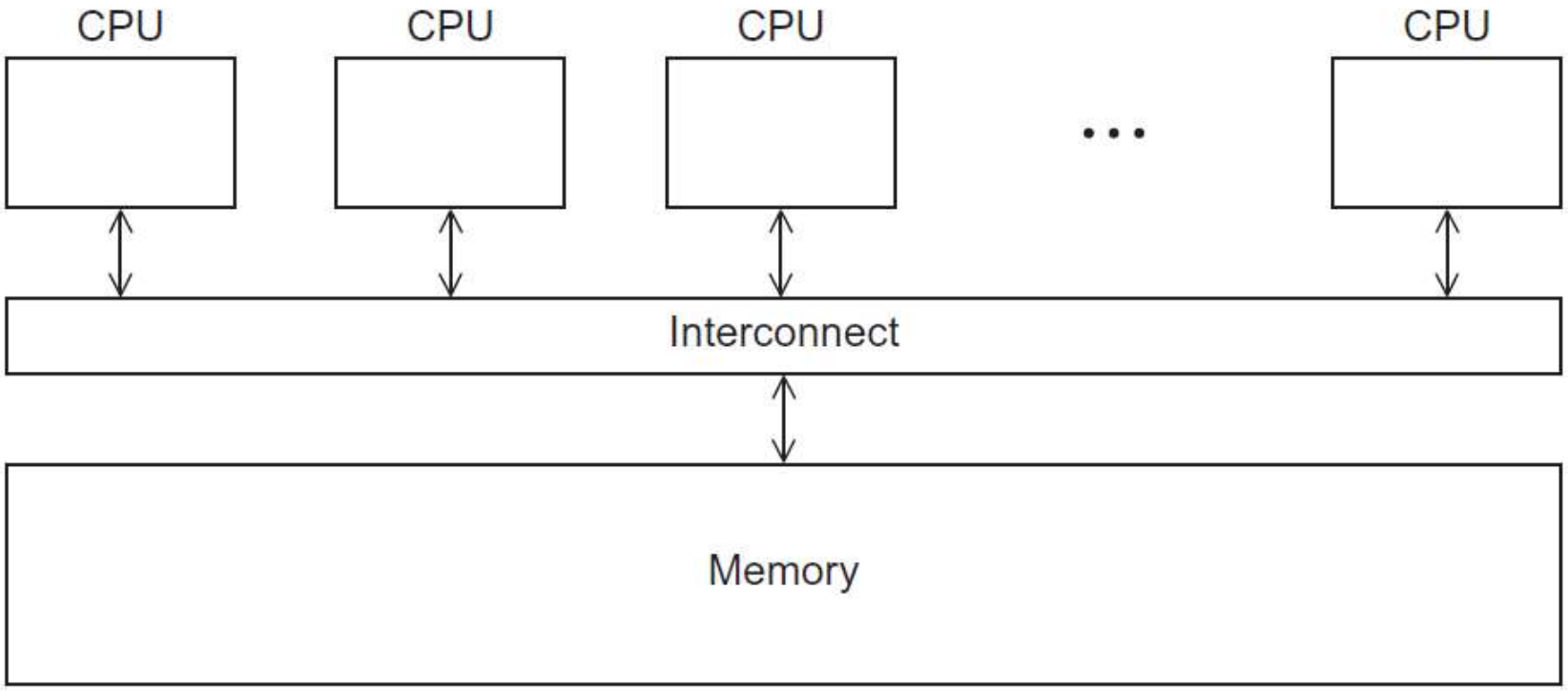
Flynn's classification is based on how instructions and data are used.

How about we classify based on how memory is designed?

Shared Memory System

- A collection of autonomous processors is connected to a memory system via an interconnection network.
- Each processor can access each memory location.
- The processors usually communicate implicitly by accessing shared data structures.

Shared Memory System



Suppose that one CPU wants to access addr1, and another CPU wants addr2, will they both see the same memory access delay?

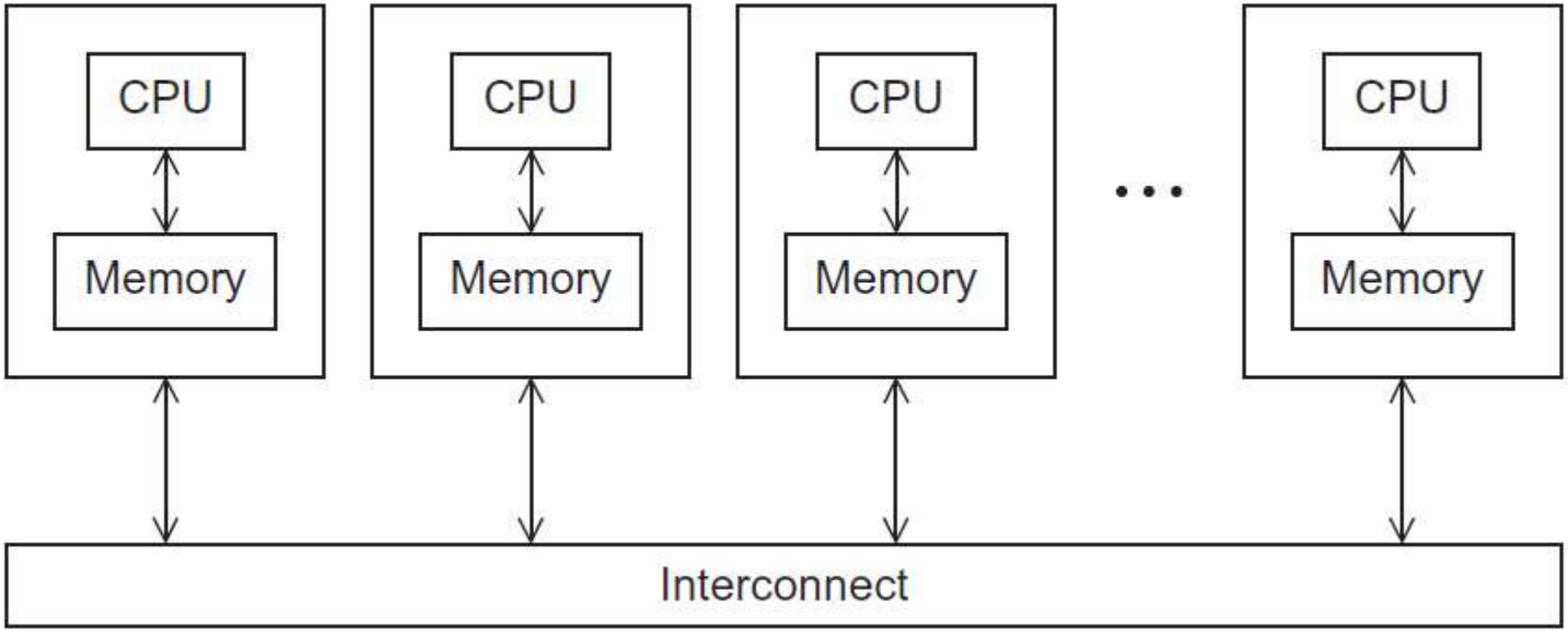
Hint: Banks!

Distributed Memory System

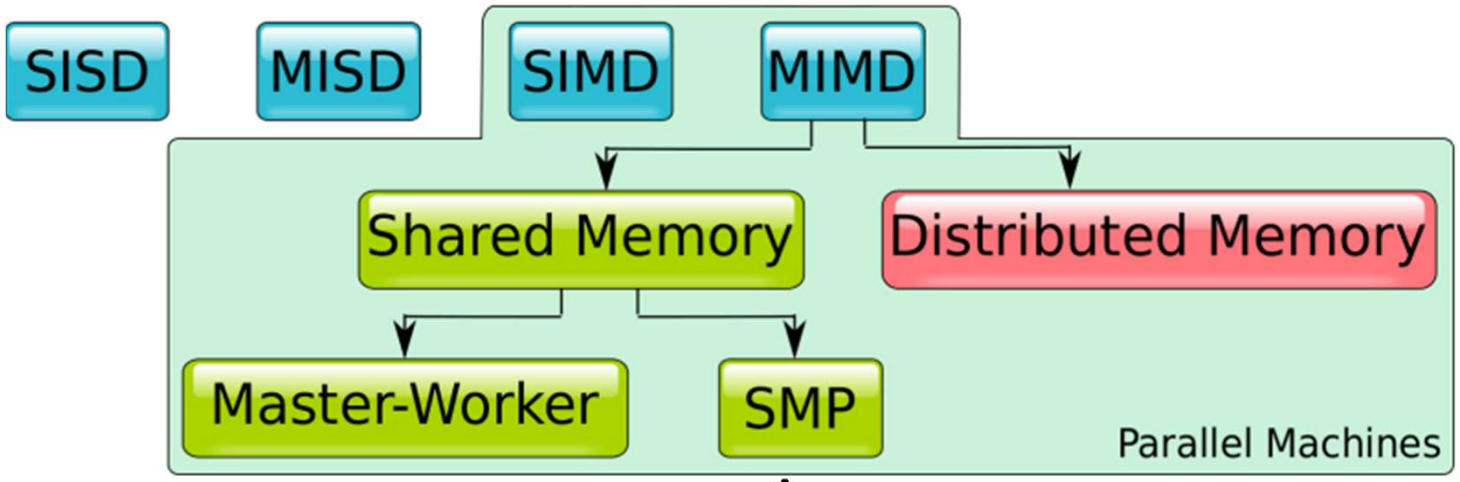
- Clusters A collection (cluster) of nodes
  - Connected by a interconnection network

- Nodes of a cluster are individual computation units.

Distributed Memory System



Let's summarize that:



One node is more important than the others.

All nodes are the same.

(SMP = Symmetric Multi-Processing)

A Brief discussion of Interconnection networks

- Affects performance of both distributed and shared memory systems.

- Two categories:
  - Shared memory interconnects
  - Distributed memory interconnects

Shared memory interconnects

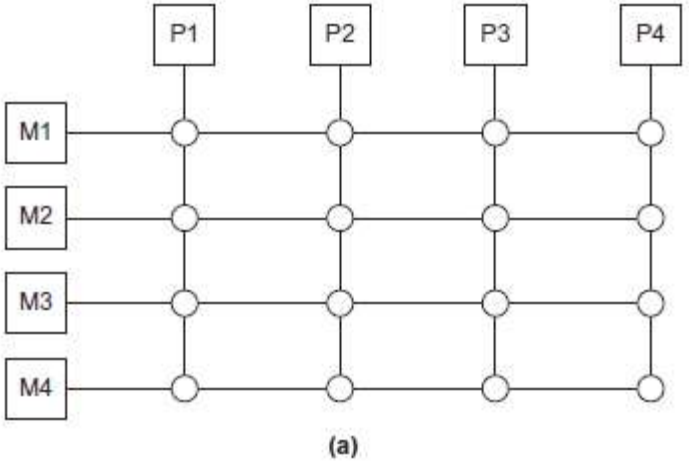
- Bus interconnect
  - A collection of parallel communication wires together with some hardware that controls access to the bus.
  - Communication wires are shared by the devices that are connected to it.
  - As the number of devices connected to the bus increases, contention for use of the bus increases, and performance decreases.

Shared memory interconnects

- Switched interconnect
  - Uses switches to control the routing of data among the connected devices.
  - Crossbar
    - Allows simultaneous communication among different devices.
    - Faster than buses.
    - But the cost of the switches and links is relatively high.

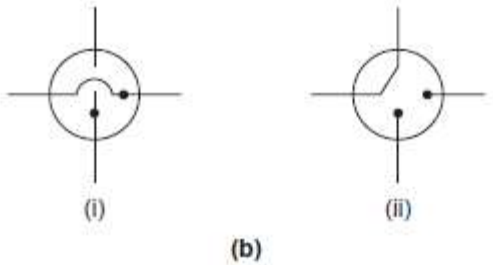
(a)

A crossbar switch connecting 4 processors (P_i) and 4 memory modules (M_j)

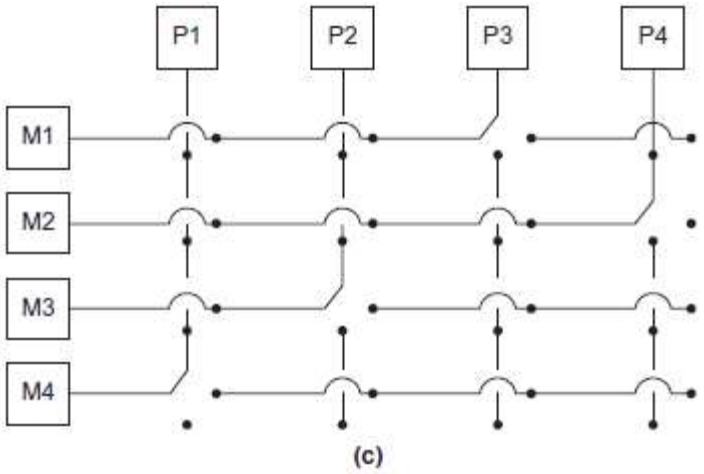


(b)

Configuration of internal switches in a crossbar



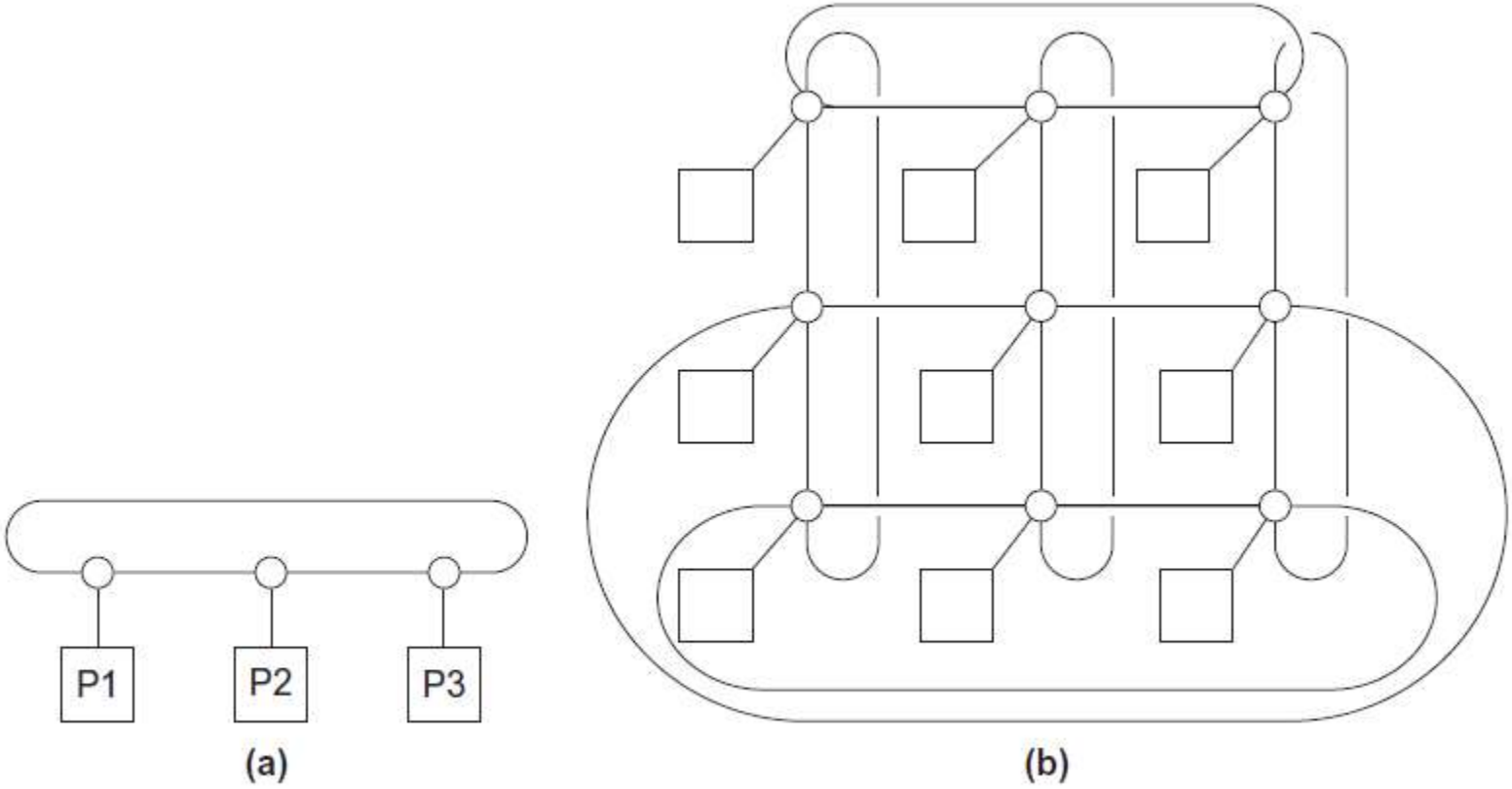
(c) Simultaneous memory accesses by the processors



Distributed memory interconnects

- Two groups
  - Direct interconnect
    - Each switch is directly connected to a processor memory pair, and the switches are connected to each other.
  - Indirect interconnect
    - Switches may not be directly connected to a processor.

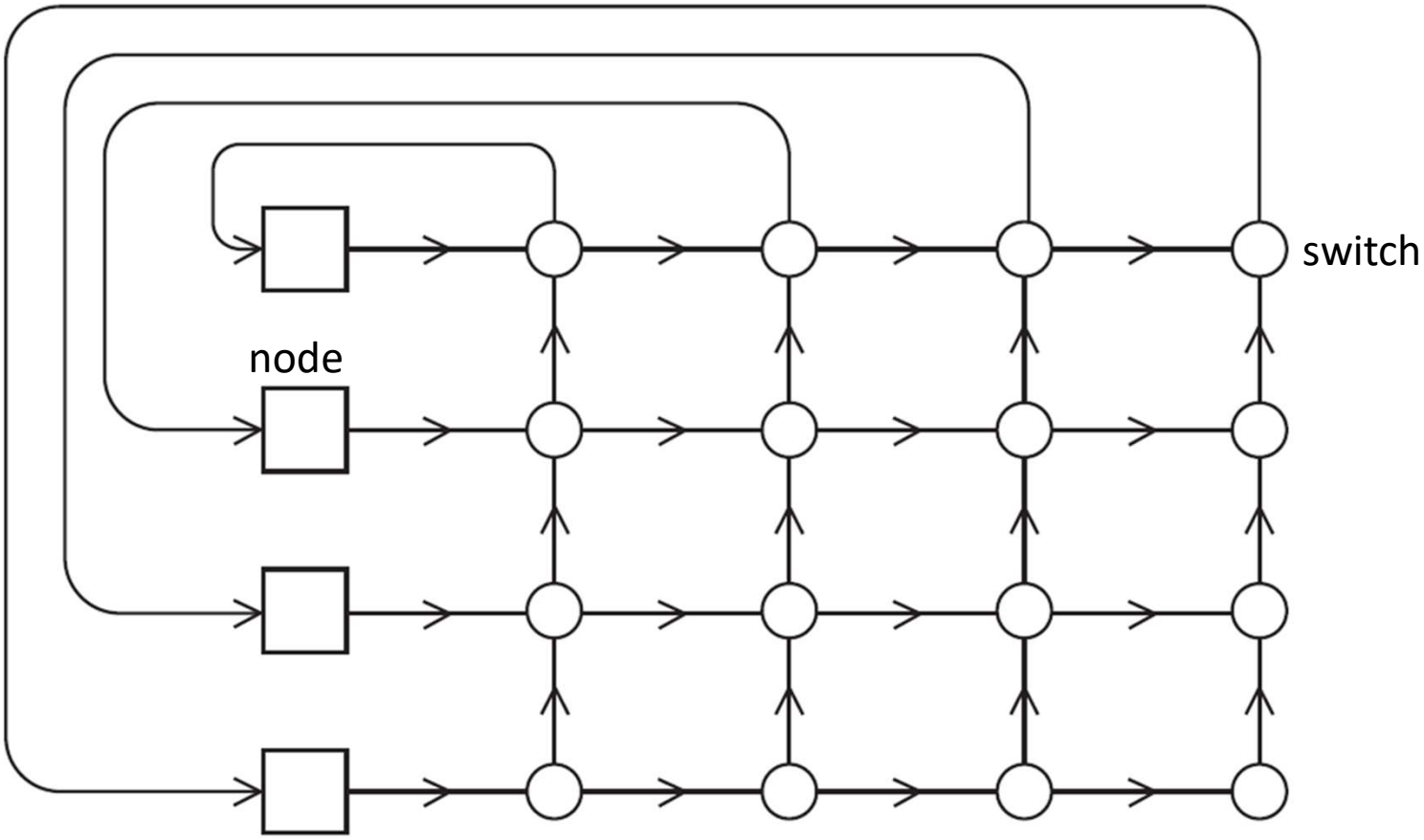
Direct Interconnect



ring

toroidal mesh

Indirect Interconnect

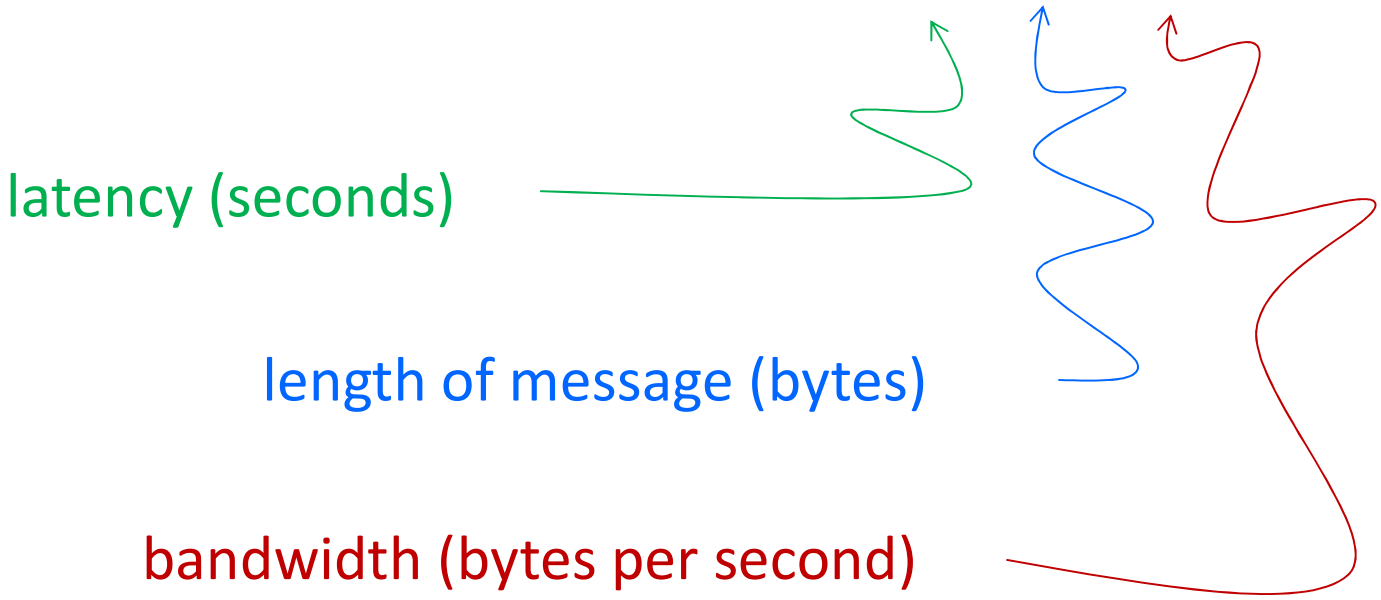


Crossbar Interconnect

Some Definitions Related to Interconnection Networks

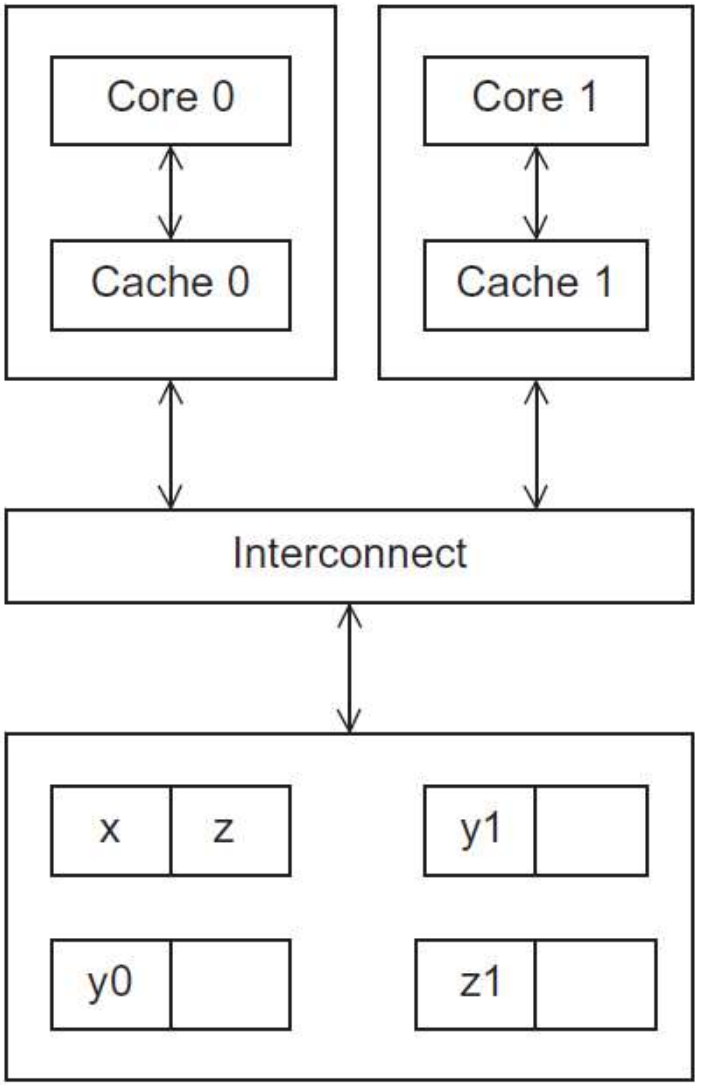
- Any time data is transmitted, we're interested in how long it will take for the data to reach its destination.
- Latency
  - The time that elapses between the source's beginning to transmit the data and the destination's starting to receive the first byte.
- Bandwidth
  - The rate at which the destination receives data after it has started to receive the first byte.

Message transmission time = l + n / b



Cache coherence

- Programmers have no control over caches and when they get updated.



Cache coherence

y0 privately owned by Core 0
y1 and z1 privately owned by Core 1

x = 2; /* shared variable */

Time	Core 0	Core 1
0	`y0 = x;`	`y1 = 3*x;`
1	`x = 7;`	Statement(s) not involving x
2	Statement(s) not involving x	`z1 = 4*x;`

y0 eventually ends up = 2
y1 eventually ends up = 6
z1 = ???

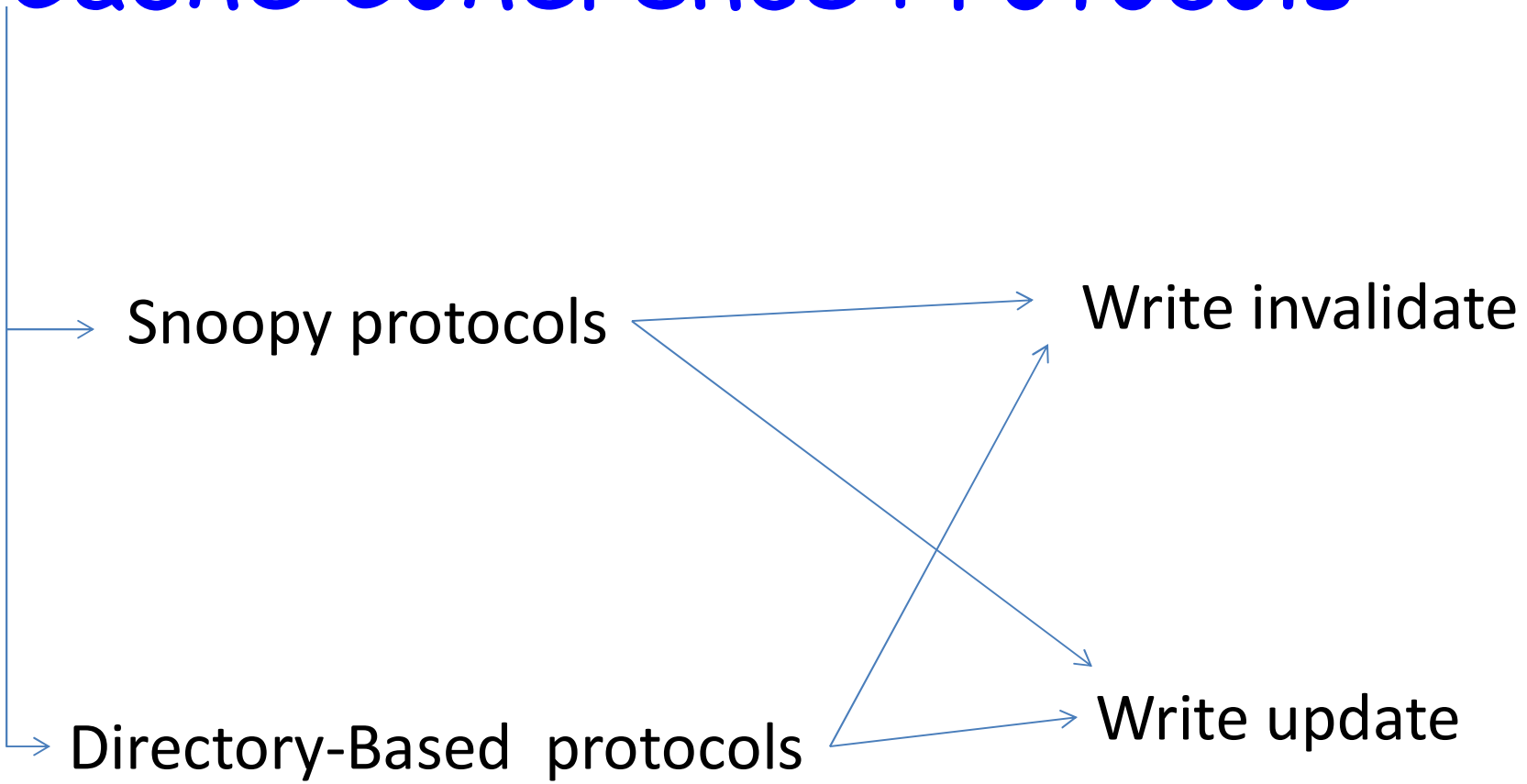
Snooping Cache Coherence

- The cores share a bus .
- Any signal transmitted on the bus can be "seen" by all cores connected to the bus.
- When core 0 updates the copy of x stored in its cache it also broadcasts this information across the bus.
- If core 1 is "snooping" the bus, it will see that x has been updated and it can mark its copy of x as invalid.

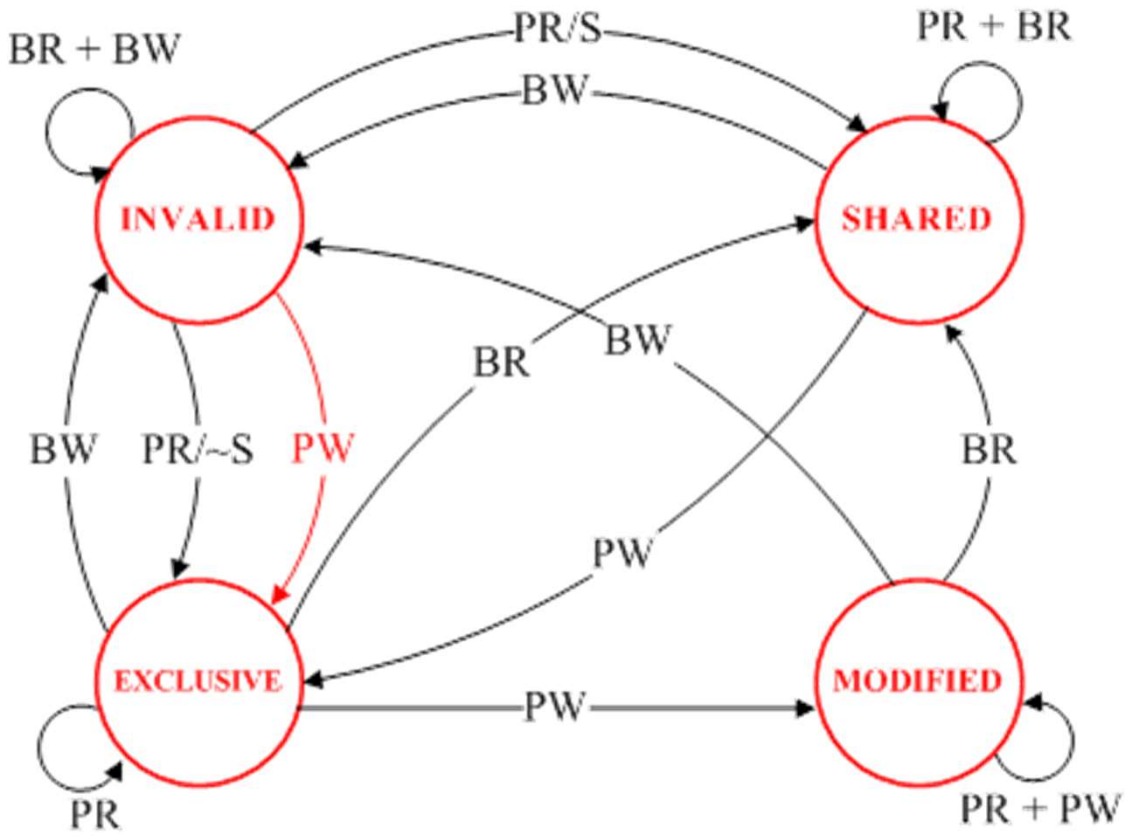
Directory Based Cache Coherence

- Uses a data structure called a directory that stores the status of each cache line.
- When a variable is updated, the directory is consulted, and the cache controllers of the cores that have that variable's cache line in their caches are invalidated.

Cache Coherence Protocols



Example: MESI Protocol



PR = processor read
PW = processor write
S/~S = shared/NOT shared

BR = observed bus read
BW = observed bus write

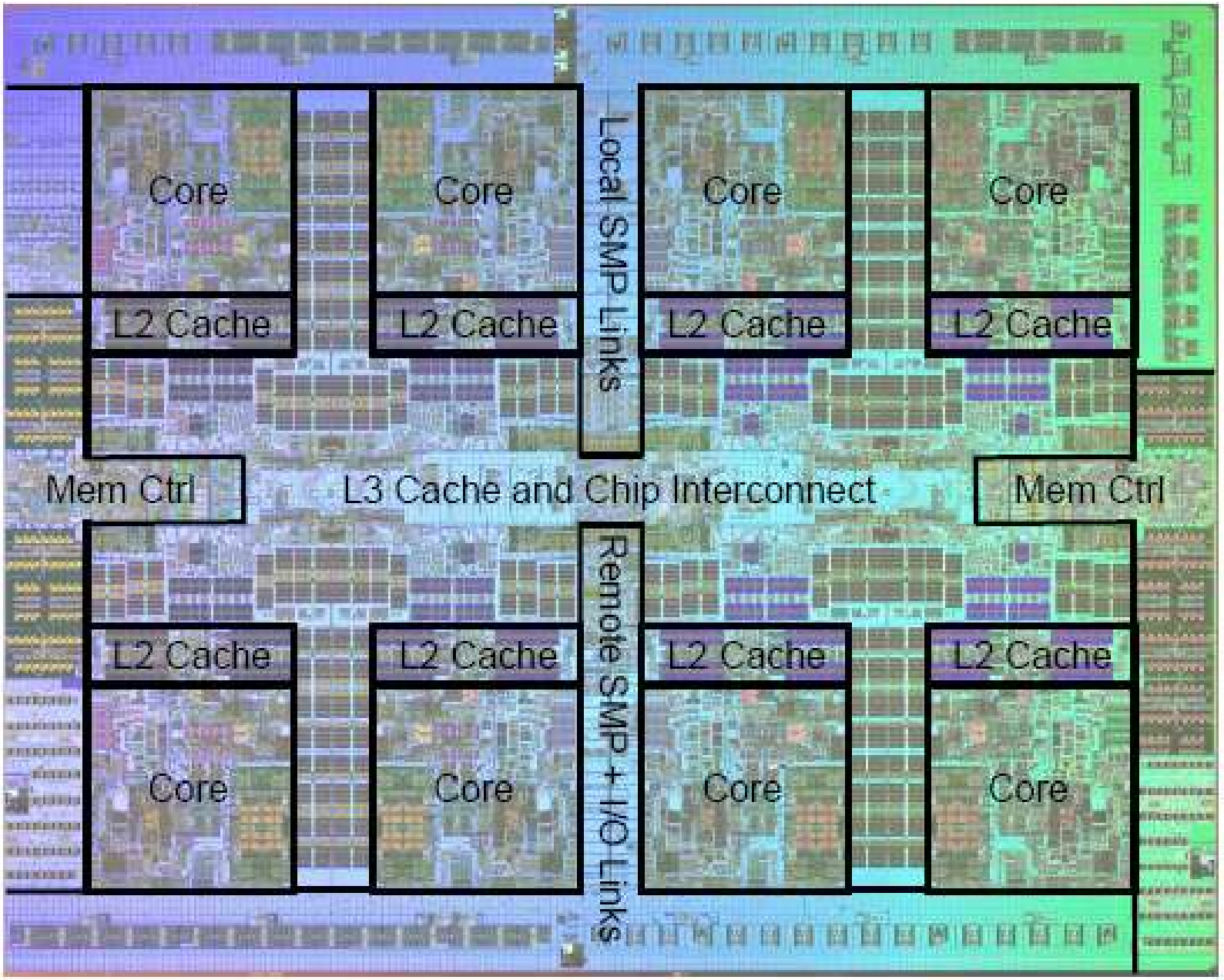
Examples from Real Life

IBM Power 7

- Supports global shared memory space for POWER7 clusters
  - So you can program a cluster as if it were a single system
- Design for power-efficiency
- ~1.2B Transistors
- Up to 8 cores and 4-way SMT
- TurboCore mode that can turn off half of the cores from an eight-core processor, but those 4 cores have access to all the memory controllers and L3 cache at increased clock speeds.
- 3.0 – 4.25 GHz clock speed

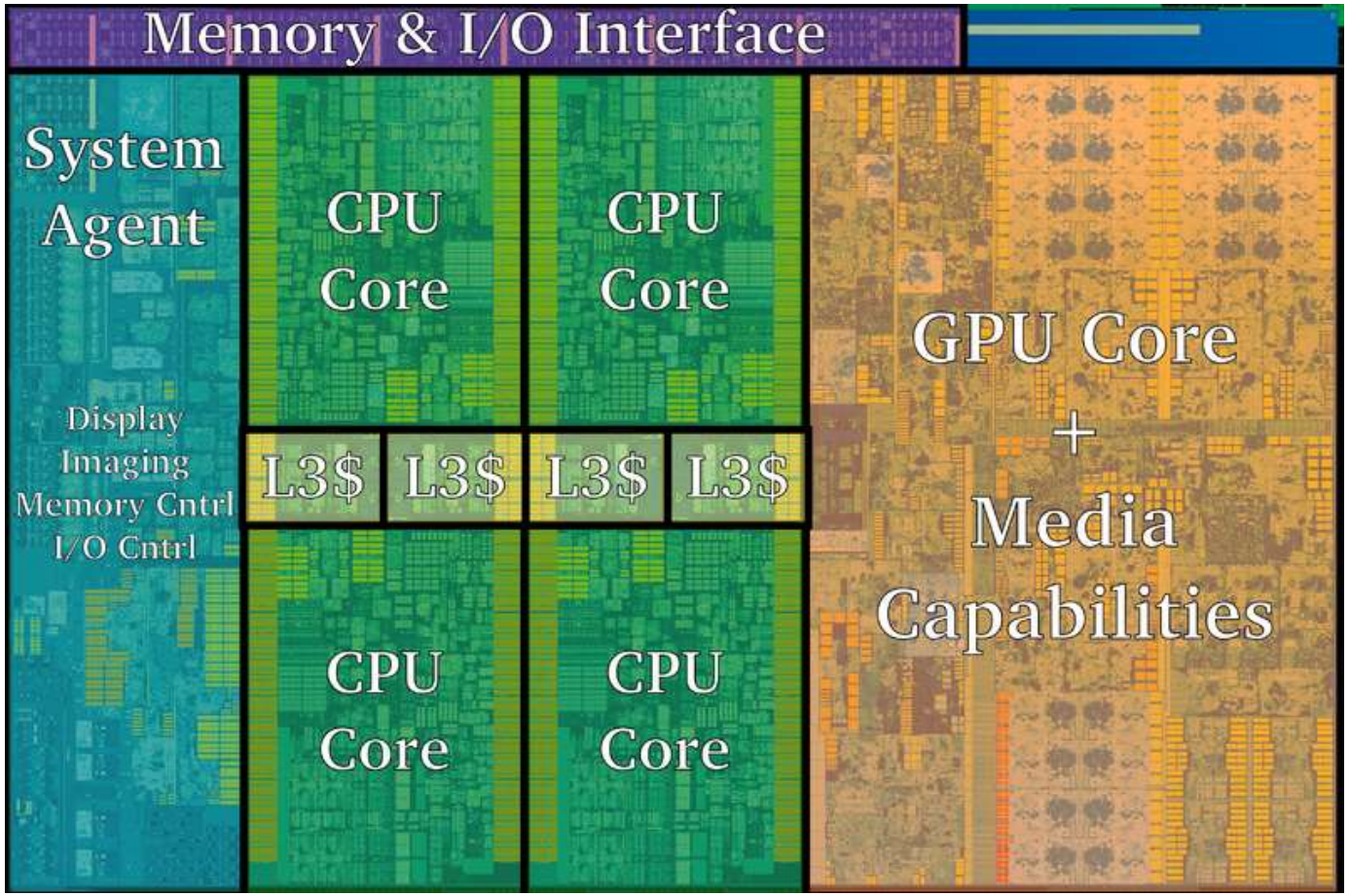
IBM Power 7: Cache Hierarchy

- 32KB DL1 and IL1 per core
- 256KB L2 per core
- eDRAM L3 4MB per core (total of 32MB)
  - Very flexible design for L3



Source: Slides from Joel M. Tendler from IBM

Intel Kaby Lake



source: wikichip

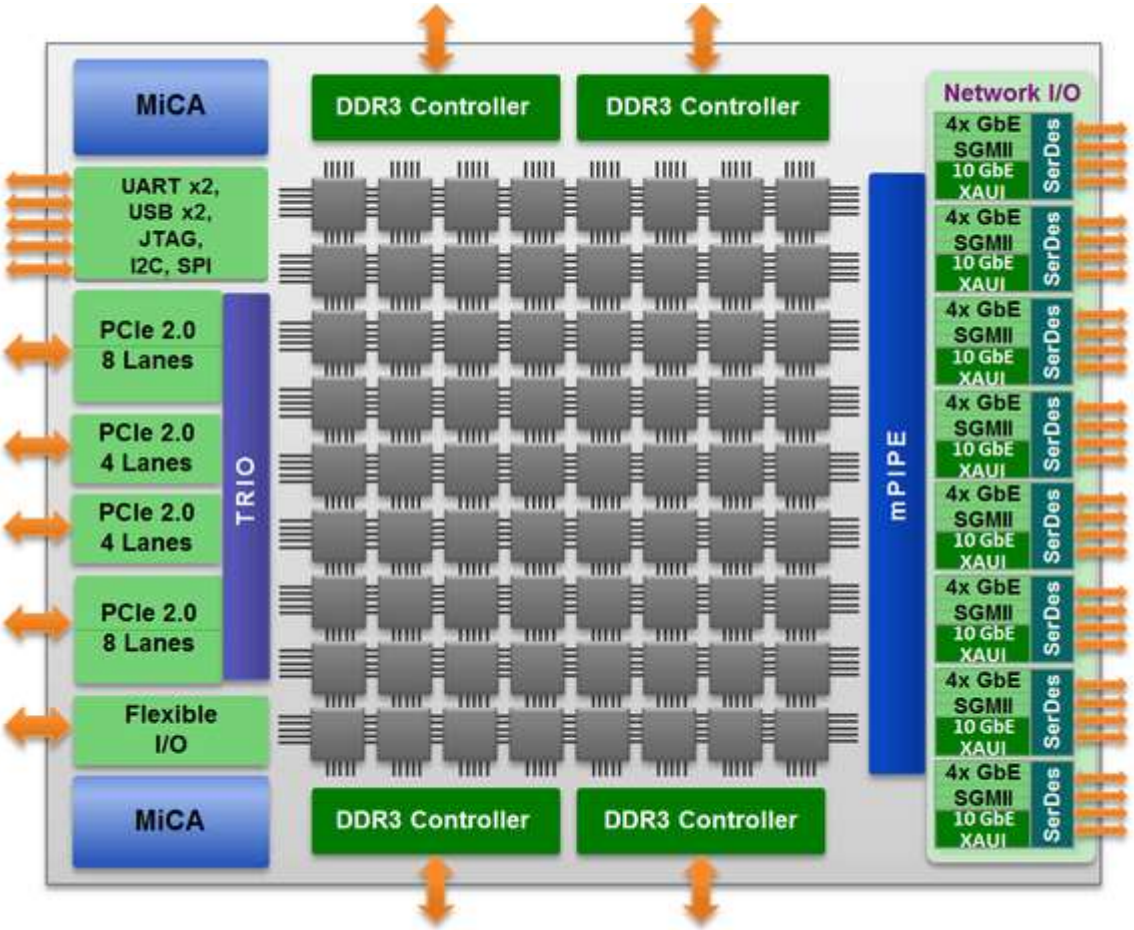
Intel Kaby Lake

- 14 nm process technology
- Hyperthreading technology: 2 threads/core
- Is the optimization phase of the newer Intel's "process-architecture-optimization" model
- L1I Cache: 32 KB 8-way set associative - 64 B line size - Write-back policy - shared by the two threads, per core
- L1D Cache: 32 KB 8-way set associative - 64 B line size - shared by the two threads, per core - 64 Bytes/cycle load bandwidth - 32 Bytes/cycle store bandwidth - Write-back policy
- L2 Cache: unified, 256 KB 4-way set associative - 64B/cycle bandwidth to L1$ - Write-back policy
- L3 Cache: Up to 2 MB Per core, shared across all cores - Up to 16-way set associative - Write-back policy
- L4 Cache (if any): 64 MB - Per package
- On-chip GPU included

TILERA: Many-core chips

- Released in August 2007.
- TILE64 offered 64 cores arranged in a 2-D grid.
- TILE-Gx8072 has 72 cores with a 2-D grid of communication channels called the iMesh Interconnect.
  - iMesh comes with five independent mesh networks that offer an aggregate bandwidth exceeding 110 Tbps.
- Each core has 32KB data and 32KB instruction L1 caches and 256KB L2 cache. A 18MB L3 coherent cache is shared between the cores. Access to the main RAM is done via four DDR3 controllers.

TILE-Gx8072 Block Diagram



How About Supercomputers?

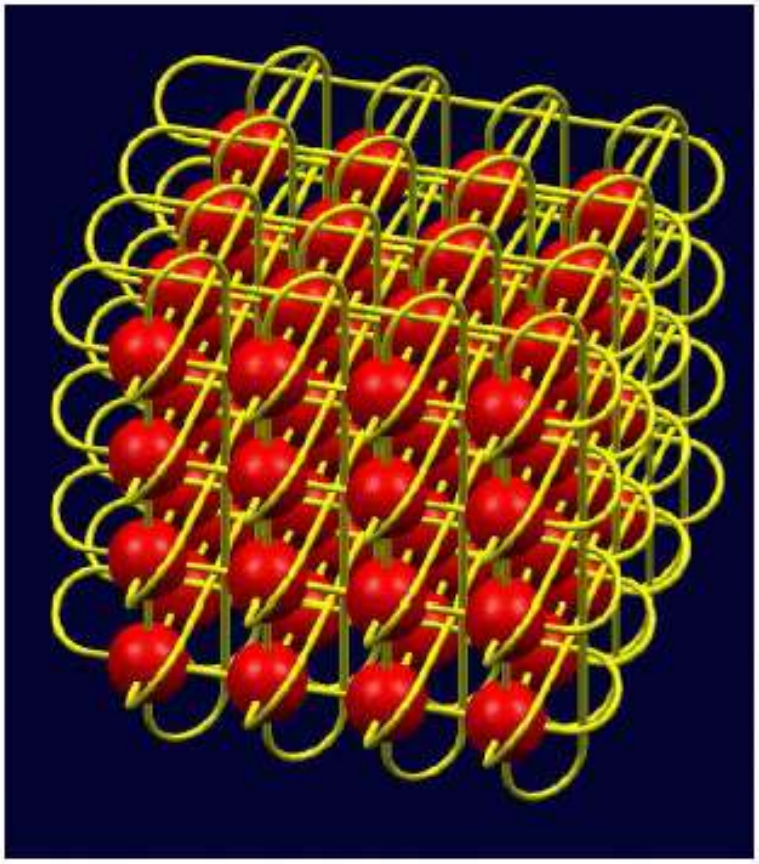
http://www.top500.org/

IBM BlueGene/L

- Started December 1999
- Main goal: to build a petaflop/s scale supercomputer to attack science problems such as protein folding.
  (Now we want exascale!!)
- Strategy: Massive collection of low-power CPUs instead of a moderate-sized collection of high-power CPUs.
- BlueGene is a family of supercomputers.
  - BlueGene/L is the first generation
  - BlueGene/P is the petaflop generation
  - BlueGene/Q is the third generation

IBM BlueGene/L

- A large number of nodes (65,536)
  - Low-power (20W) nodes for density
  - High floating-point performance
  - System-on-a-chip technology
- Nodes interconnected as 64x32x32 three-dimensional torus
  - Easy to build large systems, as each node connects only to six nearest neighbors – full routing in hardware
  - Bisection bandwidth per node is proportional to n^2/n^3
  - Auxiliary networks for I/O and global operations
- Applications consist of multiple processes with message passing
  - Strictly one process/node
  - Minimum OS involvement and overhead



- 65,536 dual-processor compute nodes
  - 700MHz IBM PowerPC 440 processors
  - 512 MB memory per compute node, 16 TB in entire system.
  - 800 TB of disk space
- 2,500 square feet

Conclusions

- The trend now is:
  - More cores per chip
  - Non-bus interconnect
  - NUMA and NUCA (Non-Uniform Memory/Cache Access)
- Communication and memory access are the two most expensive operations, NOT computations