Computer Architecture

Ch6 – Parallel Computer Architecture

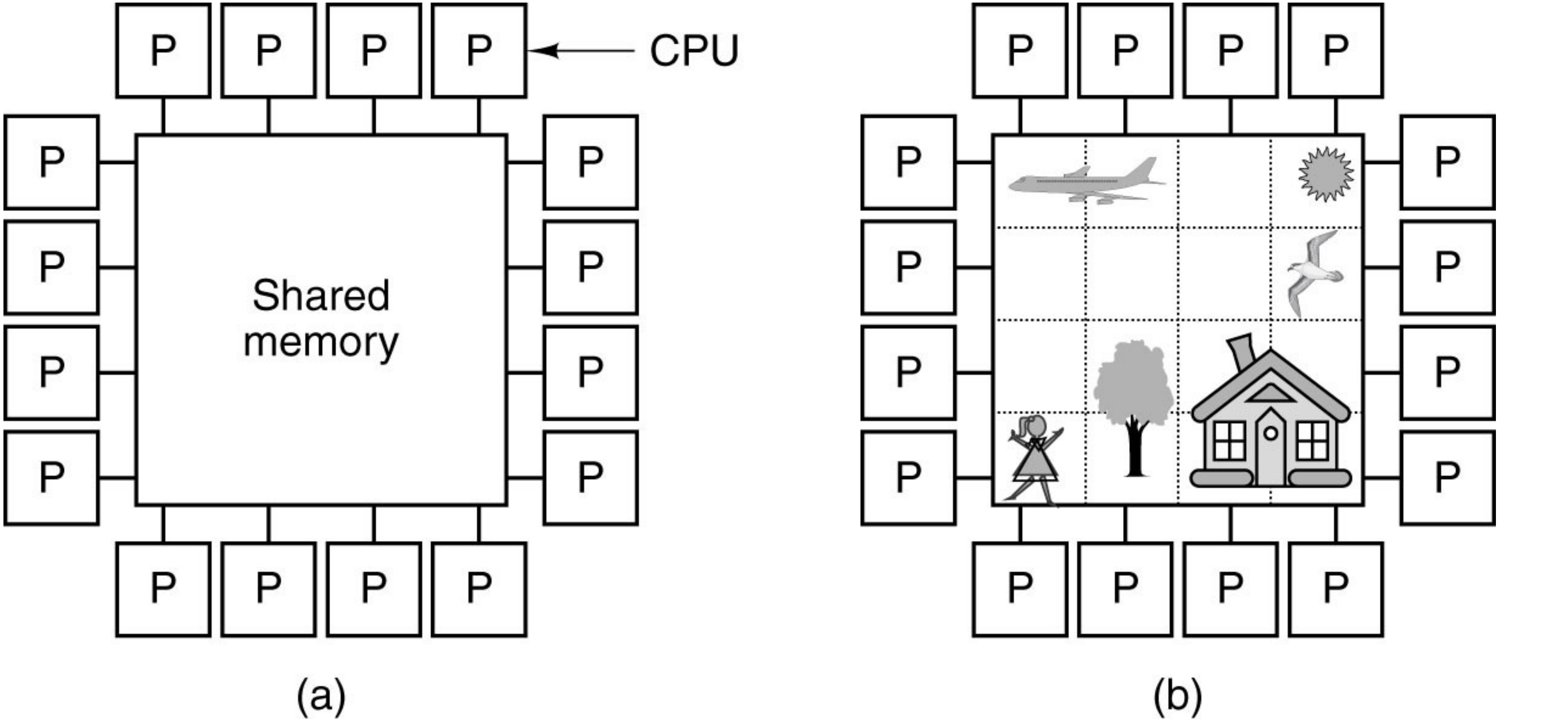
Nguyễn Quốc Đính, FIT – IUH

HCMC, Aug 2015

Significant Architecture Distinction

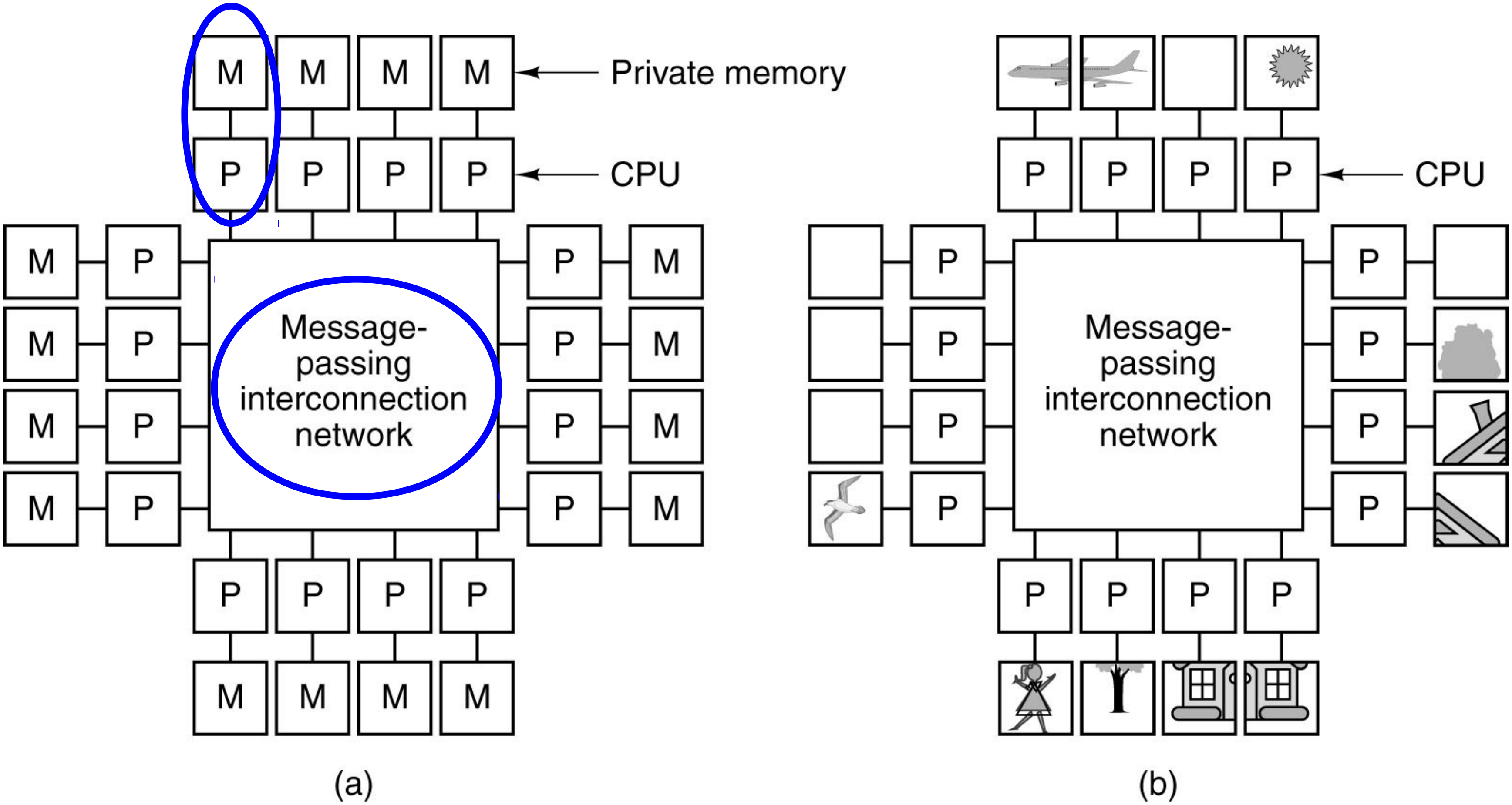
- Multiprocessor
  - Many parallel processors
  - Shared memory machines
- Multicomputer
  - Many parallel computers
  - Message passing machines (processors pass messages to share data)

Multiprocessors



(a) A multiprocessor with 16 CPUs sharing a common memory.
(b) An image partitioned into 16 sections, each being analyzed by a different CPU.

Multicomputers


M
M
M
M
Private memory
P
P
P
P
CPU
Message-passing interconnection network
(a)
P
CPU
Message-passing interconnection network
(b)

(a) A multicomputer with 16 CPUs, each with its own private memory.
(b) The bit-map image of Fig. 8-17 split up among the 16 memories.

Multicomputers (cont'd)

- Communication between processes often use software primitives such as `send` and `receive`.
- Correctly dividing up the data and placing them in the optimal locations is a major issue on a multicomputer
- Large multicomputers are much simpler and cheaper to build than multiprocessors with the same number of CPUs

The TOP500

- The TOP500 project started in 1993.
- The best performance on the *Linpack Benchmark* is used as performance measure for ranking the computer systems

  http://www.netlib.org/utk/people/JackDongarra/PAPERS/hpl.pdf
- Overview of recent supercomputer

  http://www.top500.org/static/lists/2011/11/TOP500_201111_Poster.png

TOP500® NOVEMBER 2011

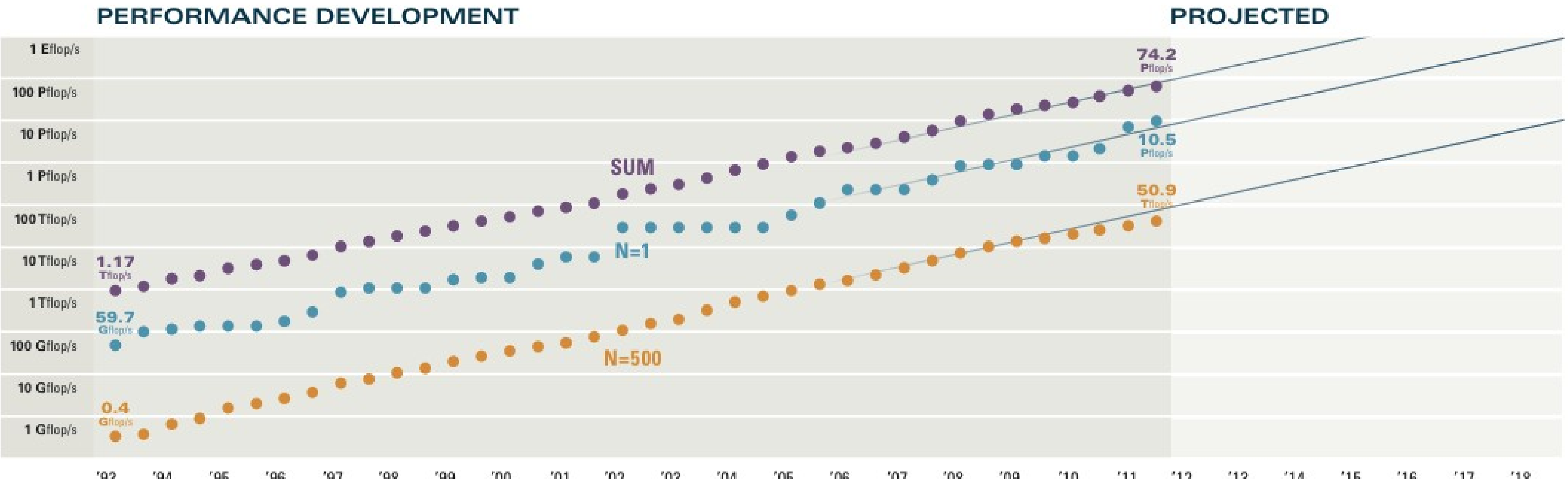
PRESENTED BY
UNIVERSITY OF MANNHEIM

ICL UT INNOVATIVE COMPUTING LABORATORY THE UNIVERSITY of TENNESSEE

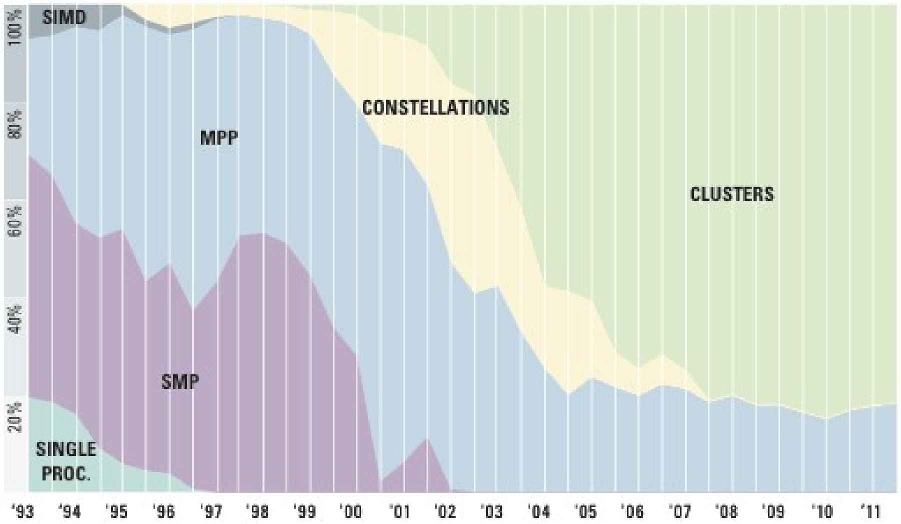
BERKELEY LAB Lawrence Berkeley National Laboratory

FIND OUT MORE AT www.top500.org

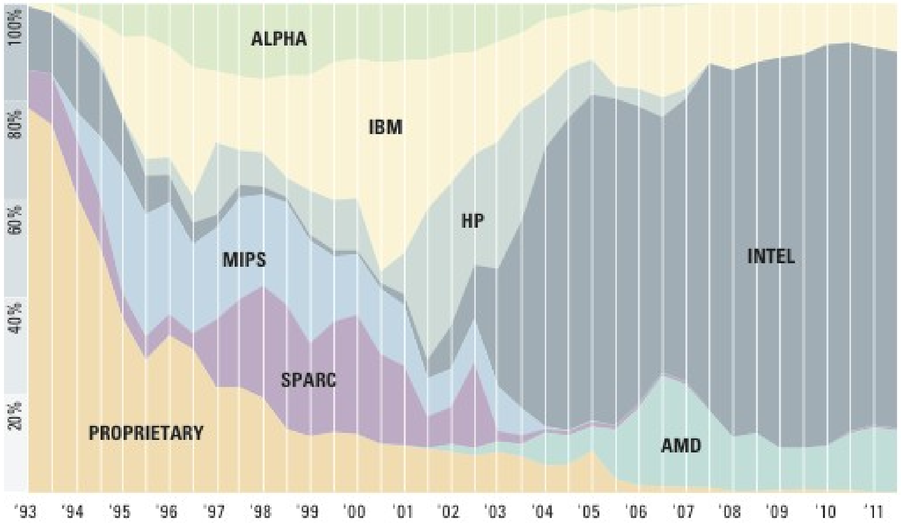
	NAME/MANUFACTURER/COMPUTER	SITE	COUNTRY	CORES	Rmax Pflop/s
1	**K computer** SPARC64 VIIIfx 2.0GHz, Tofu interconnect	RIKEN	Japan	705,024	10.5
2	**Tianhe-1A** 6-core Intel X5670 2.93 GHz + Nvidia M2050 GPU w/custom interconnect	NUDT/NSCC/Tianjin	China	186,368	2.57
3	**Jaguar** Cray XT-5 6-core AMD 2.6 GHz w/custom interconnect	DOE/OS/ORNL	USA	224,162	1.76
4	**Nebulae** Dawning TC3600 Blade Intel X5650 2.67 GHz, NVidia Tesla C2050 GPU w/ Iband	NSCS	China	120,640	1.27
5	**Tsubame 2.0** HP Proliant SL390s G7 nodes (Xeon X5670 2.93GHz) , NVIDIA Tesla M2050 GPU w/Iband	TiTech	Japan	73,278	1.19



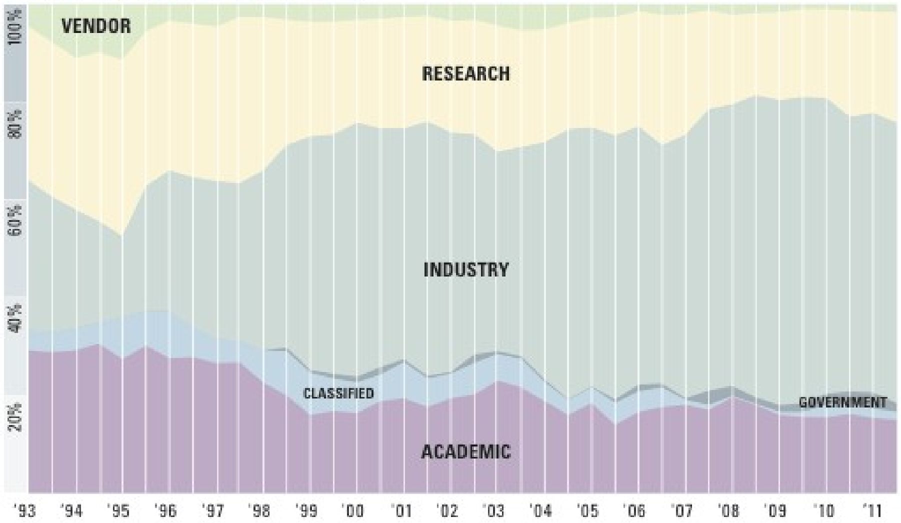
ARCHITECTURES



CHIP TECHNOLOGY



INSTALLATION TYPE



HPLINPACK

A Portable Implementation of the High Performance Linpack Benchmark for Distributed Memory Computers

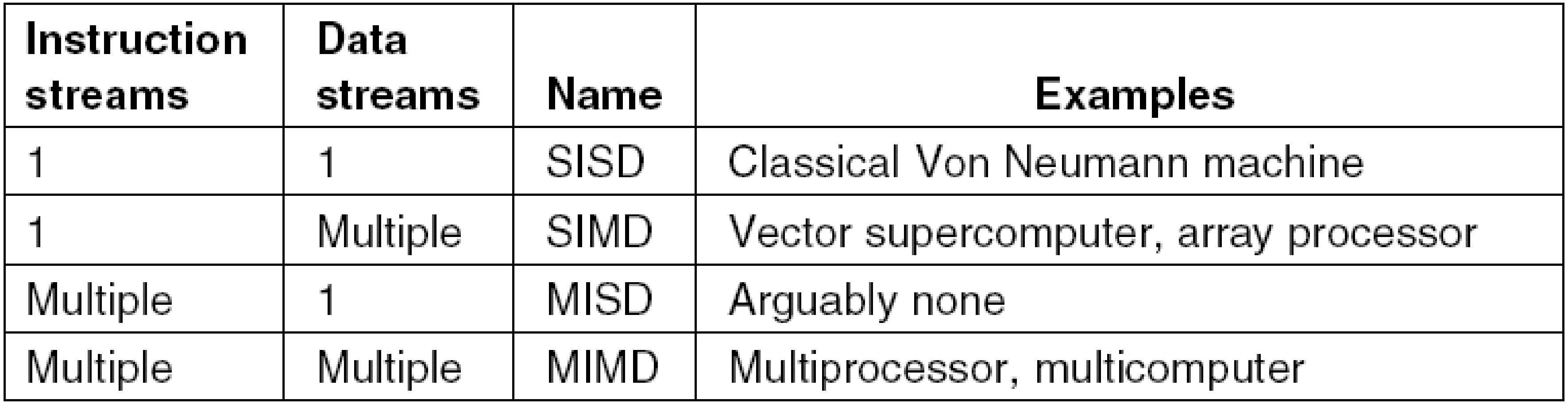
Algorithm: recursive panel factorizations, multiple lookahead depths, bandwidth reducing swapping

Easy to install, only needs MPI + BLAS or VSIPL

Highly scalable and efficient from the smallest cluster to the largest supercomputers in the world

FIND OUT MORE AT **http://icl.eecs.utk.edu/hpl/**

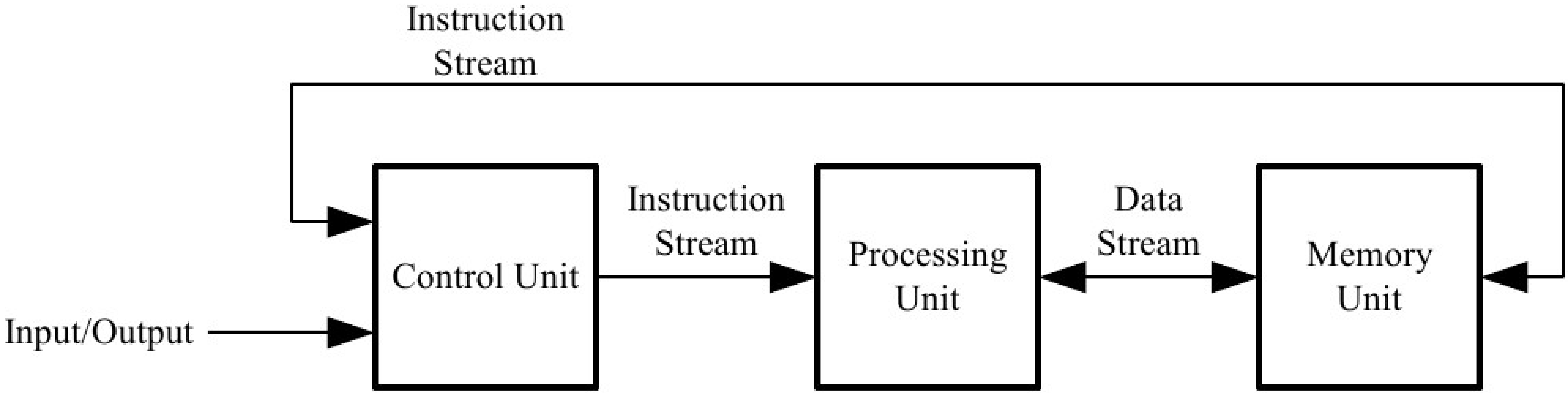
Taxonomy of Parallel Computers



Instruction streams	Data streams	Name	Examples
1	1	SISD	Classical Von Neumann machine
1	Multiple	SIMD	Vector supercomputer, array processor
Multiple	1	MISD	Arguably none
Multiple	Multiple	MIMD	Multiprocessor, multicomputer

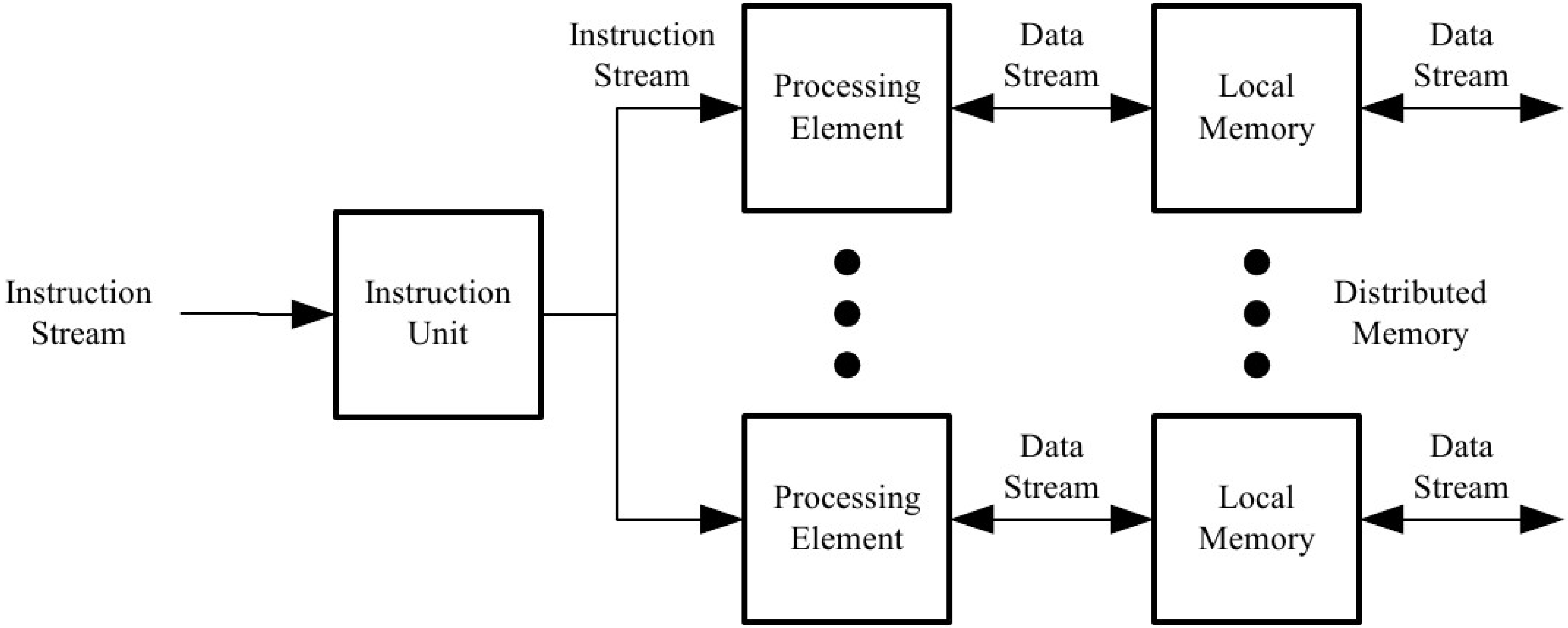
Flynn’s taxonomy of parallel computers.

Instruction streams	Data streams	Name	Examples
1	1	SISD	Classical Von Neumann machine
1	Multiple	SIMD	Vector supercomputer, array processor
Multiple	1	MISD	Arguably none
Multiple	Multiple	MIMD	Multiprocessor, multicomputer



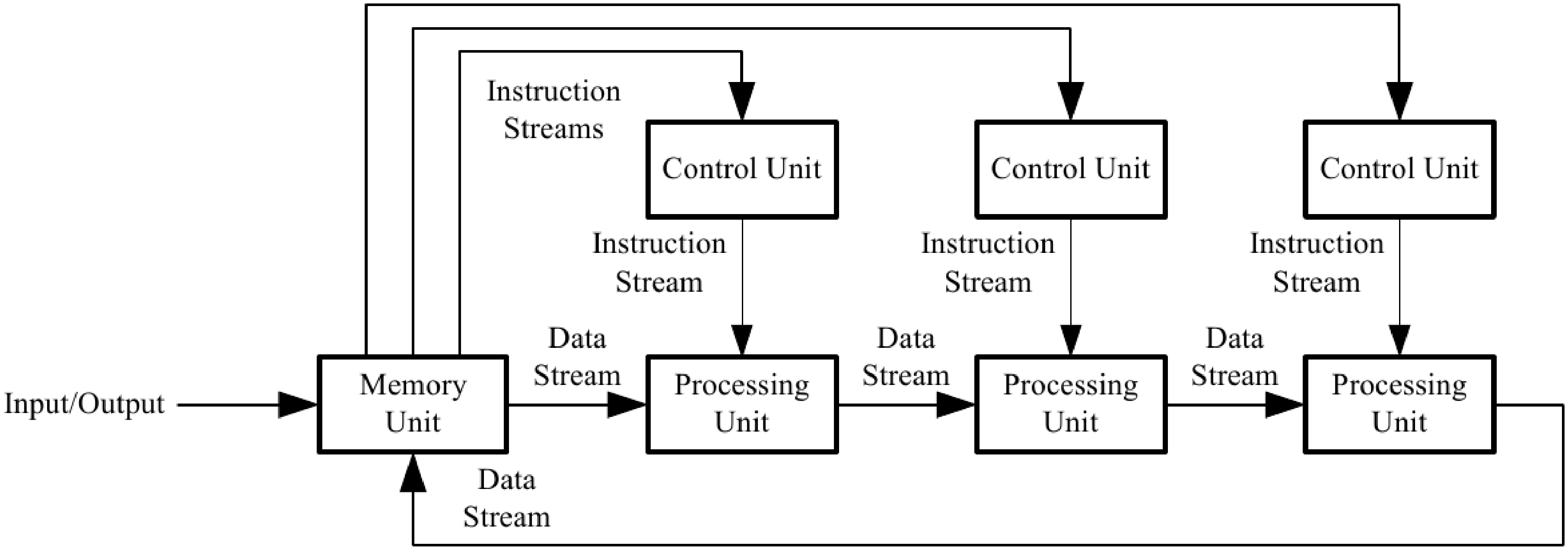
Single Instruction, Single Data (SISD)
Uniprocessor

Instruction streams	Data streams	Name	Examples
1	1	SISD	Classical Von Neumann machine
1	Multiple	SIMD	Vector supercomputer, array processor
Multiple	1	MISD	Arguably none
Multiple	Multiple	MIMD	Multiprocessor, multicomputer



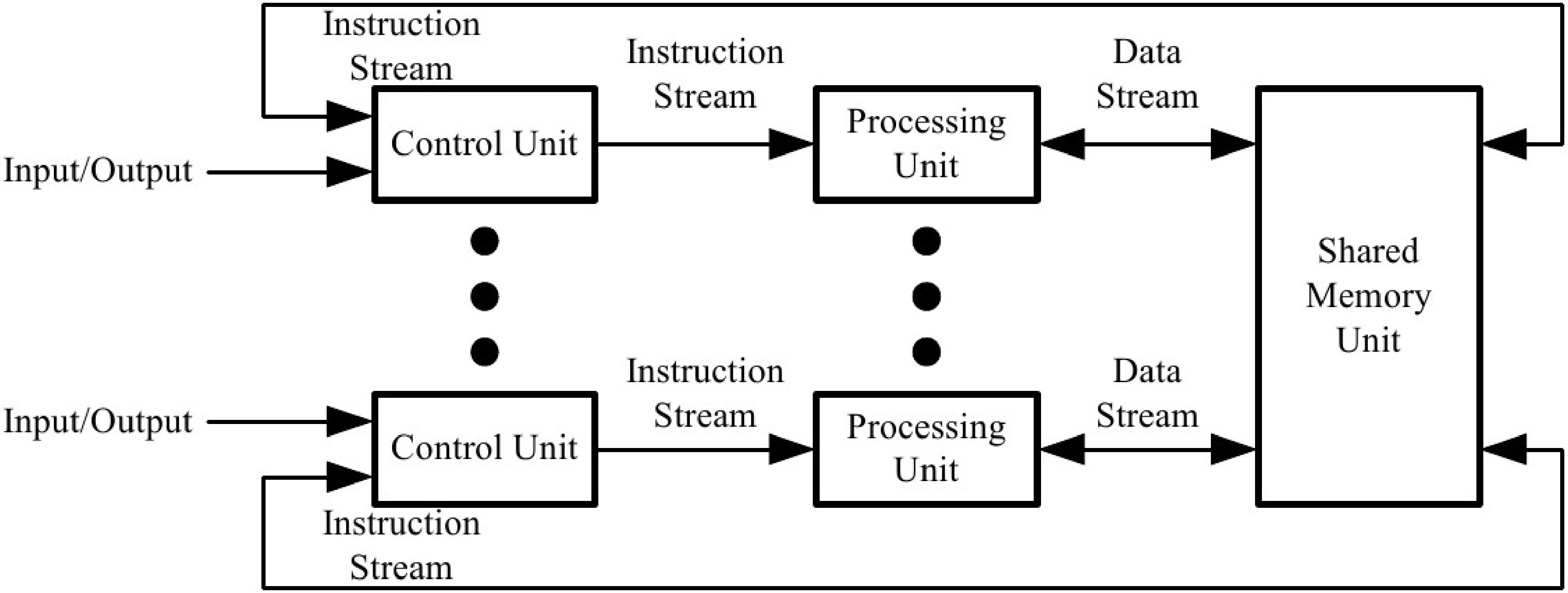
Single Instruction, Multiple Data (SIMD)
Vector Processor, Multiprocessor

Instruction streams	Data streams	Name	Examples
1	1	SISD	Classical Von Neumann machine
1	Multiple	SIMD	Vector supercomputer, array processor
Multiple	1	MISD	Arguably none
Multiple	Multiple	MIMD	Multiprocessor, multicomputer



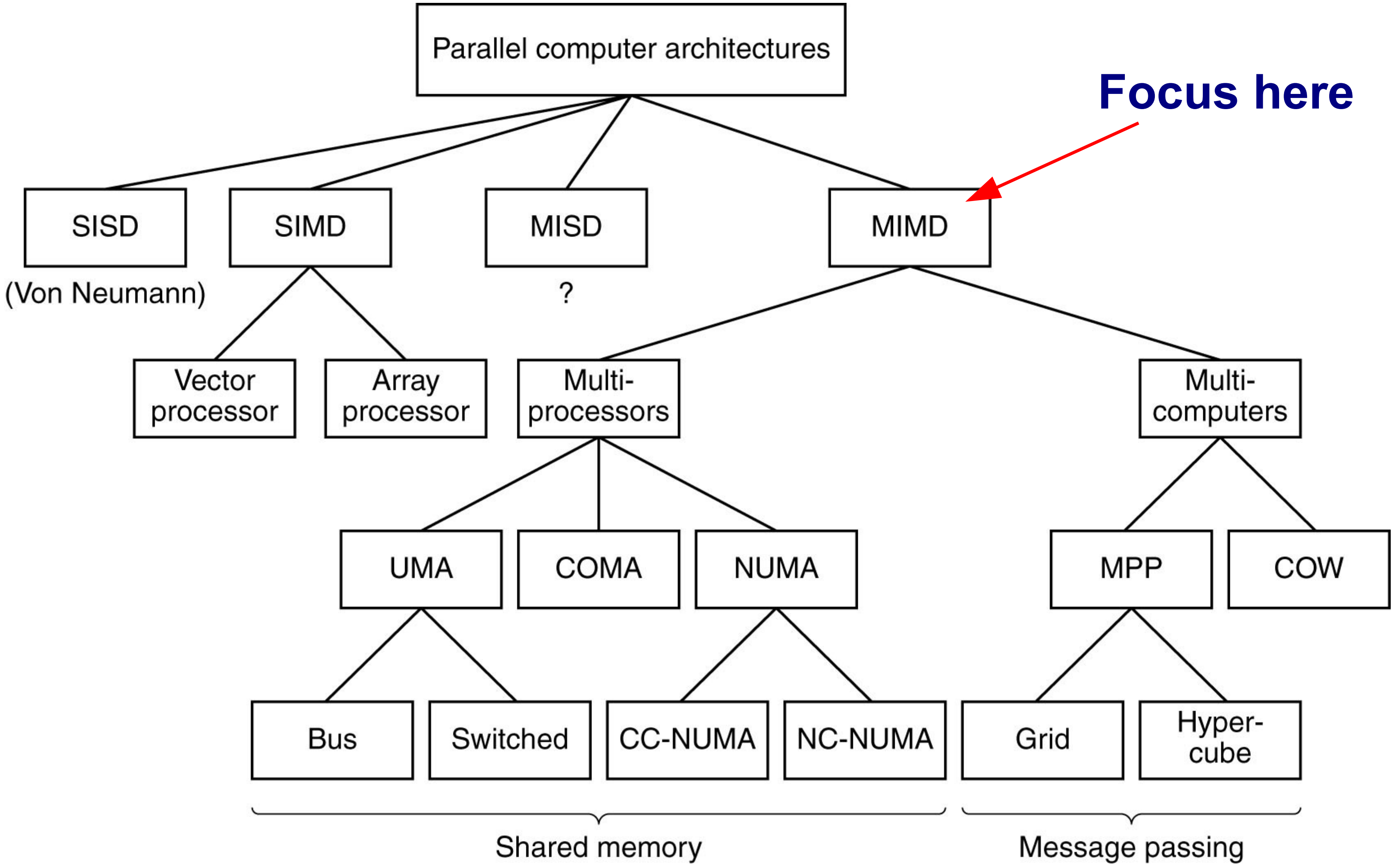
Multiple Instruction, Single Data (MISD)
Pipelined Array Processor, Systolic Array

Instruction streams	Data streams	Name	Examples
1	1	SISD	Classical Von Neumann machine
1	Multiple	SIMD	Vector supercomputer, array processor
Multiple	1	MISD	Arguably none
Multiple	Multiple	MIMD	Multiprocessor, multicomputer



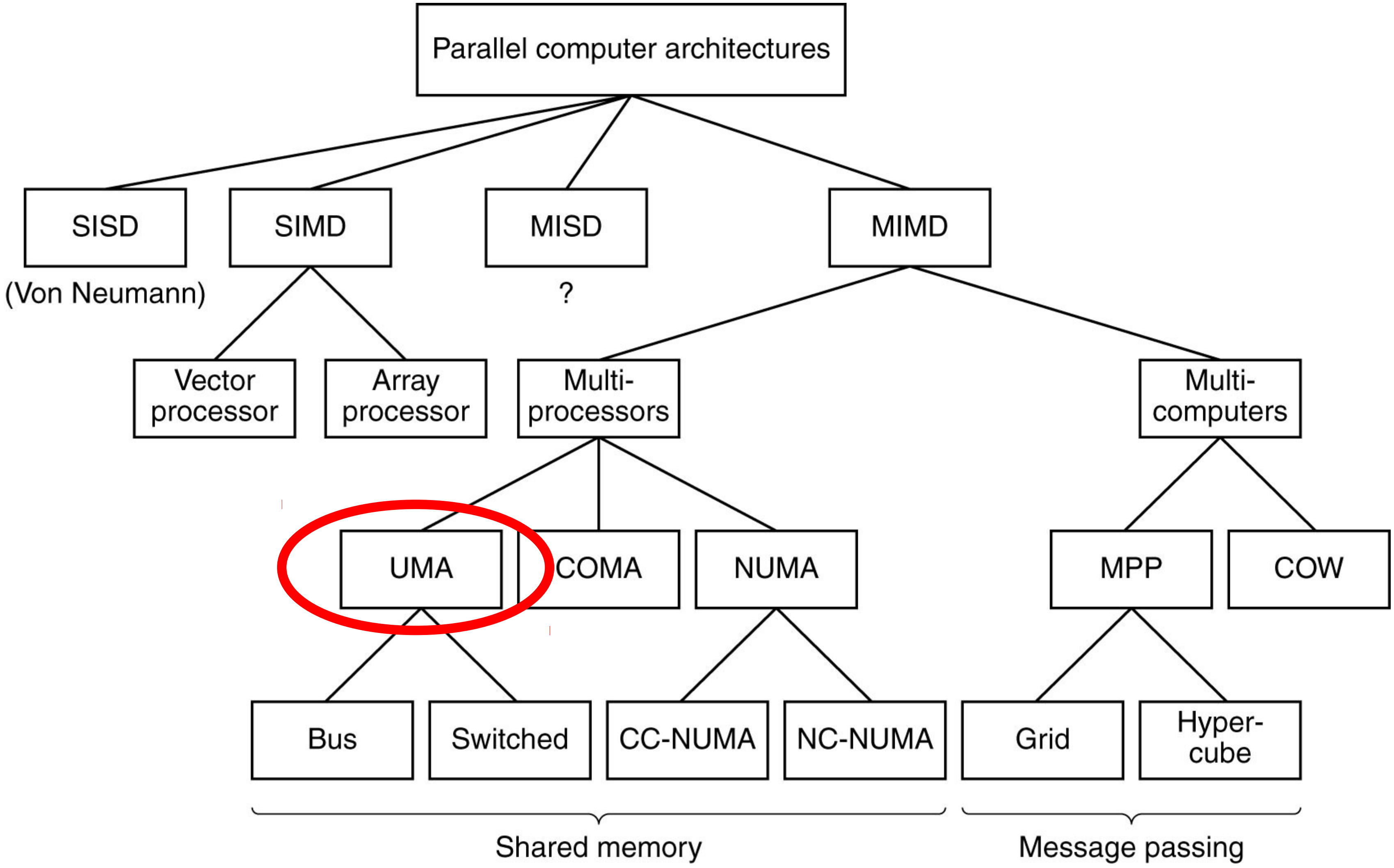
Multiple Instruction, Multiple Data (MIMD)
Multiprocessor, Multicomputer

Expanded Computer Taxonomy



A taxonomy of parallel computers.

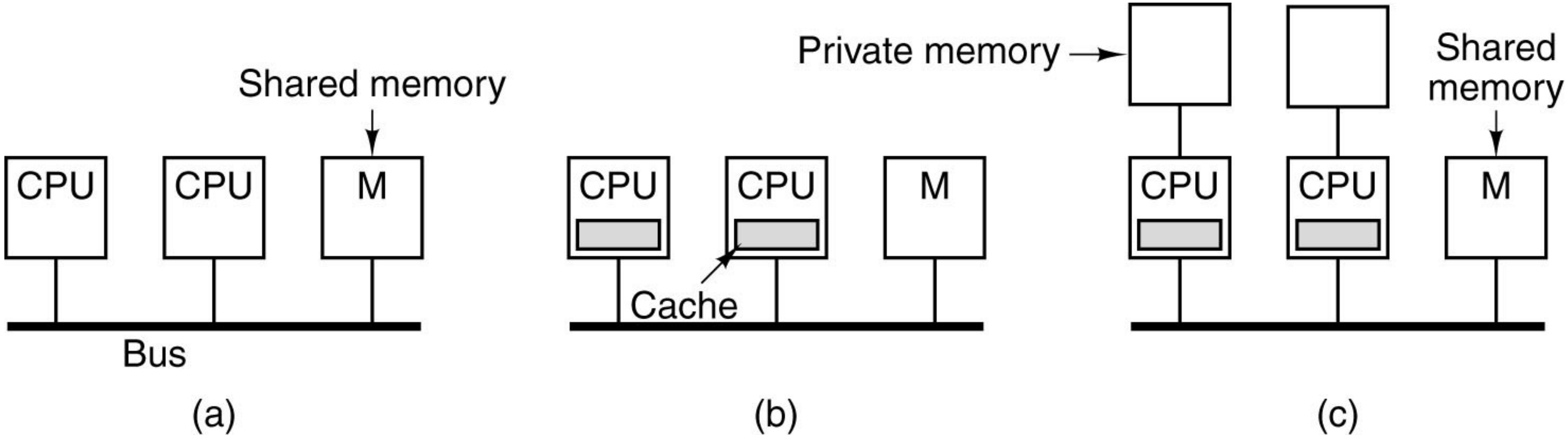
Expanded Computer Taxonomy



A taxonomy of parallel computers.

UMA Symmetric Multiprocessor Architecture

- (a) limited by the bandwidth of the bus, and most CPU will be idle most of the time
- (b) caching is a big win
- (c) each CPU has only cache, but private memory.
  - Compiler should place all the program text, strings, constants, read-only data, stacks, and local variable in the private memories.



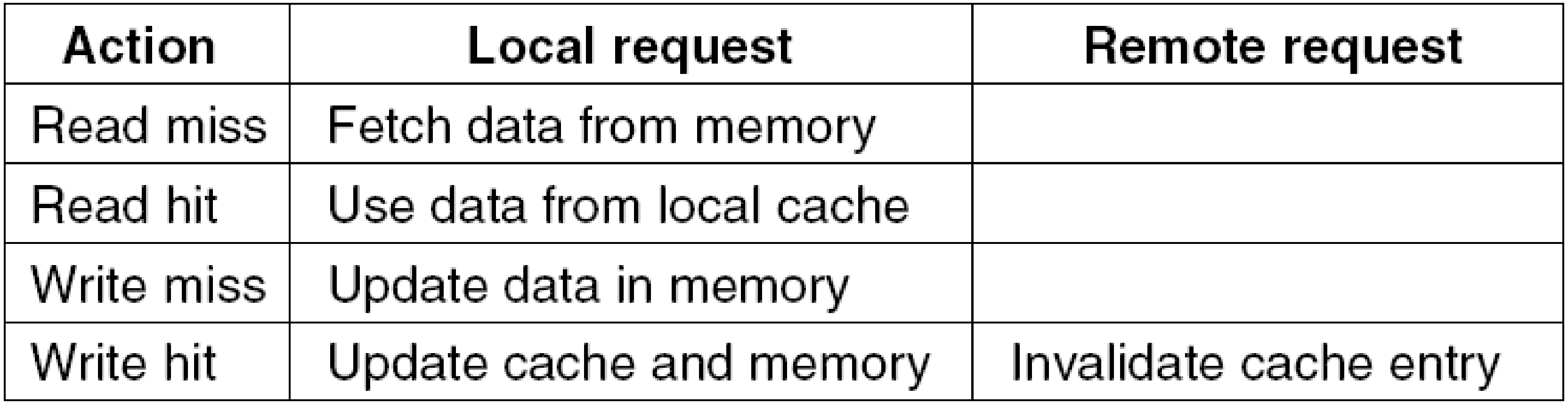
Snooping Caches

- Problem: stale data
- Solution: snooping caches/snoopy caches
  - Cache controller is designed to allow it to eavesdrop on the bus (bus requests from CPUs) and taking action in certain cases
- One simplest snooping cache protocol is called write through

→ see next slide

(Simple) Write Through

- In **write hit**:
  - when cache 1 writes a word that is present in cache 2's cache, if cache 2 does nothing, it will have stale data
  - Hence it marks the cache entry, and perform **update strategy** or **invalidate strategy**



Action	Local request	Remote request
Read miss	Fetch data from memory	
Read hit	Use data from local cache	
Write miss	Update data in memory	
Write hit	Update cache and memory	Invalidate cache entry

Then … Write Back

- Write through is simple but … inefficient
  - Bus still become a bottleneck since every write operation goes to memory over the bus.
- Idea:
  - When a cache line is modified, a bit is set inside the cache noting that the cache line is correct but memory is not.
  - Not all write go directly through memory
  - Known as a write-back protocol
  - Popular wire-back protocol called MESI

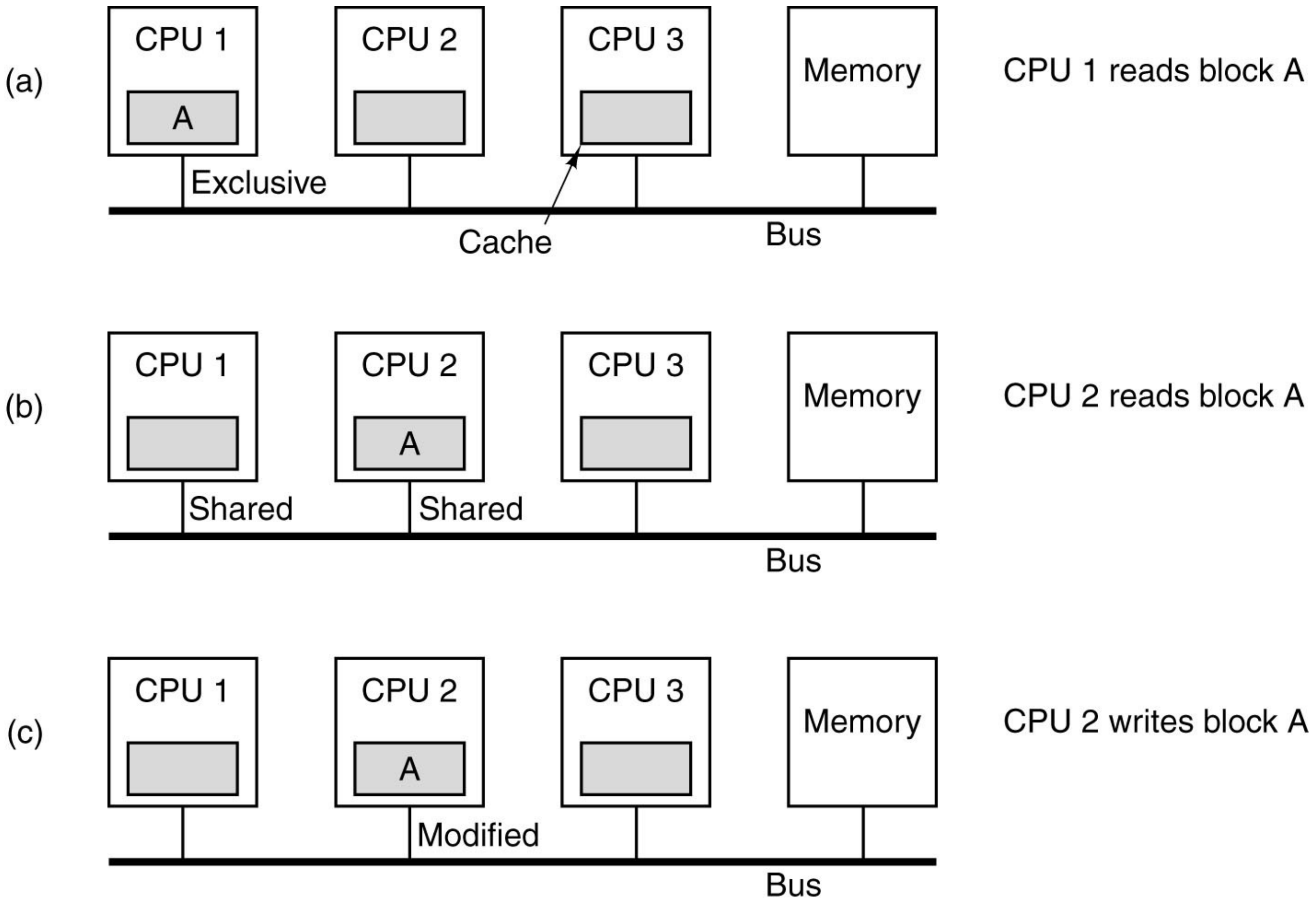
→ see next slide

The MESI Cache Coherence Protocol

- Used by Pentium 4 and many others

- **Cache lines states**
  - Invalid: line is not loaded or data is invalid
  - Shared: line is being shared by multiple processors
  - Modified: the line has been changed in the processor
  - Exclusive: the line is exclusively held by the processor cache

The MESI Cache Coherence Protocol

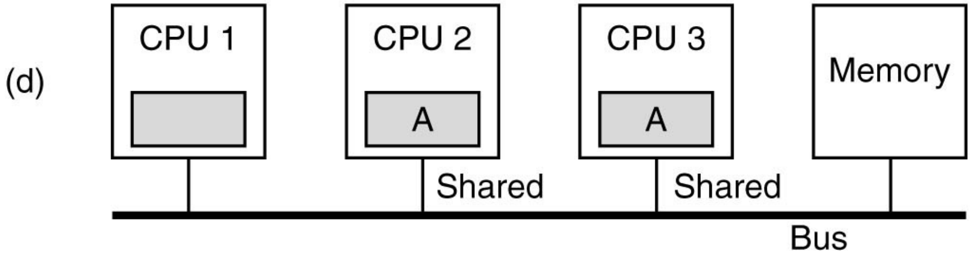


CPU1 reads the memory The line referenced is marked as being in the E state

CPU2 also fetch the same line and cache it. By snooping, CPU1 see that it is no longer alone. Both copes are marked as being in the S state

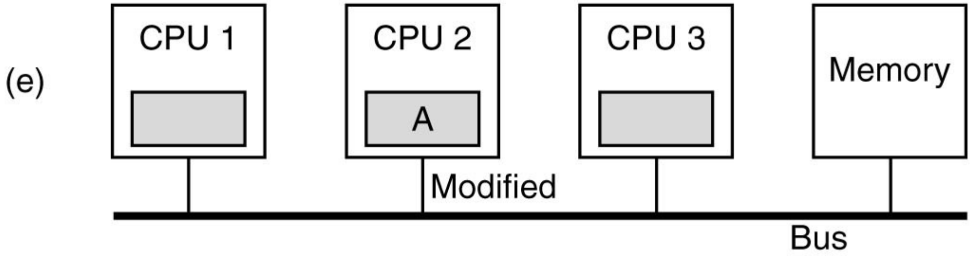
If CPU2 write to the cache line, it puts out an invalidate signal on the bus, telling other CPUs to discard their copies

The MESI Cache Coherence Protocol



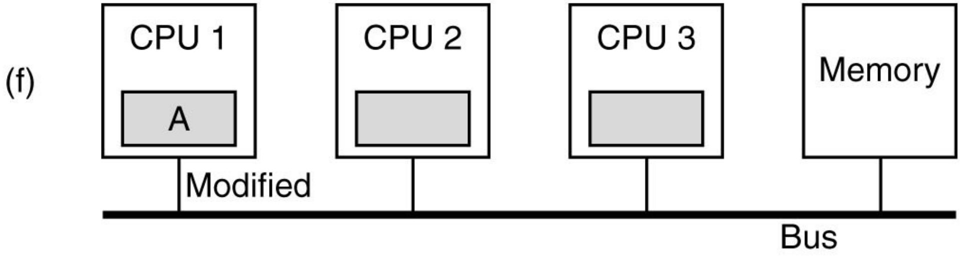
CPU 3 reads block A

When CPU3 reads the line, CPU2 knows that the copy in memory is not valid, so it asserts a signal on the bus telling CPU3 to wait until it writes its line back to memory



CPU 2 writes block A

CPU2 writes its line again, which invalidates the copy in CPU3's cache



CPU 1 writes block A

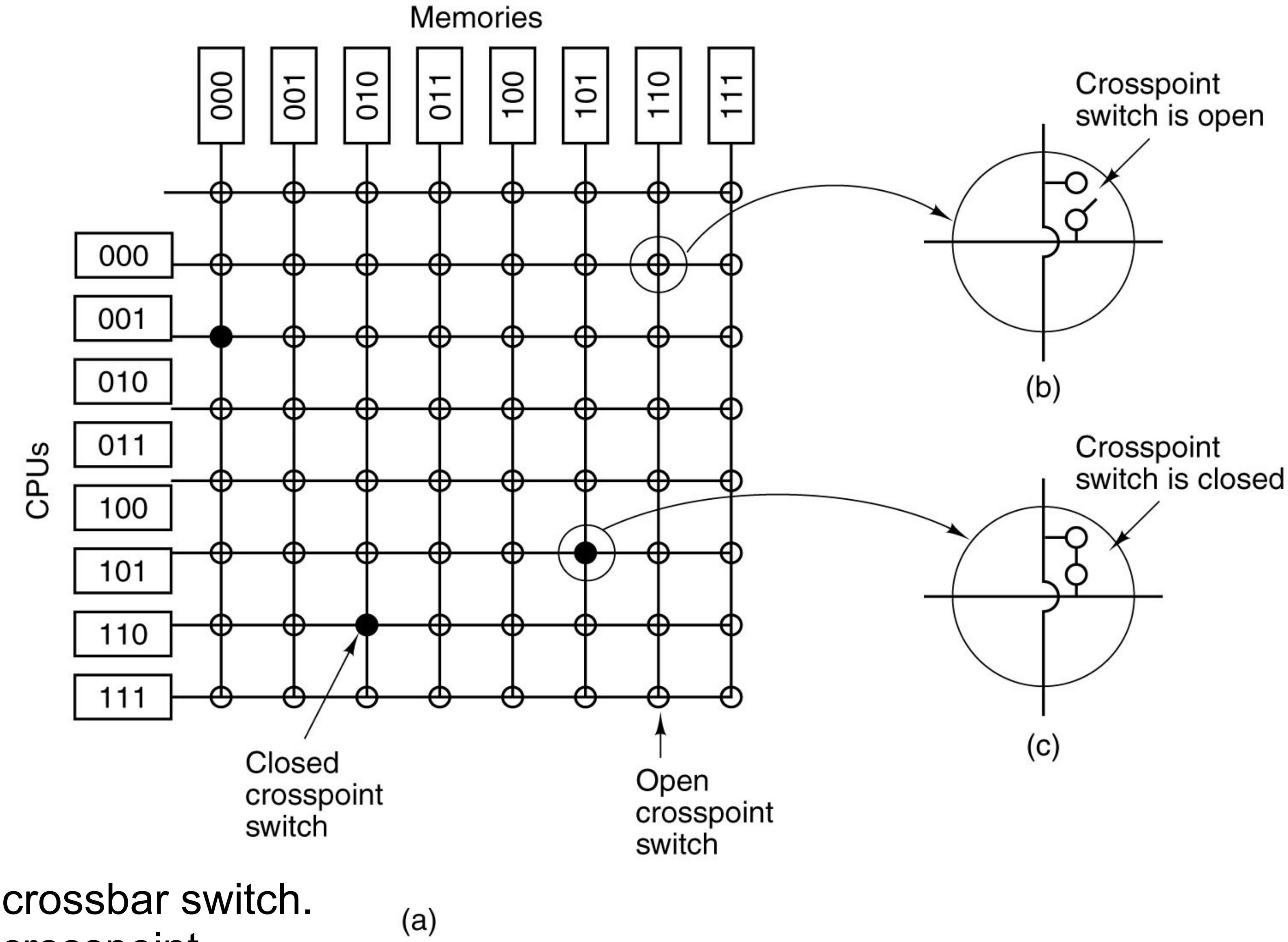
If CPU1 writes to a word in line, CPU2 tells CPU1 to wait and writes its line back to memory. When finished, it marks is own copy as invalid

Different Approaches for Interconnection Network

- **On the use of bus**:
  - The use of a single bus limits the size of a UMA multiprocessor to about 16 to 32 CPUs
  - To go beyond, different kind of interconnection network is needed.

- Others
  - <u>Crossbar</u> switch
  - <u>Multistage</u> switch

Crossbar Switch



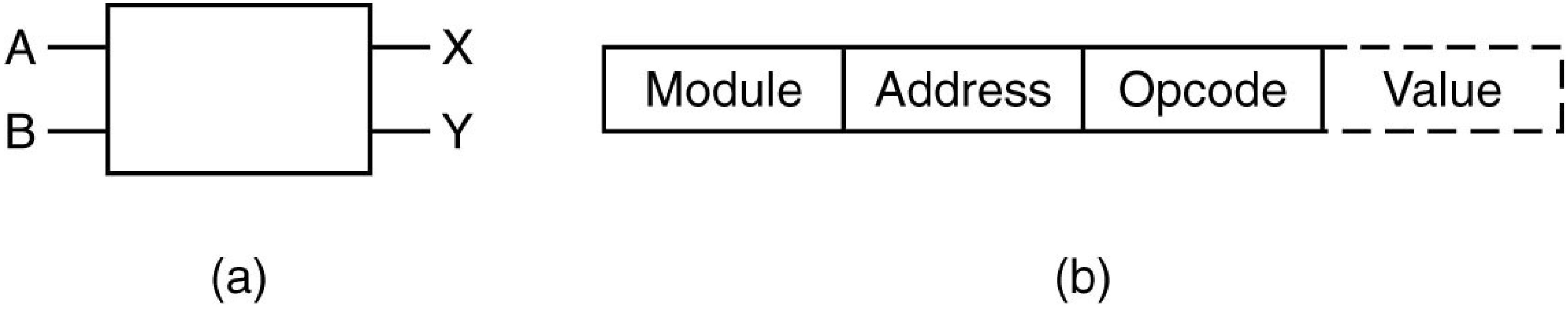
(a) An 8 × 8 crossbar switch.
(b) An open crosspoint.
(c) A closed crosspoint.

Crossbar Switch Properties

- **Nonblocking network**

- Number of switch ~ **n^2**
  - e.g. Sun Fire E25K has 1000 CPU and 1000 memory module
  - require 10 million switch, such is no feasible

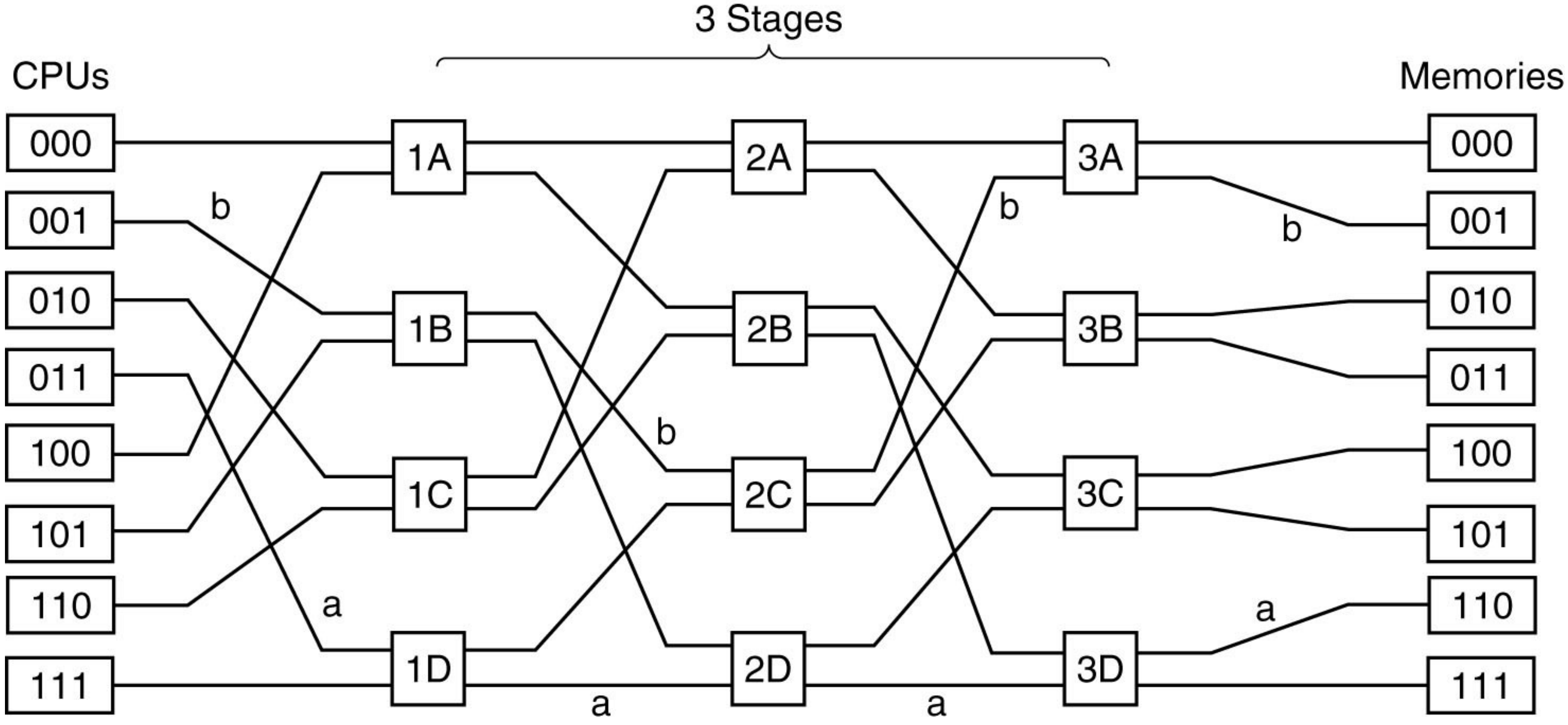
Multistage Switching Networks



(a) A 2 × 2 switch.
(b) A message format.

- Module field tells which memory to use
- Address specifies an address within a module
- Opcode is READ or WRITE
- Value contains an operand

One possibility of building larger multistage switching networks is omega switching networks



Omega Switching Network Properties

- The wiring pattern of the omega network is often called the perfect shuffle

- Omega switching networks is **blocking network**

- Question: with n CPUs and n memories, how many stages and switches per stage is needed?
  - Compare to crosspoint system

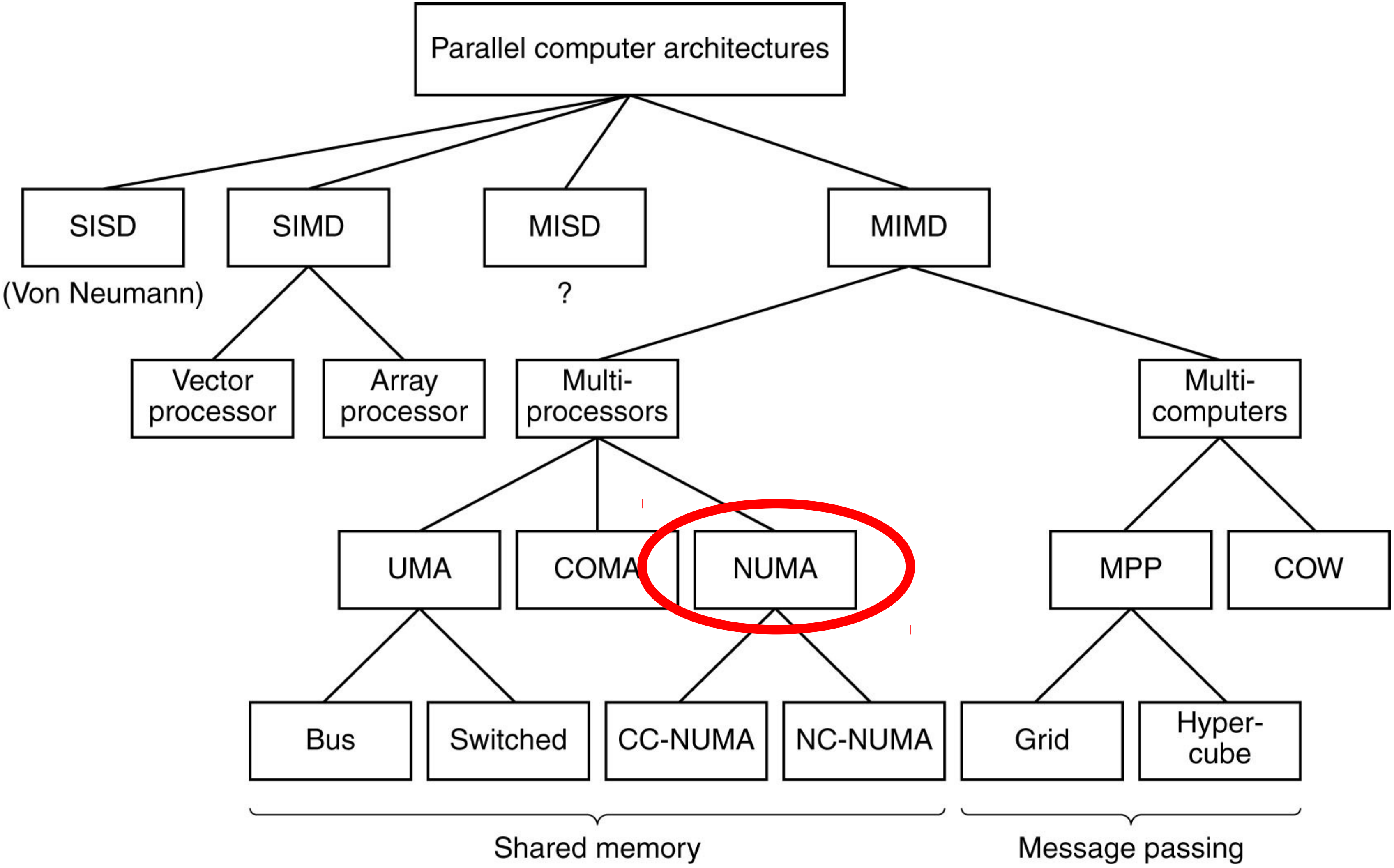
NUMA (NonUniform Memory Access) Multiprocessors

- Fact
  - UMA multiprocessors are limited to no more than few dozen CPUs

- **To get more than 100 CPUs, idea is**
  - Access to local memory is faster than access to remote one
  - All memory modules have the same access time

    → lead to NUMA

NUMA (NonUniform Memory Access) Multiprocessors



A taxonomy of parallel computers.

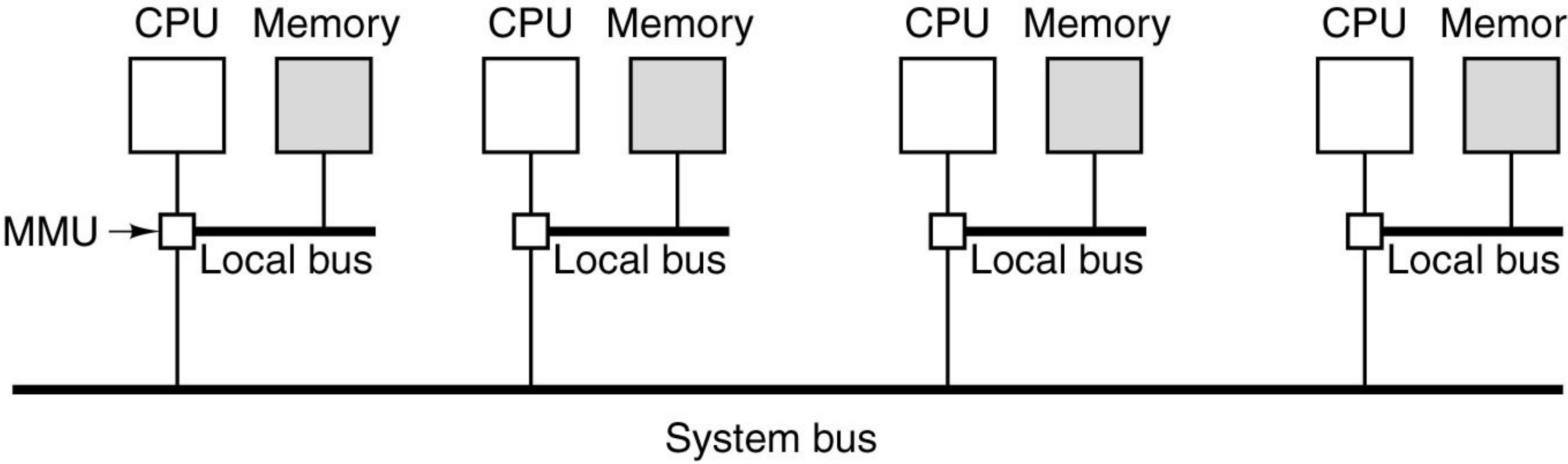
NUMA Multiprocessors

- **NUMA machines have three key characteristics:**
  - There is single address space visible to all CPUs
  - Access to remote memory is done using LOAD and STORE instructions
  - Access to remote memory is slower than access to local memory

- 2 sorts of NUMA
  - NC-NUMA (no cache NUMA)
  - CC-NUMA (cache coherent)

MMU: memory management unit

NC-NUMA



- A NUMA machine based on two levels of buses. The Carnegie-Mellon Cm* was the first multiprocessor to use this design.

- When a memory request come to the MMU, a check was made to see if the word needed was in local memory.

- Page scanner runs every few seconds to examine the usage statistics and move pages around in an attempt to improve performance

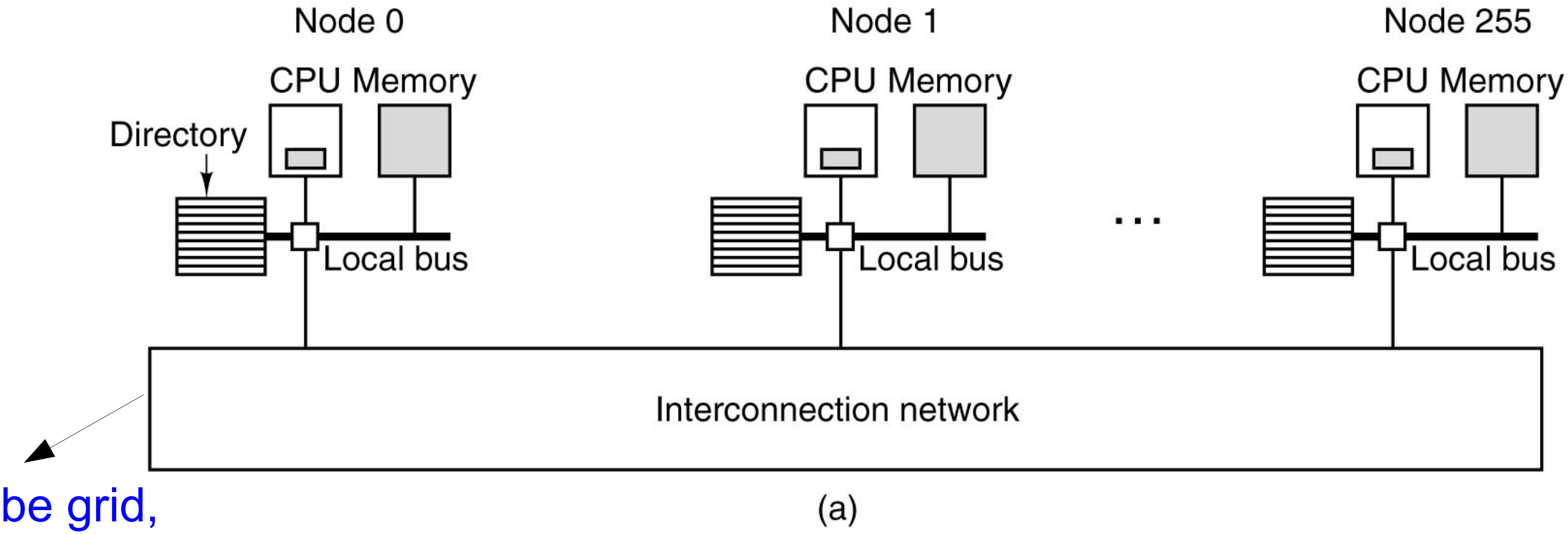
CC-NUMA

- NC-NUMA have to go to the remote memory every time a nonlocal memory is accessed is a major problem hit.
- Idea:
  - Caching. Hence cache coherent must also added. Snooping the bus is feasible?
- Popular approach for building large CC-NUMA multiprocessors currently is the **directory-based multiprocessor**
  - maintain a database telling where each cache line is and what its status is

CC-NUMA

- Consider a 256-node system
- Each node consisting of one CPU and 16 MB of RAM connected to the CPU
- The total memory is 2^32 bytes, divided up into 2^26 cache lines of 64 bytes each

CC-NUMA



Could be grid, hypercube, or others

Bits	8	18	6
	Node	Block	Offset

(b)

2^{18}-1		
≈		
4	0	
3	0	
2	1	82
1	0	
0	0	

(c)

(a) A 256-node (x16MB) directory-based multiprocessor.
(b) Division of a 32-bit memory address into fields [node, block, offset].
(c) The directory at node 36.

Directory-base Multiprocessor

- Idea: maintain a database telling where each cache line is and what its status is
- The database must be kept in extremely-fast special-purpose hardware
  - can be respond in a fraction of a bus cycle
- This model has a lot of message passing in the network

Example (with figure in previous slide)

- LOAD instruction from CPU 20 that references a cached line, which physical address is, say, 0x24000108 (by CPU200's MMU)
  - Node 36, line 4, offset 8
  - CPU 20 send a request to node 36 for line 4
- If the line is not cached, the hardware fetches line 4 from local RAM, send back to node 20, and update the directory that the line is now cached at node 20
- Assume node 20 send the second request for node 36 line 2. Fig. (c) shows that the line is now at node 82, so node 20 sends a message to node 82 to pass the line to node 20, and invalidate its cache

Overload by Directory

- Each node has 16 MB of RAM
- 2^18 * 9 bit entries to keep tract
  - Overhead is about 1.76%
- With 32 byte cache lines
  - Overhead is about 4%

- Remember cache directory must be high-speed memory

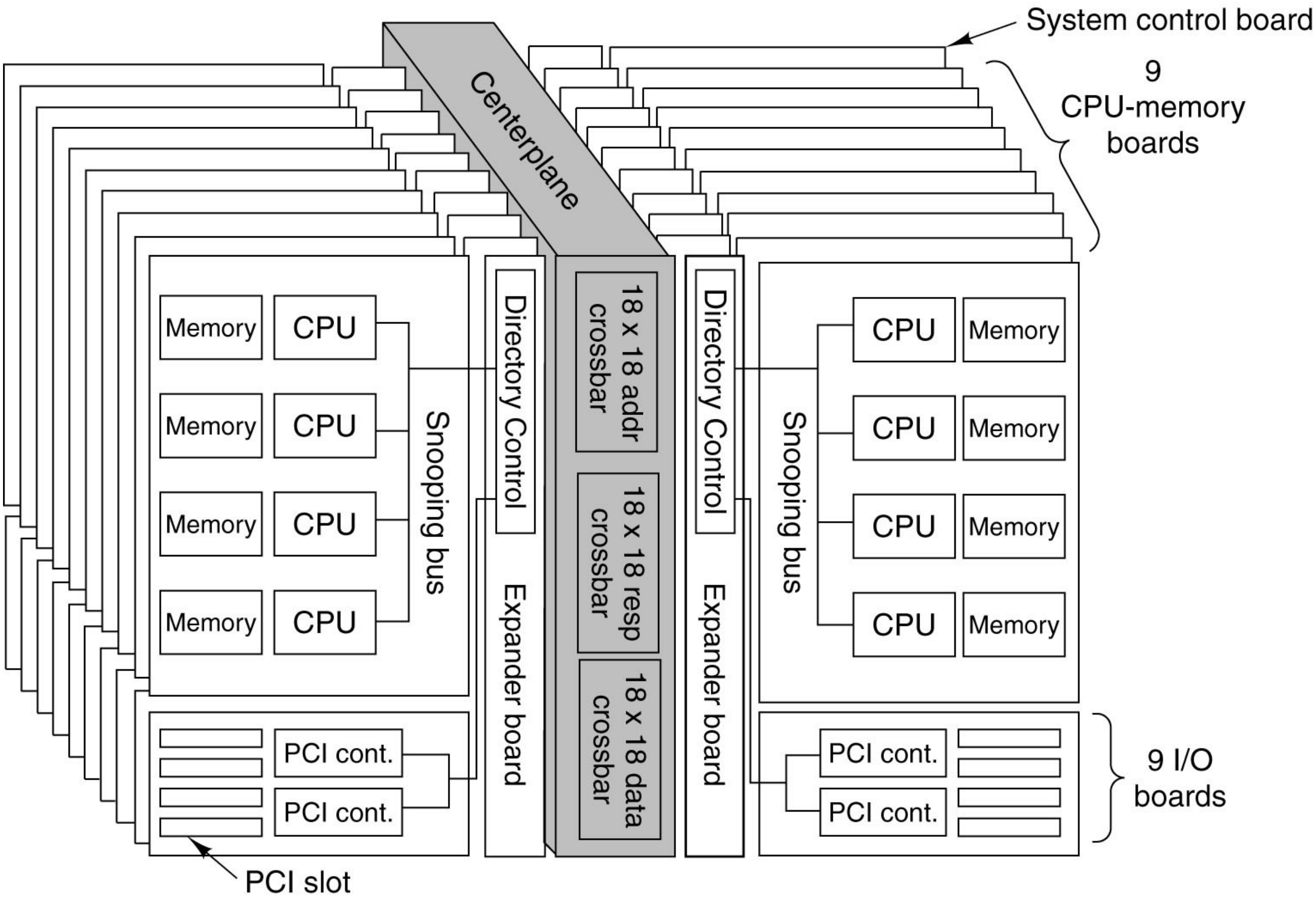
Comments

- Improvement
  - Cached at only one node → cached at multiple nodes
  - Keep tract of whether a cache line is is modified

- See (*) for other performance optimizations

(*) STENSTROM, P., HAGERSTEN, E., LILJA, D.J., MARTONOSI, M., and VENUGOPAL, M.: “Trends in Shared Memory Multiprocessing,” IEEE Computer Magazine, vol. 30, pp. 44-50, Dec. 1997.

Example of NUMA: The Sun Fire E25K



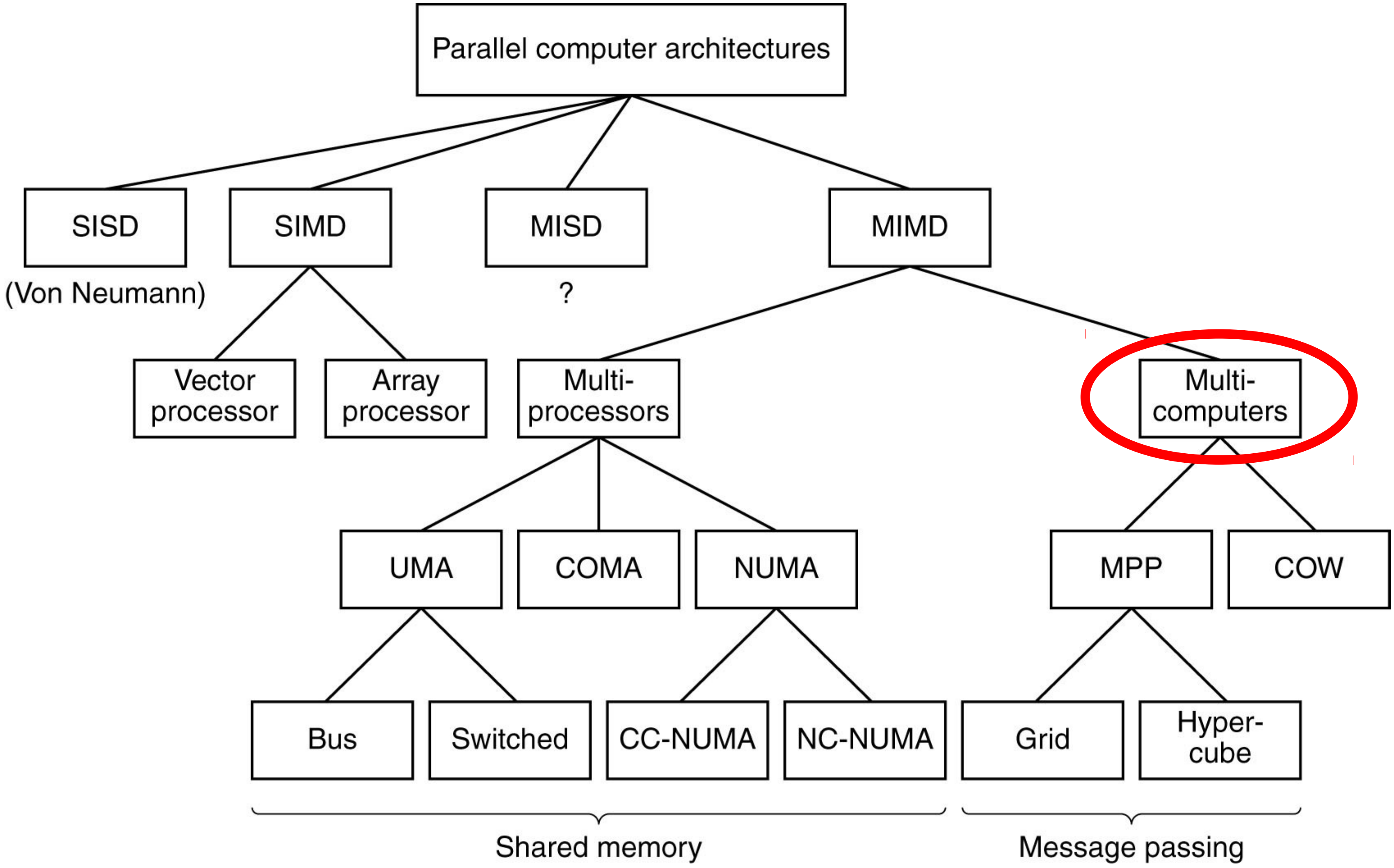
Example of NUMA: The Sun Fire E25K

- Sun Fire E25K has
  - 72 UltraSPARC IV CPU chips (dual processor)
  - 18 boardsets
  - Each memory board contains 4 CPU chips and 4x8 GB RAM modules
- Total
  - 144 CPU, 576 GB RAM, 72 PCI slots

Example of NUMA: The Sun Fire E25K

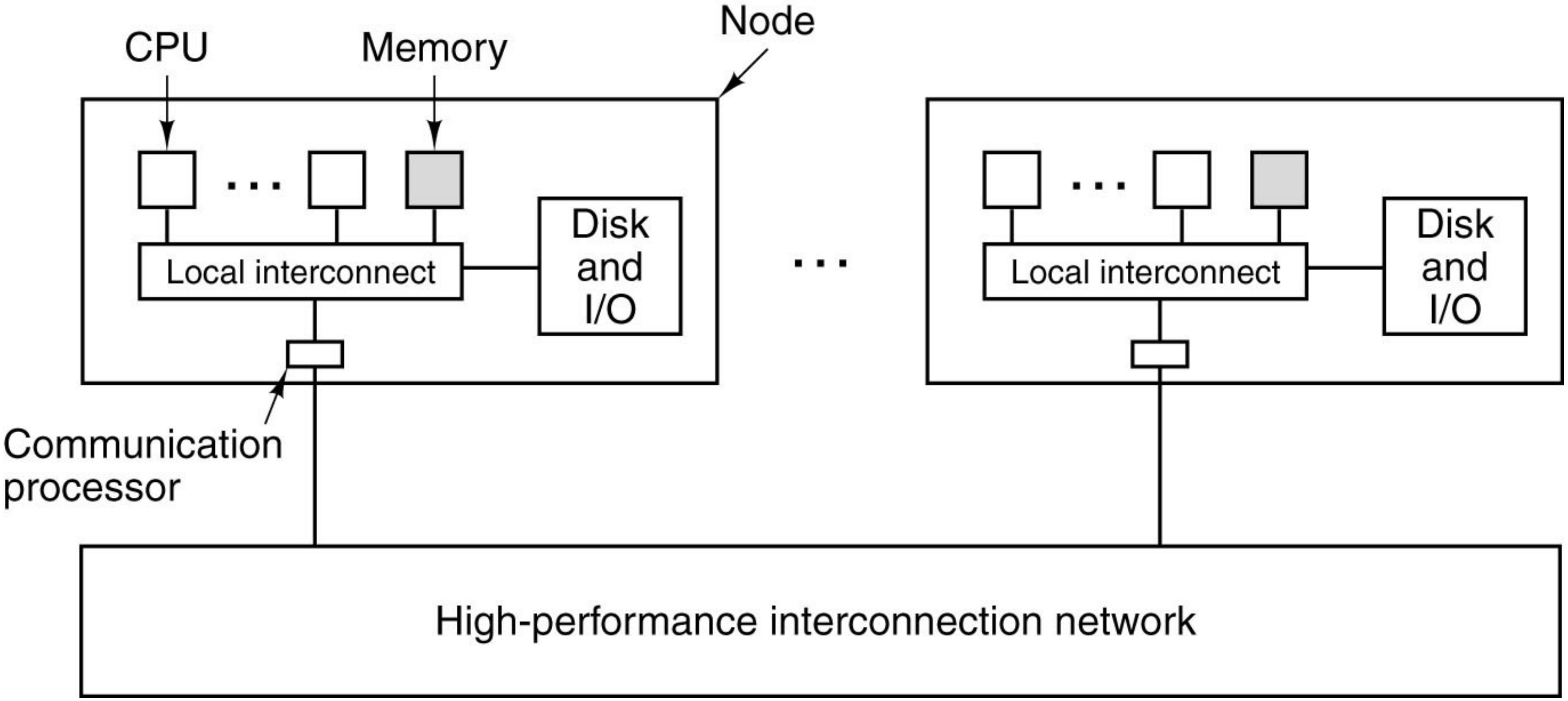
- 576 GB of memory is split into 2^29 blocks of 64 bytes each
  - When CPU needs to read/write a memory word, it first checks its own cache; if fail, it looks on its own boardset; if not, it send a request over the centerplane to asking where the memory block is.

Expanded Computer Taxonomy



A taxonomy of parallel computers.

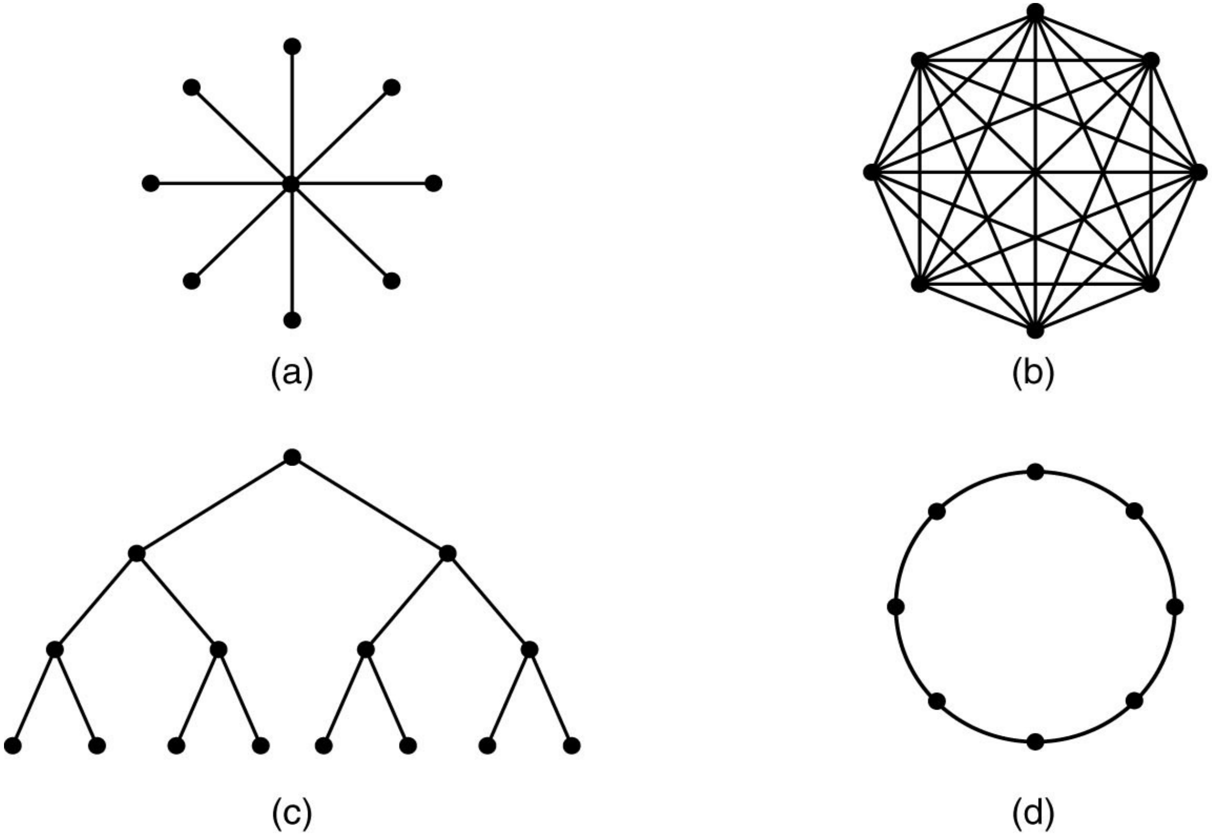
Message-Passing Multicomputers



A generic multicomputer

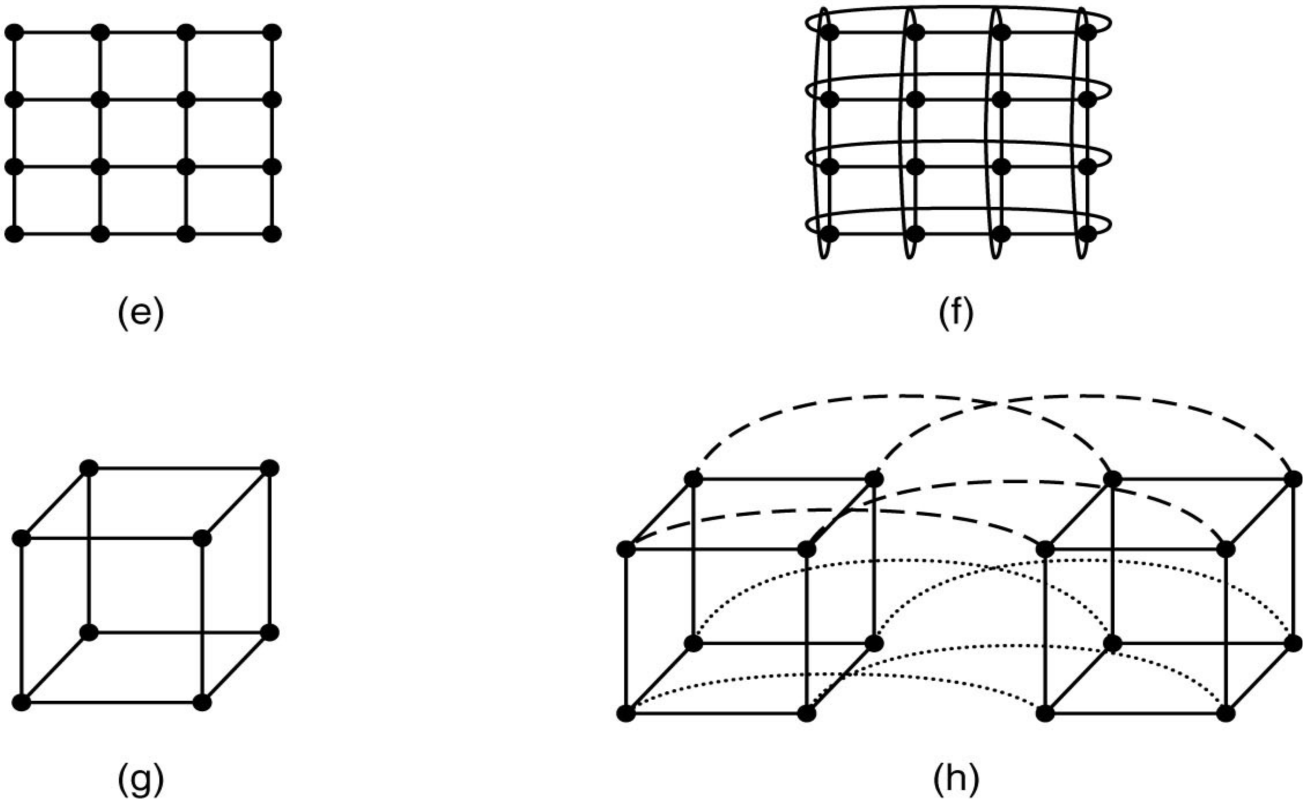
Network Topology (1)

- Multiprocessor and multicomputer surprisingly similar in this respect (network).



(a) A star. (b) A complete interconnect.
(c) A tree. (d) A ring.

Network Topology (2)



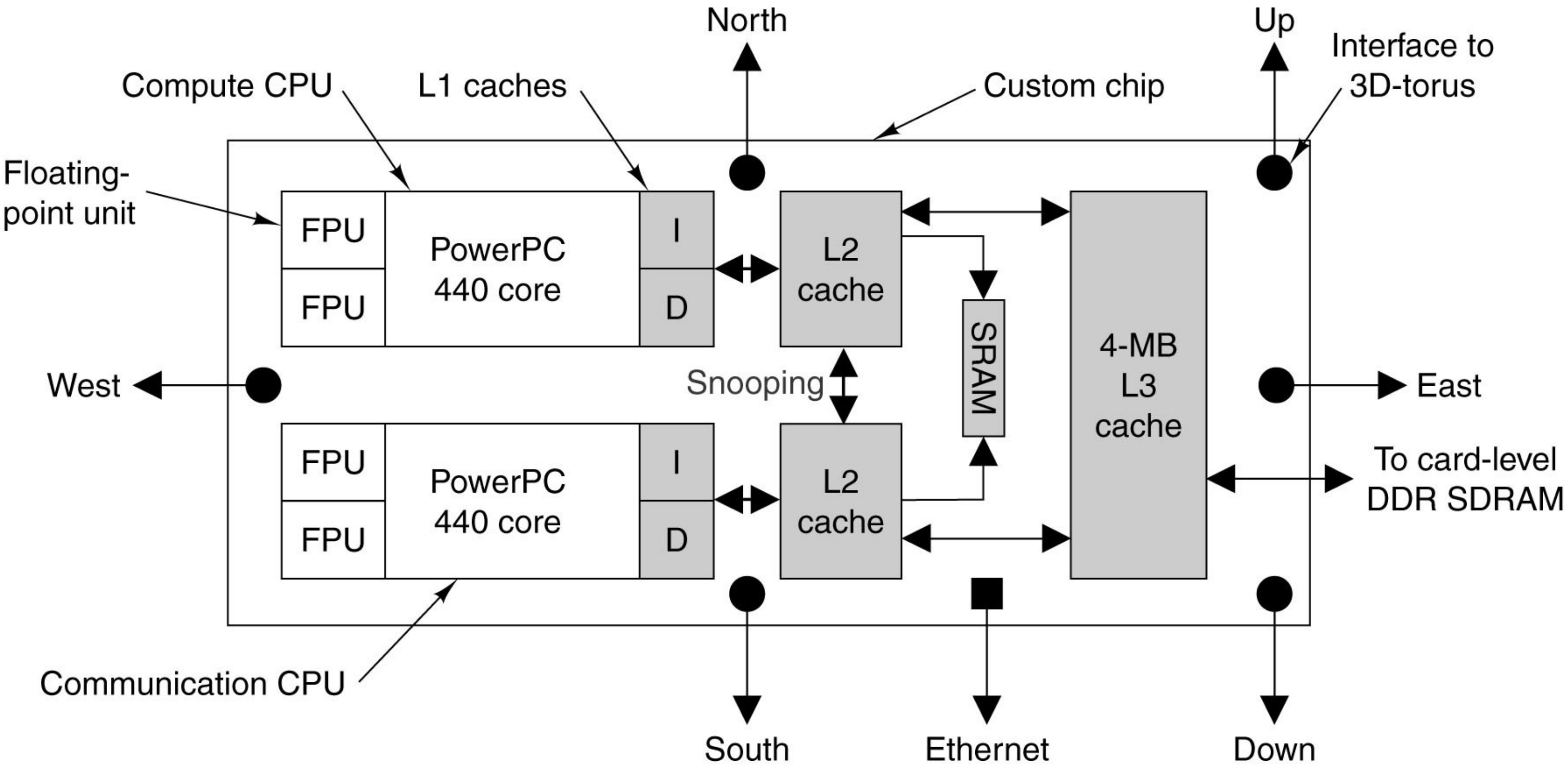
(e) A grid. (f) A double torus.
(g) A cube. (h) A 4D hypercube.

MMP – Massively Parallel Processors

- Example 1: Blue Gene (IBM)

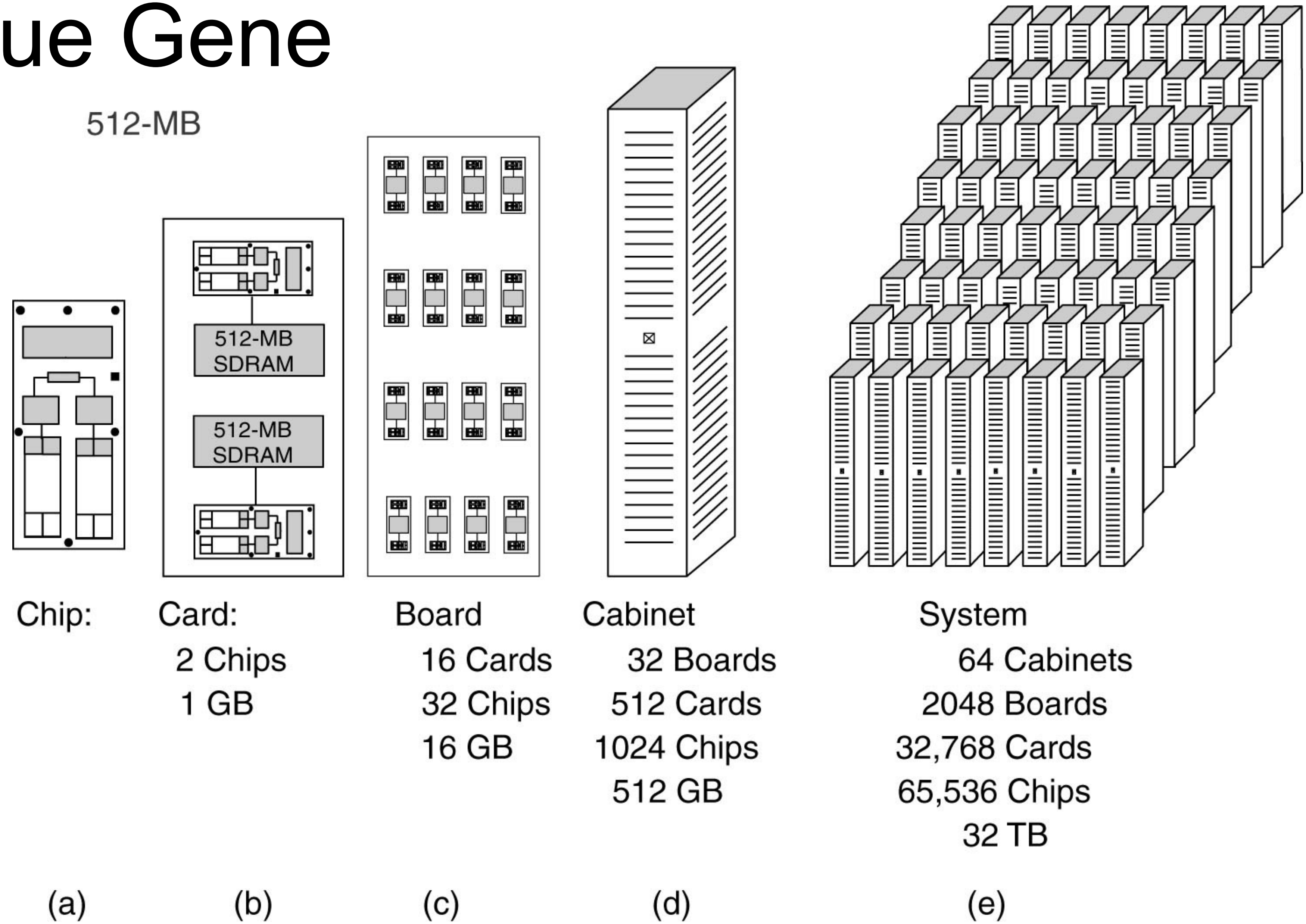
Picture copied from http://en.wikipedia.org/wiki/Blue_Gene

Blue Gene



Release in Nov. 2004.
PowerPC core runs @700Mhz, is a pipelined dual-issue superscalar processor. Each core has a pair of dual-issue floating point units. BG has capability of 71 Teraflops/sec.

Blue Gene

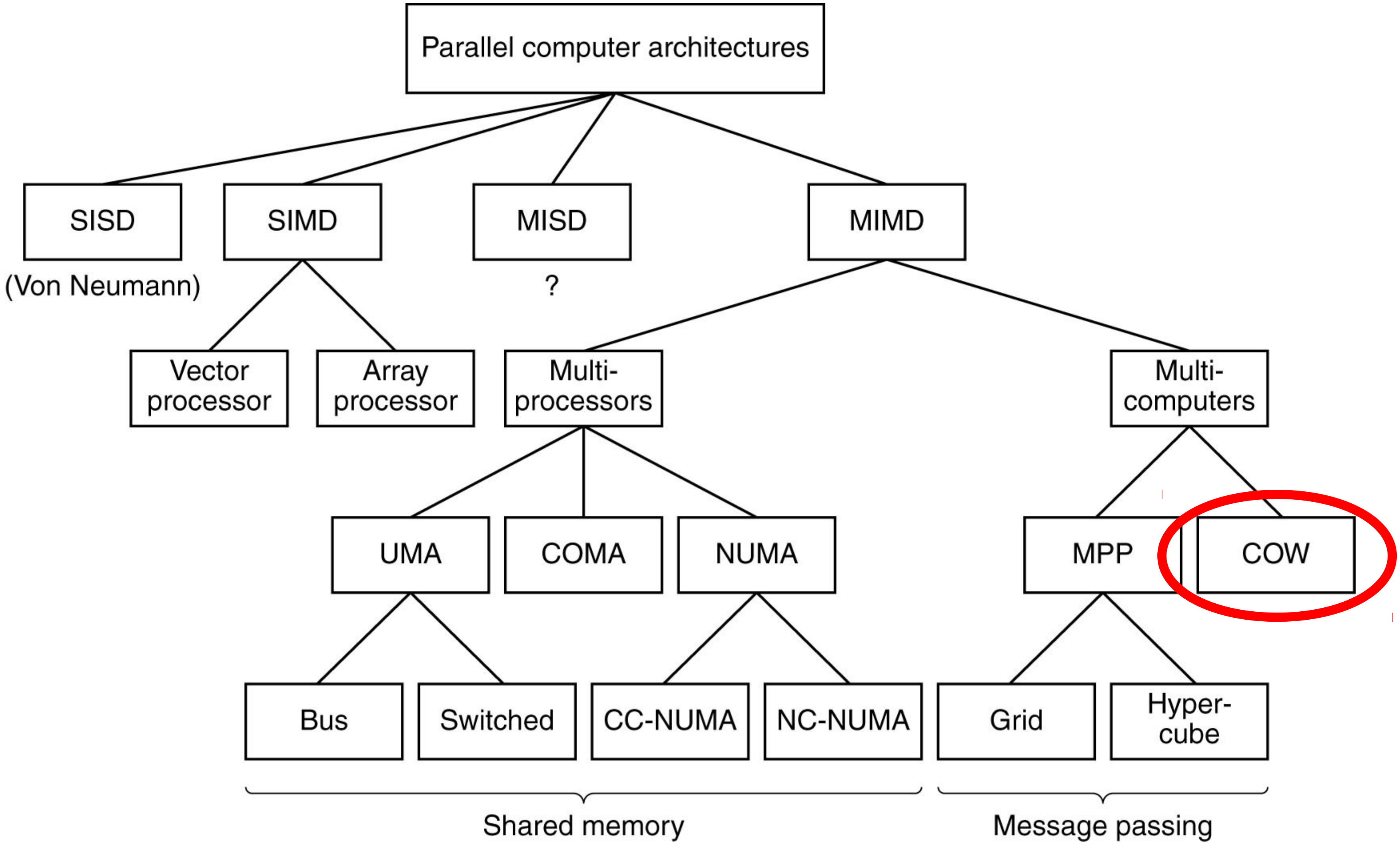


The BlueGene/L. (a) Chip. (b) Card. (c) Board. (d) Cabinet. (e) System.

Blue Gene

- On the 2 CPU cores, one for computing, the other is for handling communication among the 65536 nodes
- Interconnection
  - Three-dimensional torus, each CPU needs 6 connection
  - Each cabinet has 1024 nodes, connected by 8x8x16 torus
  - Final torus: 64 x 32 x 32
  - Point-to-point link operates at 1.4 Gbps

Expanded Computer Taxonomy

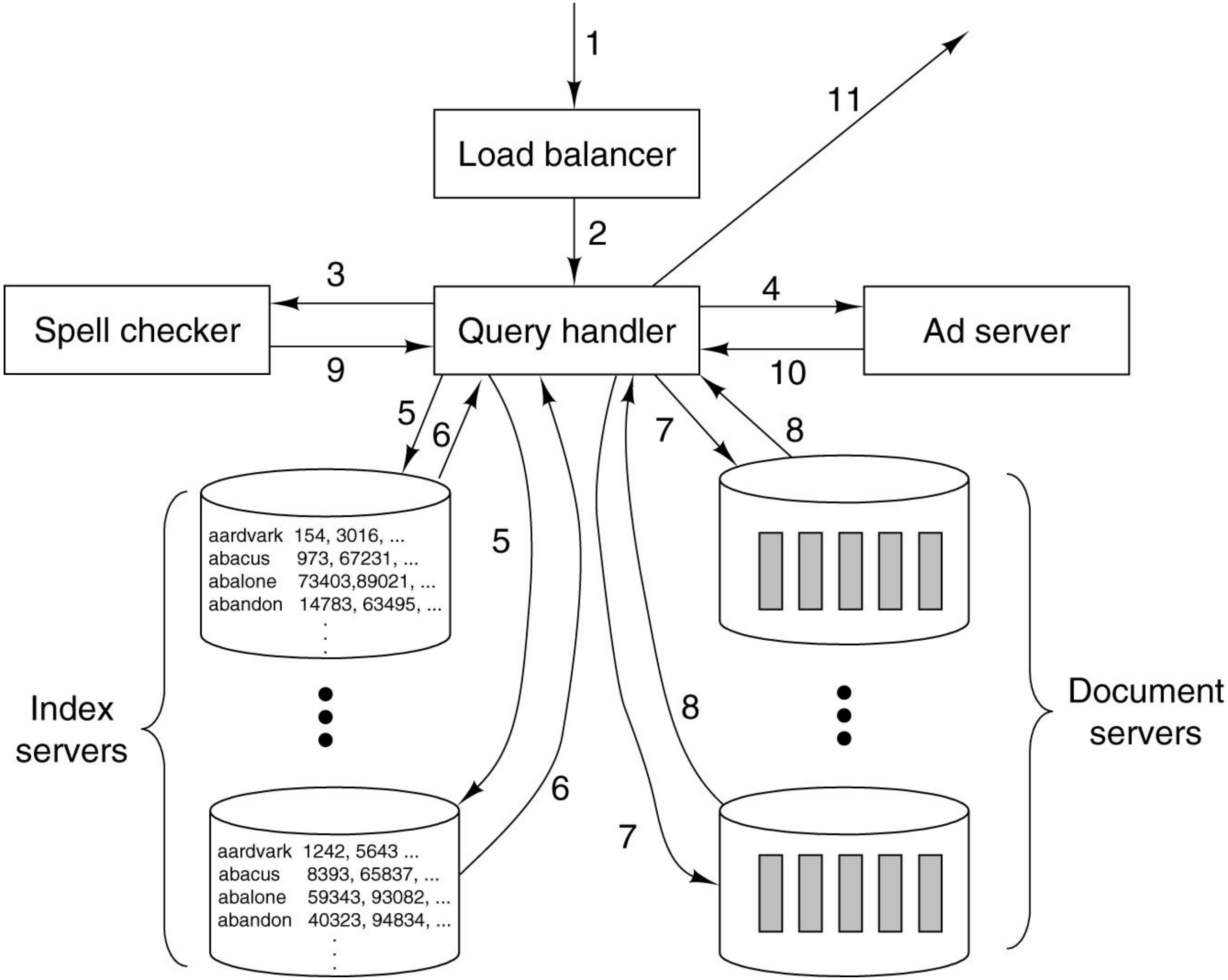


A taxonomy of parallel computers.

Cluster Computing (COW)

- MMPs specialize in their high speed interconnect to Cluster Computer
  - But the gap begins to close
- COW typically consists of hundred or thousands of Pcs/Workstations connected by a commercially-available network board
- 2 kind of clusters
  - Centralized: machines are homogeneous and have no peripherals other than network cards and disk
  - Decentralized: machines are heterogeneous, idle many hours a day, connected by LAN on campus/building

Google



Processing of a Google query.

Google

- Google operates multiple data centers around the world. Why?
- Google does not buy the biggest, fastest, and most reliable equipment when dealing with huge database, massive transaction rate, and the need for reliability.
  - They optimize price/performance, not absolute performance
  - Build the largest off-the-shelf cluster

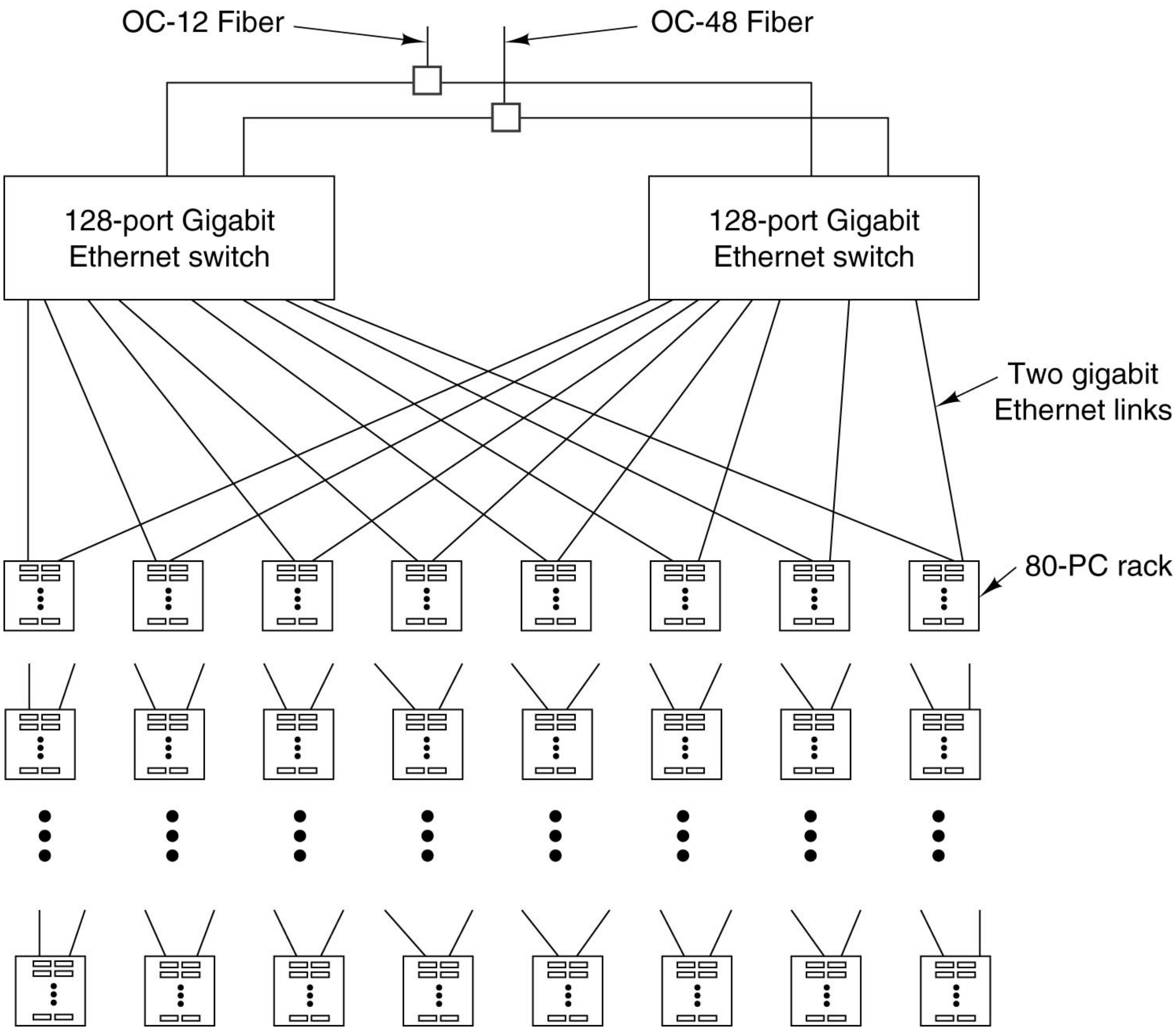
Google

- From Wiki http://en.wikipedia.org/wiki/Google_platform

  “Servers as of 2009-2010 consisted of a custom made open top systems containing two processors (each with an 2 cores[3]) a considerable amount of RAM spread over 8 DIMM slots housing double height DIMMS and two SATA hard drives connected through a non-standard ATX sized power supply.[4] According to CNET and to book by Hennessy, each server has a novel 12 volt battery to reduce costs and improve power efficiency”

A Typical Google Cluster

Typical value
128 port switch
64 racks
5120 PCs


OC-12 Fiber
OC-48 Fiber
128-port Gigabit Ethernet switch
128-port Gigabit Ethernet switch
Two gigabit Ethernet links
80-PC rack

Lessons from Google

1. **Component will fail, so plan for it**
   - Event with best equipment it will fail some day no mater 1 time a week or 2 times a week
   - Need fault-tolerant software
2. **Replicate everything for throughput and availability**
   - Both hardware and software have to be highly redundant
   - PCs, disks, ables, and switches are all replicated
3. **Optimize price/performance**
   - If the system has been designed to deal with failure, buying expensive component, e.g. RAIDs with SCSI disk, is a mistake

More about Google

- BARROSO, L.A., DEAN, J., HOLZLE, U.: “Web Search for a Planet: The Google Cluster Architecture,” IEEE Micro Magazine, vol. 23, pp. 22-28, March-April 2003.
- GHEMAWAT, S., GOBIOFF, H., and LEUNG, S.-T.: “The Google File System,” Proc. 19th Symp. on Operating Systems Principles, ACM, pp. 29-43, 2003.

Communication Software for Multicomputers

- MPI – Message Passing Interface
  - Your third project (*)

Copyright note

- Slides are adopted form lecture notes of Tanenbaum, Structured Computer Organization, Fifth Edition, (c) 2006 Pearson Education, Inc.