AMS 250: An Introduction to High Performance Computing

Parallel Computer Architecture

Shawfeng Dong

shaw@ucsc.edu

(831) 502-7743

Applied Mathematics & Statistics

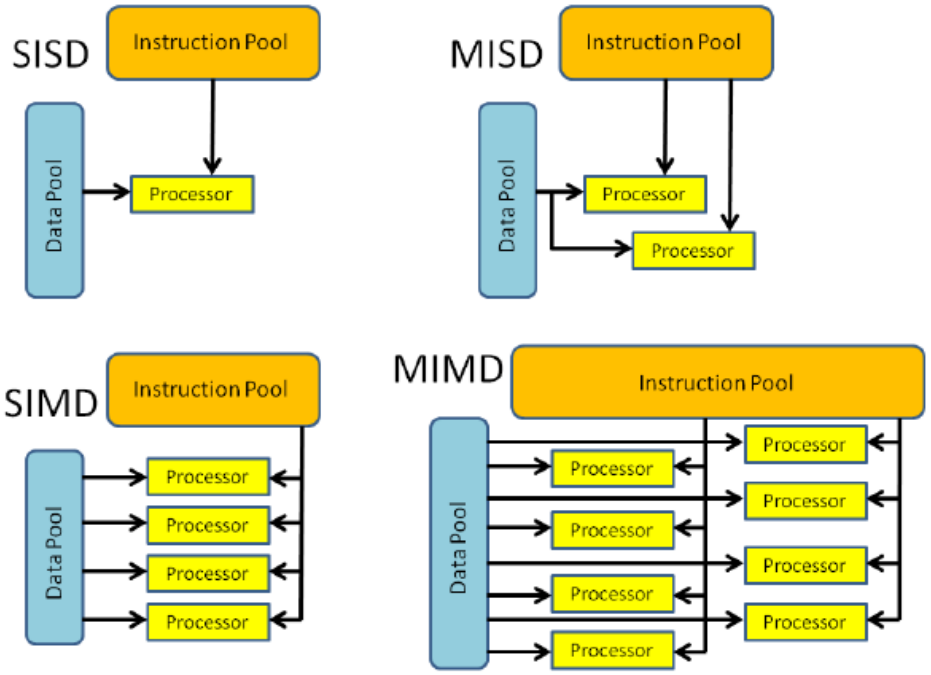
University of California, Santa Cruz

Outline

- Parallel architecture types
- Instruction-level parallelism
- Vector processing
- SIMD
- Shared memory
  - Memory organization: UMA, NUMA
  - Coherency: CC-UMA, CC-NUMA
- Interconnection networks
- Distributed memory
- Clusters
- Fastest computers in history

Classifying Parallel Systems – Flynn's Taxonomy

- Distinguishes multi-processor computer architectures along the two independent dimensions
  - ***Instruction*** and ***Data***
  - Each dimension can have one state: ***Single*** or ***Multiple***
- SISD: Single Instruction, Single Data
  - Serial (non-parallel) machine
- SIMD: Single Instruction, Multiple Data
  - Processor arrays and vector machines
  - SIMT (*T: threads*) for GPUs
- MISD: Multiple Instruction, Single Data (weird)
- MIMD: Multiple Instruction, Multiple Data
  - Most common parallel computer systems
  - SPMD & MPMD (*P: program*)

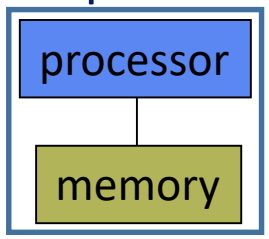


Phases of Supercomputing (Parallel) Architecture

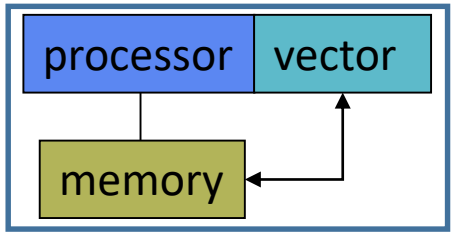
- Phase 1 (1950s): sequential instruction execution
- Phase 2 (1960s): sequential instruction issue
  - Pipeline execution, reservations stations
  - Instruction Level Parallelism (ILP)
- Phase 3 (1970s): vector processors
  - Pipelined arithmetic units
  - Registers, multi-bank (parallel) memory systems
- Phase 4 (1980s): SIMD and SMPs
- Phase 5 (1990s): MPPs and clusters
  - Communicating sequential processors
- Phase 6 (>2000): many cores, accelerators, scale, …

Parallel Architecture Types

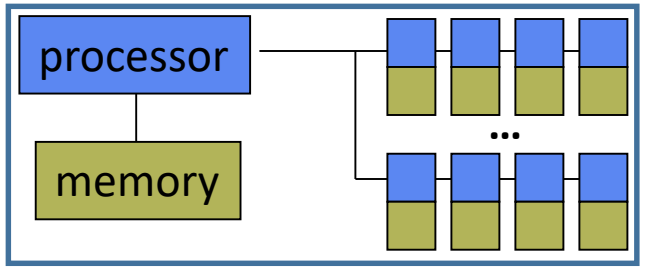
- **Uniprocessor**
  - Scalar processor



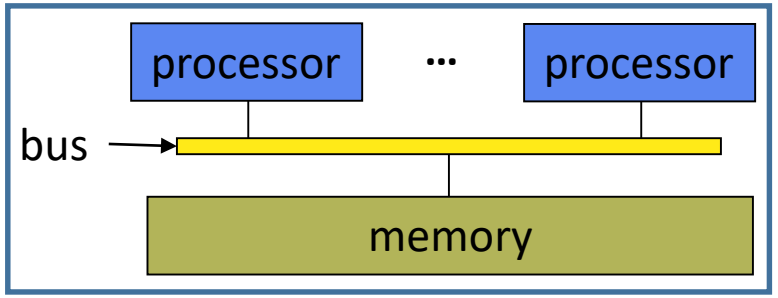
  - Vector processor



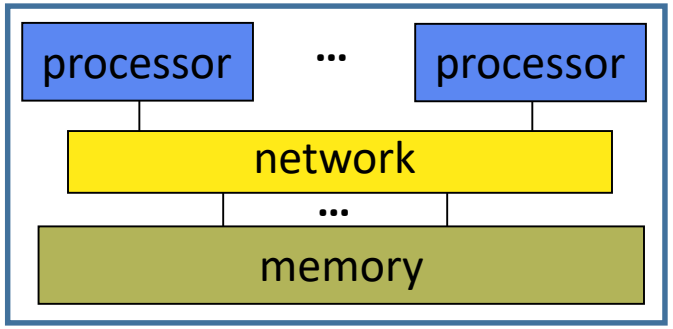
  - Single Instruction Multiple Data (SIMD)



- **Shared Memory Multiprocessor (SMP)**
  - Shared memory address space
  - Bus-based memory system

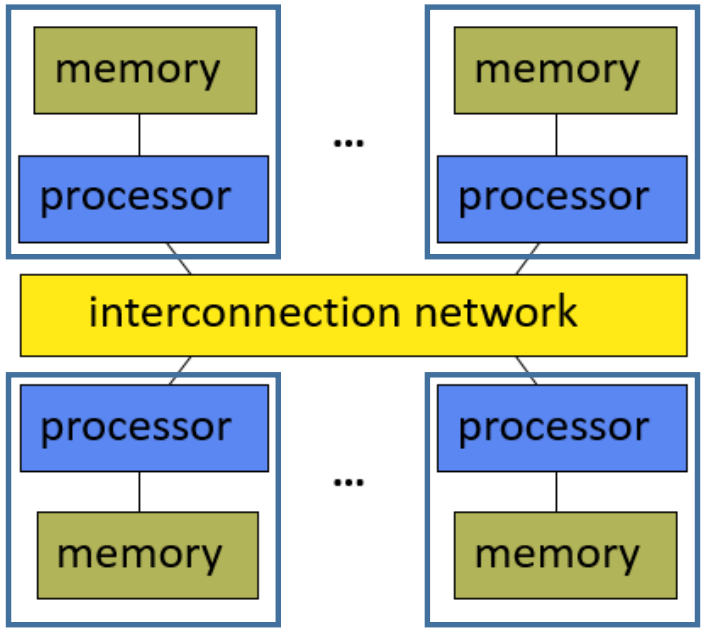


  - Interconnection network



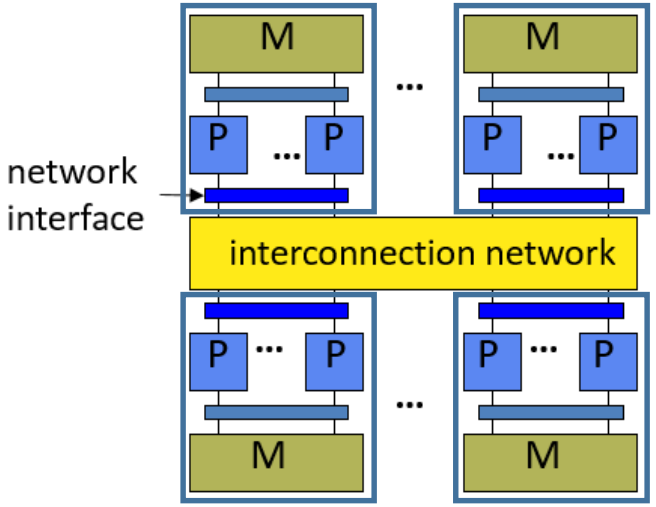
Parallel Architecture Types (2)

- **Distributed Memory Multiprocessor**
  - Message passing between nodes



  - Massively Parallel Processor (MPP)
    - Many, many processors

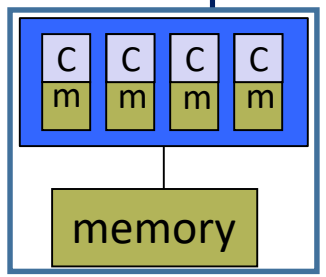
- **Cluster of SMPs**
  - Shared memory addressing within SMP node
  - Message passing between SMP nodes



  - Can also be regarded as MPP if processor number is large

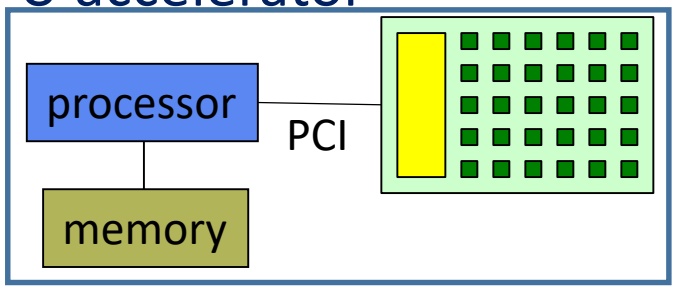
Parallel Architecture Types (3)

- **Multicore**
  - Multicore processor

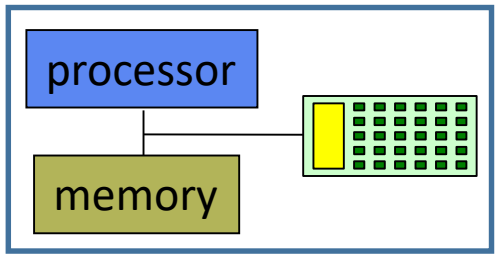


cores can be hardware multithreaded (hyper-threading)

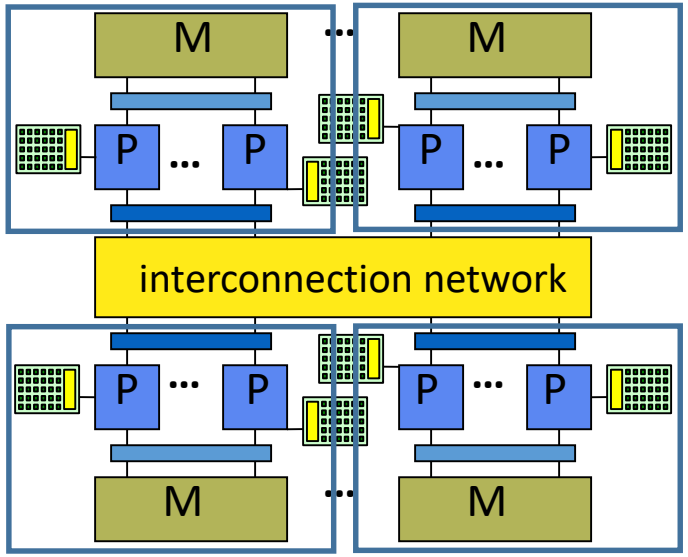
  - GPU accelerator



  - "Fused" processor accelerator



- **Multicore SMP+GPU Cluster**
  - Shared memory addressing within SMP node
  - Message passing between SMP nodes
  - GPU accelerators attached



How do you get parallelism in the hardware?

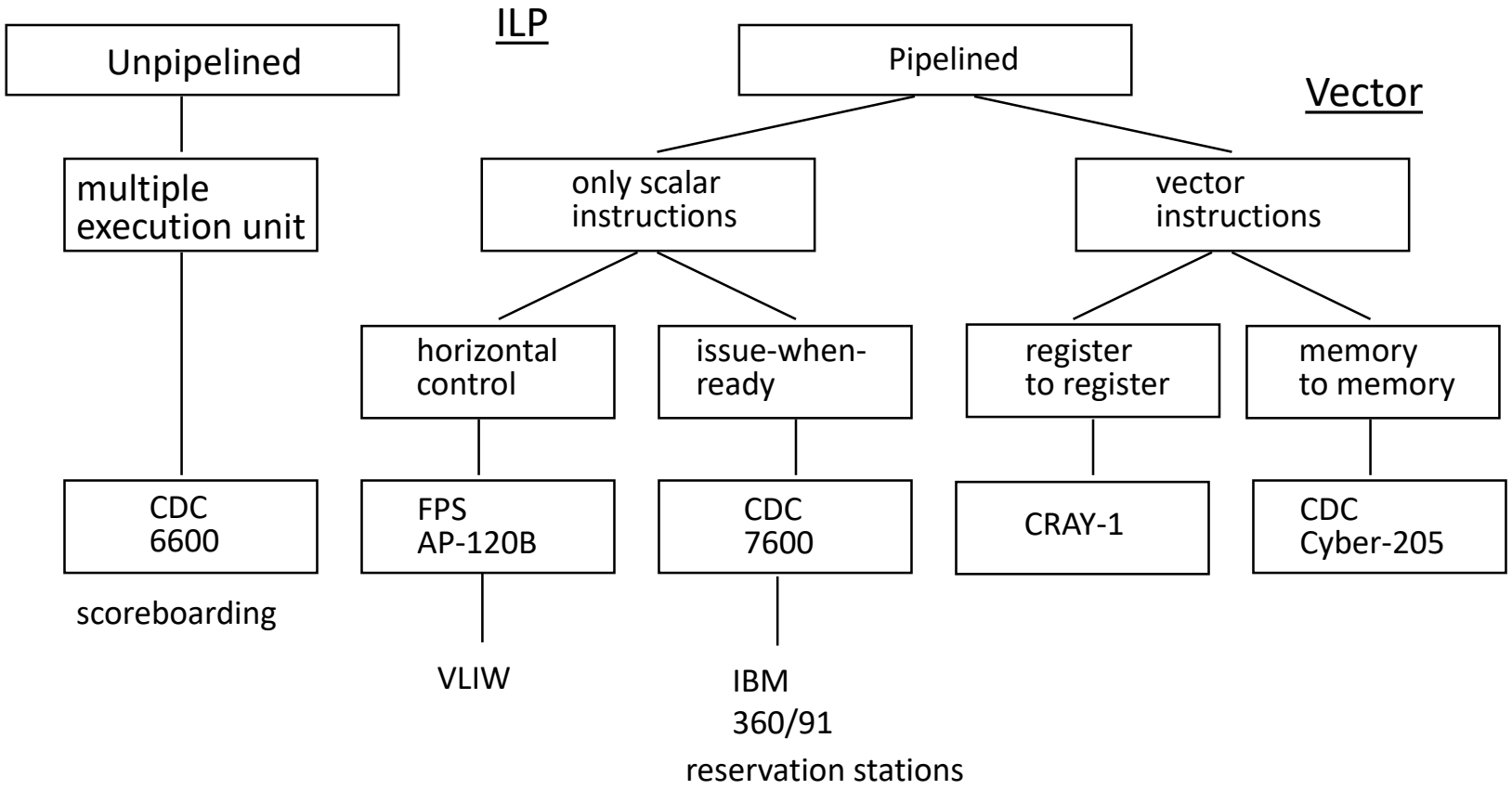
- Instruction-Level Parallelism (ILP)
- Data parallelism
  - Increase amount of data to be operated on at same time
- Processor parallelism
  - Increase number of processors
- Memory system parallelism
  - Increase number of memory units
  - Increase bandwidth to memory
- Communication parallelism
  - Increase amount of interconnection between elements
  - Increase communication bandwidth

Instruction-Level Parallelism

- Opportunity for splitting up instruction processing is called **Instruction-Level Parallelism (ILP)** - https://en.wikipedia.org/wiki/Instruction-level_parallelism
- Micro-architectural techniques that are used to exploit ILP include:
  - Instruction pipelining
  - Overlapped execution
  - Multiple functional units
  - Out-of-order execution
  - Multi-issue execution
  - Superscalar processing
  - Speculative execution (but beware of the Meltdown and Spectre vulnerabilities!)
  - Branch prediction
  - Very Long Instruction Word (VLIW)
  - EPIC
  - Register renaming
  - Hardware multithreading (hyper-threading)
- ILP is exploited by both the *hardware* and the *compilers*.

Parallelism in Single Processor Computers

History of processor architecture innovation



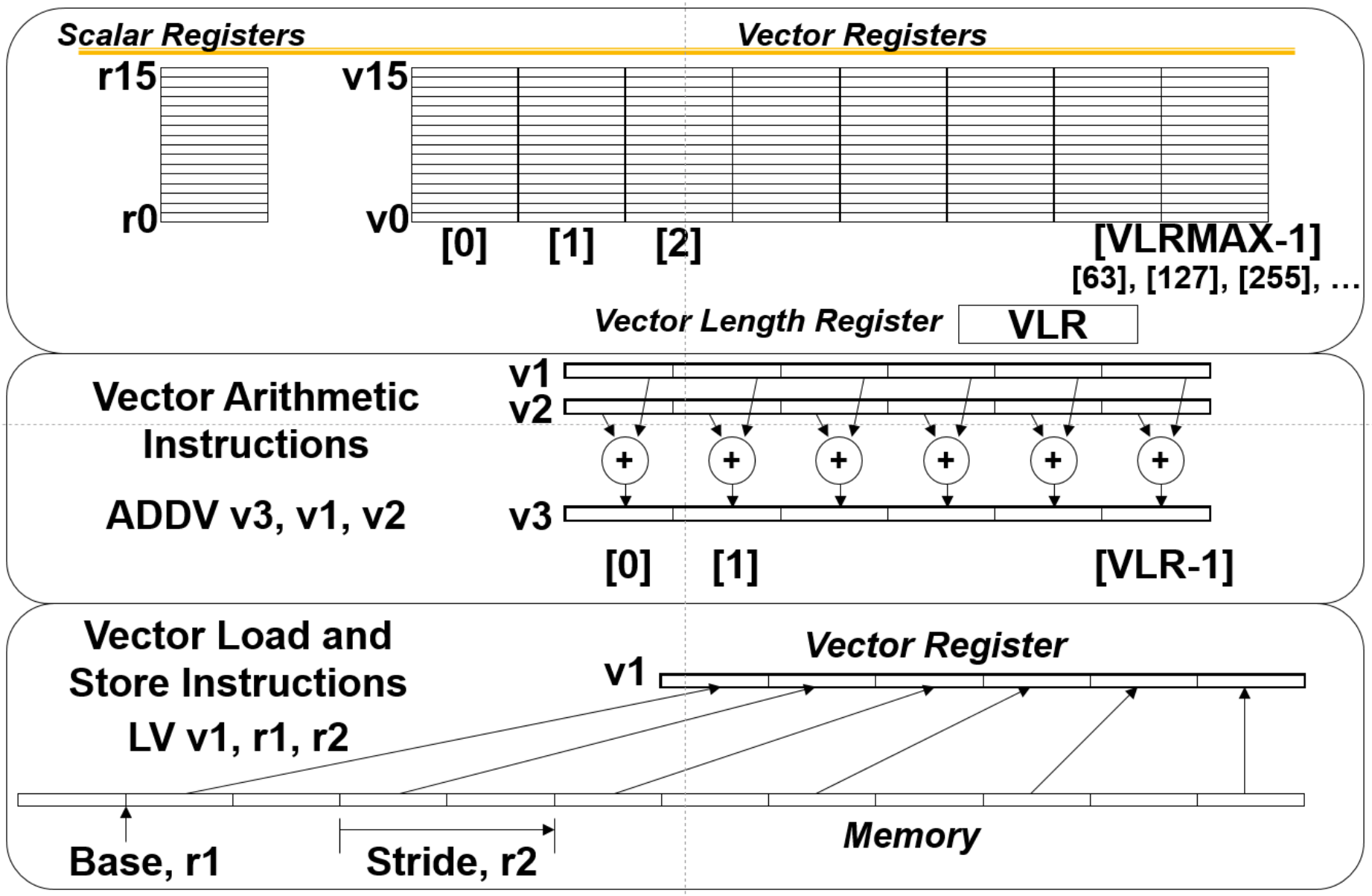
Vector Processing

- Scalar processing
  - Processor instructions operate on scalar values
  - integer registers and floating point registers
- Vectors
  - Set of scalar data
  - Vector registers
    - integer, floating point (typically)
  - Vector instructions operate on vector registers (SIMD)
- Vector unit pipelining
- Multiple vector units
- Vector chaining

Liquid-cooled with inert fluorocarbon

Cray-2

Vector Programming Model



Vector Code Example

```
# C code
for (i=0; i<64; i++)
    C[i] = A[i] + B[i];

! Fortran code
C(1:64) = A(1:64) + B(1:64)
! or
C = A + B
```

```
# Scalar Code
  LI     R4, 64
loop:
  L.D    F0, 0(R1)
  L.D    F2, 0(R2)
  ADD.D  F4, F2, F0
  S.D    F4, 0(R3)
  DADDIU R1, 8
  DADDIU R2, 8
  DADDIU R3, 8
  DSUBIU R4, 1
  BNEZ   R4, loop
```

```
# Vector Code
  LI      VLR, 64
  LV      V1, R1
  LV      V2, R2
  ADDV.D  V3, V1, V2
  SV      V3, R3
```

Vector Instruction Set Advantages

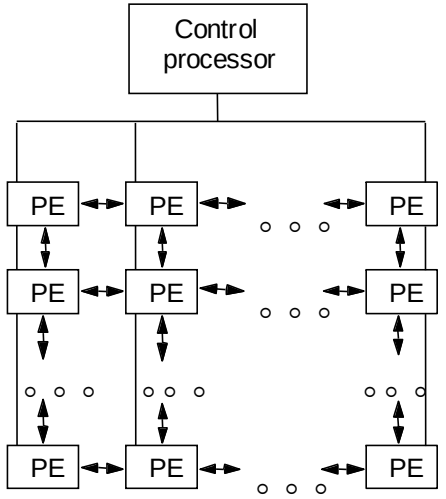
- Compact
  - one short instruction encodes N operations => N*Flop bandwidth
- Expressive, tells hardware that these N operations:
  - are independent
  - use the same functional unit
  - access disjoint registers
  - access registers in the same pattern as previous instructions
  - access a contiguous block of memory (unit-stride load/store)
  - or access memory in a known pattern (strided load/store)
- Scalable
  - can run same object code on more parallel pipelines or *lanes*

Properties of Vector Processors

- Each result independent of previous result
  - => long pipeline, compiler ensures no dependencies
  - => high clock rate
- Vector instructions access memory with known pattern
  - => highly interleaved memory
  - => amortize memory latency of 64-plus elements
  - => no (data) caches required! (but use instruction cache)
- Reduces branches and branch problems in pipelines
- Single vector instruction implies lots of work (≈ loop)
  - => fewer instruction fetches

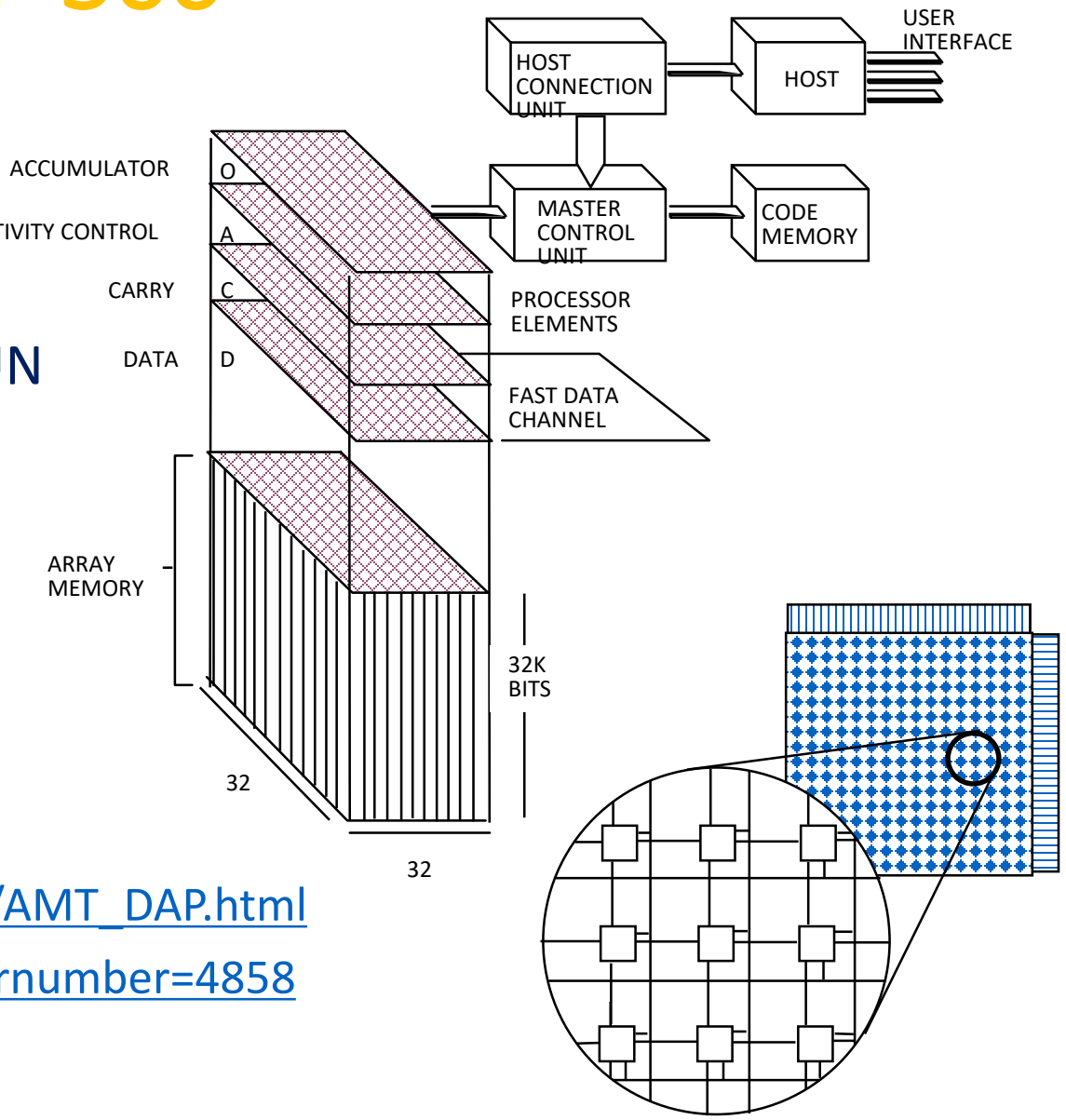
Data Parallel Architectures

- SIMD (Single Instruction Multiple Data)
  - Logical single thread (instruction) of control
  - Processor associated with data elements
- Architecture
  - Array of simple processors with memory
  - Processors arranged in a regular topology
  - Control processor issues instructions
    - All processors execute same instruction (some maybe disabled)
  - Specialized synchronization and communication
  - Specialized reduction operations
  - Array processing



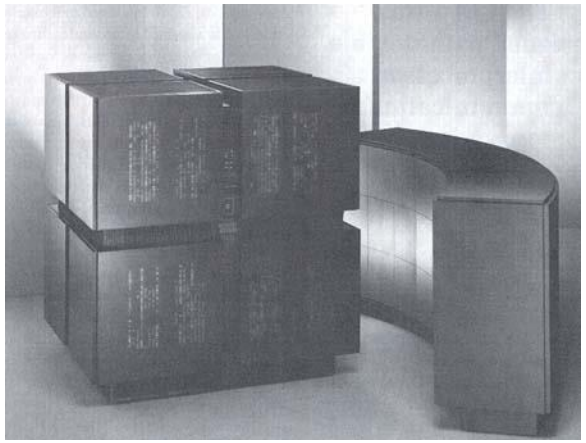
AMT DAP 500

- Applied Memory Technology (AMT)
- Distributed Array Processor (DAP)
  - 32x32 array of bit-organized processors
  - attached to a host computer such as SUN
- ~ 6 – 60 MFLOPS (32-bit)
- first delivered in 1987



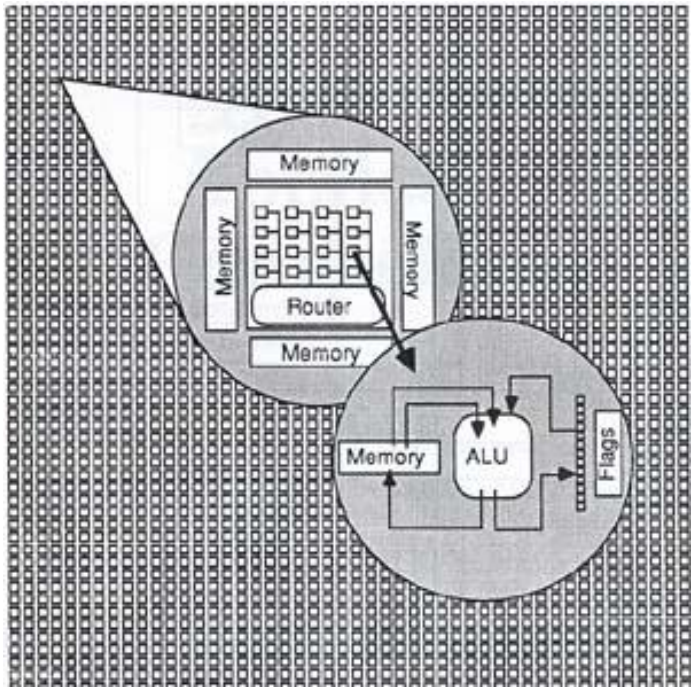
- http://www.computermuseum.org.uk/fixed_pages/AMT_DAP.html
- http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=4858

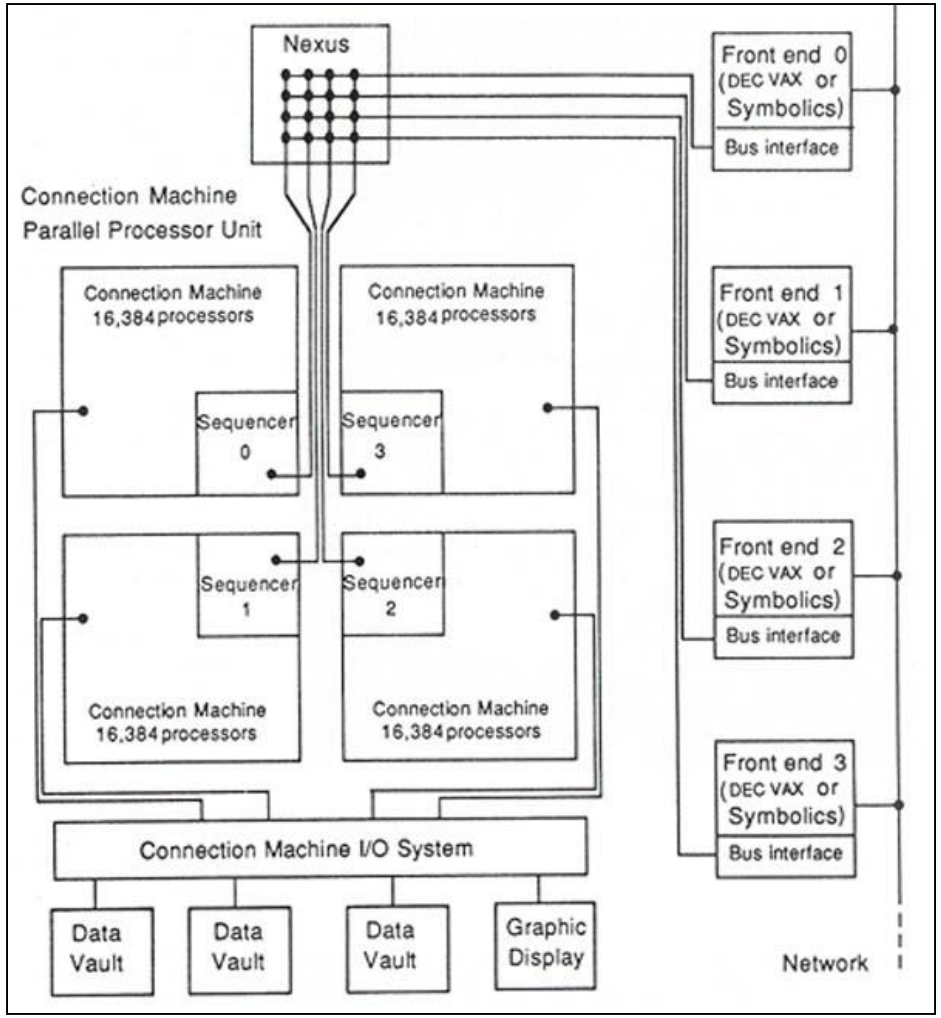
Thinking Machines Connection Machine



16,000 processors!!!

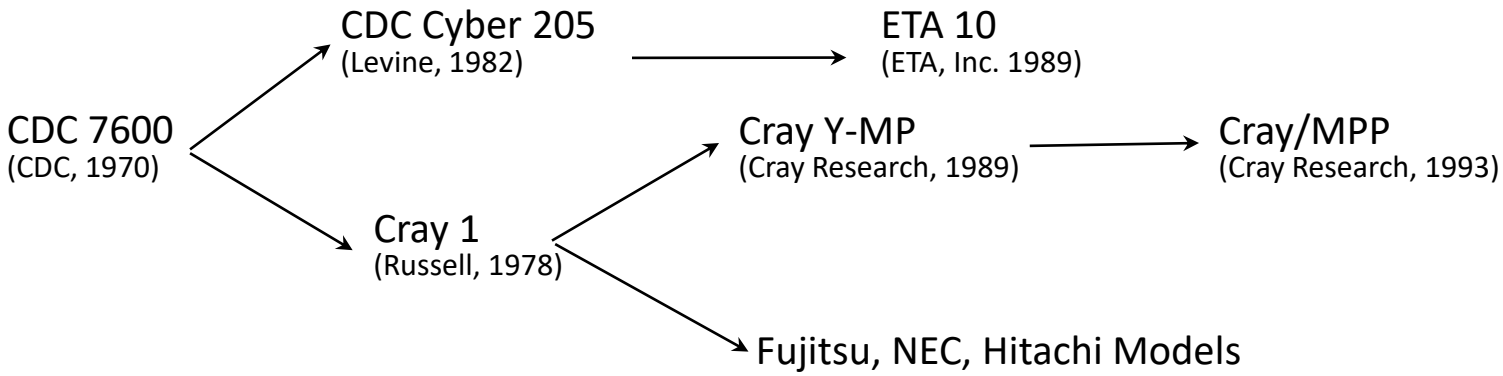
(Tucker, IEEE Computer, Aug. 1988)



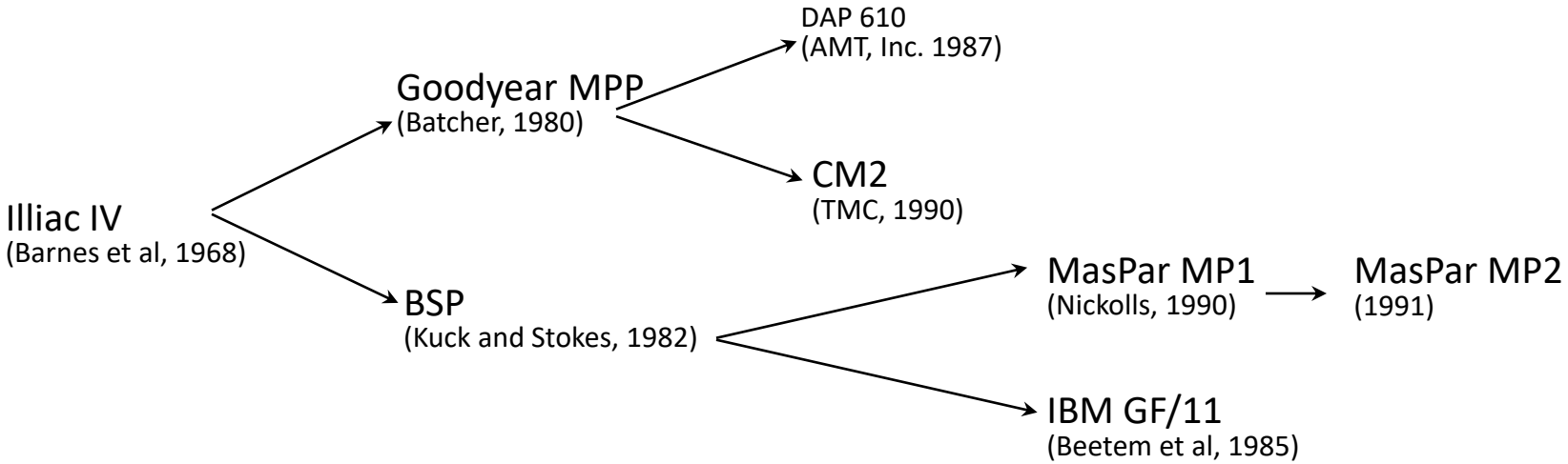


https://en.wikipedia.org/wiki/Connection_Machine

Vector and SIMD Processing Timeline



(a) Multivector track



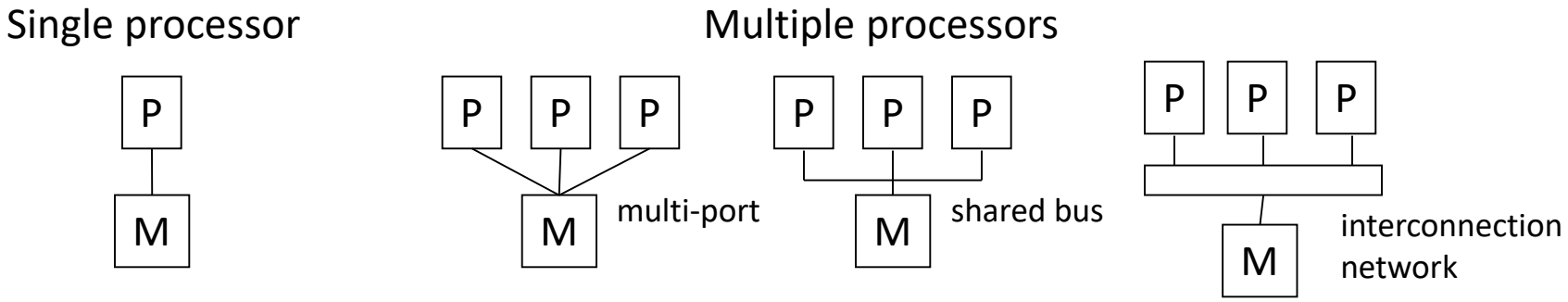
(b) SIMD track

Shared Physical Memory

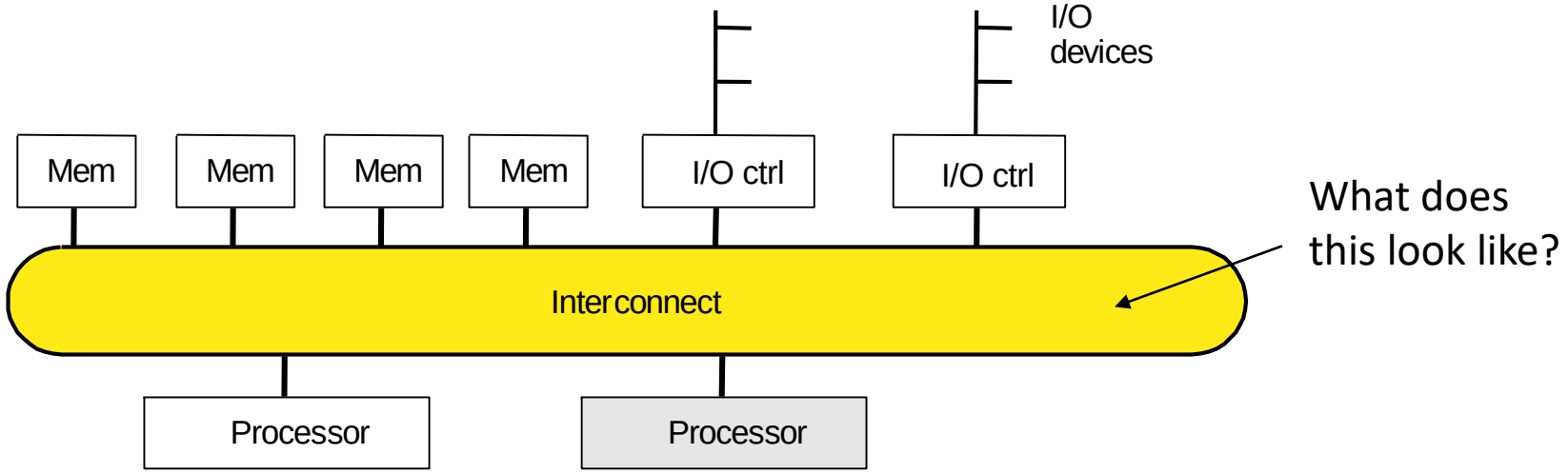
- Add processors to single processor computer system
- Processors *share* computer system resources
  - Memory, storage, …
- Sharing physical memory
  - Any processor can reference any memory location
  - Any I/O controller can reference any memory address
  - Single physical memory address space
- Operating system runs on any processor, or all
  - OS see single memory address space
  - Uses shared memory to coordinate
- Communication occurs as a result of loads and stores

Shared Memory Multiprocessors (SMP)

- Architecture types



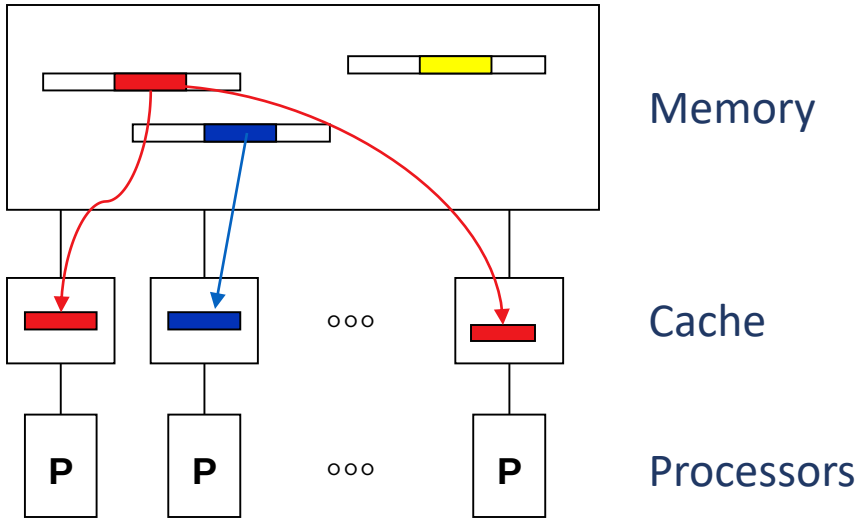
- Differences lie in memory system interconnection



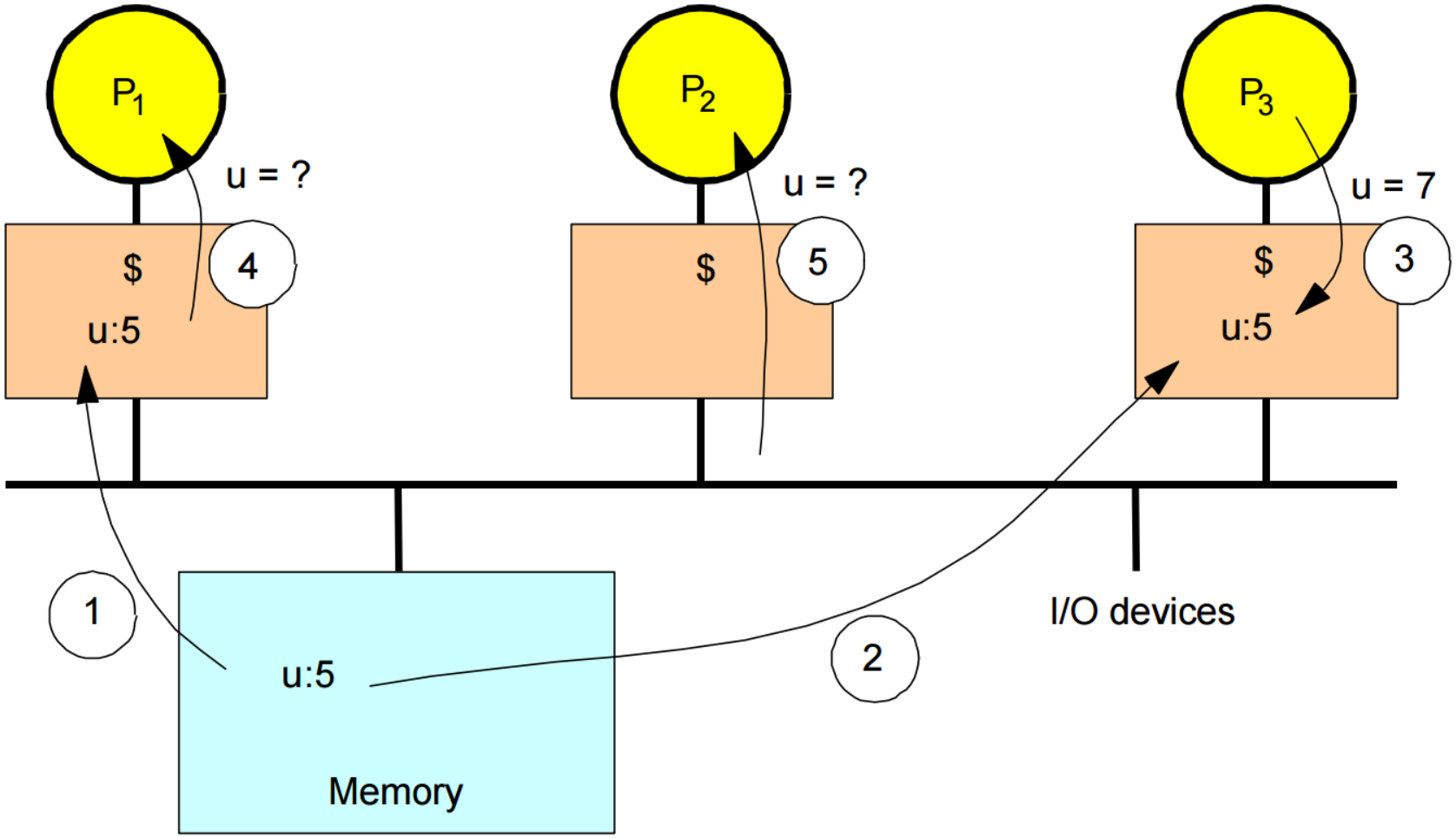
Caching in Shared Memory Systems

- Reduce average latency
  - automatic replication closer to processor
- Increase average bandwidth
- Data is logically transferred from producer to consumer to memory
  - store: reg → mem
  - load: reg ← mem
- Processors can share data efficiently
- What happens when store and load are executed on different processors?

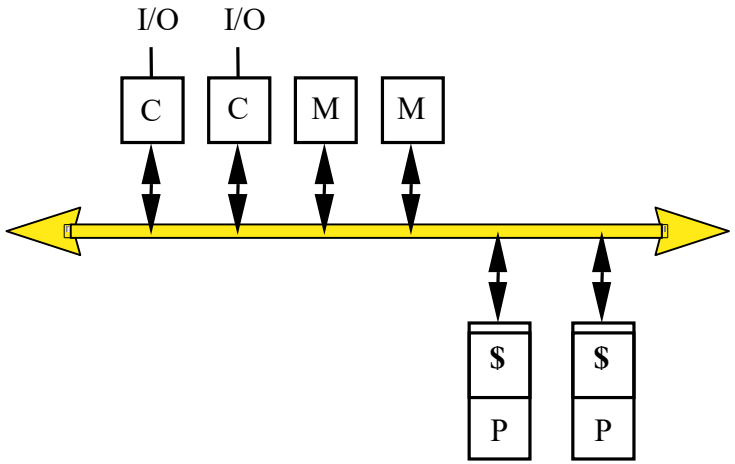
  => Cache coherence problems



Cache Coherence Problem

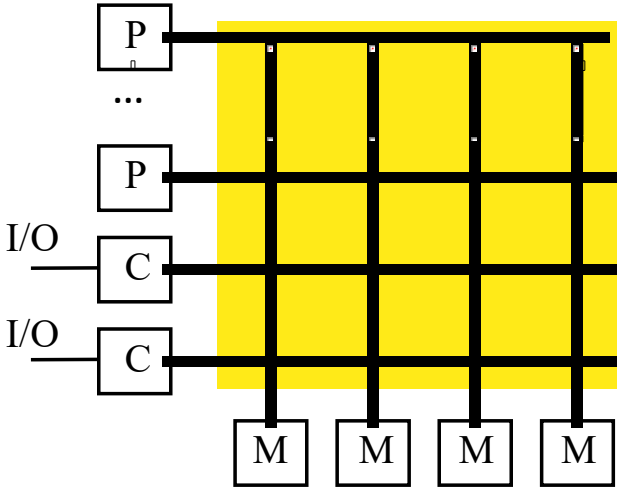


Bus-based SMP



- Memory bus handles all memory read/write traffic
- Processors share bus
- *Uniform Memory Access* (*UMA*)
  - Memory (not cache) uniformly equidistant
  - Take same amount of time (generally) to complete
- May have multiple memory modules
  - Interleaving of physical address space
- Caches introduce memory hierarchy
  - Lead to data consistency problems
  - Cache coherency hardware necessary (*CC-UMA*)

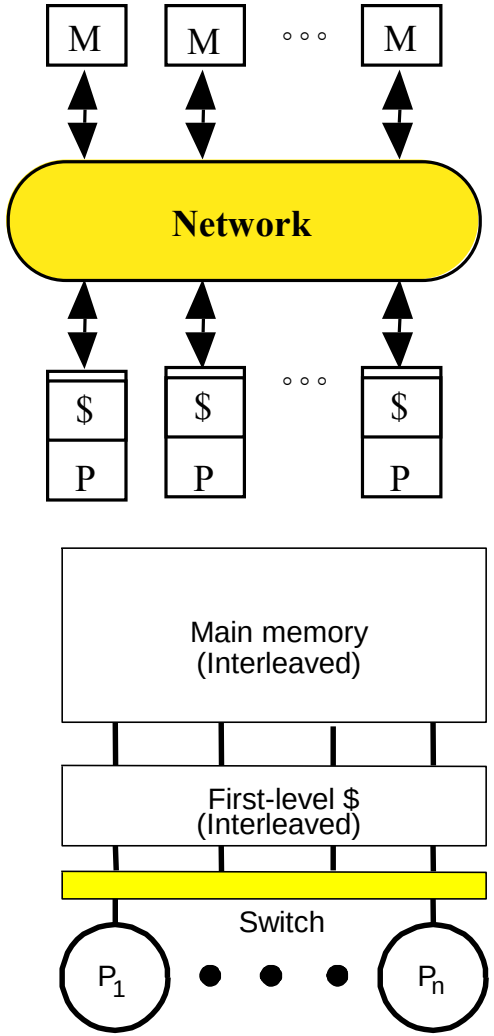
Crossbar SMP



- Replicates memory bus for every processor and I/O controller
  - Every processor has direct path
- UMA SMP architecture
- Can still have cache coherency issues
- Multi-bank memory or interleaved memory
- Advantages
  - Bandwidth scales linearly (no shared links)
- Problems
  - High incremental cost (cannot afford for many processors)
  - One solution is to use switched multi-stage interconnection network

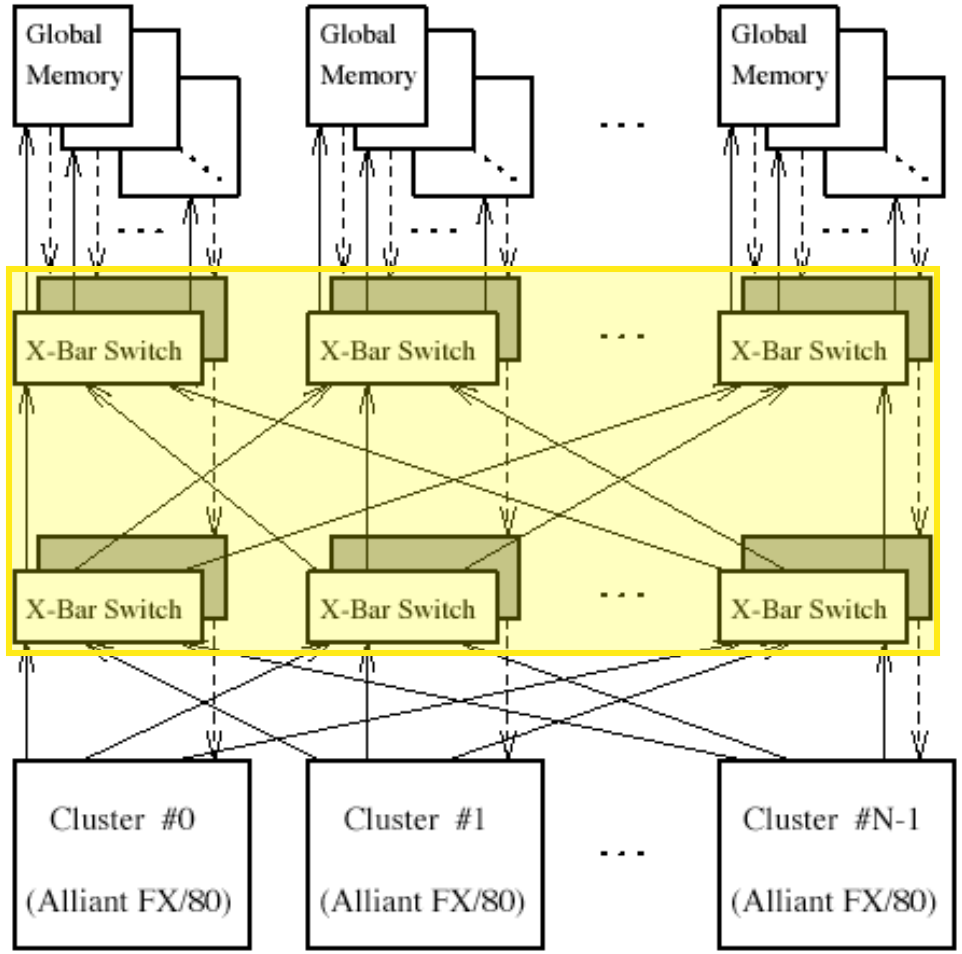
“Dance Hall” SMP and Shared Cache SMP

- “Dance Hall” SMP
  - Interconnection network connects processors to memory
  - Centralized memory (UMA)
  - Network determines performance
    - Continuum from bus to crossbar
    - Scalable memory bandwidth
  - Memory is physically separated from processors
  - Could have cache coherence problems
- Shared cache SMP reduces coherence problem and provides fine grained data sharing

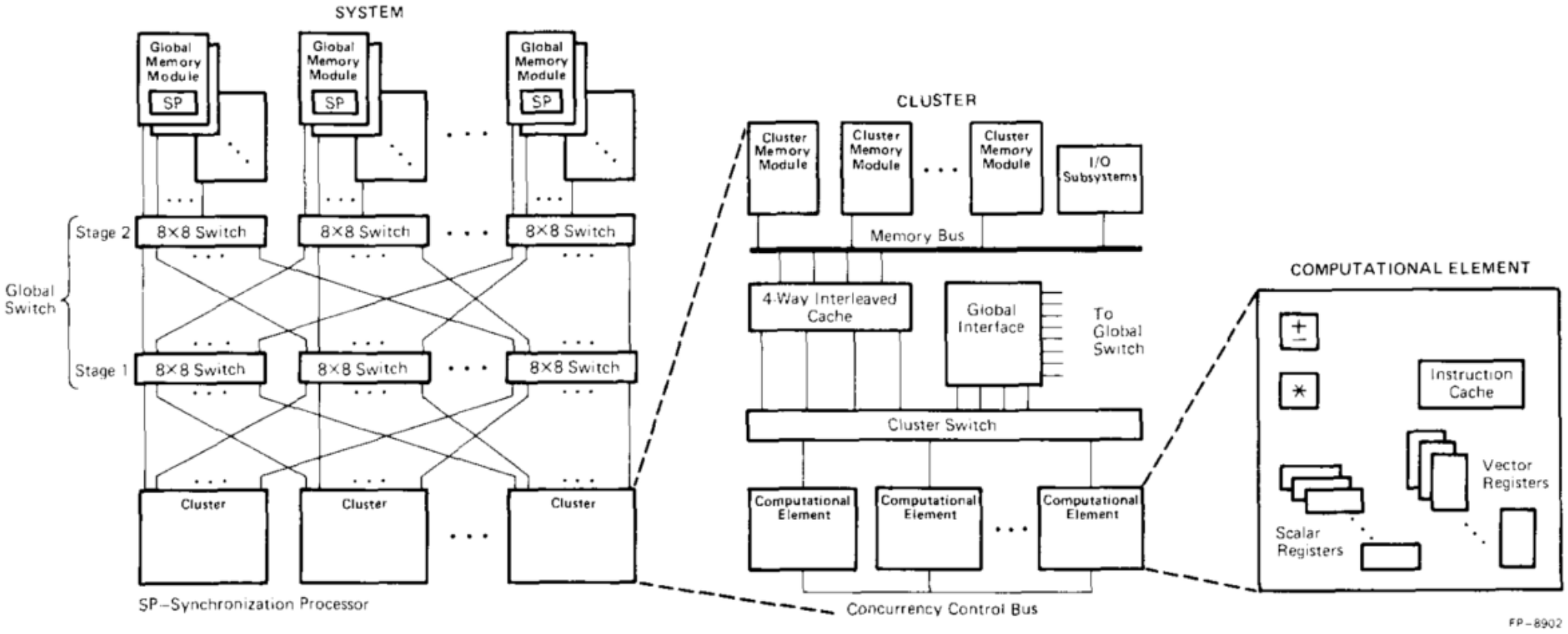


University of Illinios CSRD Cedar Machine

- Center for Supercomputing Research and Development
- Multi-cluster scalable parallel computer
- Alliant FX/80
  - 8 processors w/ vectors
  - Shared cache
  - HW synchronization
- Omega switching network
- Shared global memory
- SW-based global memory coherency
- 1988

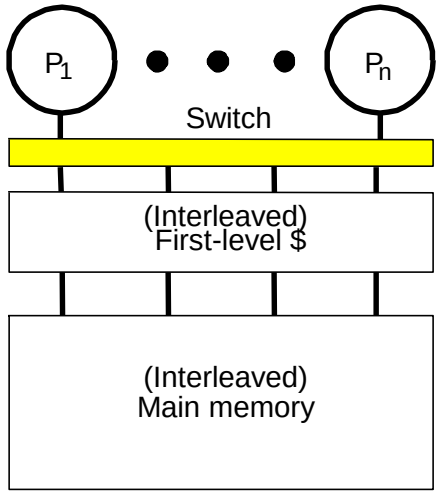


Organization of the Cedar Parallel Processor

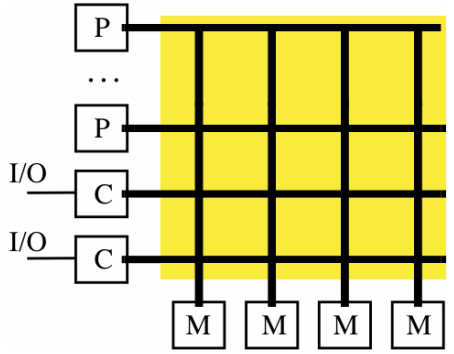


http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=10362

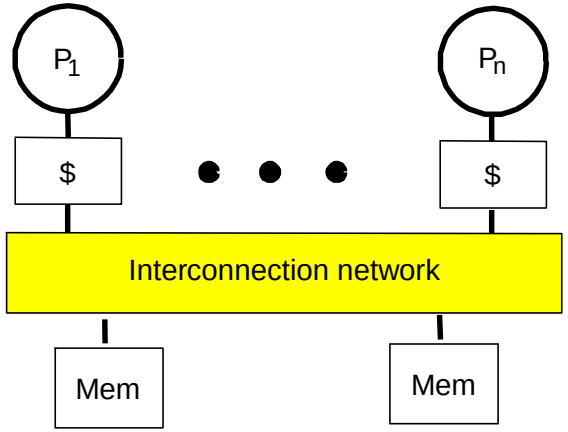
Natural Extensions of the Memory System



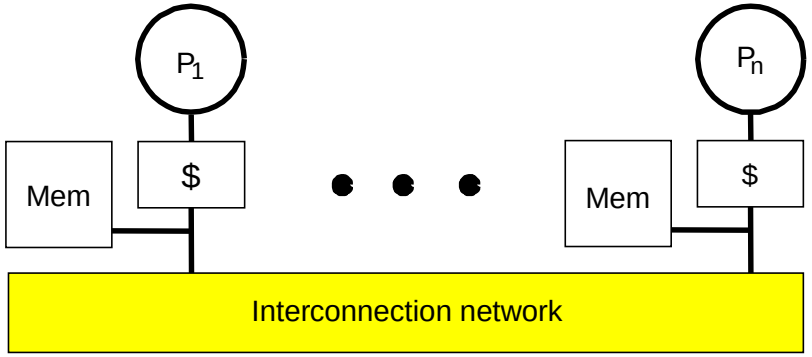
(a) Shared Cache



(b) Crossbar, Interleaved

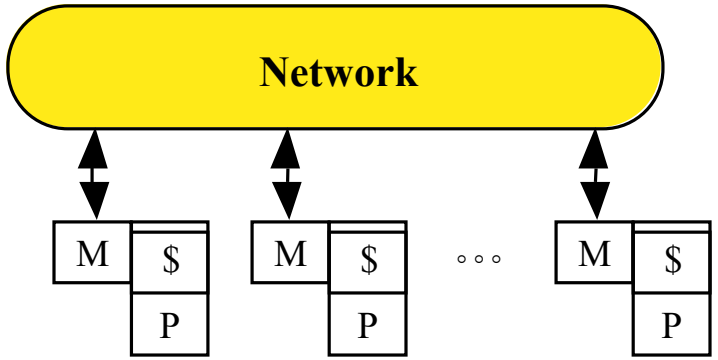


(c) Centralized Memory Dance Hall, UMA



(d) Distributed Memory (NUMA)

Non-Uniform Memory Access (NUMA) SMPs



- Distributed memory
- Memory is physically resident close to each processor
- Memory is still *shared*
- *Non-Uniform Memory Access* (NUMA)
  - Local memory and remote memory
  - Access to local memory is faster, remote memory slower
  - Access is non-uniform
  - Performance will depend on data locality
- Interconnection network architecture is more scalable
- Cache coherency is still an issue (more serious)

Definitions

- Memory operations (load, store, read-modify-write, …)
- Processor perspective
  - Write: subsequent reads return the value
  - Read: subsequent writes cannot affect the value
- *Coherent memory system*
  - There exists a serial order of memory operations on each location such that
    - operations issued by a process appear in order issued
    - value returned by each read is that written by previous write

    $\Rightarrow$ write propagation + write serialization

Motivation for Memory Consistency

- Coherence implies that writes to a location become visible to all processors in the same order
- But when does a write become visible?
- How do we establish orders between a write and a read by different processors?
  - Use event synchronization
- Implement hardware protocol for cache coherency
- Protocol will be based on model of memory consistency

Memory Consistency

- Specifies constraints on the order in which memory operations (from any process) can appear to execute with respect to each other
  - What orders are preserved?
  - Given a load, constrains the possible values returned by it
- Implications for both programmer and system designer
  - Programmer uses to reason about correctness
  - System designer can use to constrain how much accesses can be reordered by compiler or hardware
- Contract between programmer and system
- Need coherency systems to enforce memory consistency

Sequential Consistency

- Total order achieved by interleaving accesses from different processes
  - Maintains *program order*
  - Memory operations (from all processes) appear to issue, execute, and complete atomically with respect to others
  - As if there was a single memory (no cache)

"A multiprocessor is sequentially consistent if the result of any execution is the same as if the operations of all the processors were executed in some sequential order, and the operations of each individual processor appear in this sequence in the order specified by its program." [Lamport, 1979]

Sequential Consistency (Sufficient Conditions)

- There exist a total order consistent with the memory operations becoming visible in program order
- Sufficient Conditions
  - every process issues memory operations in program order
  - after write operation is issued, the issuing process waits for write to complete before issuing next memory operation (atomic writes)
  - after a read is issued, the issuing process waits for the read to complete and for the write whose value is being returned to complete (globally) before issuing its next memory operation
- Cache-coherent architectures implement consistency

Requirements of a Cache Coherent System

- Provide set of states, state transition diagram, and actions
- Manage coherence protocol
  - (0) Determine when to invoke coherence protocol
  - (a) Find info about state of block in other caches to determine action
  - (b) Locate the other copies
  - (c) Communicate with those copies (invalidate/update)
- (0) is done the same way on all systems
  - state of the line is maintained in the cache
  - protocol is invoked if an “access fault” occurs on the line
- Different approaches distinguished by (a) to (c)

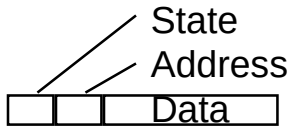
Bus-based Cache Coherence

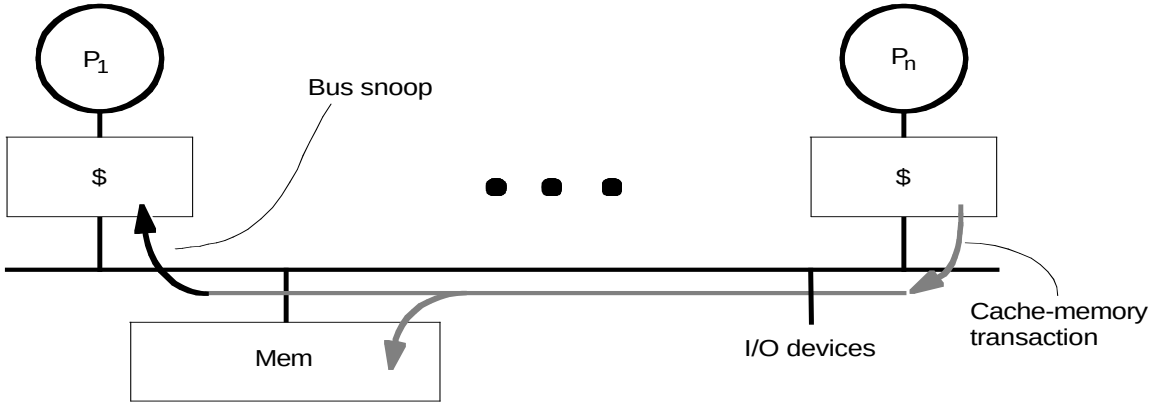
- All of (a), (b), (c) done through broadcast on bus
  - faulting processor sends out a “search”
  - others respond to the search probe and take necessary action
- Could do it in scalable network too
- Conceptually simple, but broadcast doesn’t scale
  - on bus, bus bandwidth doesn’t scale
  - on scalable network, every fault leads to at least p network transactions
- Scalable coherence:
  - can have same cache states and state transition diagram
  - different mechanisms to manage protocol

Bus-based Cache-Coherent Architecture

- Bus Transactions
  - Single set of wires connect several devices
  - Bus protocol: arbitration, command/addr, data
  - Every device observes every transaction
- Cache block state transition diagram
  - FSM (Finite Series Model) specifying how disposition of block changes
    - invalid, valid, dirty
  - *Snoopy protocol*
- Basic Choices
  - Write-through vs Write-back
  - Invalidate vs. Update

Snoopy Cache-Coherency Protocols





- Bus is a broadcast medium
- Caches know what they have
- Cache controller "snoops" all transactions on shared bus
  - relevant transaction if for a block its cache contains
  - take action to ensure coherence
    - invalidate, update, or supply value
  - depends on state of the block and the protocol

Basic Snoopy Protocols

- Write Invalidate :
  - Multiple readers, single writer
  - Write to shared data: an invalidate is sent to all caches
  - Read Miss:
    - Write-through: memory is always up-to-date
    - Write-back: snoop in caches to find most recent copy
- Write Broadcast (typically write through):
  - Write to shared data: broadcast on bus, snoop and update
- Write serialization: bus serializes requests!
- Write Invalidate versus Broadcast

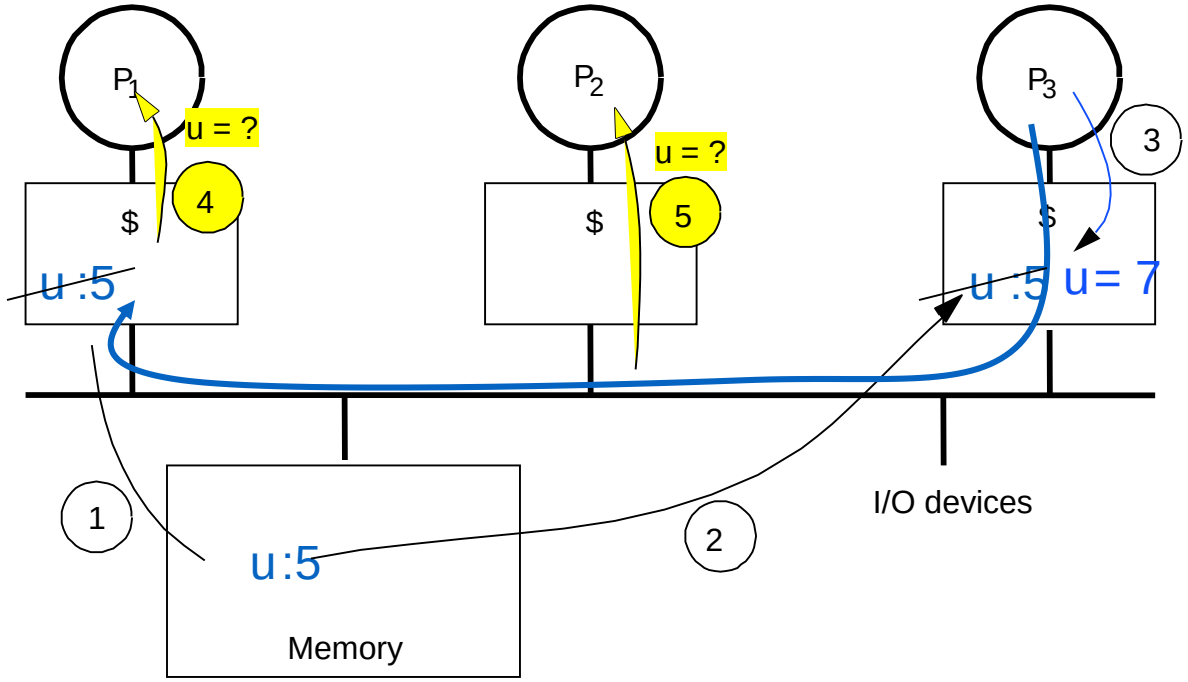
Snooping Cache Variations

Basic Protocol	Berkeley Protocol	Illinois Protocol	MESI protocol
Exclusive Shared Invalid	Owned Exclusive Owned Shared Shared Invalid	Private Dirty Private Clean Shared Invalid	Modified (private,!=Memory) Exclusive (private,=Memory) Shared (shared,=Memory) Invalid

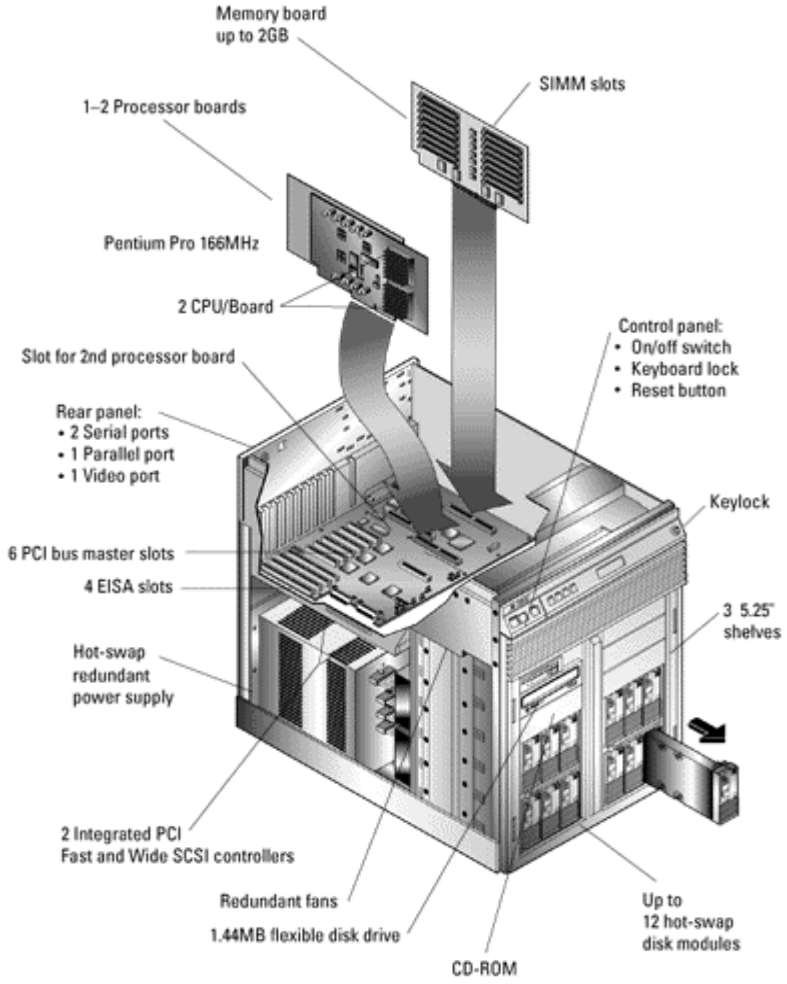
Owner can update via bus invalidate operation
Owner must write back when replaced in cache

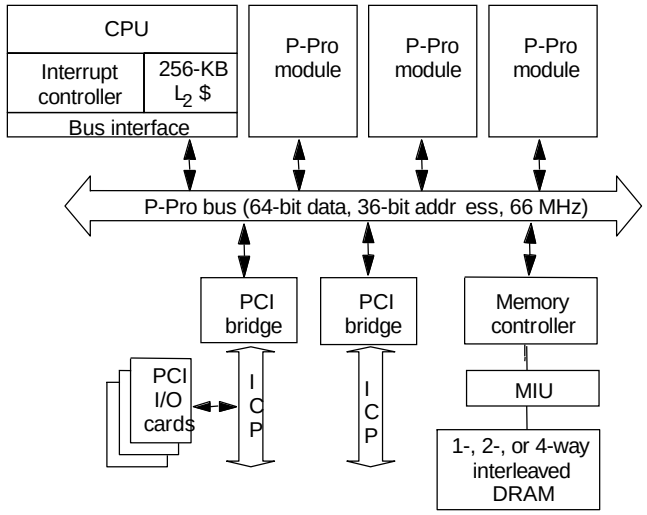
If read sourced from memory, then Private Clean
if read sourced from other cache, then Shared
Can write in cache if held private clean or dirty

Example: Write-back Invalidate



Intel Pentium Pro Quad Processor



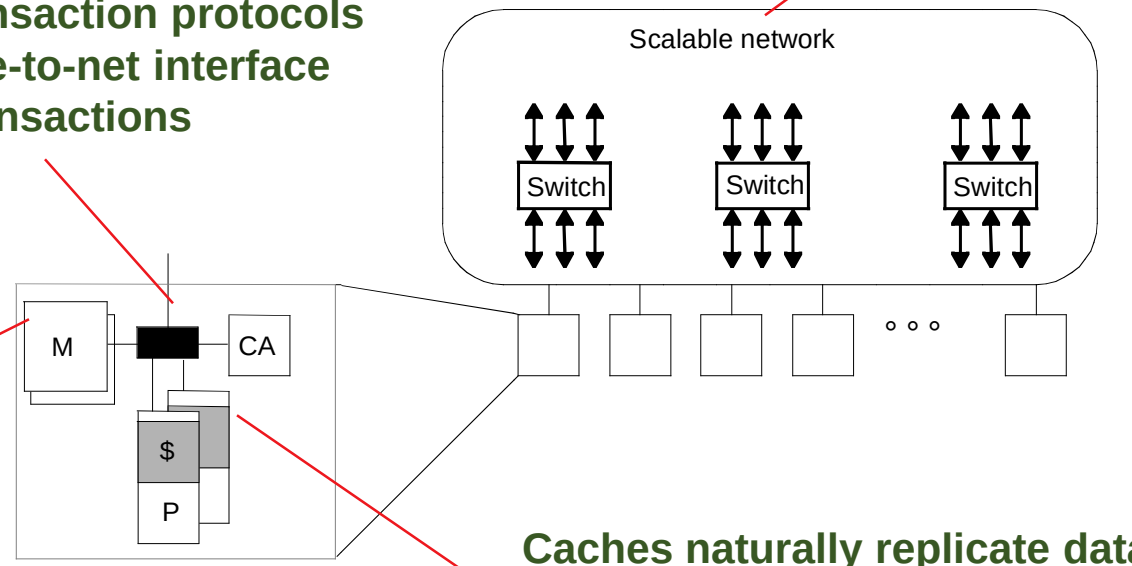


- All coherence and multiprocessing glue in processor module
- Highly integrated, targeted at high volume
- Low latency and bandwidth
- 1995 - 1998

Context for Scalable Cache Coherence

Scalable Networks
- many simultaneous transactions

Realizing programming models through net transaction protocols
- efficient node-to-net interface
- interprets transactions

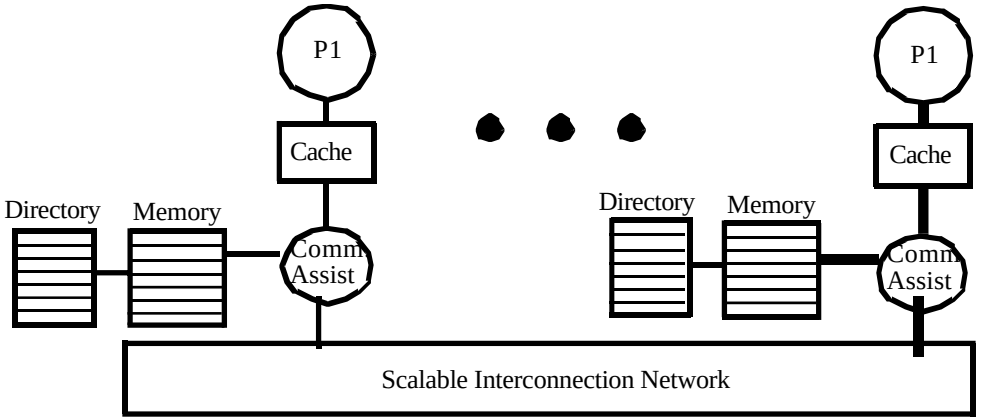


Scalable distributed memory

Caches naturally replicate data
- coherence through bus
- snooping protocols
- consistency

Need cache coherence protocols that scale!
- no broadcast or single point of order

Generic Solution: Directories

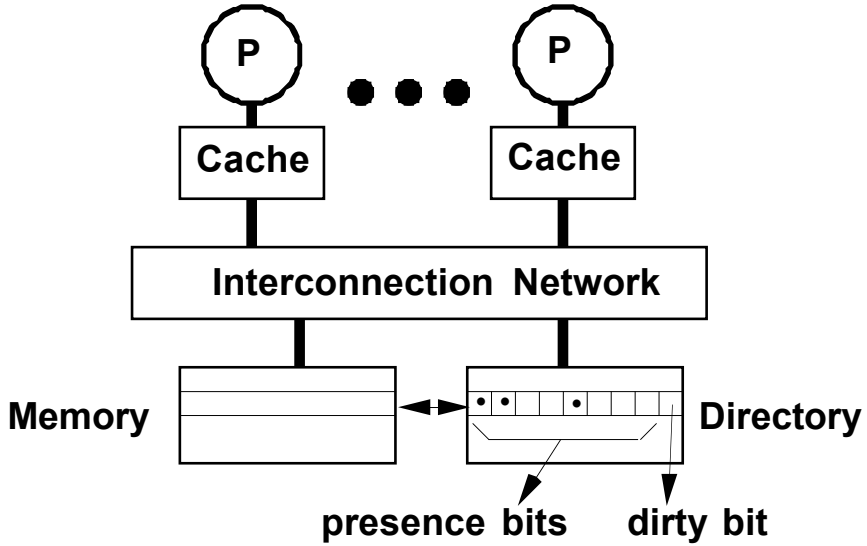


- Maintain state vector explicitly
  - associate with memory block
  - record state of block in each cache
- On miss, communicate with directory
  - determine location of cached copies
  - determine action to take
  - conduct protocol to maintain coherence

Scalable Approach: Directories

- Every memory block has associated directory information
  - keeps track of copies of cached blocks and their states
  - on a miss, find directory entry, look it up, and communicate only with the nodes that have copies if necessary
  - in scalable networks, communication with directory and copies is through network transactions
- Many alternatives for organizing directory information
- https://en.wikipedia.org/wiki/Directory-based_cache_coherence

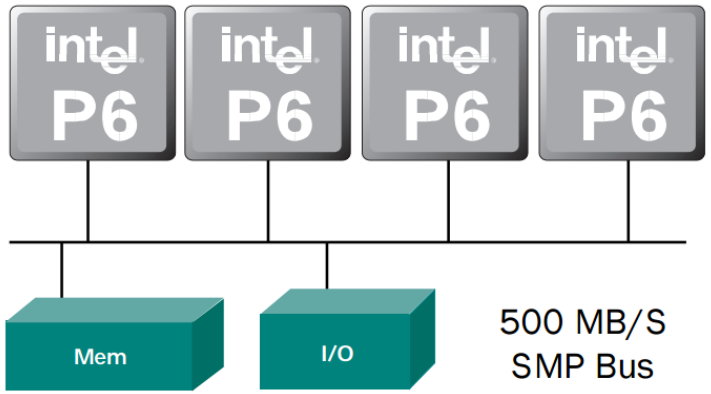
Basic Operation of Directory

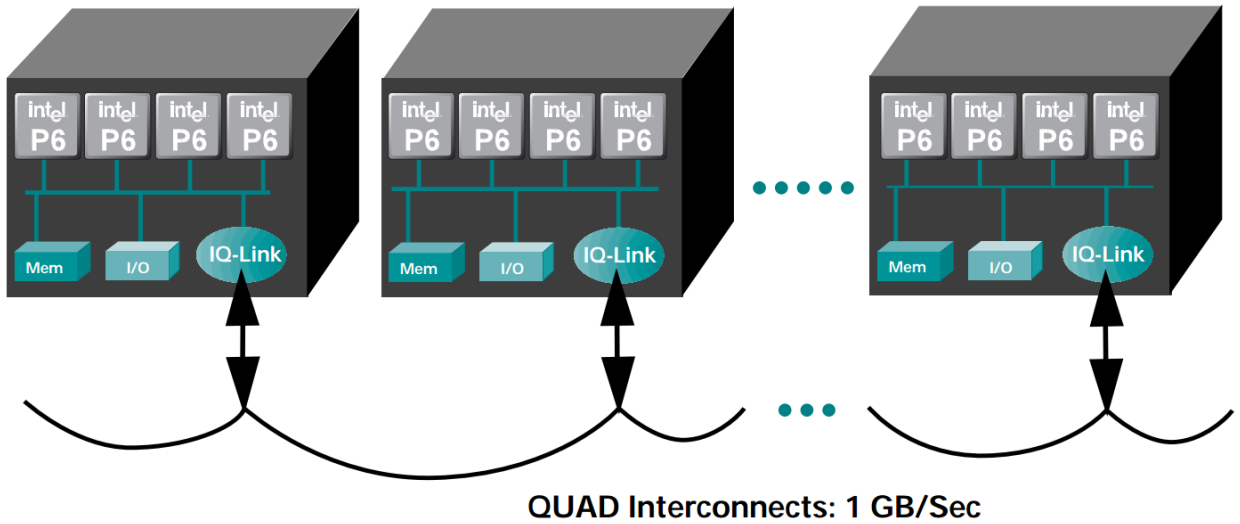


- k processors
- Each cache-block in memory
  - k presence bits and 1 dirty bit
- Each cache-block in cache
  - 1 valid bit and 1 dirty (owner) bit

- Read from memory
  - dirty bit ON
- Write to memory
  - dirty bit OFF

Sequent NUMA-Q



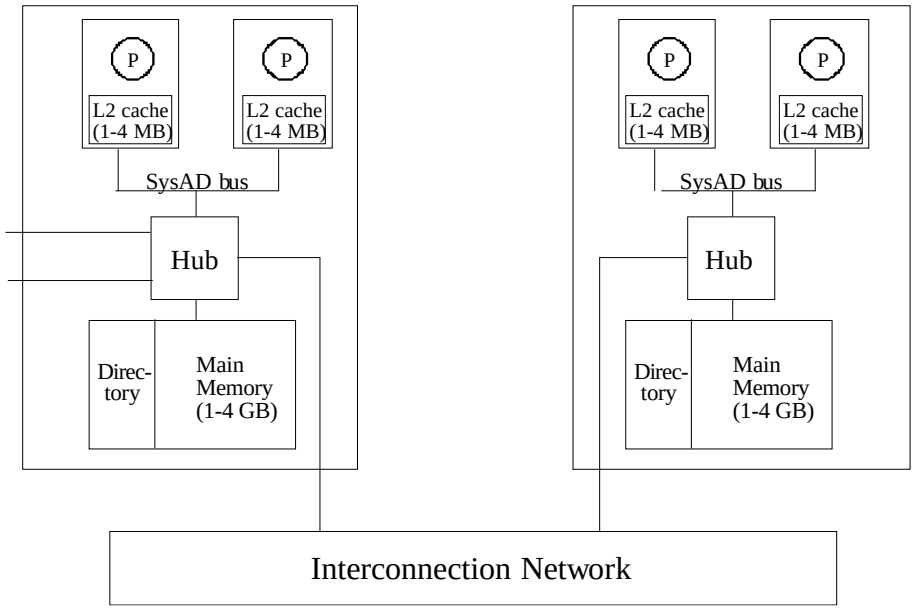


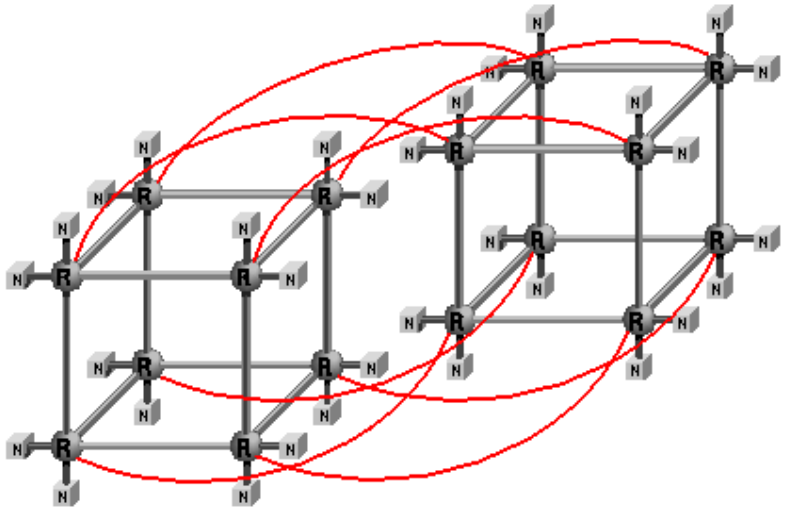
- Ring-based SCI (Scalable Coherent Interface) network
  - 1 GB/second
  - Built-in coherency
- Commodity SMPs as building blocks
  - Extend coherency mechanism
- Split transaction bus
- 1996

https://parallel.ru/sites/default/files/ftp/computers/sequent/Numa_Arch_wp.pdf

SGI Origin 2000

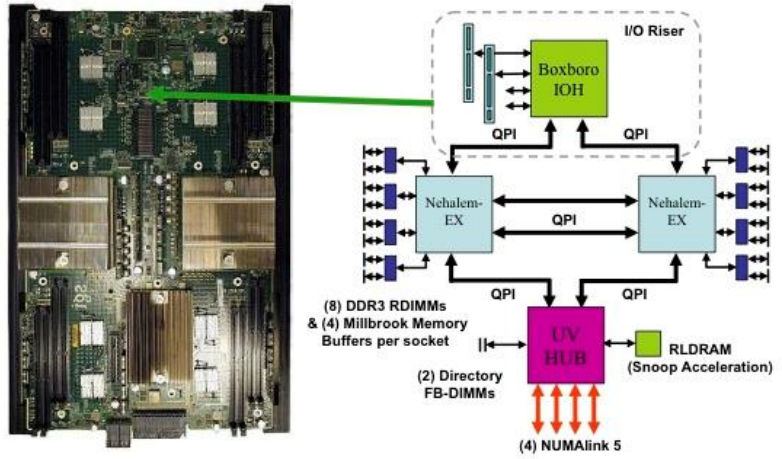
- Scalable shared memory multiprocessor
- MIPS R10000 CPUs @ 250MHz
- NUMAlink router
- Directory-based cache coherency (MESI)
- ASCI Blue Mountain (at LANL)
  - 3.072 TFLOPS
  - #2 in June 1999

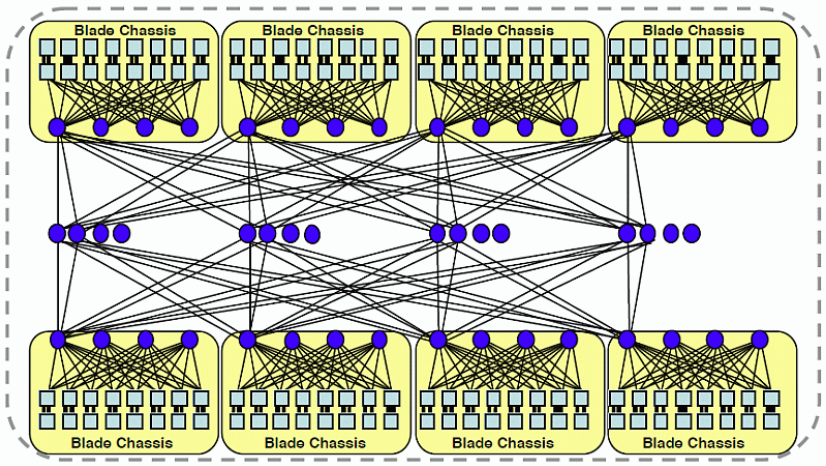




SGI UV

- Latest generation scalable shared memory architecture
- Scaling up to 256 sockets
  - Intel Xeon E5-4600 v3 processors
- Architectural provisioning for up to 262,144 cores
- Up to 64TB of cache-coherent shared memory in a single system image (SSI)
- NUMAlink 6 interconnect (6.7GB/s bidirectional)

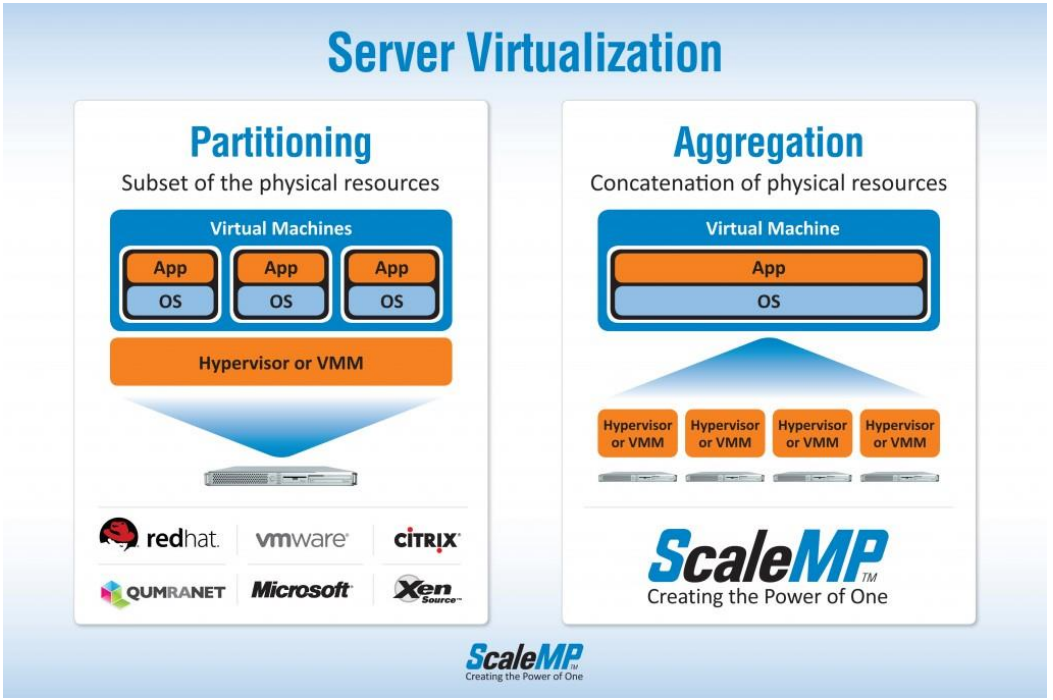




https://www.sgi.com/pdfs/4555.pdf

Virtual SMP

- Use *hypervisor* to combine a cluster of servers into a *virtual* Shared Memory Multiprocessor (SMP) system
  - Virtualization for Aggregation
  - Very large coherent memory
  - Single System Image (SSI)
- ScaleMP vSMP
  - Versatile SMP (vSMP) Architecture
  - Requires InfiniBand infrastructure
- TidalScale
  - https://www.youtube.com/watch?v=f-ug6B6ycng
  - Ike Nassi, the founder, is a UCSC adjunct professor

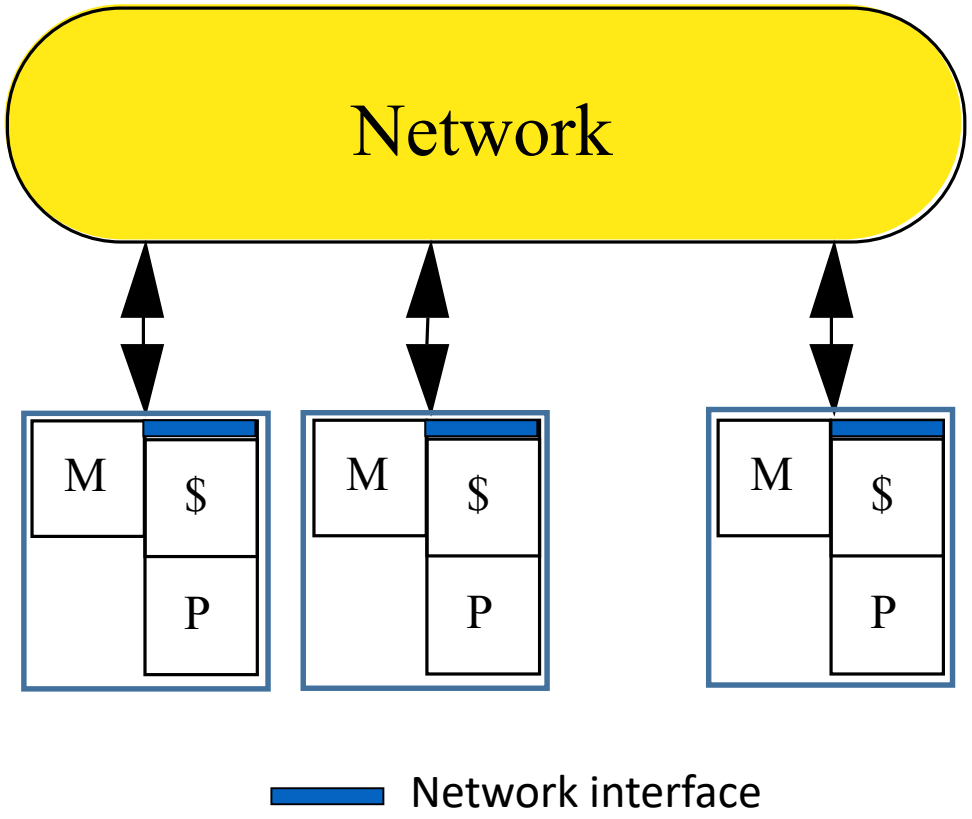


Distributed Memory Multiprocessors

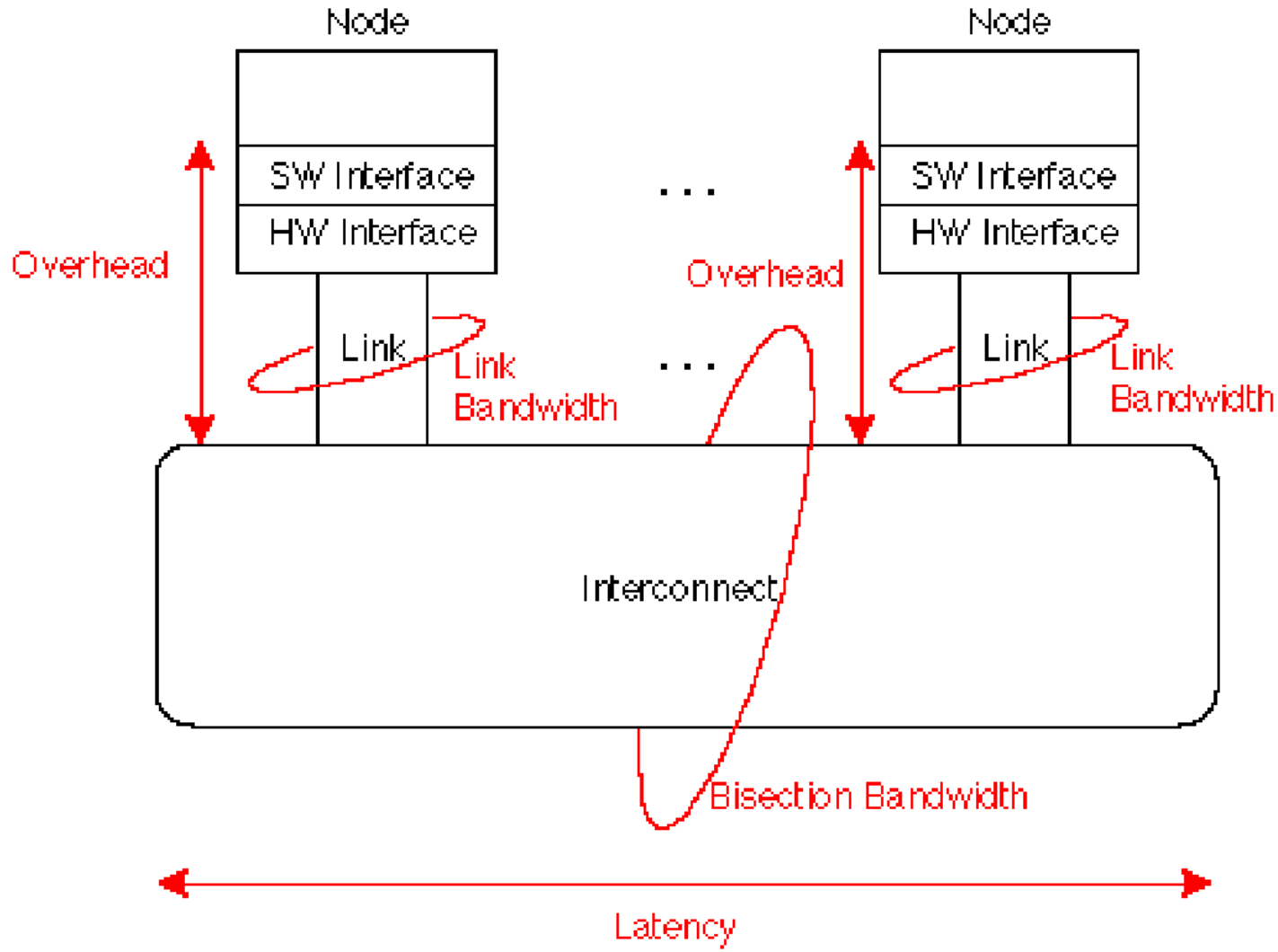
- Each processor has a local memory
  - Physically separated memory address space
- Processors must communicate to access non-local data
  - Message communication (message passing)
    - *Message passing architecture*
  - Processor interconnection network
- Parallel applications must be partitioned across
  - Processors: execution units
  - Memory: data partitioning
- Scalable architecture
  - Small incremental cost to add hardware (cost of node)

Distributed Memory Architecture

- Nodes are complete computer systems
  - Including I/O
- Nodes communicate via interconnection network
  - Standard networks
  - Specialized networks
- Network interfaces
- Easier to build



Network Performance Measures



Overhead: latency of interface vs. Latency: network

Performance Metrics: Latency and Bandwidth

- Bandwidth
  - Need high bandwidth in communication
  - Match limits in network, memory, and processor
  - Network interface speed vs. network bisection bandwidth
- Latency
  - Performance affected since processor may have to wait
  - Harder to overlap communication and computation
  - Overhead to communicate is a problem in many machines
- Latency hiding
  - Increases programming system burden
  - Examples: communication/computation overlaps, prefetch

Scalable, High-Performance Interconnect

- Interconnection network is core of parallel architecture
- Requirements and tradeoffs at many levels
  - Elegant mathematical structure
  - Deep relationship to algorithm structure
  - Hardware design sophistication
- Little consensus
  - Performance metrics?
  - Cost metrics?
  - Workload?
  - …

What Characterize an Interconnection Network?

- Topology (what)
  - Interconnection structure of the network graph
- Routing Algorithm (which)
  - Restricts the set of paths that messages may follow
  - Many algorithms with different properties
- Switching Strategy (how)
  - How data in a message traverses a route
  - *circuit switching* vs. *packet switching*
- Flow Control Mechanism (when)
  - When a message or portions of it traverse a route
  - What happens when traffic is encountered?

Topological Properties

- Routing distance
  - Number of links on route from source to destination
- Diameter
  - Maximum routing distance
- Average distance
- Partitioned network
  - Removal of links resulting in disconnected graph
  - Minimal cut
- Scaling increment
  - What is needed to grow the network to next valid degree

Interconnection Topologies

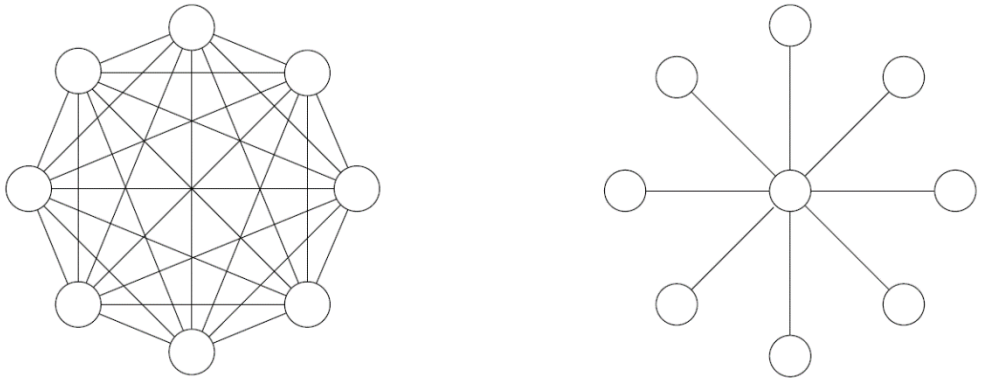


Figure 2.14 (a) A completely-connected network of eight nodes; (b) a Star connected network of nine nodes.

Introduction to Parallel Computing, Ananth Grama, et al., Addison Wesley, 2nd Ed., 2003

Interconnection Topologies (cont'd)

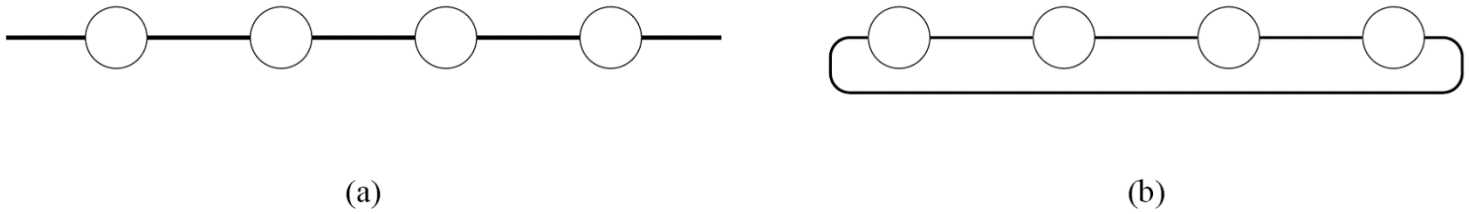


Figure 2.15 Linear arrays: (a) with no wraparound links; (b) with wraparound link.

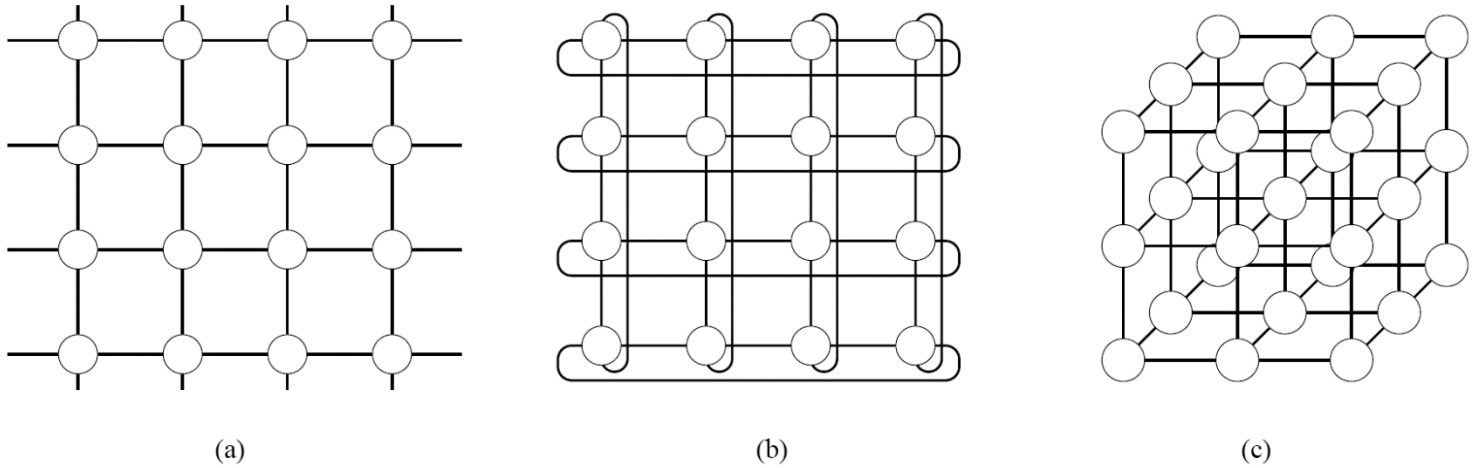
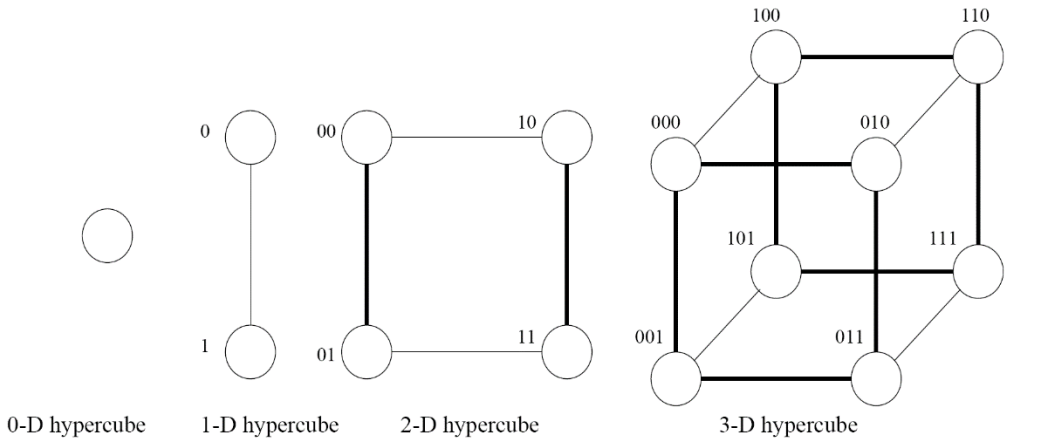


Figure 2.16 Two and three dimensional meshes: (a) 2-D mesh with no wraparound; (b) 2-D mesh with wraparound link (2-D torus); and (c) a 3-D mesh with no wraparound.

Introduction to Parallel Computing, Ananth Grama, et al., Addison Wesley, 2nd Ed., 2003

Interconnection Topologies (cont'd)



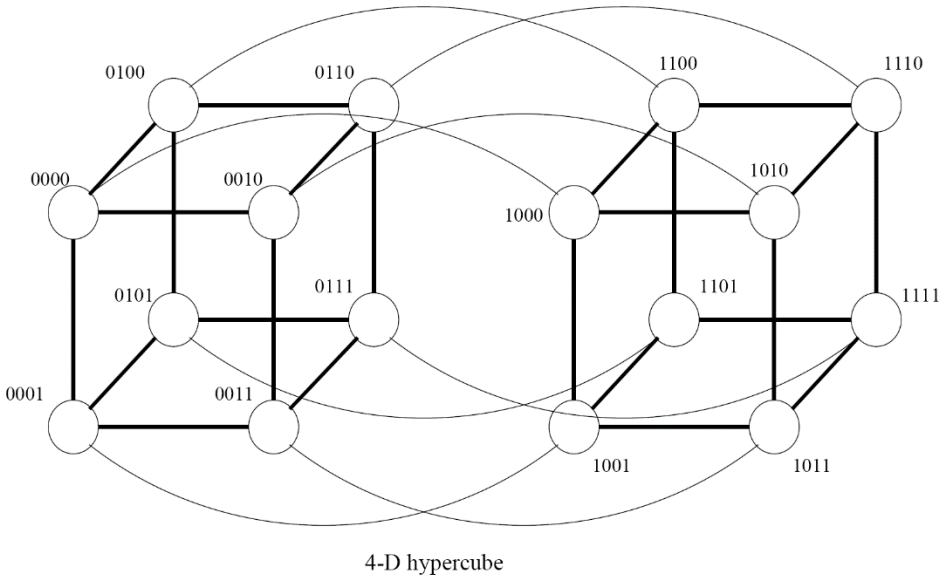


Figure 2.17 Construction of hypercubes from hypercubes of lower dimension.

Introduction to Parallel Computing, Ananth Grama, et al., Addison Wesley, 2nd Ed., 2003

Interconnection Topologies (cont'd)

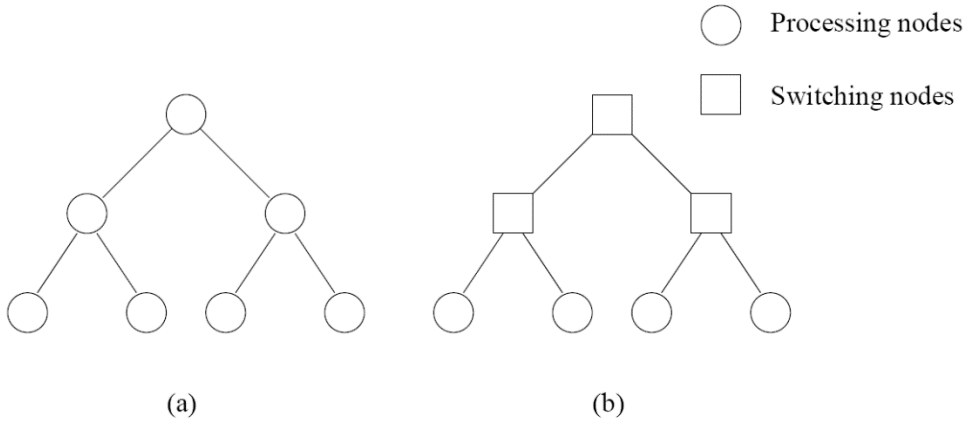


Figure 2.18 Complete binary tree networks: (a) a static tree network; and (b) a dynamic tree network.

Introduction to Parallel Computing, Ananth Grama, et al., Addison Wesley, 2nd Ed., 2003

Interconnection Topologies (cont'd)

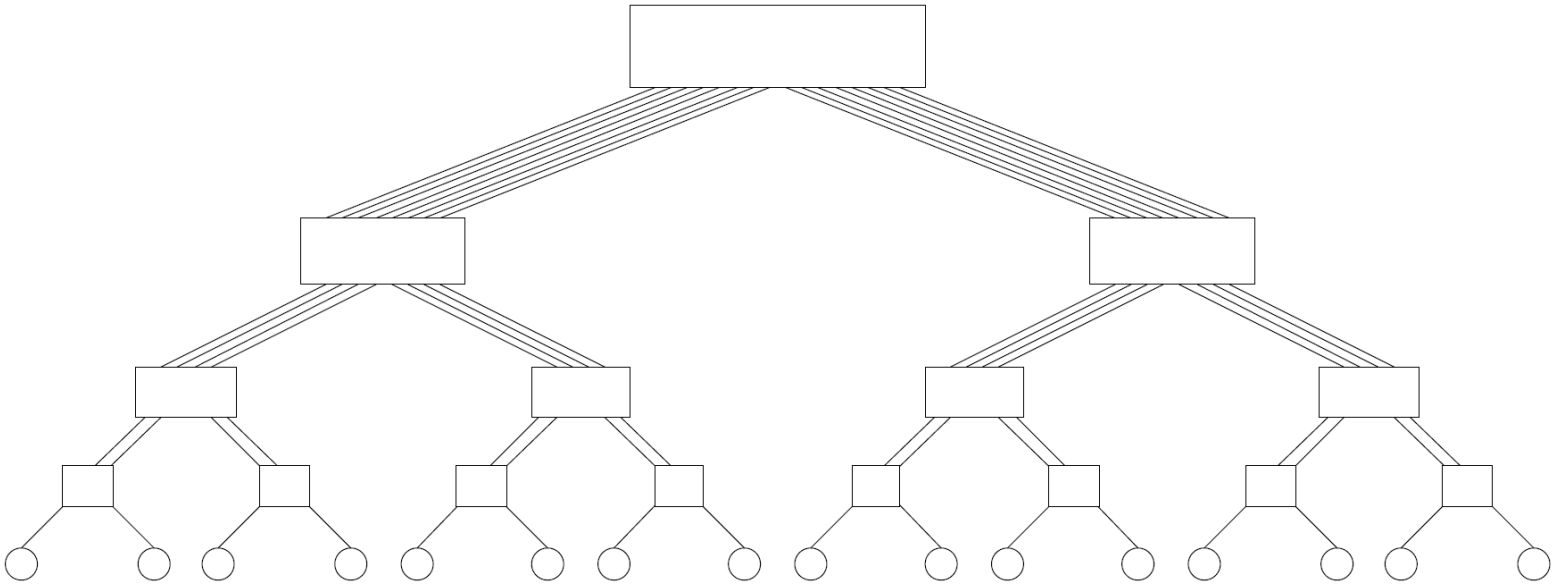


Figure 2.19 A fat tree network of 16 processing nodes.

Introduction to Parallel Computing, Ananth Grama, et al., Addison Wesley, 2nd Ed., 2003

Interconnection Topologies (cont'd)

Table 2.1 A summary of the characteristics of various static network topologies connecting p nodes.

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	1	$p-1$
Complete binary tree	$2\log((p+1)/2)$	1	1	$p-1$
Linear array	$p-1$	1	1	$p-1$
2-D mesh, no wraparound	$2(\sqrt{p}-1)$	$\sqrt{p}$	2	$2(p-\sqrt{p})$
2-D wraparound mesh	$2\lfloor\sqrt{p}/2\rfloor$	$2\sqrt{p}$	4	$2p$
Hypercube	$\log p$	$p/2$	$\log p$	$(p\log p)/2$
Wraparound k-ary d-cube	$d\lfloor k/2\rfloor$	$2k^{d-1}$	$2d$	dp

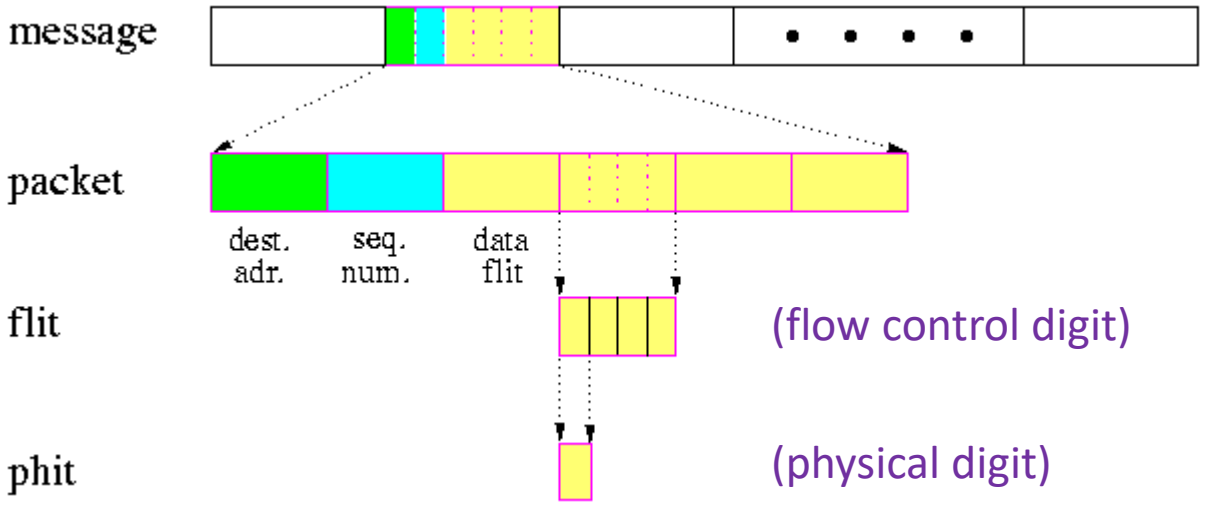
Introduction to Parallel Computing, Ananth Grama, et al., Addison Wesley, 2nd Ed., 2003

Communication Performance

- **Time(n)$_{s-d}$ = overhead + routing delay + channel occupancy + contention delay**

- **occupancy = (n + n_h) / b**

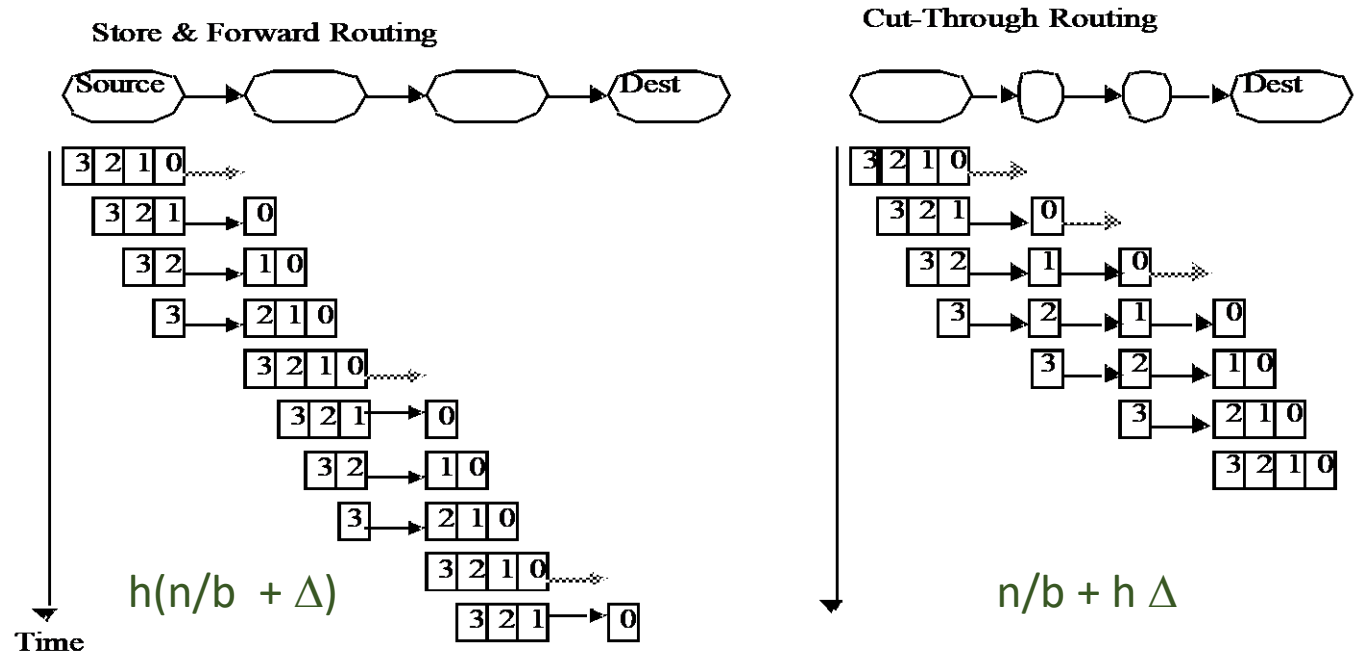
  n = size of message
  n_h = size of header
  b = bitrate of communication link



- What is the routing delay?
- Does contention occur and what is the cost?

Store-and-Forward vs. Cut-Through Routing

Flow Control:



- ***Store-and-Forward***: head flits waits at router until entire packet is buffered before being forwarded to the next hop
- ***Cut-Though***: flits can proceed to next hop before tail flit has been received by current router; but only if next router has enough buffer space for entire packet
- ***Wormhole***: just like cut-through, but with buffers allocated per flit (not channel)

Message Passing Model

- Hardware maintains send and receive message buffers
- Send message (synchronous)
  - Build message in local message send buffer
  - Specify receive location (processor id)
  - Initiate send and wait for receive acknowledge
- Receive message (synchronous)
  - Allocate local message receive buffer
  - Receive message byte stream into buffer
  - Verify message (e.g., checksum) and send acknowledge
- Memory to memory copy with acknowledgement and pairwise synchronization

Advantages of Shared Memory Architectures

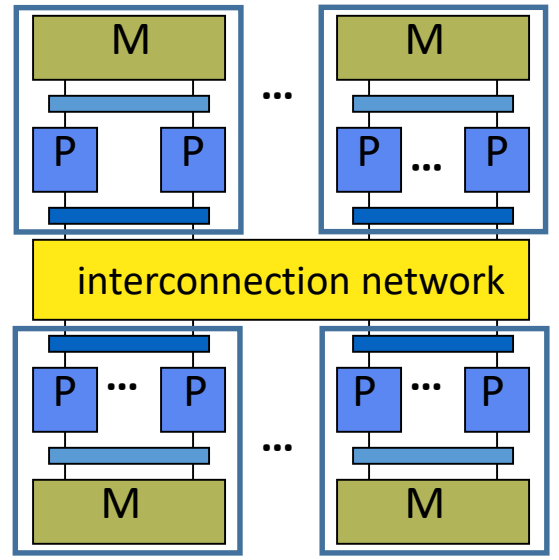
- Compatibility with SMP hardware
- Ease of programming when communication patterns are complex or vary dynamically during execution
- Ability to develop applications using familiar SMP model, attention only on performance critical accesses
- Lower communication overhead, better use of BW for small items, due to implicit communication and memory mapping to implement protection in hardware, rather than through I/O system
- HW-controlled caching to reduce remote communication by caching of all data, both shared and private

Advantages of Distributed Memory Architectures

- The hardware can be simpler (especially versus NUMA) and is more scalable
- Communication is explicit and simpler to understand
- Explicit communication focuses attention on costly aspect of parallel computation
- Synchronization is naturally associated with sending messages, reducing the possibility for errors introduced by incorrect synchronization
- Easier to use sender-initiated communication, which may have some advantages in performance

Clusters of SMPs

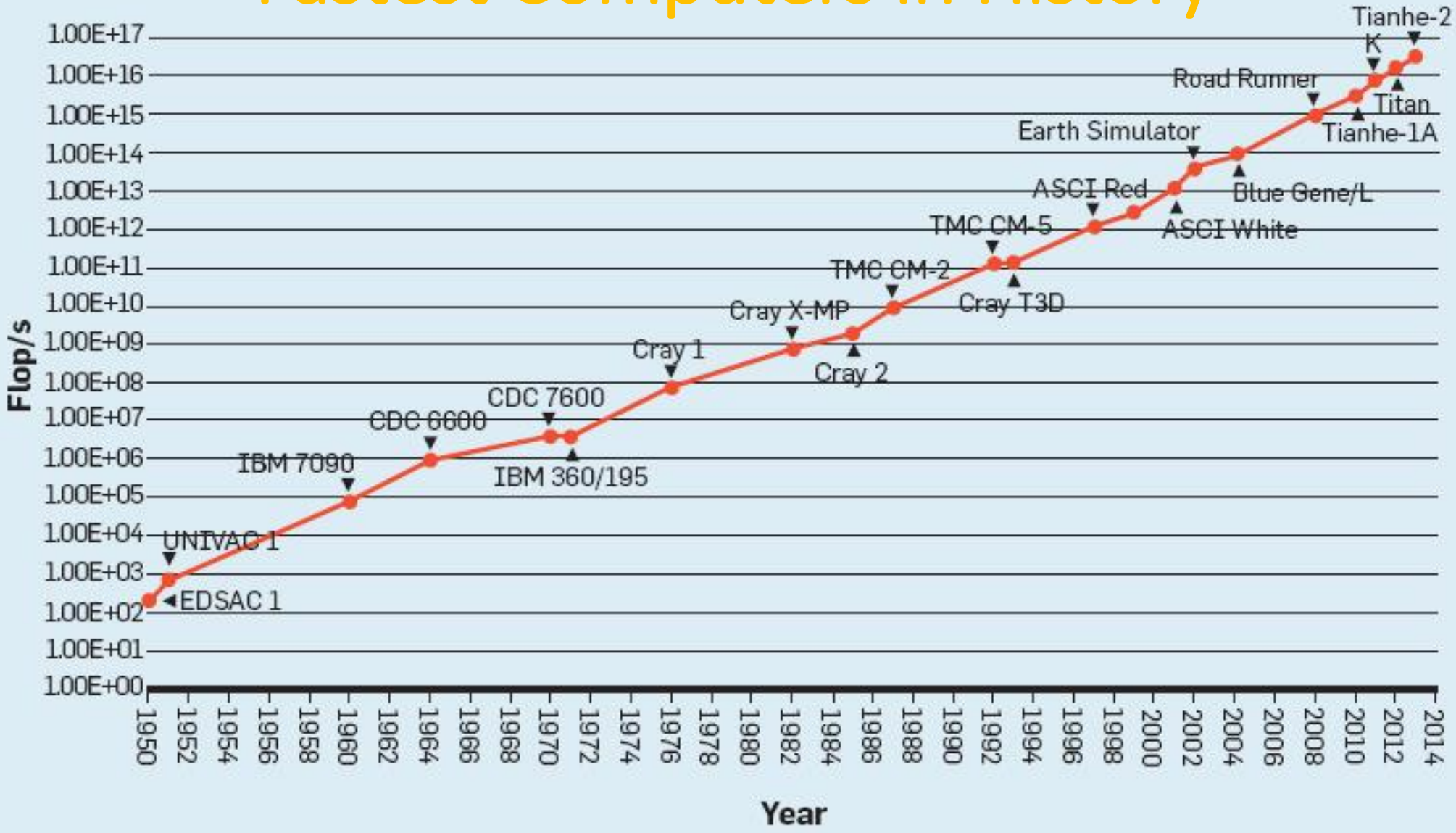
- Clustering
  - Integrated packaging of nodes
- Motivation
  - Ammortize node costs by sharing packaging and resources
  - Reduce network costs
  - Reduce communications bandwidth requirements
  - Reduce overall latency
  - More parallelism in a smaller space
  - Increase node performance
- Scalable parallel systems today are built as SMP clusters



Beowulf Cluster

- A cluster of *commodity-grade* computers
- Originally referred to a cluster built in 1994 by Thomas Sterling and Donald Becker at NASA
- Similar to Berkeley NOW (Network of Workstations)
- *Commodity* Interconnects
  - Ethernet, Myrinet, SCI, InfiniBand, etc.
- *Commodity* software
  - Unix-like OS, such as Linux, BSD, Solaris, etc.
  - Cluster toolkit, such as Rocks, Warewulf, etc.
  - Libraries and programs, such as MPI, PVM, etc.

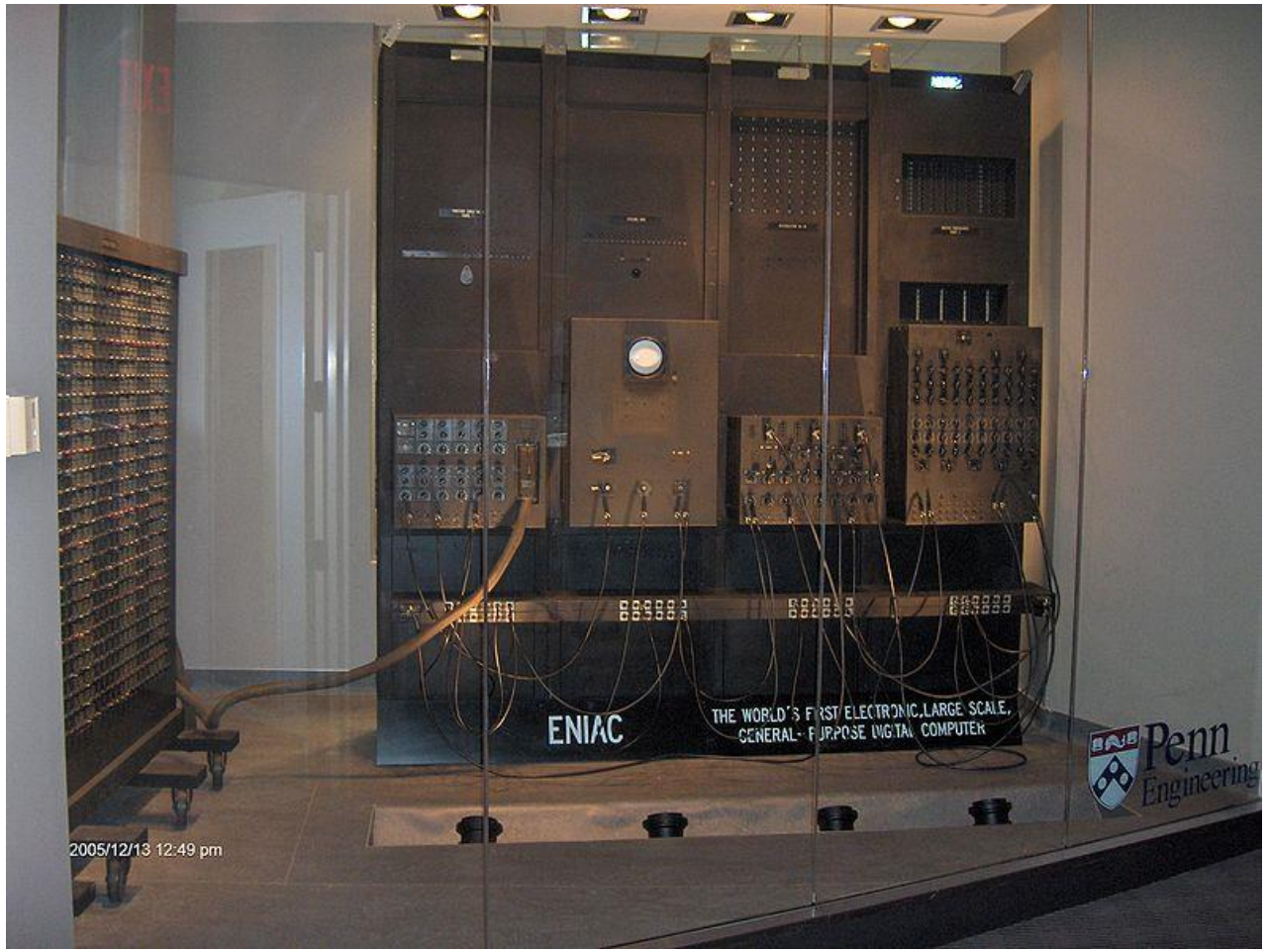
Fastest Computers in History



Systems ranked #1 since 1976

- NRCPC Sunway TaihuLight (National Supercomputing Center in Wuxi China, June 2016 – present)
- NUDT Tianhe-2A (National Supercomputing Center of Guangzhou China, June 2013 – June 2016)
- Cray Titan (Oak Ridge National Laboratory United States, November 2012 – June 2013)
- IBM Sequoia Blue Gene/Q (Lawrence Livermore National Laboratory United States, June 2012 – November 2012)
- Fujitsu K computer (Riken Advanced Institute for Computational Science Japan, June 2011 – June 2012)
- NUDT Tianhe-IA (National Supercomputing Center of Tianjin China, November 2010 – June 2011)
- Cray Jaguar (Oak Ridge National Laboratory United States, November 2009 – November 2010)
- IBM Roadrunner (Los Alamos National Laboratory United States, June 2008 – November 2009)
- IBM Blue Gene/L (Lawrence Livermore National Laboratory United States, November 2004 – June 2008)
- NEC Earth Simulator (Earth Simulator Center Japan, June 2002 – November 2004)
- IBM ASCI White (Lawrence Livermore National Laboratory United States, November 2000 – June 2002)
- Intel ASCI Red (Sandia National Laboratories United States, June 1997 – November 2000)
- Hitachi CP-PACS (University of Tsukuba Japan, November 1996 – June 1997)
- Hitachi SR2201 (University of Tokyo Japan, June 1996 – November 1996)
- Fujitsu Numerical Wind Tunnel (National Aerospace Laboratory of Japan Japan, November 1994 – June 1996)
- Intel Paragon XP/S140 (Sandia National Laboratories United States, June 1994 – November 1994)
- Fujitsu Numerical Wind Tunnel (National Aerospace Laboratory of Japan Japan, November 1993 – June 1994)
- TMC CM-5 (Los Alamos National Laboratory United States, June 1993 – November 1993)
- NEC SX-3/44 (Japan, 1992–1993)
- Fujitsu VP2600/10 (Japan, 1990–1991)
- Cray Y-MP/832, (United States, 1988–1989)
- Cray-2, (United States, 1985–1987)
- Cray X-MP, (United States, 1983–1985)
- Cray-1, (United States, 1976–1982)

ENIAC



- 1946
- 1st *electronic general-purpose* computer
- Vacuum tube circuitry
- Could make a 10-digit by 10-digit multiplication in 2800 µs
- ~ 357 *single-precision* FLOPS
  (floating-point operations per second)
- https://en.wikipedia.org/wiki/ENIAC

UNIVAC I

- 1951
- 1st commercial computer in US
- Multiplication time was 2150 µs
- ~ 465 *single-precision* FLOPS
- Originally priced at $159,000
- Raised to $1.25 - $1.5 million
- https://en.wikipedia.org/wiki/UNIVAC_I

IBM 704

- 1954
- 1st mass-produced computer with floating-point arithmetic hardware
- Fortran & Lisp were 1st developed for IBM 704
- ~ 12 kFLOPS
- $2 million
- https://en.wikipedia.org/wiki/IBM_704

IBM 7090

- 1959
- *Transistorized* version of IBM 709 vacuum tube mainframe
- Double-precision floating-point instructions were introduced on IBM 7094
- ~ 100 kFLOPS
- $2.9 million
- https://en.wikipedia.org/wiki/IBM_7090

CDC 6600

- 1965
- 1st successful supercomputer
- Designed by Seymour Cray
- CPU, peripheral processors (PPs) and I/O operated *in parallel*
- 6600 CPU had multiple functional units that could operate *in parallel*
- ~ 3 MFLOPS
- $6 - $10 million
- https://en.wikipedia.org/wiki/CDC_6600

CDC 7600

- Fastest from 1969 to 1975
- Designed by Seymour Cray
- An architecture landmark
  - Instruction pipeline
  - Reduced Instruction Set Computer (RISC)
- ~ 10 MFLOPS on hand-compiled code
- 36 MFLOPS peak performance
- $5 million
- https://en.wikipedia.org/wiki/CDC_7600

Cray-1

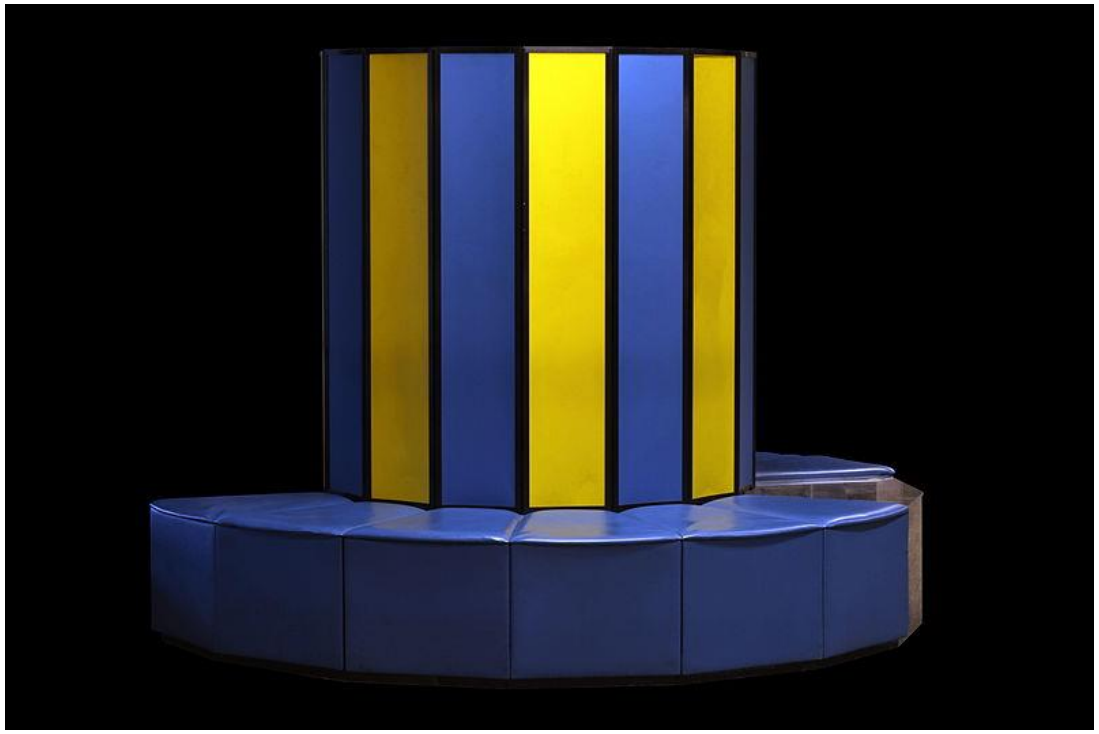
- 1975
- One of the best known and most successful supercomputers in history
- 1st Cray design to use *integrated circuits* (ICs)
- 64-bit
- Vector processor, with 12 pipelined functional units
- ~ 160 MFLOPS, with 250 MFLOPS peak
- $8.86 million (1977)
- https://en.wikipedia.org/wiki/Cray-1

IBM PC

- IBM PC 5150 was released in 1981
- Intel 8088 CPU at 4.77 MHz
- 16 kB – 256 kB of memory
- ~ 50 kFLOPS with Intel 8087 floating-point coprocessor
- $1,565 ~ $3,000

$$\frac{R_{max}(\text{Cray}-1)}{R_{max}(\text{IBM 5150})} = \frac{250\ MFLOPS}{50\ kFLOPS} = 5000$$

Cray X-MP



- 1982
- Shared-memory parallel vector processor supercomputer
- 2 vector processors at 105 MHz
- 400 MFLOPS peak performance
- $15 million
- https://en.wikipedia.org/wiki/Cray_X-MP

Cray Y-MP

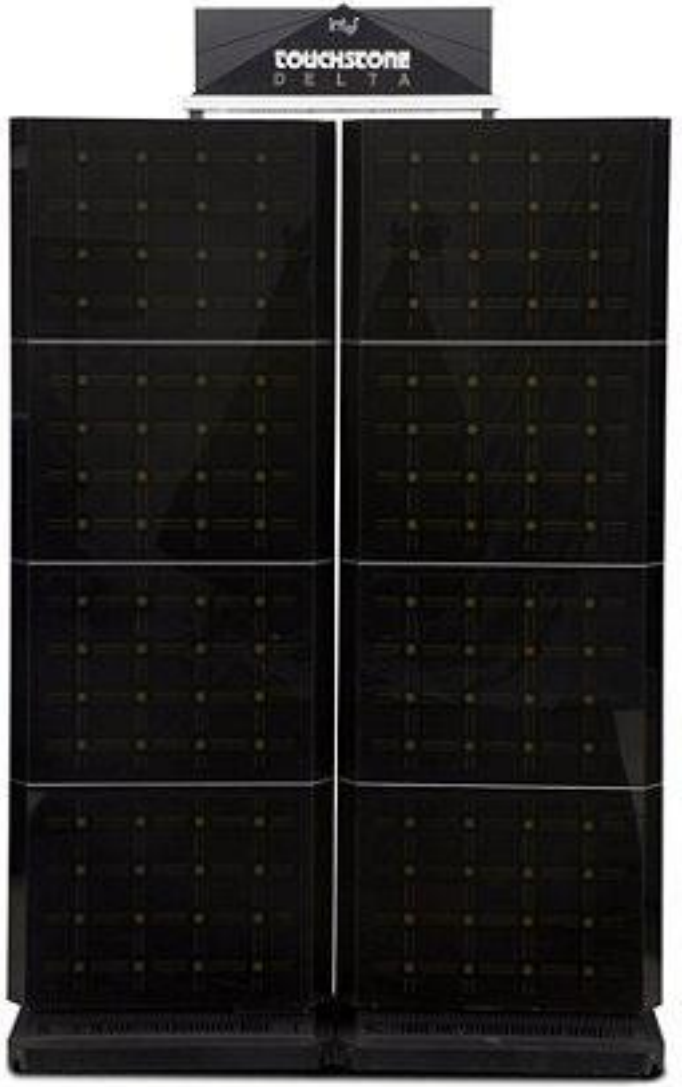
- 1988
- 2, 4, or 8 vector processors (with 2 functional units each) at 167 MHz
- 2.144 GFLOPS (measured) & 2.667 GLOPS (peak)
- $10 million
- https://en.wikipedia.org/wiki/Cray_Y-MP
- Cray C90 was a development of the Y-MP architecture, launched in 1991

Thinking Machines CM-1

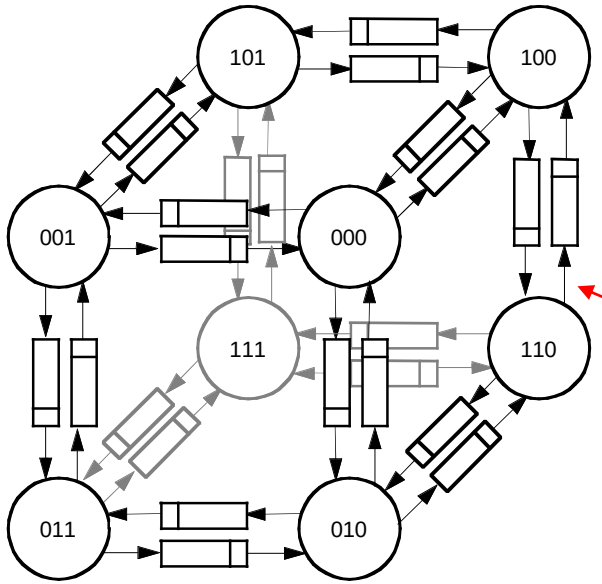
- 1985
- SIMD supercomputer
- 65,536 simple single-bit processors
- Each CM-1 processor had its own 4 kilobits of RAM
- Connected in a hypercubic routing network
- ~ 1 GFLOPS
- $5 million
- https://en.wikipedia.org/wiki/Connection_Machine

Intel Paragon

- Massively parallel supercomputers by Intel in the 1990s
- Based on the Intel i860 RISC microprocessors
- Up to 2048 (later, up to 4000) i860s are connected in a 2D grid
- The prototype was the Touchstone Delta, funded by DARPA and installed at Caltech in 1990
  - 16x32 mesh of i860 processors with a wormhole routing interconnection network
  - 40 GFLOPS



Caltech Cosmic Cube



FIFO on each link
Store and Forward

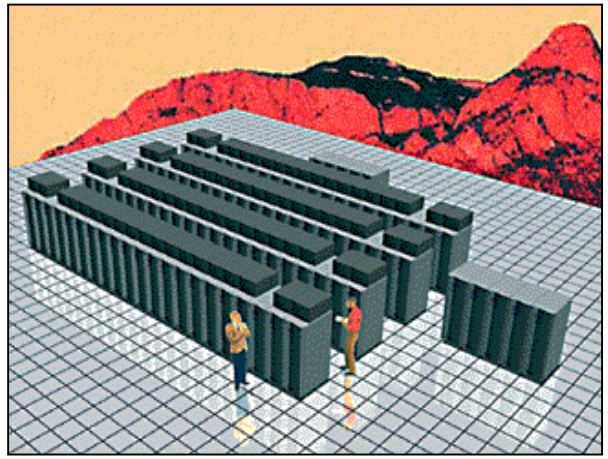
Compute nodes

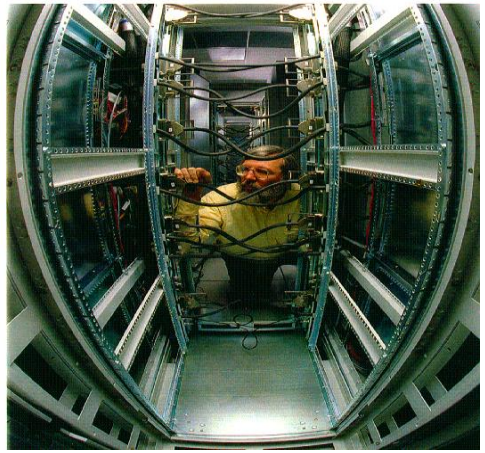
- 1st distributed memory message passing system
- Developed by Charles Seitz and Geoffrey Fox from 1981 onward
- 64 Intel 8086/8087 processors
- 6-D hypercube network
- Inspired Intel iPSC in 1980s and 1990s

http://groups.csail.mit.edu/cag/PSRG/papers/p22-seitz.pdf

ASCI Red

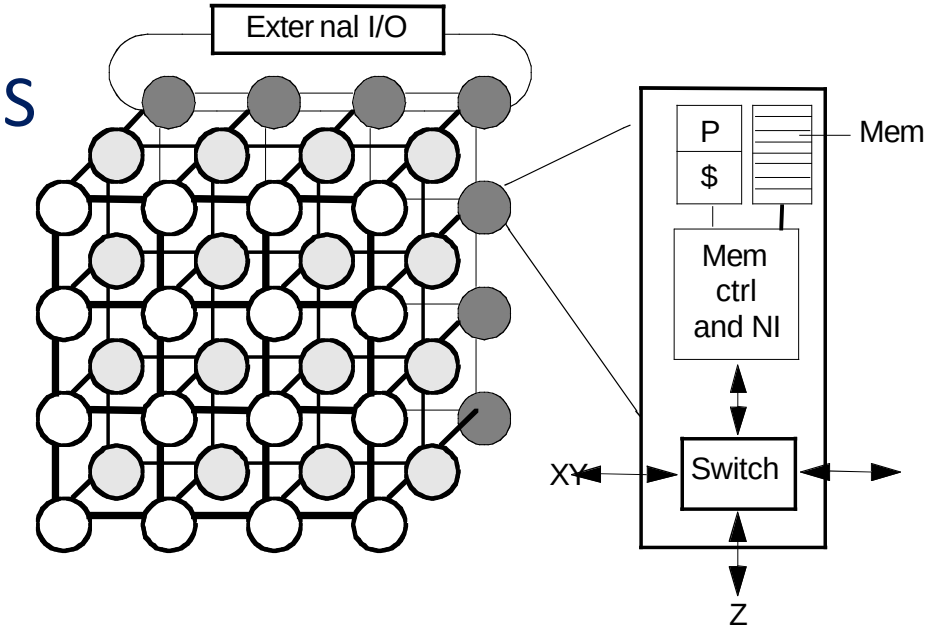
- Based on Intel Paragon
- 1st Supercomputer to achieve over 1 teraflops
- 7,264 Pentium Pro processors @ 200 MHz in 06/1997, R_{max} = 1.068 TFLOPS
- 9,152 Pentium Pro processors @ 200 MHz in 11/1997, R_{max} = 1.338 TFLOPS
- 9,632 Pentium II Overdrive processors @ 333 MHz in 11/1999, R_{max} = 2.379 TFLOPS
- Network topology: 2 dimensional 2-plane mesh (38x32x2)
  - 2 compute nodes (4 processors) on a “Kestrel” board with a NIC
  - Link bandwidth: 800 MB/s
  - Bisection bandwidth: 51 GB/s
- #1 from June 1997 to June 2000





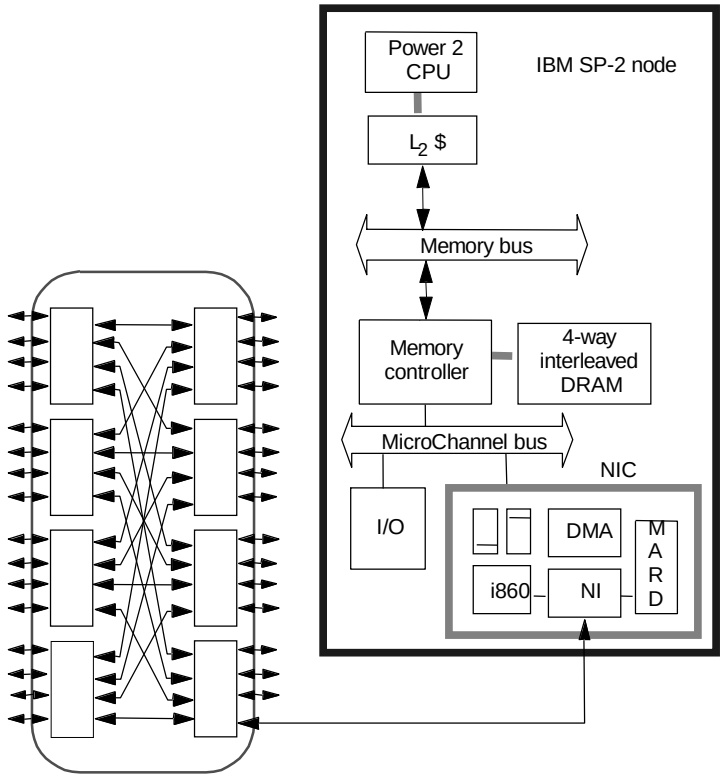
Cray T3E

- Launched in November 1995
- From 8 to 2,176 Processing Elements (PEs)
  - DEC Alpha processors @ 300, 450, 600 or 675 MHz
- 3D torus network
  - 6-way interconnect router with 480 MB/s in each direction
- 1st supercomputer to achieve over 1 TFLOPS running science simulation, in 1998
- Distributed memory machine
  - Message passing
  - SHMEM (Symmetric Hierarchical Memory)
    - *put* / *get* – one-sided operations



ASCI White

- IBM SP-2
- Made out of commercial RS/6000 workstations
  - 512 nodes
  - 16 PowerPC processors @ 375 MHz per node
  - 8,192 processors in total
- Network interface integrated in I/O bus
- SP network very advanced
  - Formed from 8-port switches
- R_{max} = 7.226 TFLOPS, R_{peak} = 12.3 TFLOPS
- #1 from 11/2000 – 11/2001
- 3MW for computer & 3MW for cooling
- Costed $110 million



NASA Columbia

- 20 interconnected SGI Altix 3700 nodes, each with:
  - 512 Itanium 2 2-core processors @ 1.6 GHz
  - 1TB memory
- NUMAflex architecture
  - NUMAlink “fat tree” network
  - 20TB of single shared memory!!!
- Nodes were also connected with InfiniBand SDR and DDR cabling
- R_{max} = 51.87 TFLOPS, R_{peak} = 60.96 TFLOPS
- #2 in November 2004

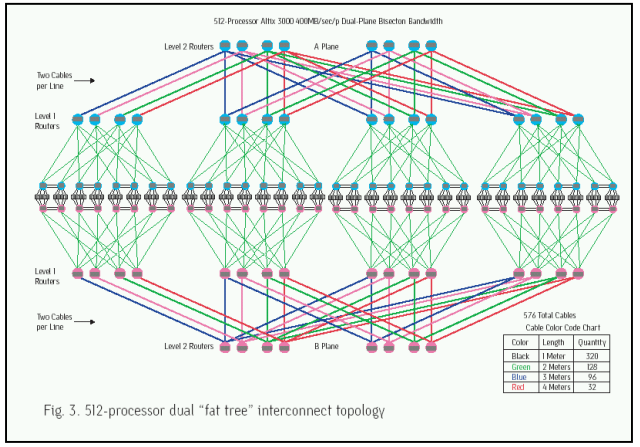
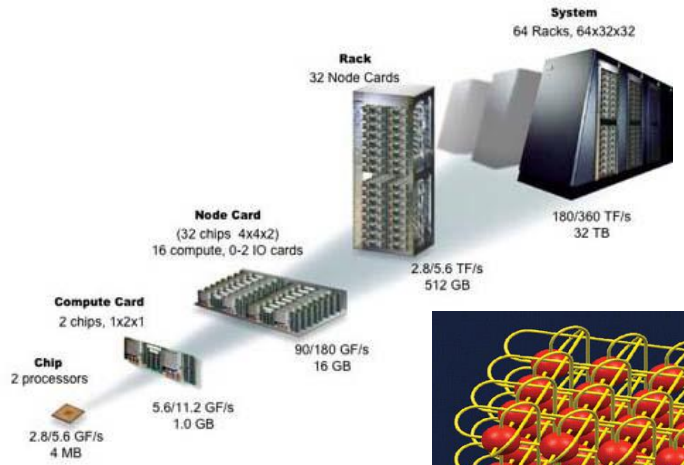
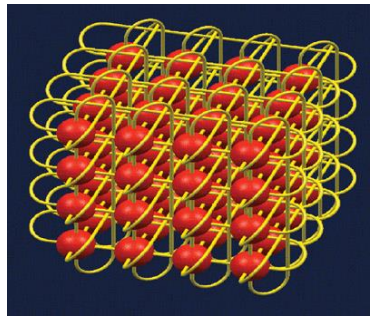


Fig. 3. 512-processor dual “fat tree” interconnect topology

LLNL Blue Gene/L

- Initially 65,536 dual-processor compute nodes
- Later upgraded to 106,496 compute nodes, each with
  - Two PowerPC 440 2-core processors @ 700MHz
  - Two working modes: co-processor mode & virtual-node mode
  - Double “hummer” floating point unit (FPU) per core
- One I/O node every 32 compute nodes
- 3D torus network (initially 32x32x64)
- Global reduction tree
- Global barrier and interrupt networks
- Scalable tree network for I/O
- R_{max} = 478.2 TFLOPS, R_{peak} = 596.4 TFLOPS
- #1 from November 2004 to June 2008

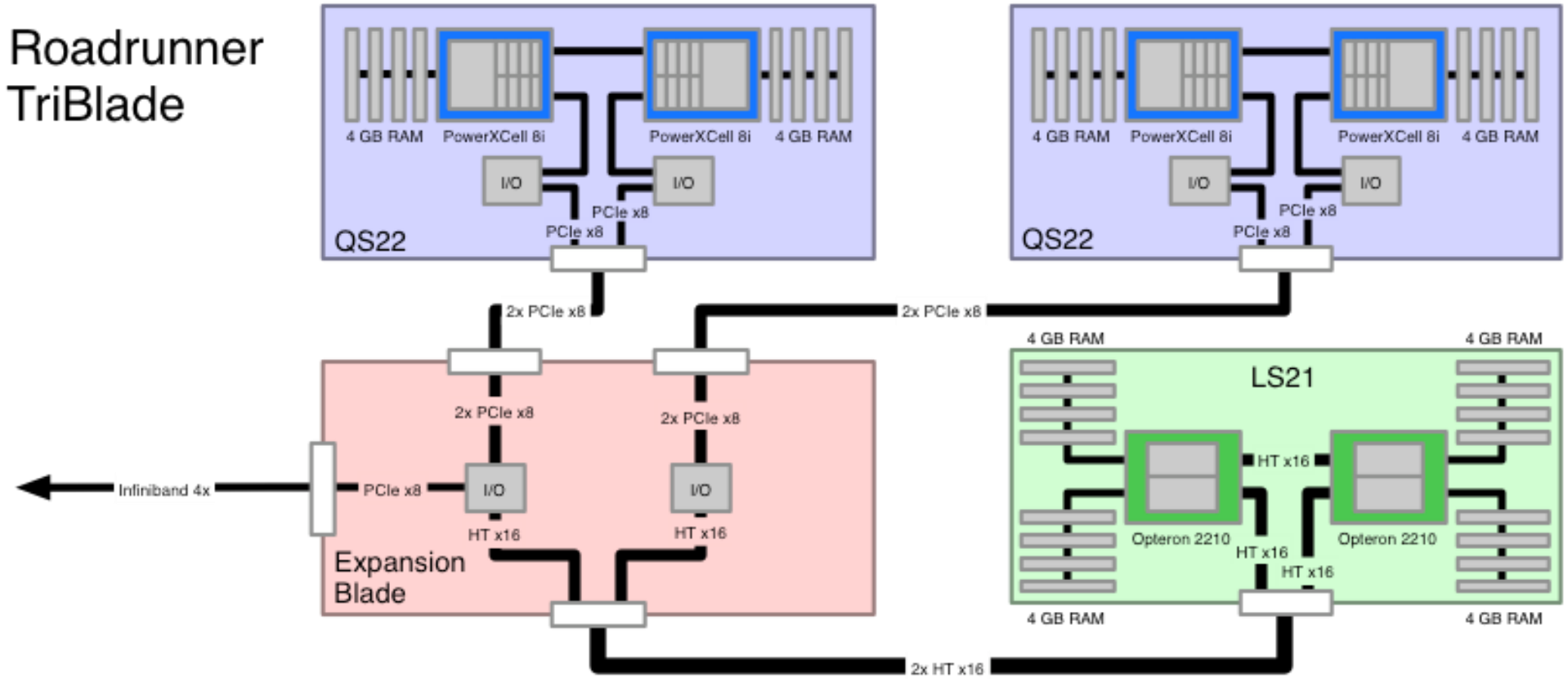




IBM Roadrunner

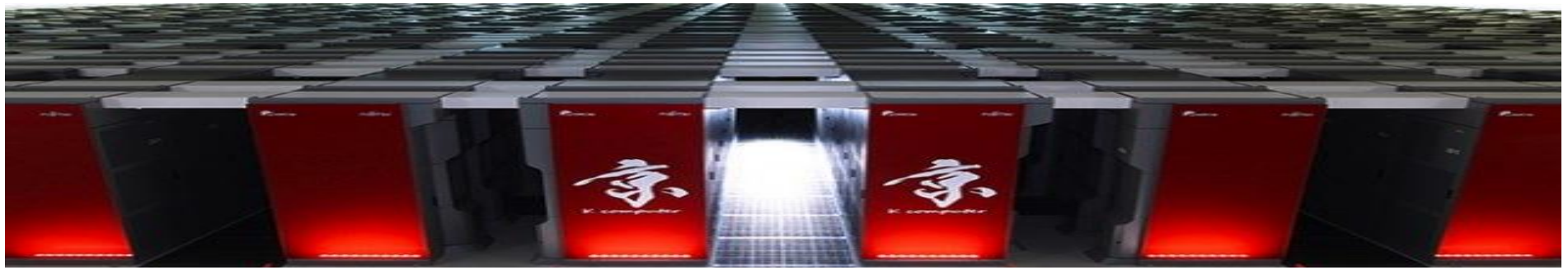
- 1st petaflops Supercomputer
- Hybrid architecture:
  - Opteron cluster with Cell accelerators
- 3,240 TriBlades, each with
  - One LS21 Opteron blade, with two dual-core AMD Opteron 2210 processors @1.8 GHz and 16GB memory
  - Two QS22 Cell blades, each with two PowerXCell 8i CPUs (@3.2GHz) and 8GB memory
    Peak performance = 102.4 GFLOPS per PowerXCell 8i
  - One expansion blade, connecting the two QS22 via four PCIe x8 links to the LS21
- 26 288-port Voltaire ISR 2012 InfiniBand 4x DDR switches
  - fat tree topology
- R_{max} = 1.026 PFLOPS, R_{peak} = 1.376 PFLOPS, Power = 2.345 MW
- #1 in June & November 2008

IBM Roadrunner TriBlade

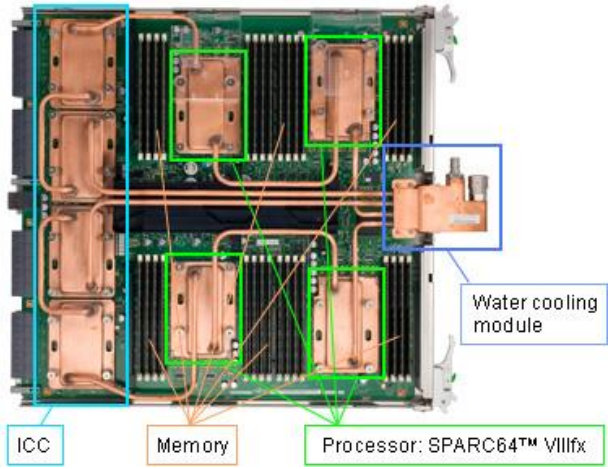


Roadrunner: Hardware and Software Overview
http://www.redbooks.ibm.com/redpapers/pdfs/redp4477.pdf

RIKEN K Computer



- 82,944 (96/cabinets x 864 cabinets) Compute Nodes, each with:
  - One 8-core SPARC64 VIIIfx @ 2.0 GHz
  - 16 GB of memory
- 5,184 (6/cabinets x 864 cabinets) I/O Nodes
- 6-dimensional torus interconnect (*Tofu*)
- Fujitsu Exabyte File System (*FEFS*), based on Lustre
- #1 in June 2011

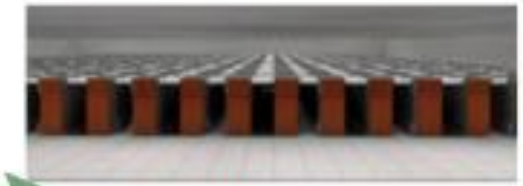


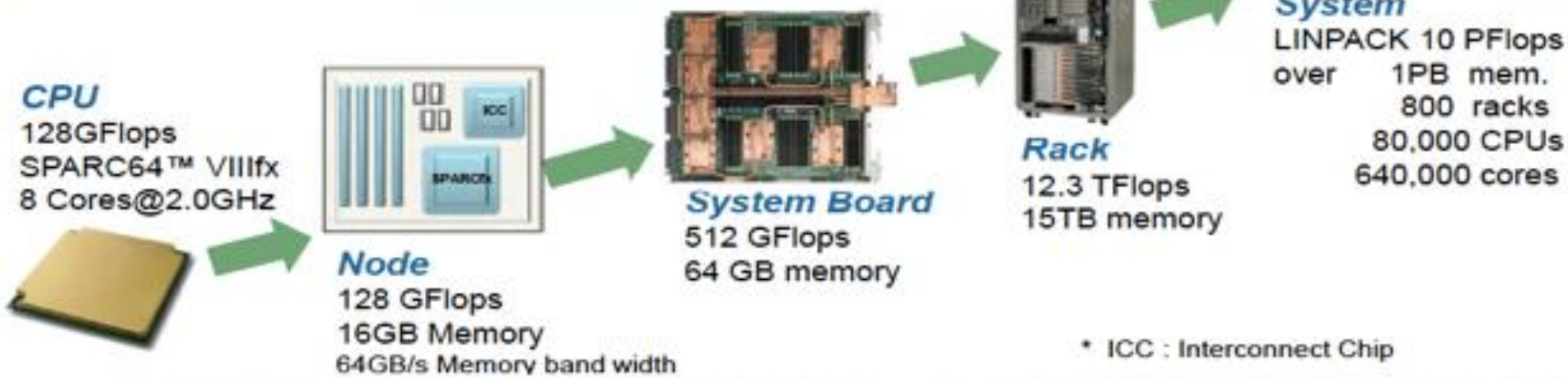
R_{peak} = 11.280 PFLOPS
R_{max} = 10.510 PFLOPS
Power = 12.6 MW
Cost > 100 billion Yen ($1.25b)

K computer Specifications

CPU (SPARC64 VIIIfx)	Cores/Node	8 cores (@2GHz)
	Performance	128GFlops
	Architecture	SPARC V9 + HPC extension
	Cache	L1(I/D) Cache : 32KB/32KB L2 Cache : 6MB
	Power	58W (typ. 30 C)
	Mem. bandwidth	64GB/s.
Node	Configuration	1 CPU / Node
	Memory capacity	16GB (2GB/core)
System board(SB)	No. of nodes	4 nodes /SB
Rack	No. of SB	24 SBs/rack
System	Nodes/system	> 80,000

Inter-connect	Topology	6D Mesh/Torus
	Performance	5GB/s. for each link
	No. of link	10 links/ node
	Additional feature	H/W barrier, reduction
	Architecture	Routing chip structure (no outside switch box)
Cooling	CPU, ICC*	Direct water cooling
	Other parts	Air cooling

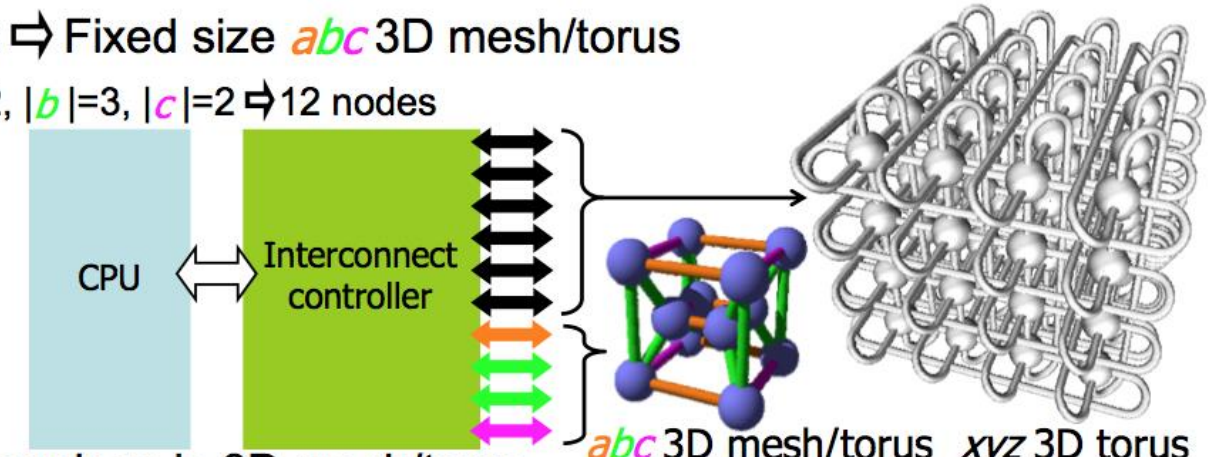




New Linpack run with 705,024 cores at 10.51 Pflop/s (88,128 CPUs)

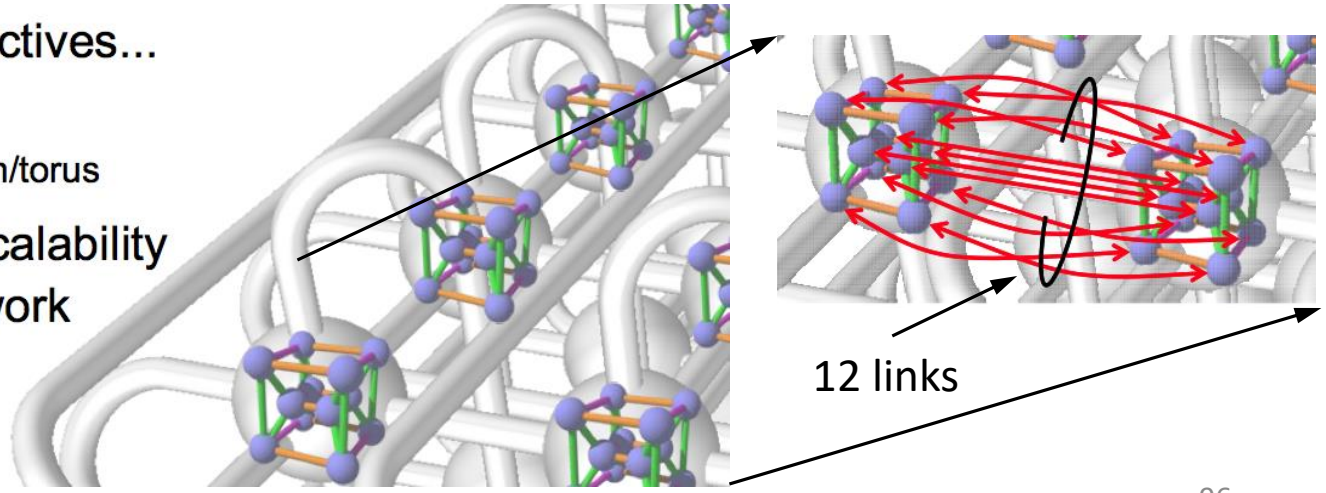
K Computer – Interconnect

- 6 links ⇨ Scalable *xyz* 3D torus
- 4 links ⇨ Fixed size *abc* 3D mesh/torus
  - |*a*|=2, |*b*|=3, |*c*|=2 ⇨ 12 nodes



- Total topology is 6D mesh/torus
  - Cartesian product of *xyz* and *abc* mesh/torus

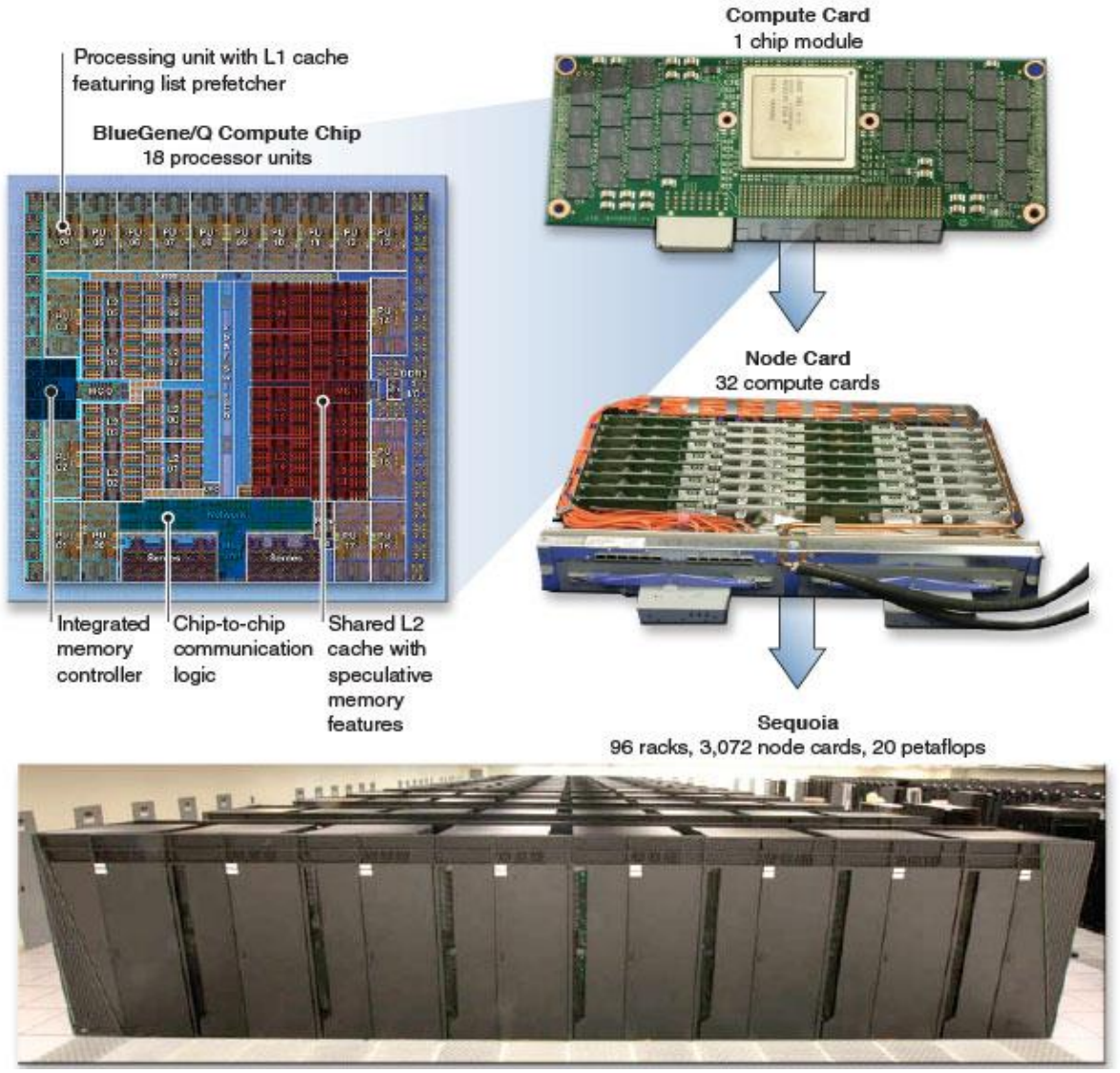
- From the other perspectives...
  - Overlaid twelve *xyz* torus
  - X x Y x Z array of *abc* mesh/torus
- Twelve times higher scalability than the 3D torus network



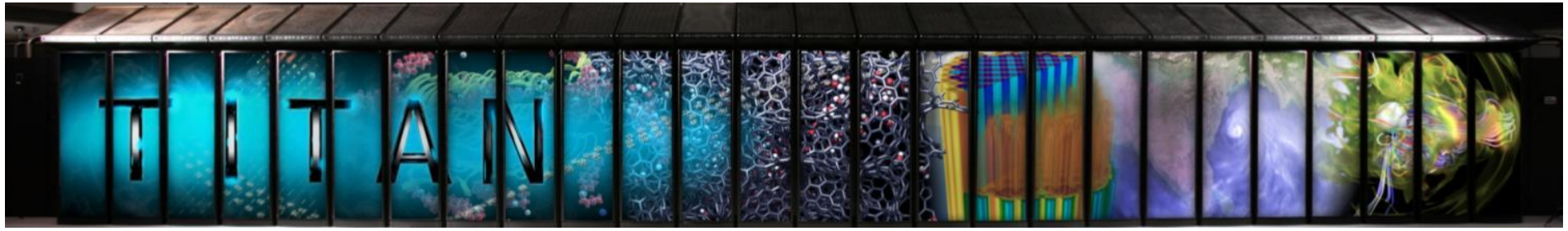
LLNL Sequoia

- IBM Blue Gene/Q design
- 98,304 (1024/rack x 96 racks) Compute Cards, each with:
  - 18-core PowerPC A2 processor @ 1.6 GHz, with 16 cores used for computing
  - 16 GB of DDR3 memory
- 5-dimensional torus interconnect
- 55 PB of Lustre storage (with ZFS backend)
- #1 in June 2012

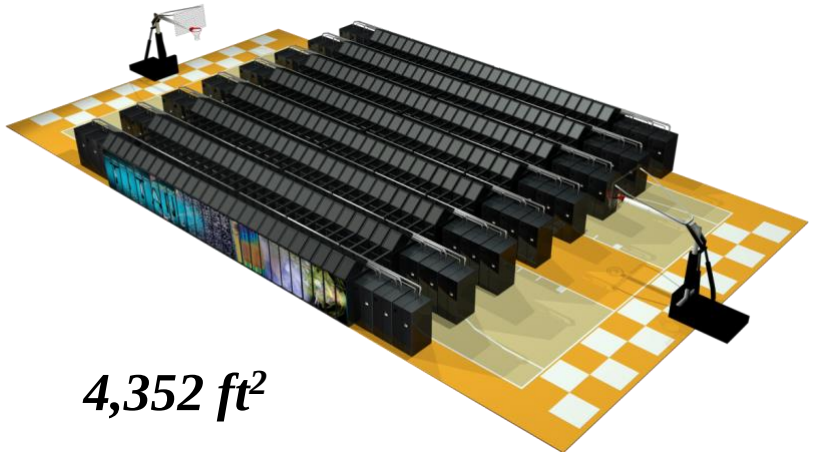
R_{peak} = 20.133 PFLOPS
R_{max} = 17.173 PFLOPS
Power = 7.9 MW
Cost = $655.4 million



ORNL Titan



- 18,688 Compute Nodes (Cray XK7), each with:
  - One AMD Opteron 6274 16-core CPU @ 2.2 GHz
  - One NVIDIA Tesla K20X GPU
  - Memory: 32 GB host + 6GB device
- 512 Service and I/O nodes
- Cray Gemini 3D Torus Interconnect
- 40 PB of Lustre storage, with an aggregate transfer rate of 1.4 TB/s
- 200 Cabinets
- #1 in November 2012



R_{peak} = 27.1 PFLOPS = 24.5 GPU + 2.6 CPU

R_{max} = 17.590 PFLOPS

Power = 8.2 MW

Cost = $97 million

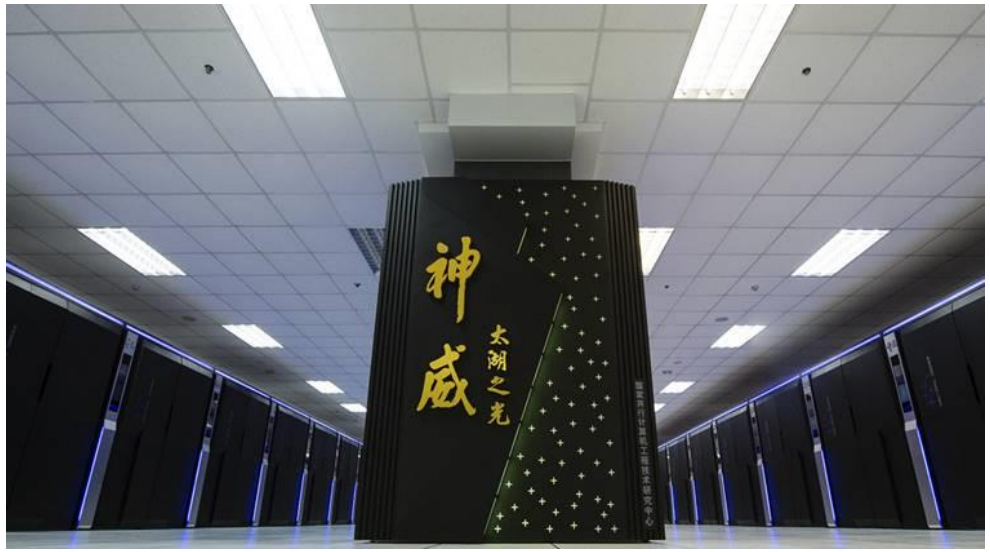
NUDT Tianhe-2 (Milky Way 2)

- 16,000 Compute Nodes, each with:
  - Two Intel Ivy Bridge Xeon E5-2692v2 12C 2.2GHz
  - Three Intel Xeon Phi 31S1P
  - Memory: 64 GB host + 24 GB devices (3 x 8GB)
  - 3.432 TFLOPS
- Front-End Node
  - 4096 Galaxy FT-1500 CPUs (a SPARC derivative)
  - Each FT-1500 has 16 cores, and runs @ 1.8 GHz
- Proprietary interconnect
  - TH2 express, in a fat tree topology
- 12.4PB of global shared parallel storage
- # 1 June 2013 — November 2015

R_{peak} = 54.902 PFLOPS
R_{max} = 33.863 PFLOPS
Power = 17.6 MW (24 MW)
Cost = 2.4 billion Yuan = $390m

NRCPC Sunway TaihuLight

- "Homegrown" Chinese supercomputer
- 40,960 Compute Nodes, each with:
  - One SW26010 manycore 64-bit RISC processor @ 1.45 GHz
  - 32GB of DDR3 memory
  - 3.0624 TFLOPS peak performance
- 40 cabinets
  - Each cabinet is composed of 4 Supernode
  - Each Supernode is composed of 256 nodes
- Custom interconnect (?)
- Sunway RaiseOS 2.0.5 (based on Linux)
- # 1 June 2016 – present



R_{peak} = 125.436 PFLOPS
R_{max} = 93.015 PFLOPS
Power = 15 MW (Linpack)
Cost = 1.8 billion Yuan (US $273m)

http://www.netlib.org/utk/people/JackDongarra/PAPERS/sunway-report-2016.pdf

2 ACM Gordon Bell Prizes

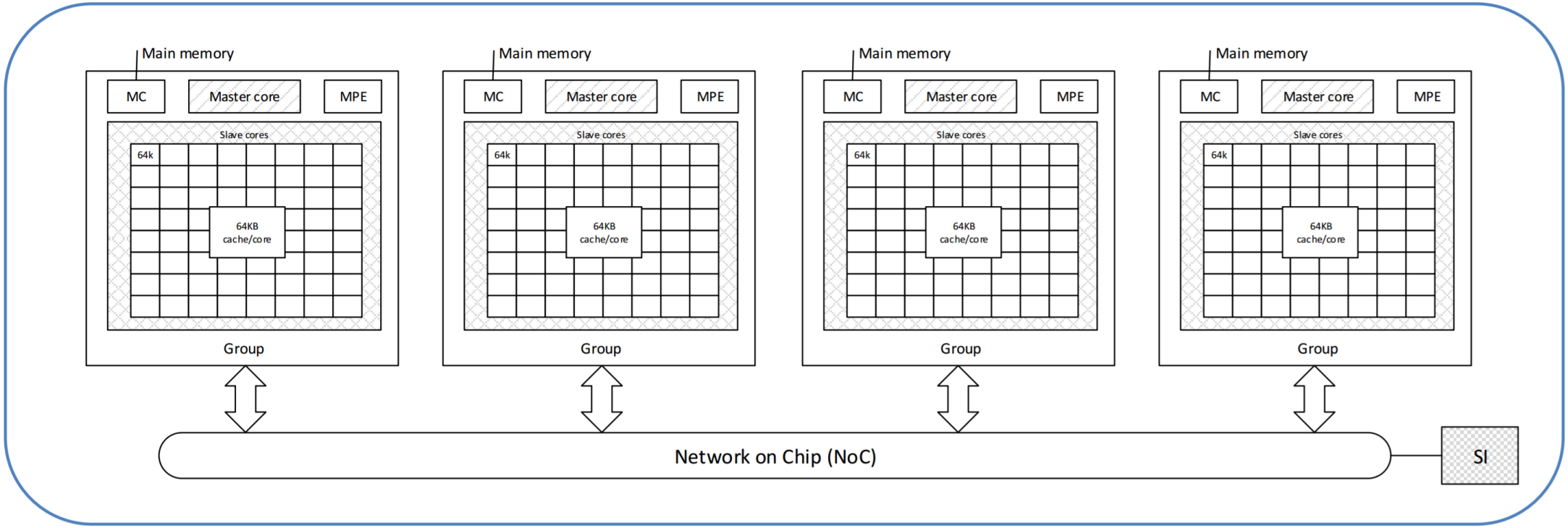
- 2016 — 10M-Core Scalable Fully-Implicit Solver for Nonhydrostatic Atmospheric Dynamics

  "The fully-implicit solver successfully scales to the entire system of the Sunway TaihuLight supercomputer with over 10.5M heterogeneous cores, sustaining an aggregate performance of 7.95 PFLOPS in double-precision"

- 2017 — 18.9-Pflops nonlinear earthquake simulation on Sunway TaihuLight: enabling depiction of 18-Hz and 8-meter scenarios

  "a customized parallelization scheme that employs the 10 million cores efficiently at both the process and the thread levels"

SW26010 Processor



SW26010 processor chip is composed of 4 core groups, each includes:

- 1 Management Processing Elements (**MPE**s), supporting both user and system modes
- 64 (8x8) Computing Processing Elements (**CPE**s), supporting only user mode
- Memory Controller (MC)

SW26010 Processor

MPE

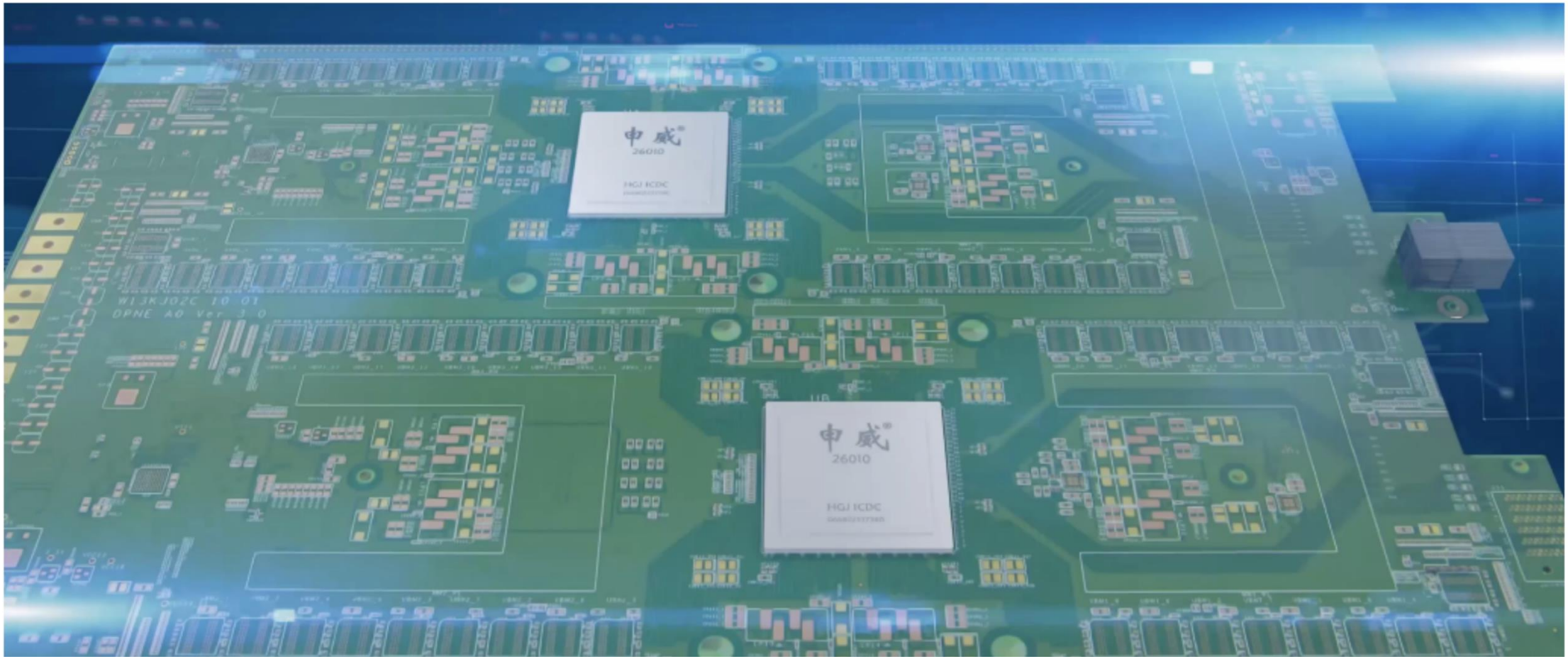
- 64-bit RISC core
- 256-bit vector instructions
- 32 KB L1 instruction cache
- 32 KB L1 data cache
- Dual pipelines

CPE

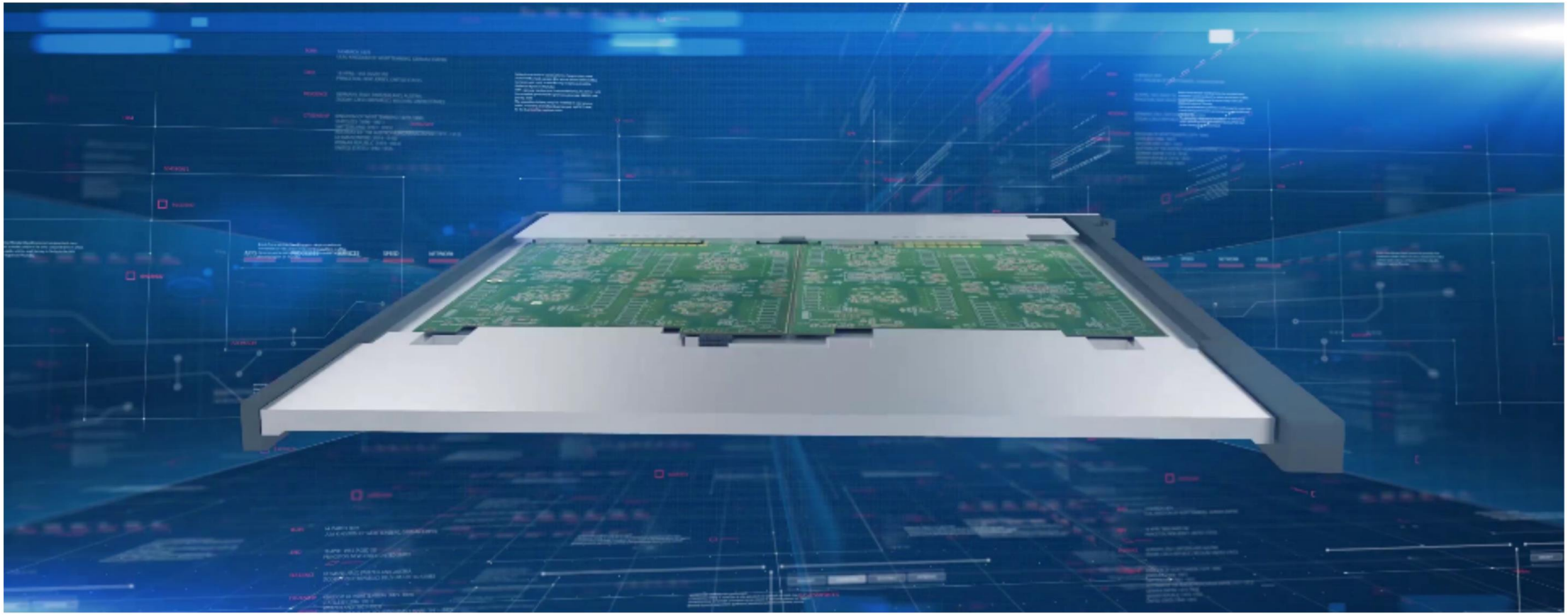
- 64-bit RISC core, user mode only
- 256-bit vector instructions
- 16 KB L1 instructions
- 64 KB Scratch Pad Memory (SPM)!
- 256 (64x4) CPEs per processor

Peak Performance of a CPE = 8 flops/cycle * 1.45 GHz = 11.6 Gflop/s
Peak Performance of a MPE = 8 flops/cycle * 2 * 1.45 GHz = 23.2 Gflop/s
Peak Performance of SW26010 = 256 * 11.6 Gflop/s + 4 * 23.2 Gflop/s
= 3.0624 Tflop/s

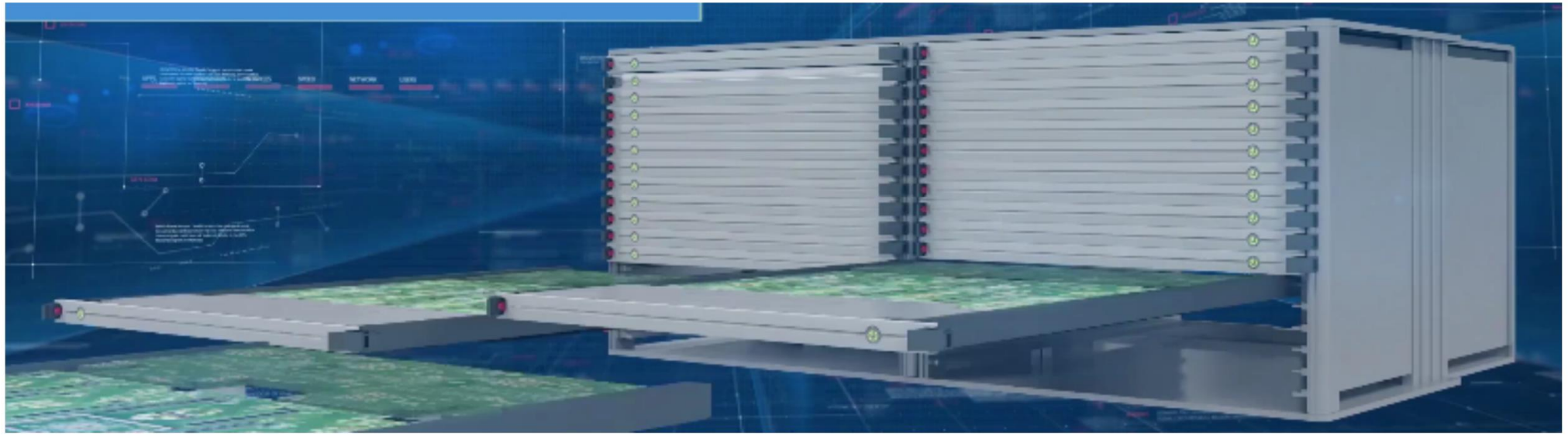
2 Compute Nodes on a Card



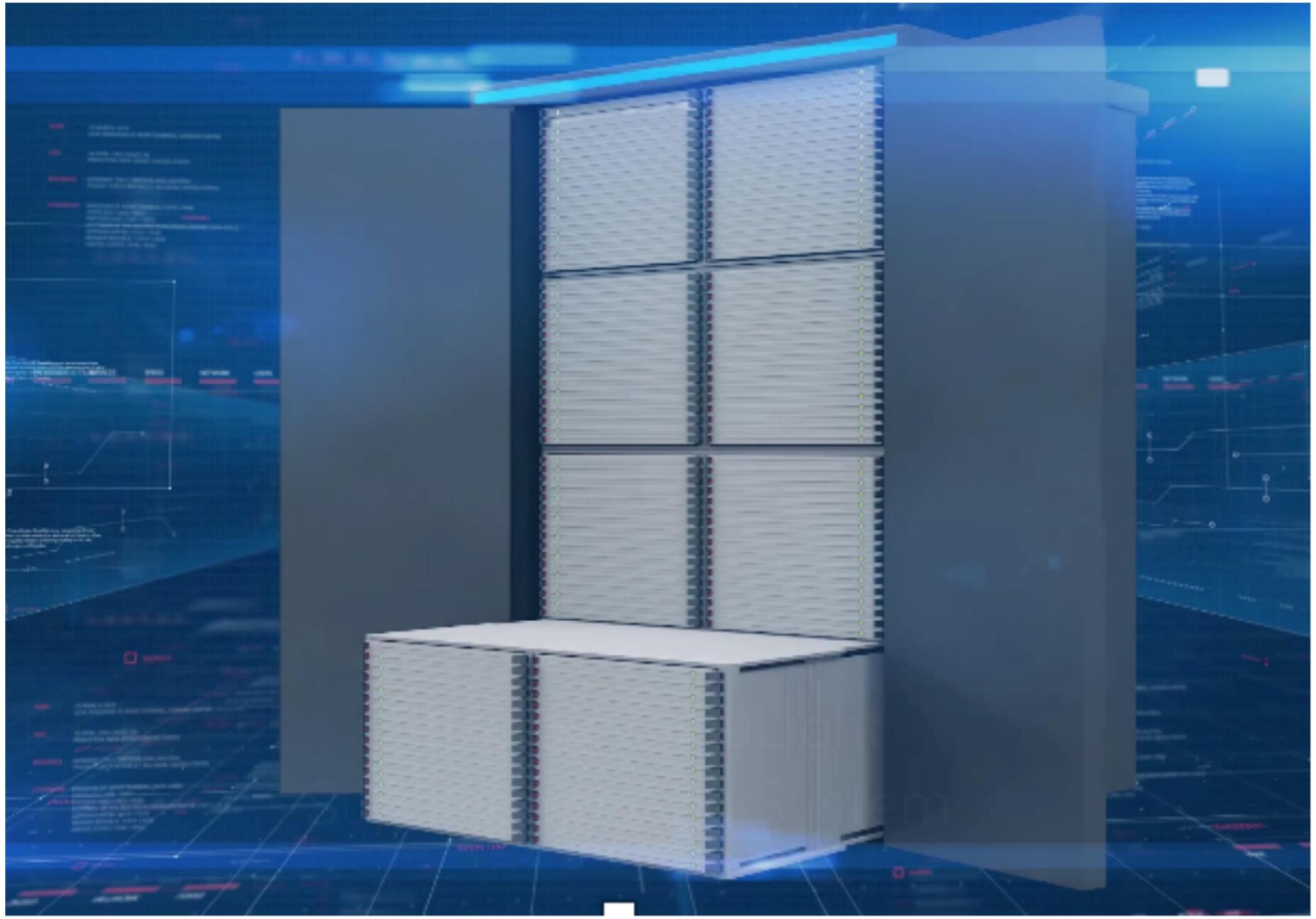
4 Cards on a Board (2 per side)



A Supernode composed of 32 Boards or 256 Nodes

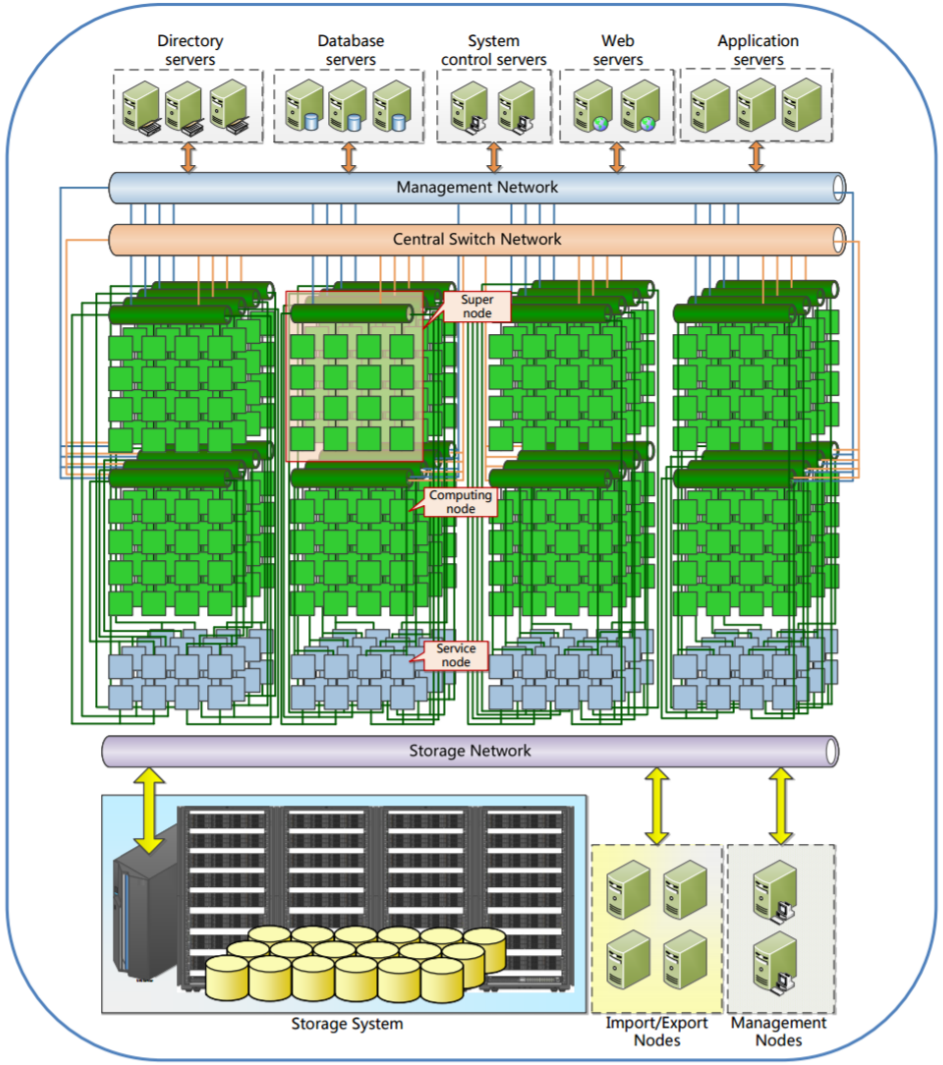


A Cabinet composed of 4 Supernodes or 1024 Nodes



Interconnect

- So-called Sunway Network
- Mellanox Host Channel Adapters (HCA) and switch chips
- InfiniBand (?)
- Fat tree topology (?)
- Bisection network bandwidth: 70TB/s
- Network diameter: 7
- Communication between nodes via MPI is at 12GB/s, and a latency about 1 us



Software Stack