

---

# CS 267 Applications of Parallel Computers

## Lecture 4: Parallel Architectures: Large-Scale Multiprocessors

**1/29/97**

**David E. Culler**

**<http://www.cs.berkeley.edu/cs267/>**

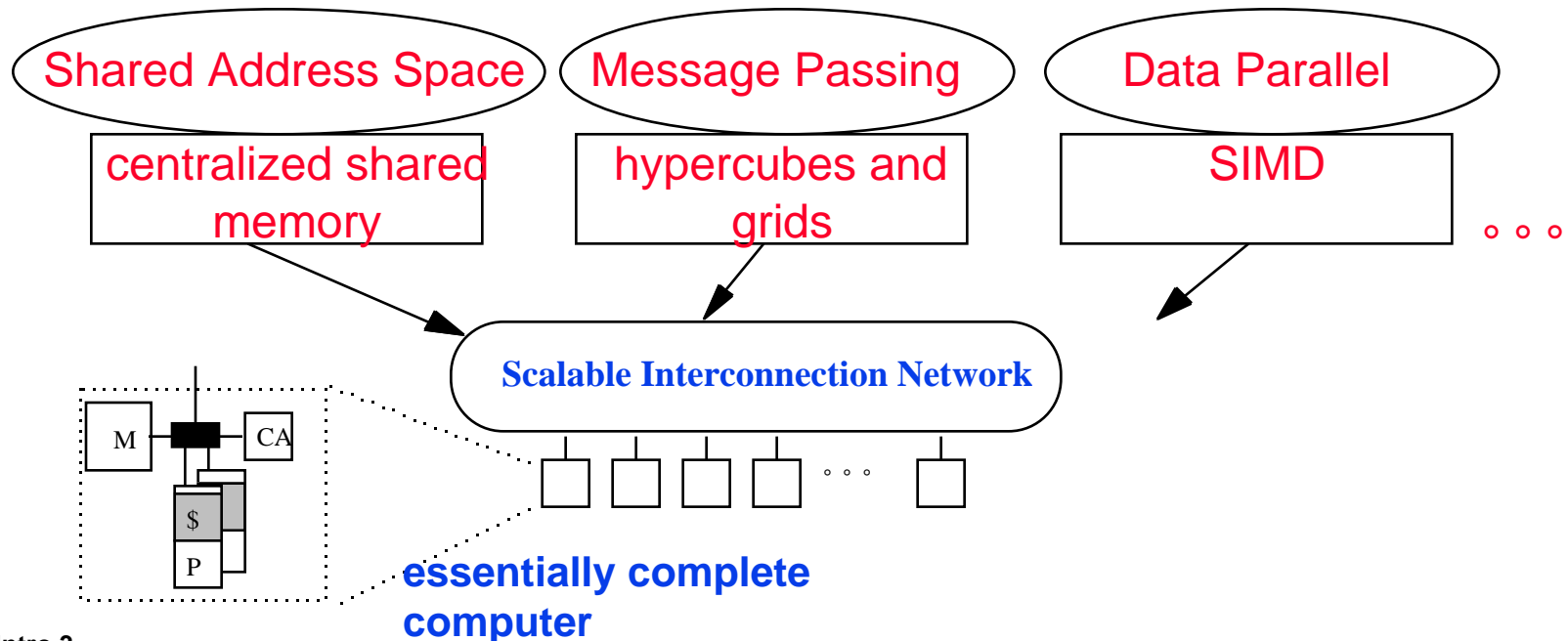
## Outline

---

- **Recap**
- **Evolution of Message Passing Machines**
- **Scalable Shared-Address Multiprocessors**
- **Evolution of Data-Parallel Machines**
- **Directions**

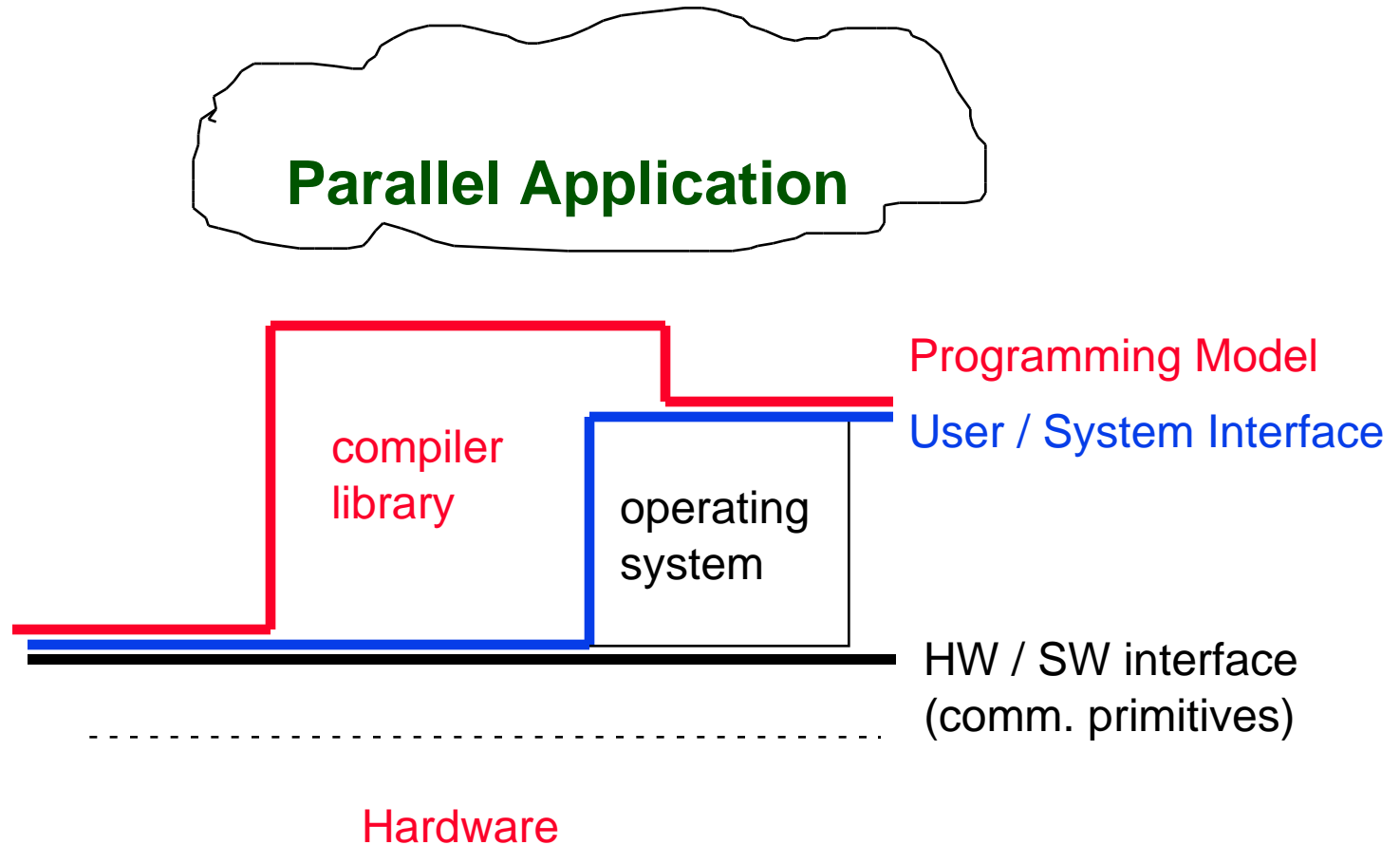
# Recap: Historical Perspective

- Diverse spectrum of parallel machines designed to implement a particular programming model directly
- Technological convergence on collections of microprocessors on a scalable interconnection network
- Map any programming model to simple hardware
  - with some specialization

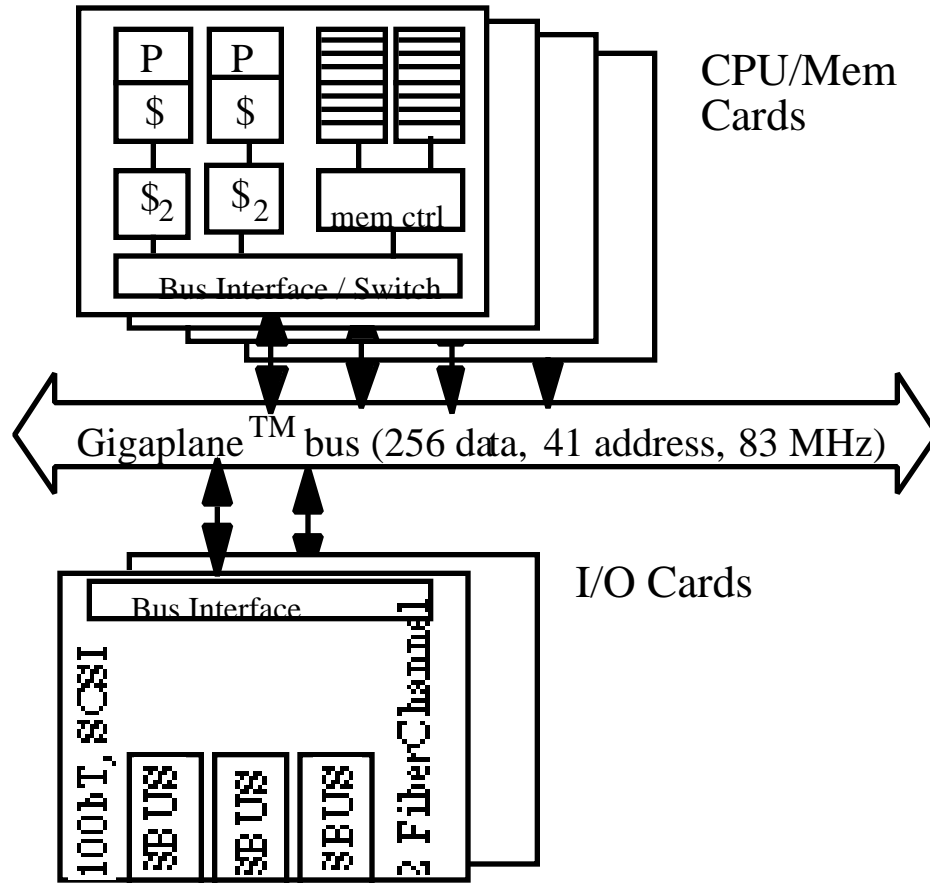


# Recap: Architecture and Programming Model

---

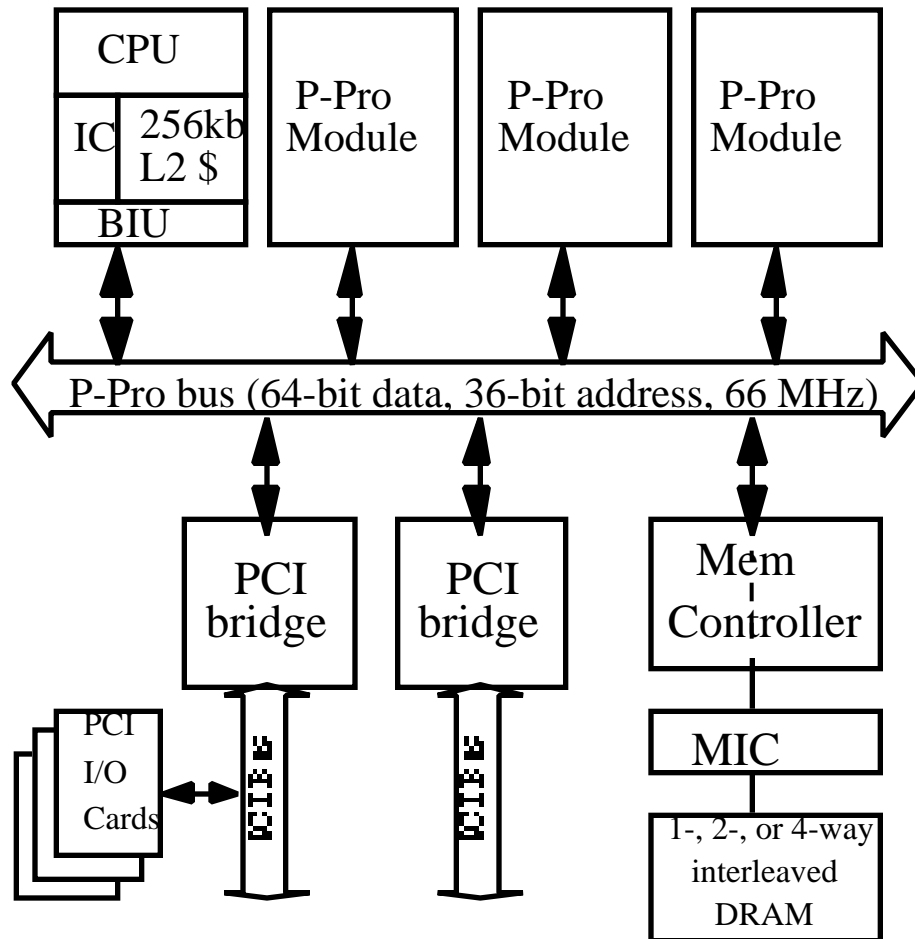


# 90's Pushing the bus to the limit: Sun Enterprise



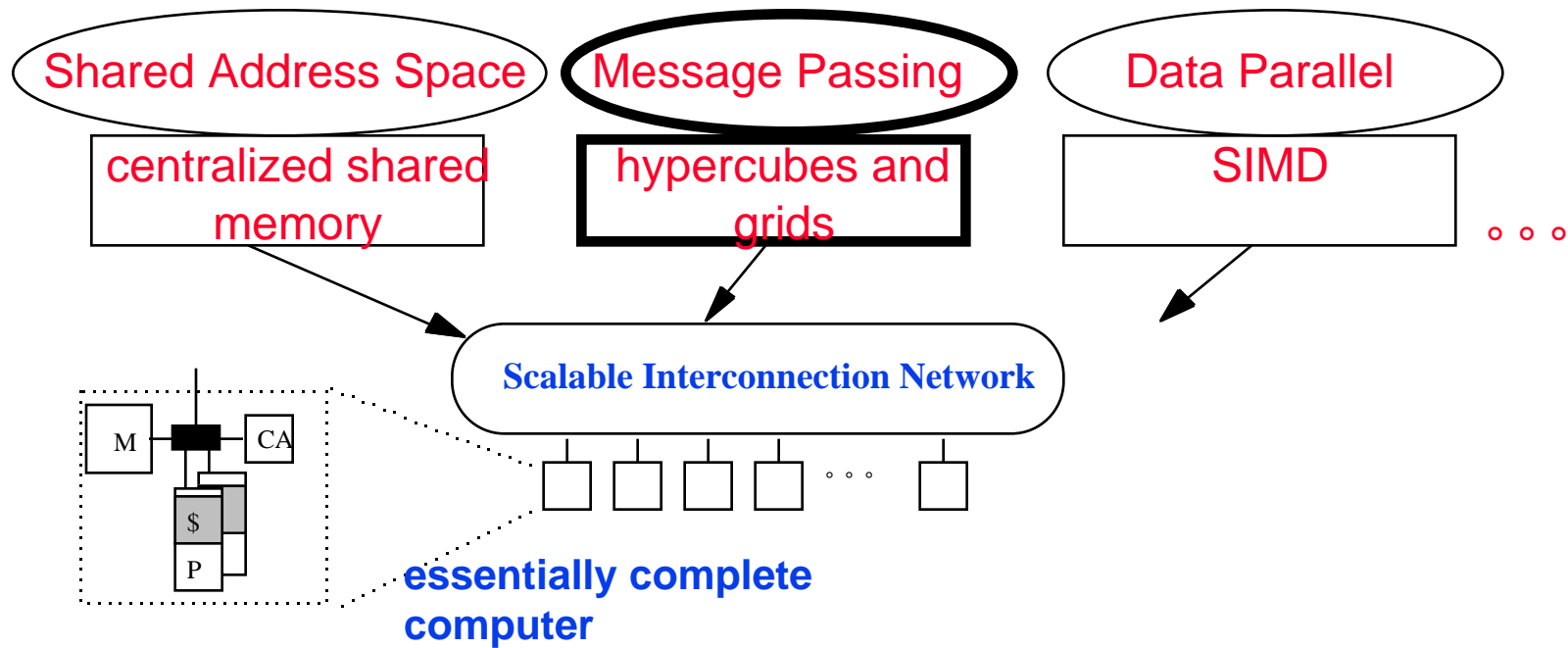
# 90's Pushing the SMP to the masses

---



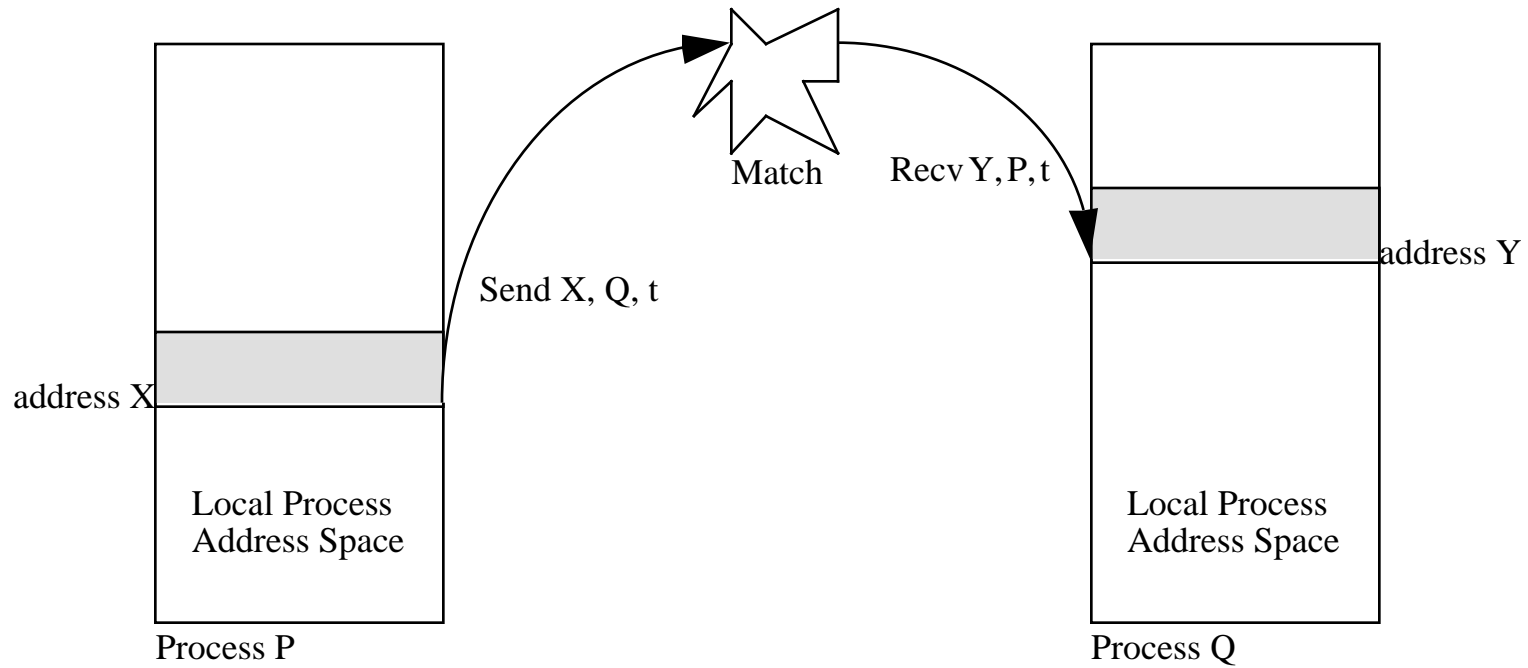
# Message Passing Machines

---



# Recall: Programming Model

---

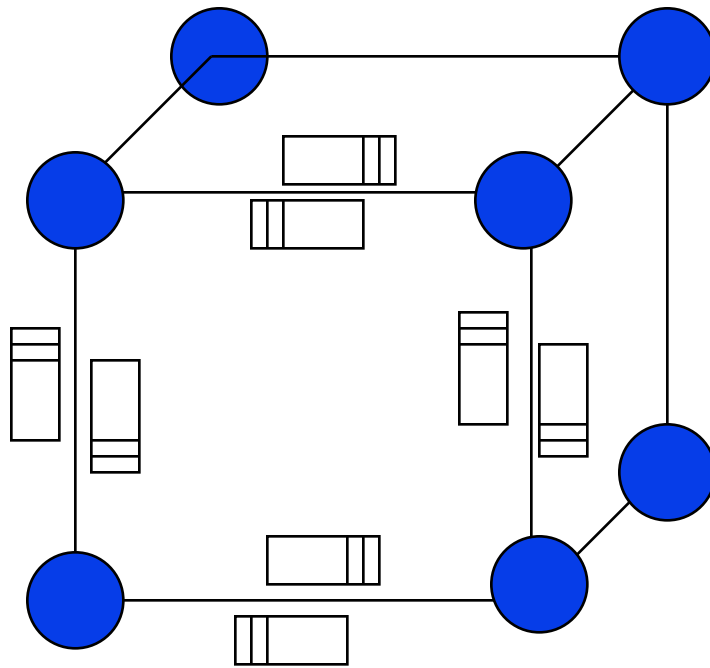


# Historical Perspective

---

- Early machines were built as a collection of microprocessors with bidirectional FIFOs between neighbors.
- Message between two nodes was forwarded along multiple hops

=> strong emphasis on network topology



## Aside: Networks

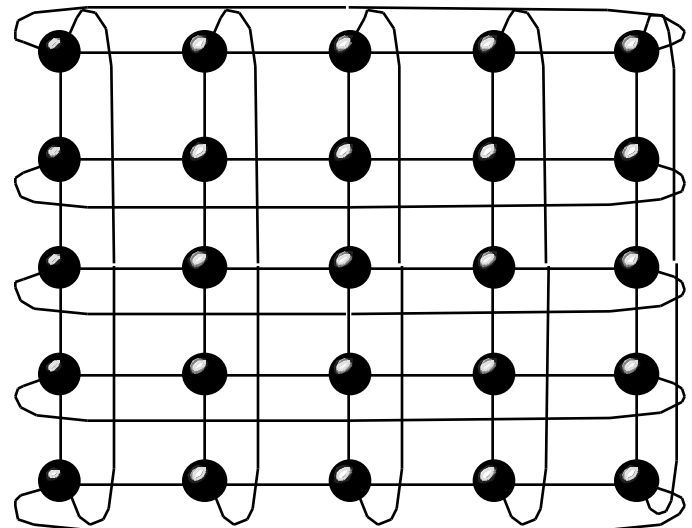
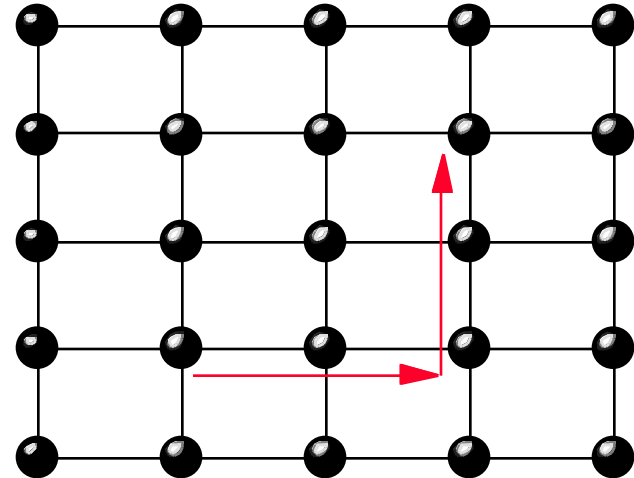
---

- To have a large number of data transfers occurring at once, they must use a large number of distinct wires
- Networks are like streets (or railways)
- Link = Street
- Switch = Intersection
- Distance (hops) = number of blocks traveled
- Routing algorithm = travel plan (for everyone)
- Properties:
  - latency
  - bandwidth
    - link bandwidth
    - aggregate
      - sum of the links
      - bisection

# Meshes and Tori

---

- **Distance**
  - $2\sqrt{n}$  hops average distance
  - Short wires
- $\sqrt{n}$  bisection bandwidth
- Typical route:  $\Delta x, \Delta y$
  
- about half the distance
  
- both usually used as **direct networks**
  - hosts connect to switches



# Hypercubes

◦ Also called binary n-cubes. # of nodes =  $N = 2^n$ .

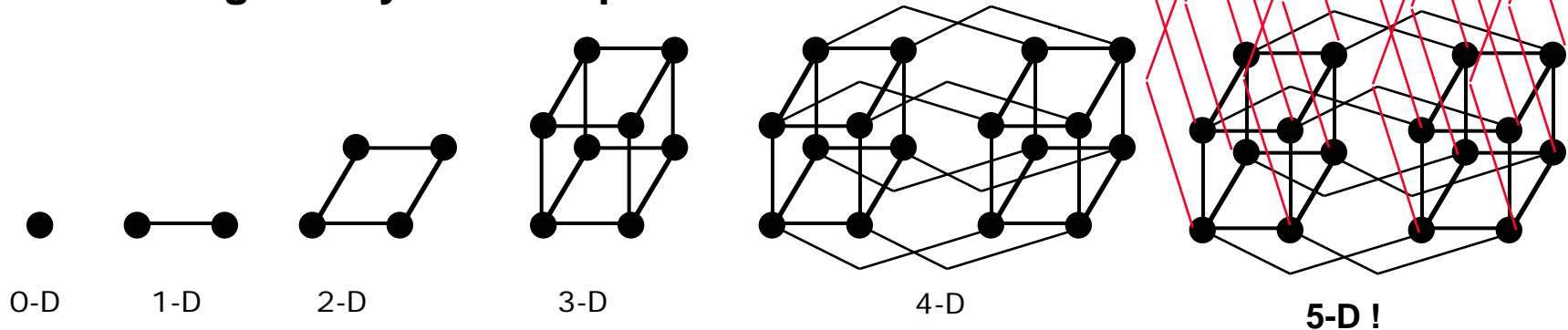
◦ Distance

- $O(\log N)$  Hops

◦ Good bisection BW

◦ Complexity

- Out degree of PE is  $O(\log N)$
- tough to layout in 3-space

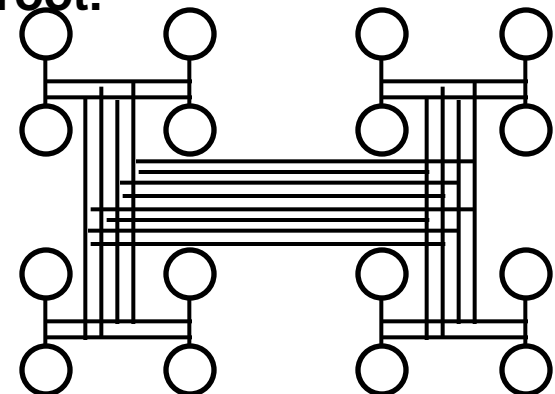
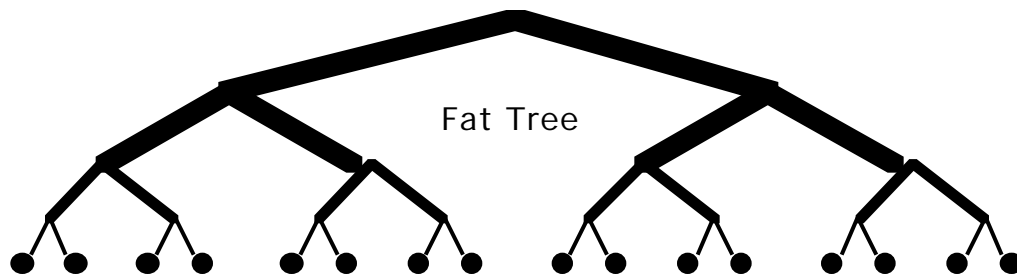
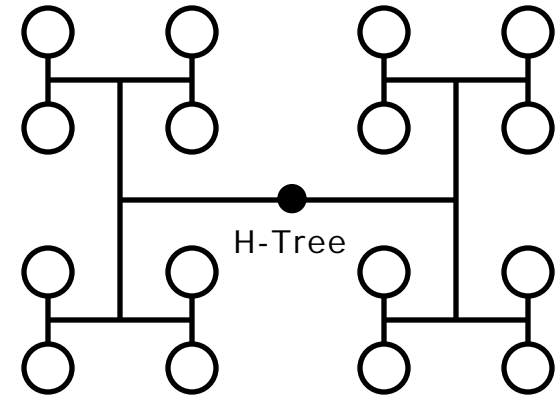


◦ Popular in early message-passing computers

- (e.g., intel iPSC, NCUBE)

# Trees

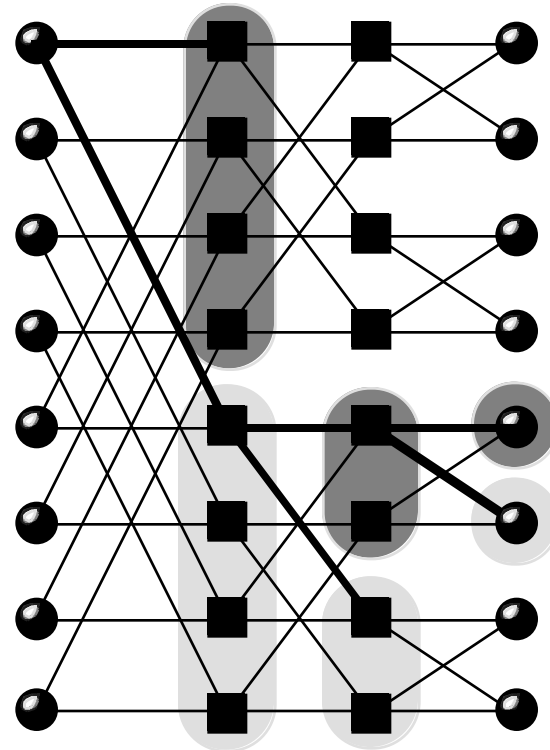
- Cheap: Cost is  $O(N)$ .
- Latency is  $O(\log N)$ .
- Easy to layout as planar graphs
  - (e.g., H-Trees)
- Up/Down routing
- Good partitioning into subtrees
- For random permutations, root can become bottleneck.
- To avoid root being bottleneck => Fat-Trees (used in CM-5)
  - channels are wider as you move towards root.
  - actually more of them



# Butterflies

---

- Split into up & down at each stage
  - A xor B in order
- Log n hops
- Bisection bandwidth:  $O(n)$
- Some long wires
- Easy to partition



See the FFT !!

# Routing Strategies and Latency

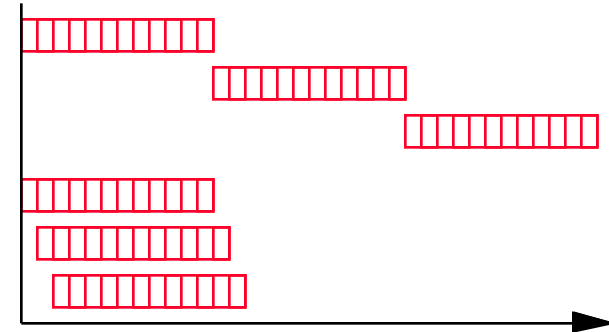
---

- **Store-and-Forward routing:**

- $T_{sf} = T_c \cdot (D \cdot L / W)$ ,  
where  $L$  = msg length,  $D$  = # of hops

- **Wormhole routing:**

- $T_{wh} = T_c \cdot (D + L / W)$
- # of hops is an additive rather than multiplicative factor
- => *nota bena*, fewer hops vs. fatter wires !!!!
- routing delay per hop also decreases with width



- **Virtual Cut-Through routing:**

- Older and similar to wormhole.
- When blockage occurs, however, message is removed from network and buffered.

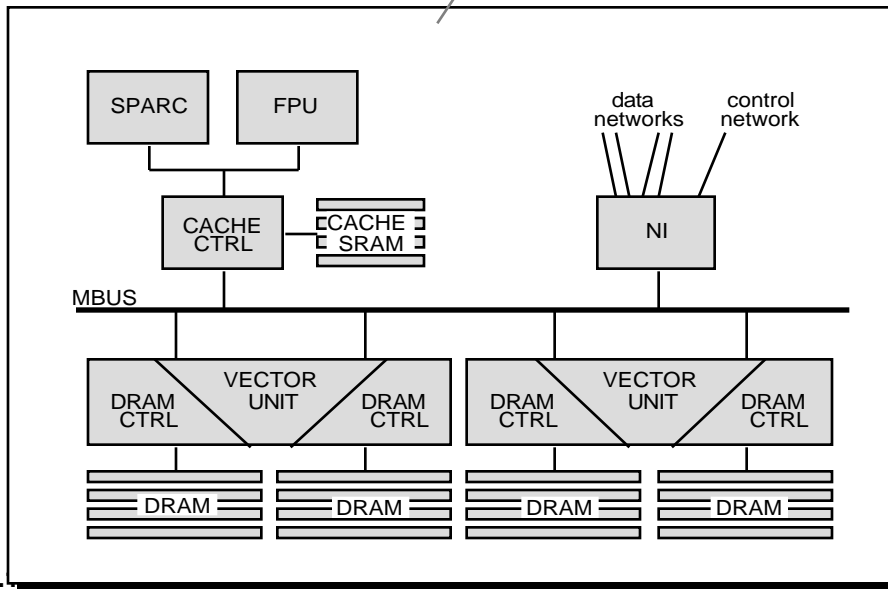
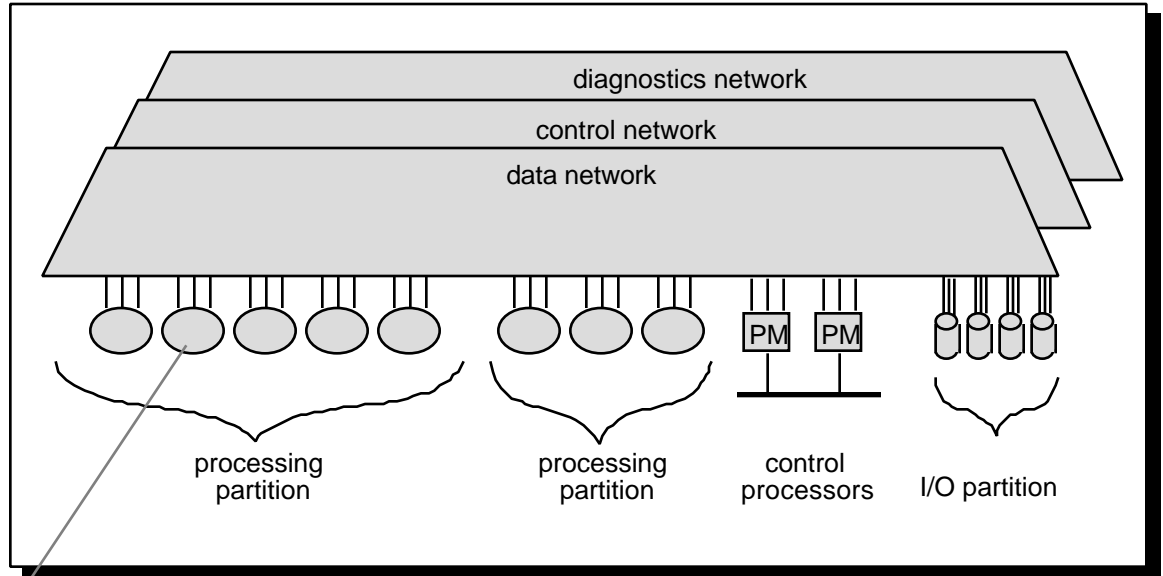
- **Consider 64 byte message on 1 byte links with 5 bytes of routing delay per hop**

## Evolution of Message Passing Machines

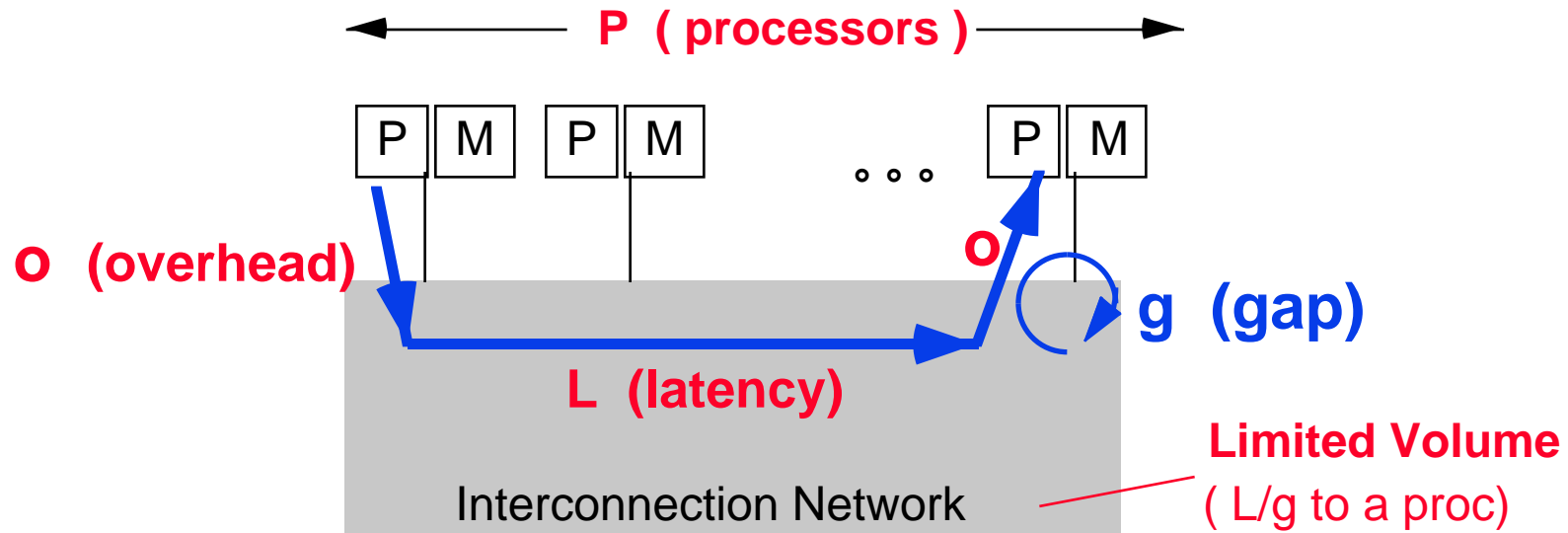
---

- **Direct access FIFO almost immediately replaced by traditional DMA (direct memory access)**
  - processor starts up transfer and does other things
- **Message passing libraries provided store-and-forward in software**
  - can send/recv between any nodes by latency proportional to distance
  - $H*(O+n)$
- **Hardware store-and-forwarding in the (direct connect) routers**
  - fixed overhead +  $H*N$
- **Wormhole routing in hardware**
  - $O + N + Hr$

# Example: CM-5

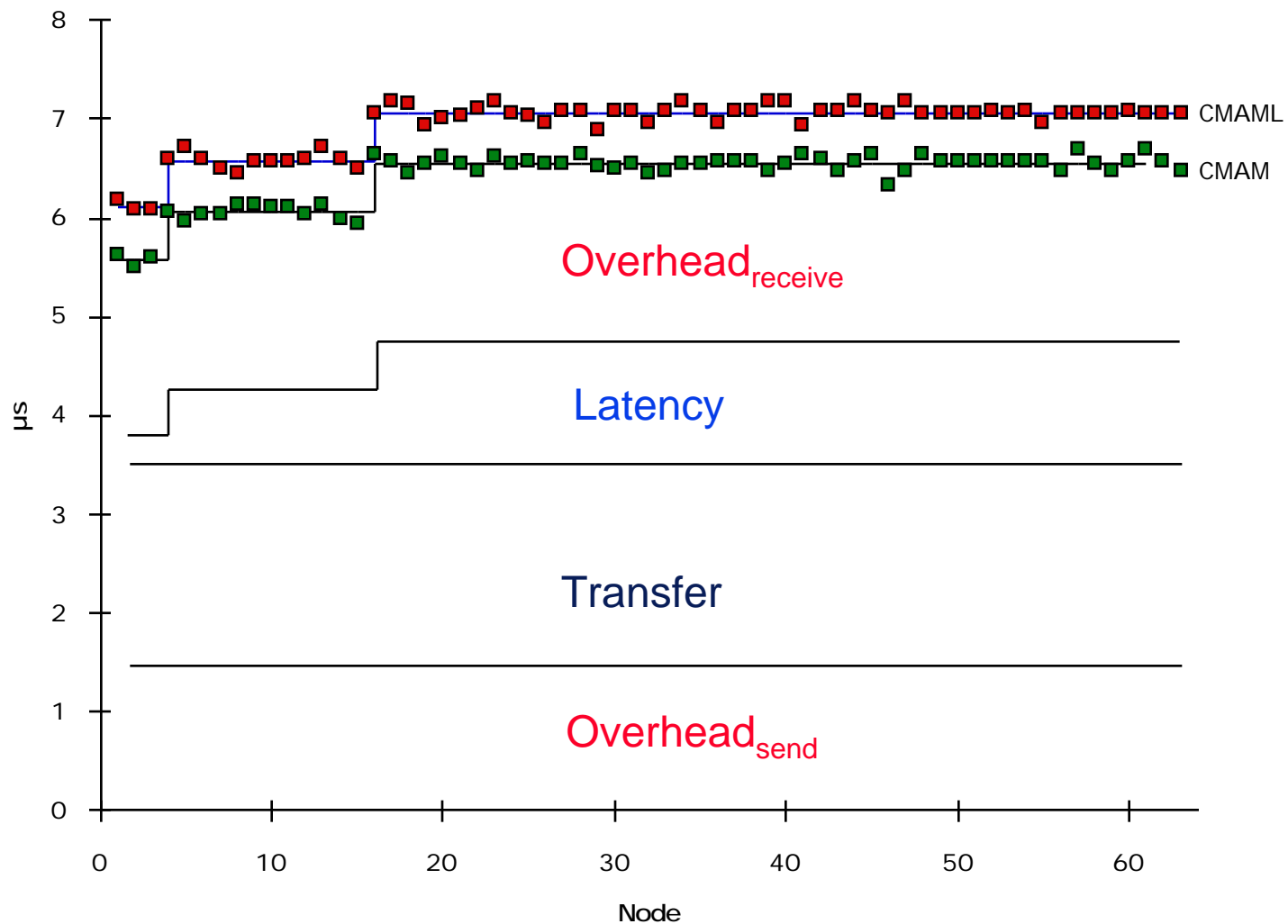


# Abstract Model of Scalable Machines: LogP

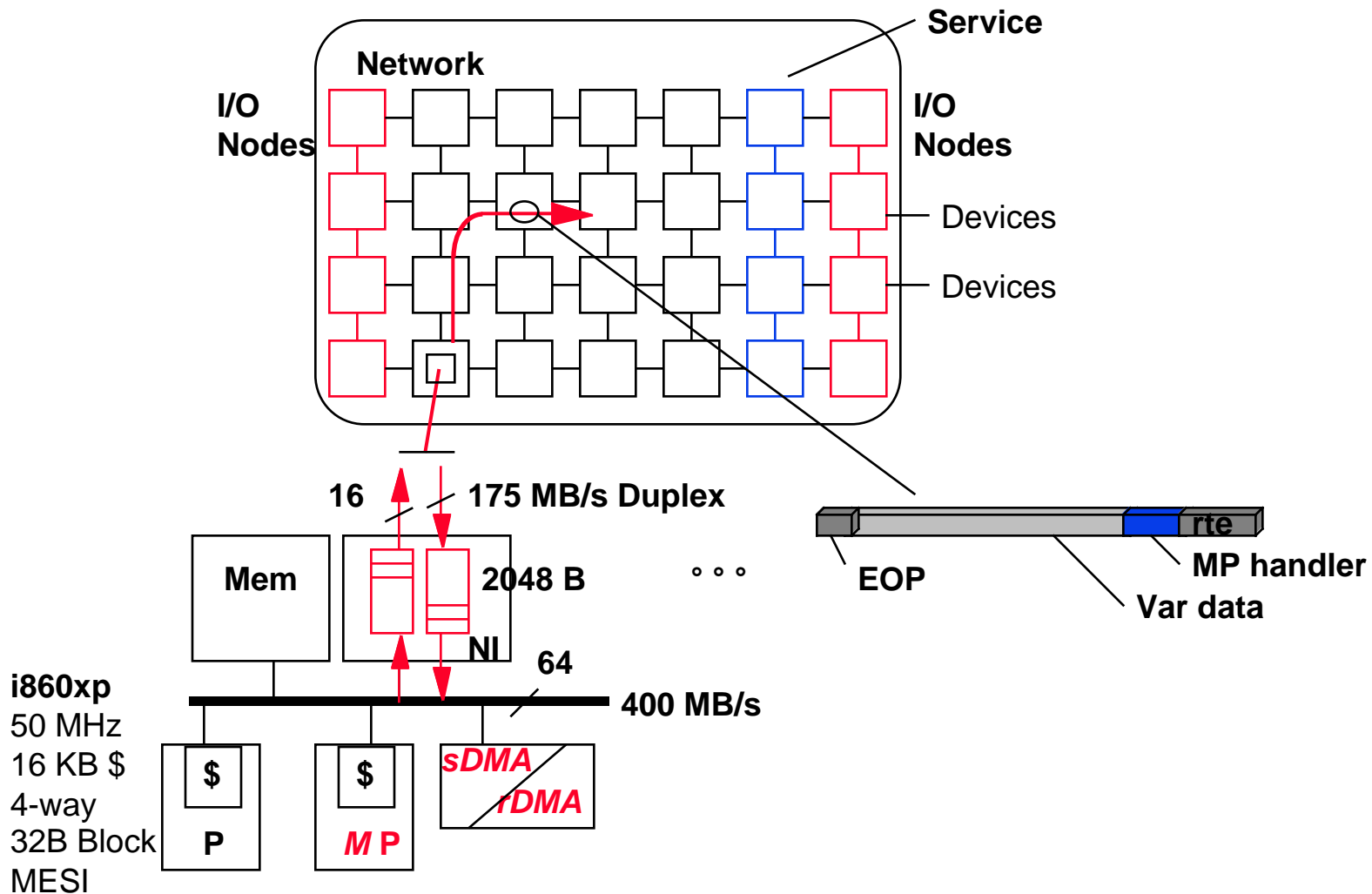


- $L$  (latency) in sending a (small) message between modules
- $o$  (overhead) felt by the processor on sending or receiving msg
- $g$  (gap) between successive sends or receives ( $1/\text{rate}$ )
- $P$  (processors)

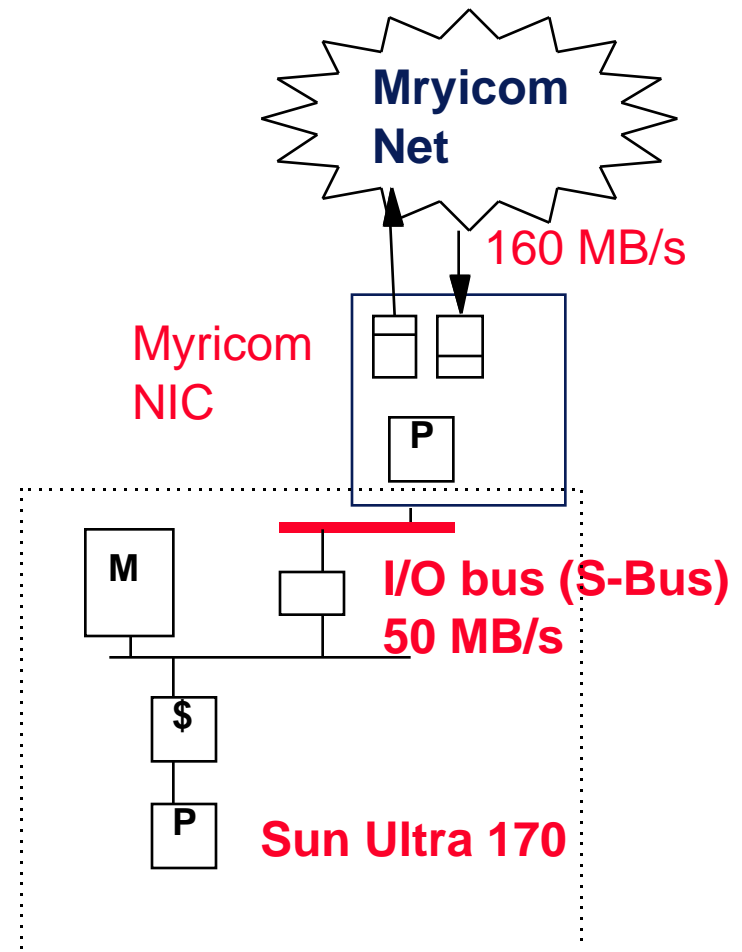
# CM-5 One-way Message Timing



# Example: Intel Paragon

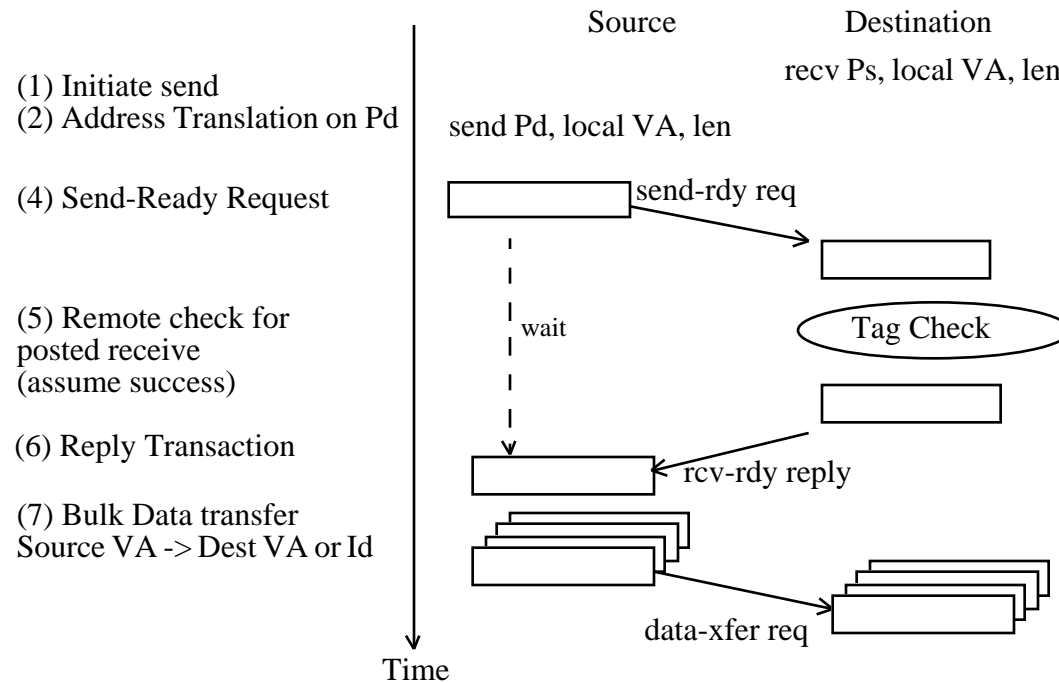


# Example: NOW

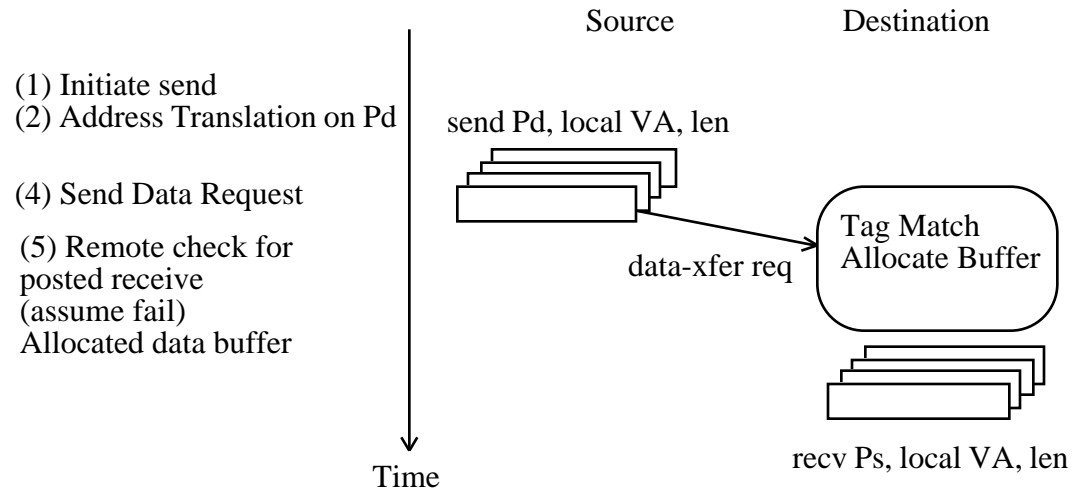


# Realizing Message Passing (synchronous)

- Send completes after matching recv and source data sent
- Receive completes after data transfer complete from matching send

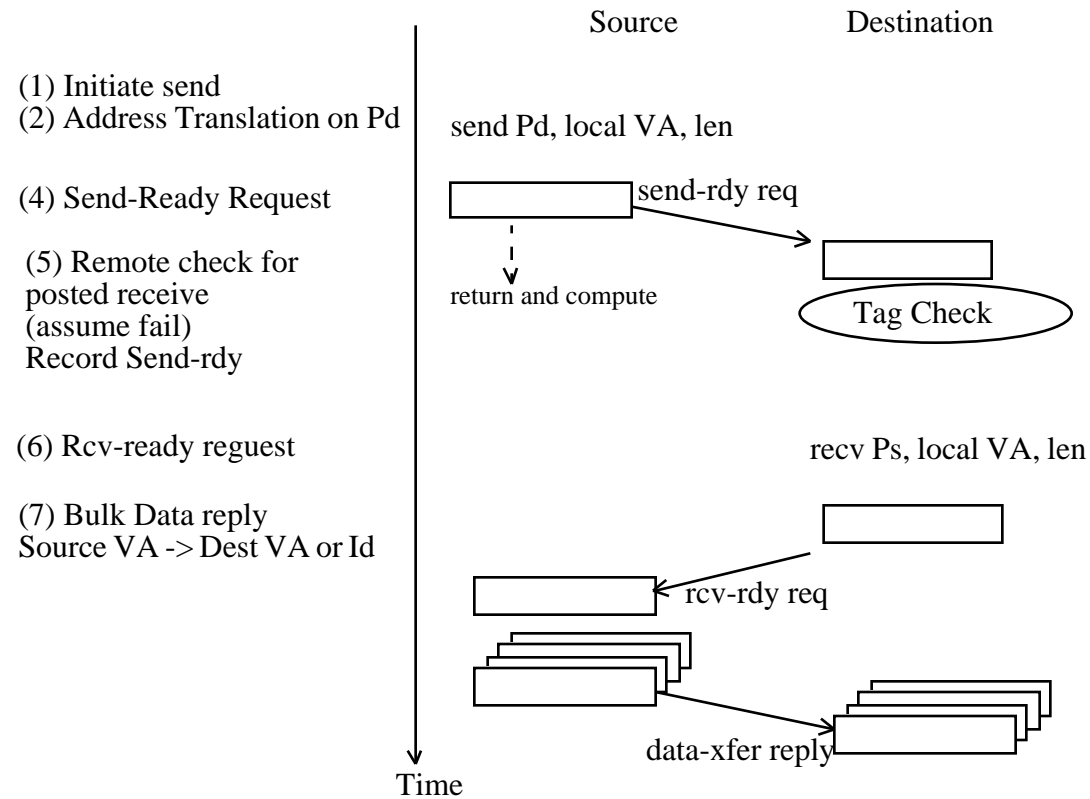


# Message Passing (Asynchronous)



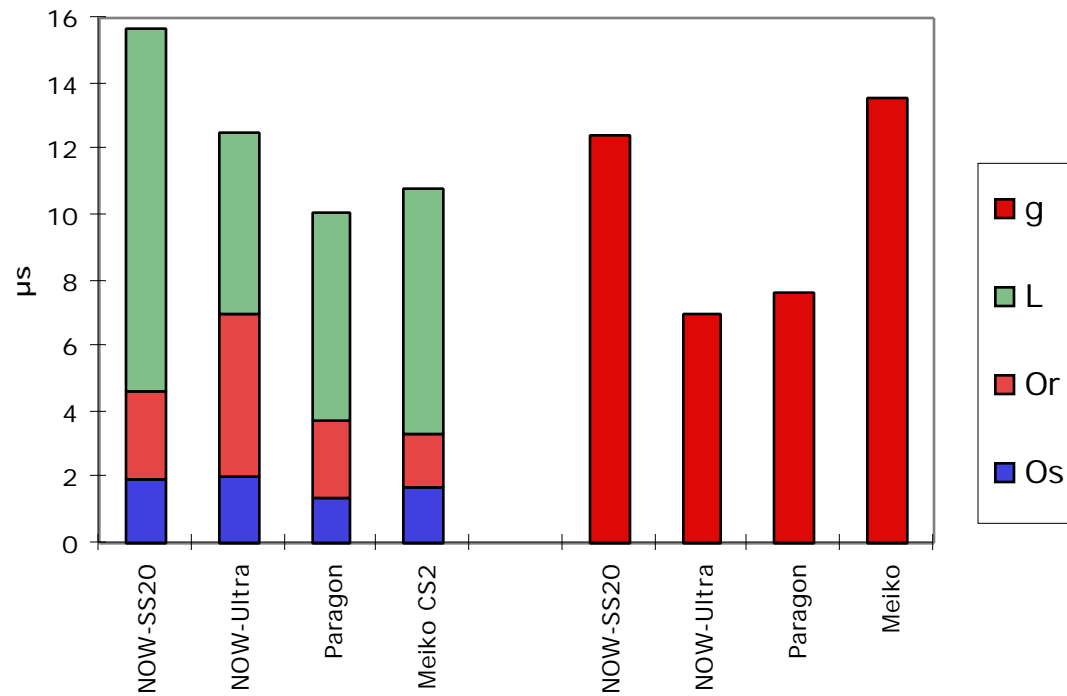
- **Send variants complete after source area free or after command has been accepted**
- **Optimistic single-phase protocols assumes the destination can buffer send data with no matching receive *on demand*.**
  - amplified input buffering problem
  - fix is to use 3-phase protocol (plus fixed credits for small msgs) and buffer at the source

# Asynchronous (Safer) Msg Passing

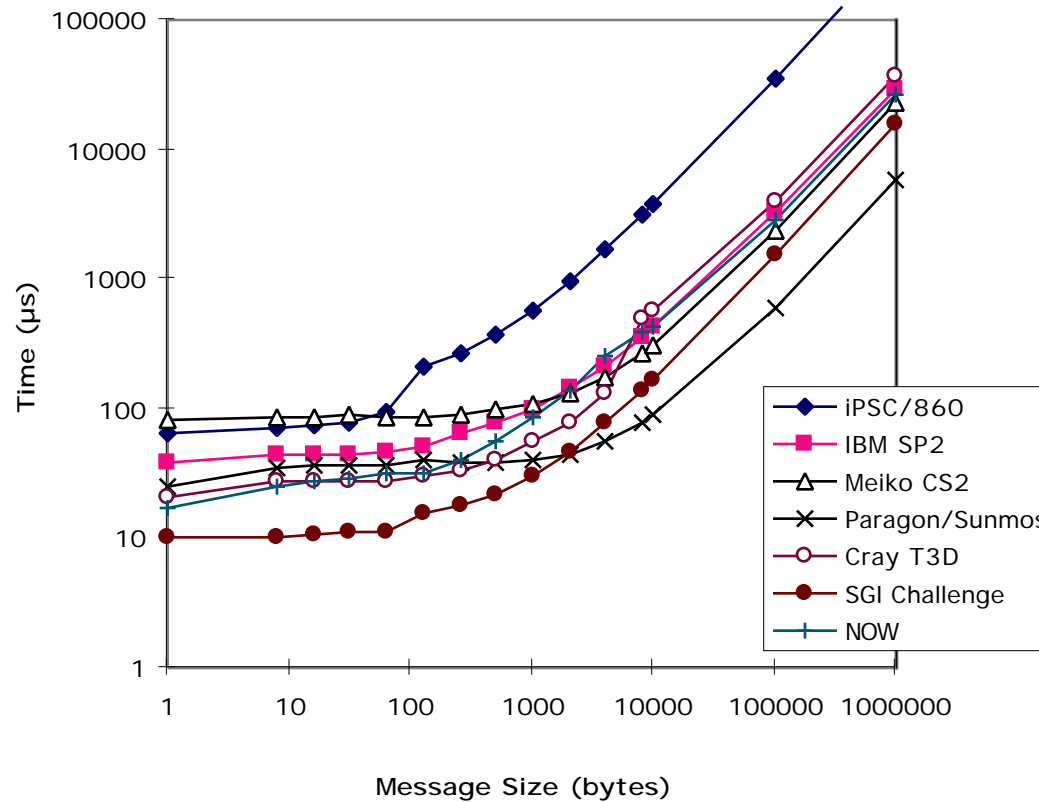


# Comparison of Base Message Event

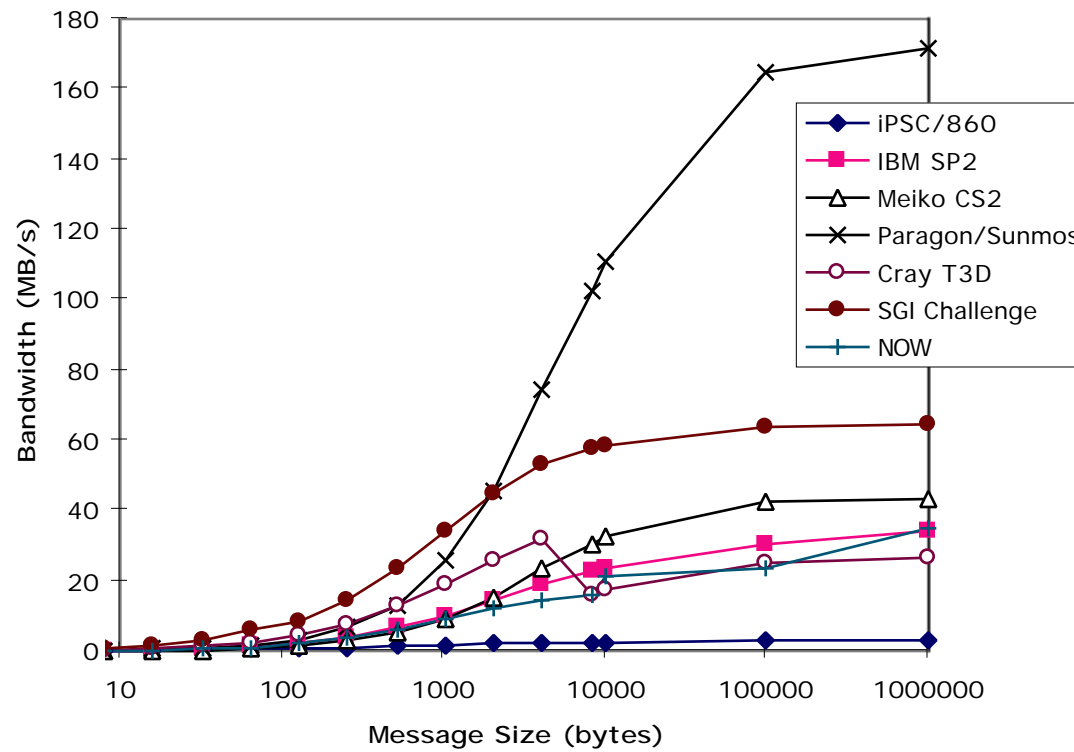
---



# Comparison of Message Passing Performance

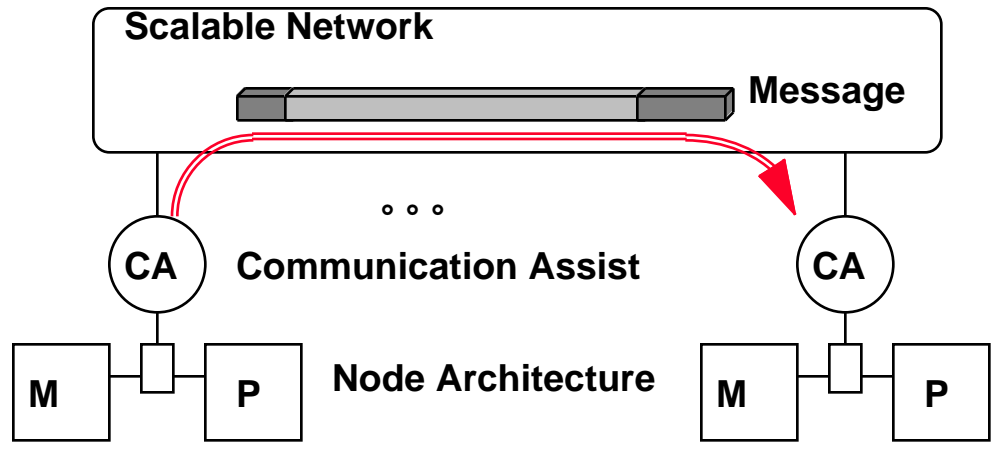


# Comparison: 1-1 Bandwidth

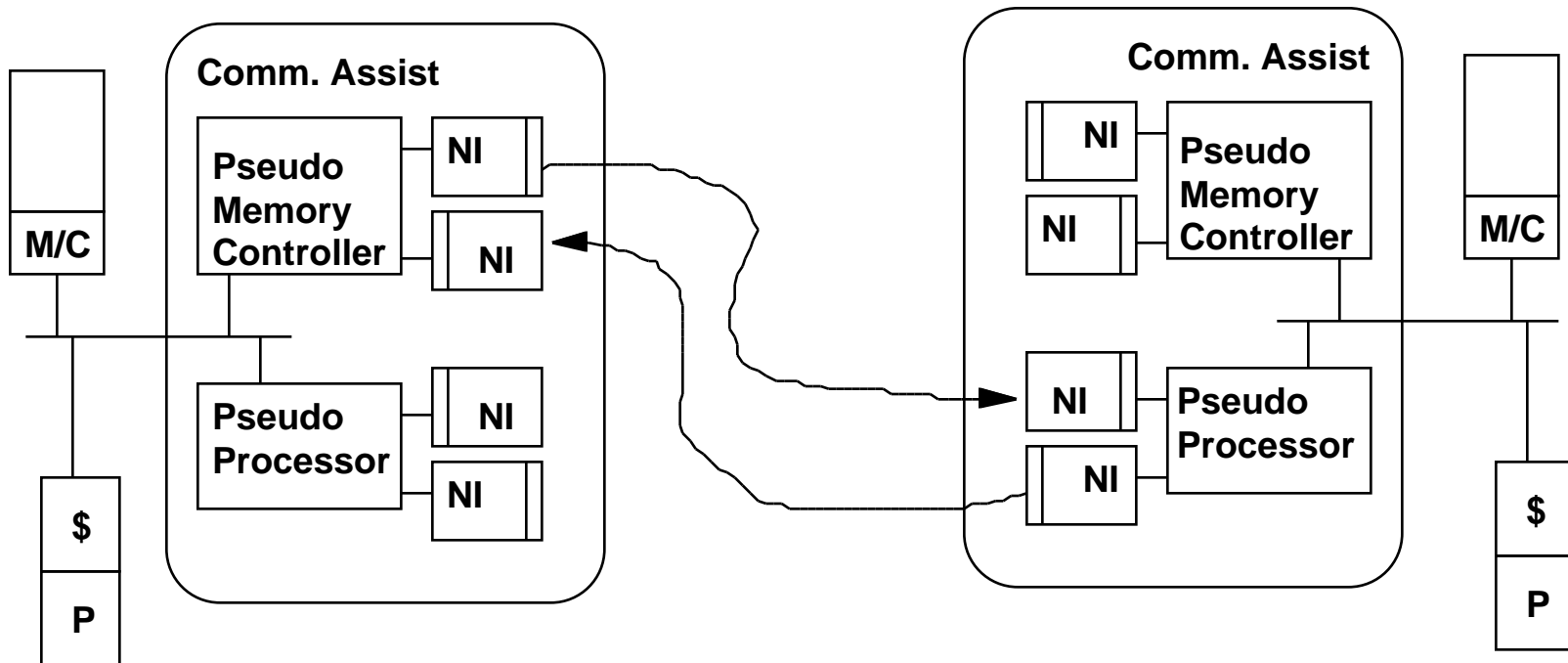


# Large Scale Shared Address Space

---

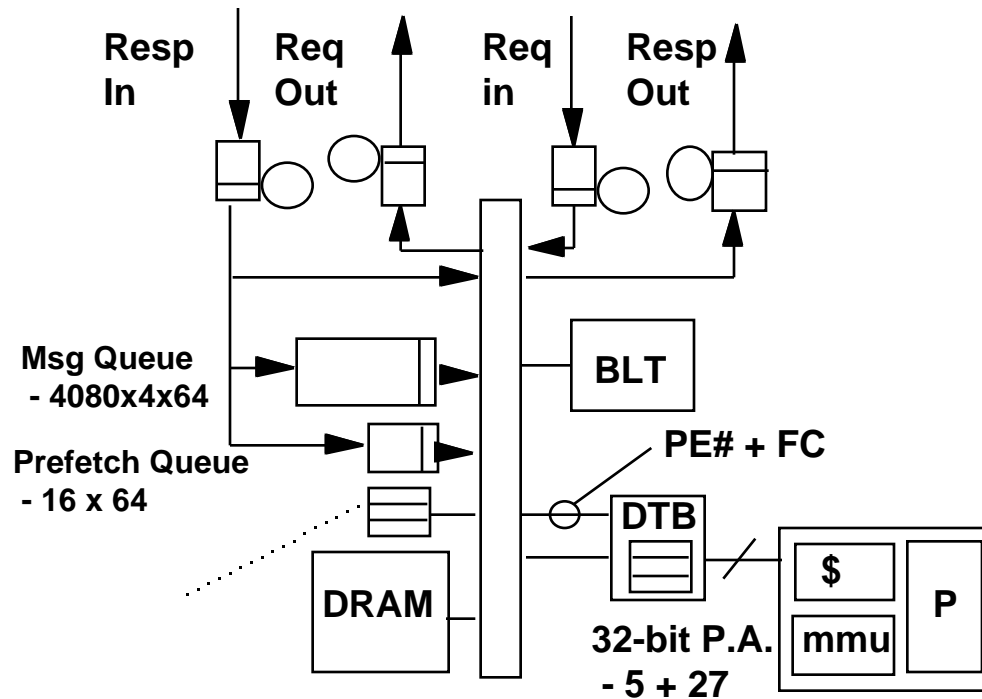


# Large Scale Shared Physical Address

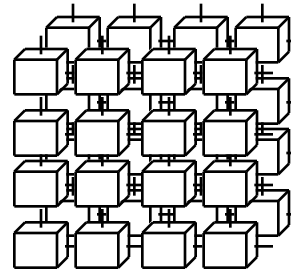


- Processor performs load
- Pseudo-memory controller turns it into a message transaction with a remote controller, which performs the memory operation and replies with the data.
- Examples: BBN butterfly, Cray T3D

# Cray T3D



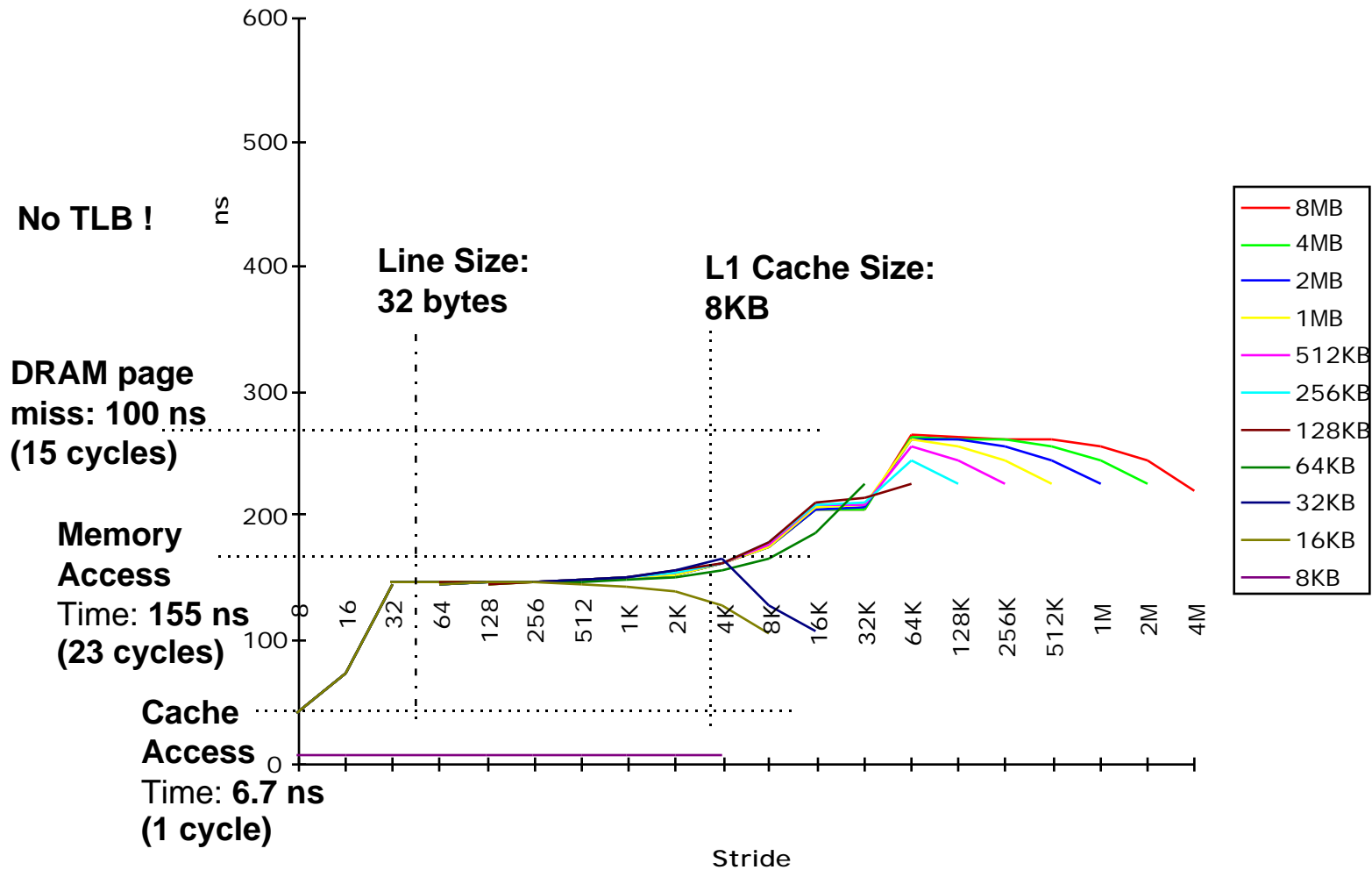
- Special Registers**
- swaperand
  - fetch&add
  - barrier



- 3D Torus of Pair of PEs**
- share net & BLT
  - upto 2048
  - 64 MB each

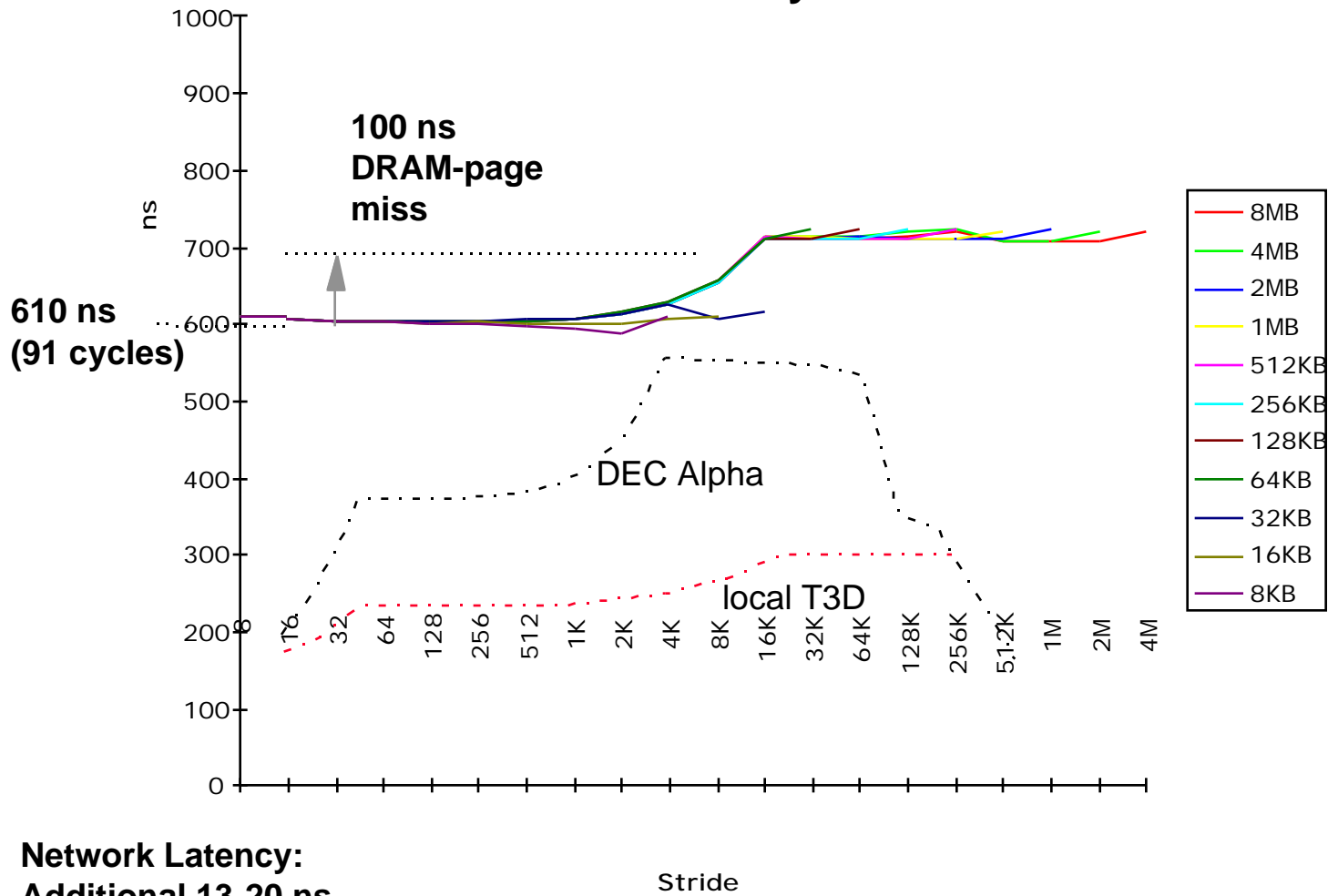
- 150 MHz Dec Alpha (64-bit)
- 8 KB Inst + 8 KB Data
- 43-bit Virtual Address
- 32 & 64 bit mem + byte operations
- Non-blocking stores + mem-barrier
- Prefetch
- Load-lock, Store Conditional

# T3D Local Read (average latency)



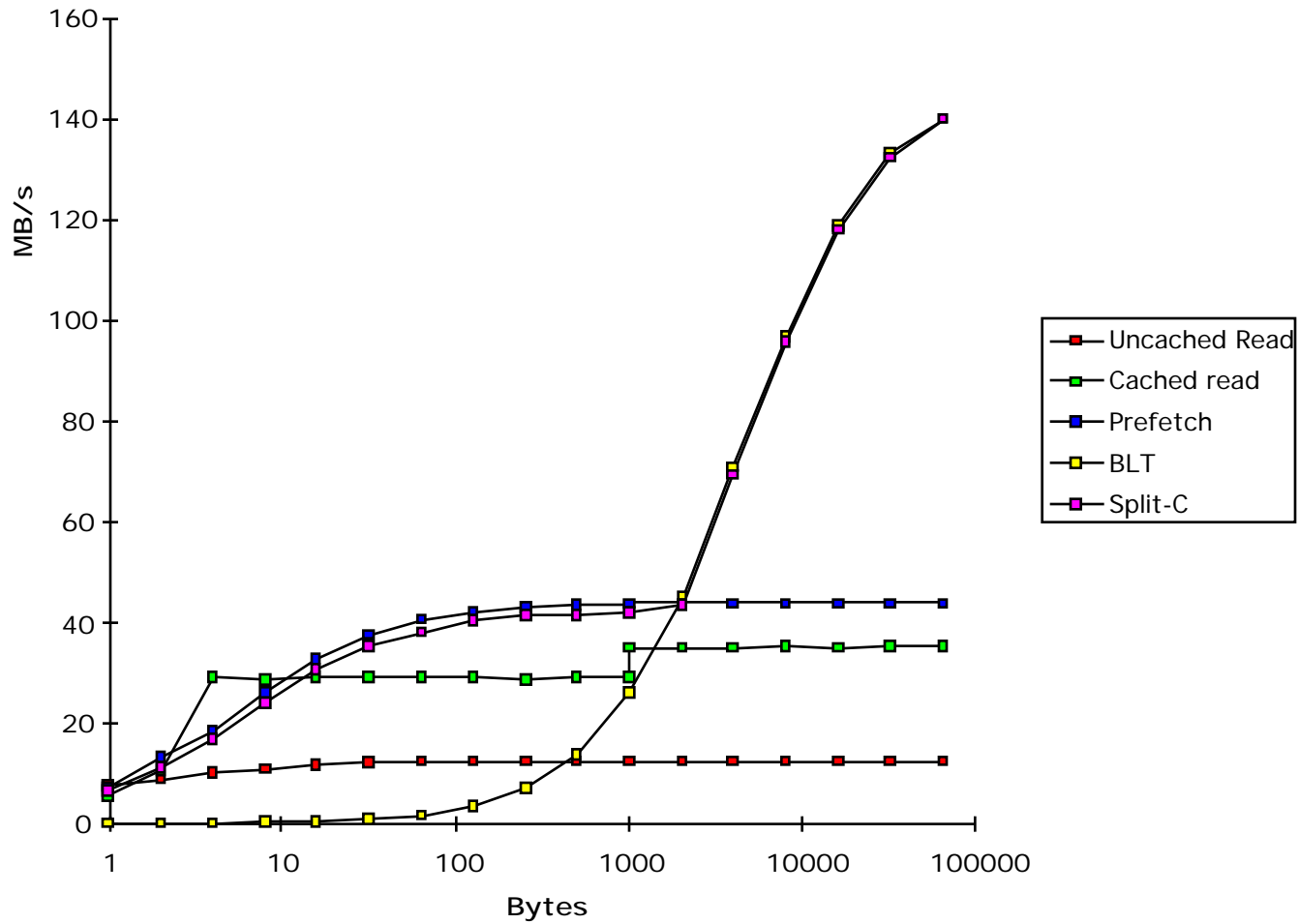
# T3D Remote Read Uncached

## 3 - 4x Local Memory Read !



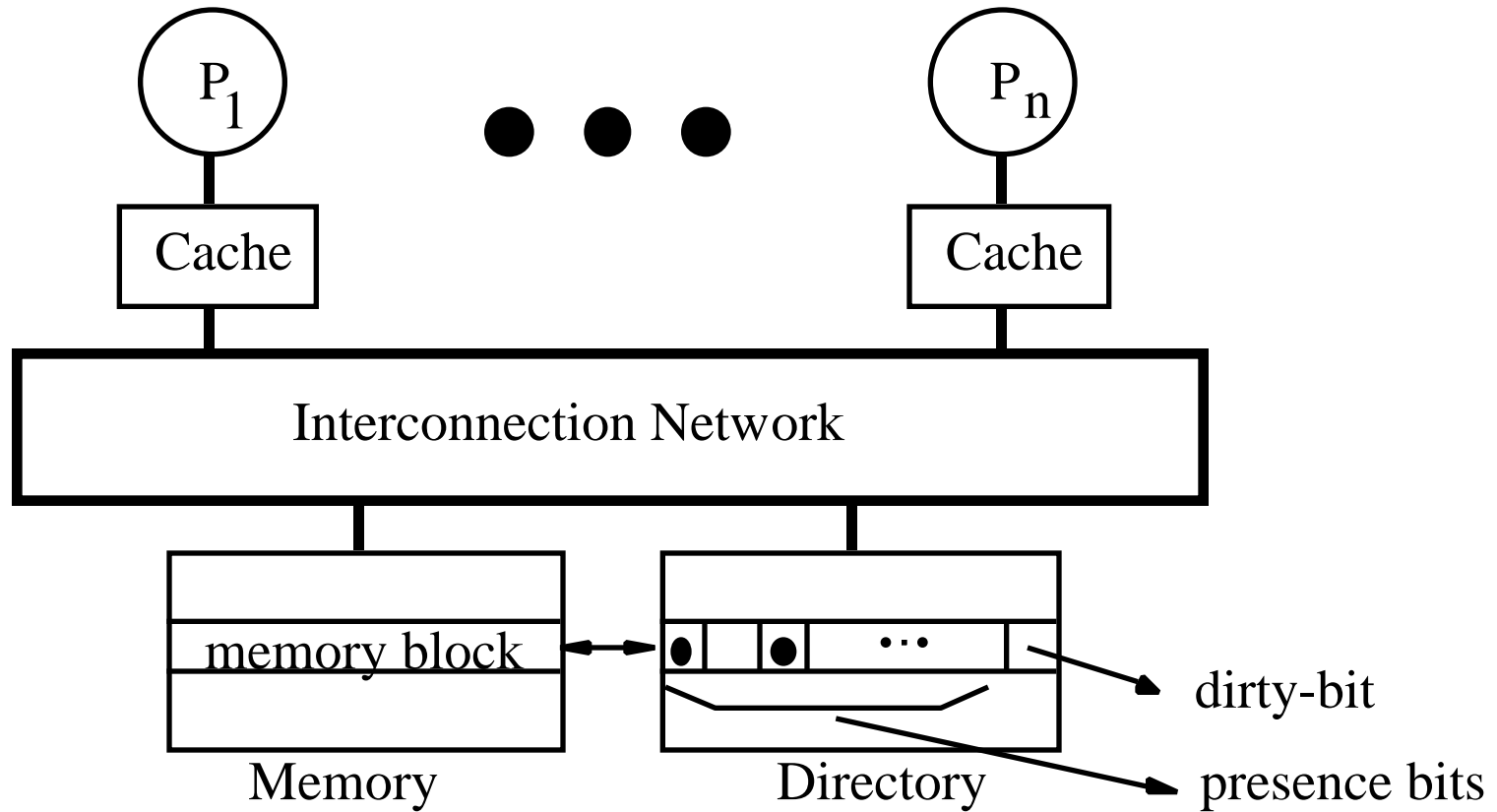
Network Latency:  
Additional 13-20 ns  
(2-3 cycles) per hop

# Bulk Read Options



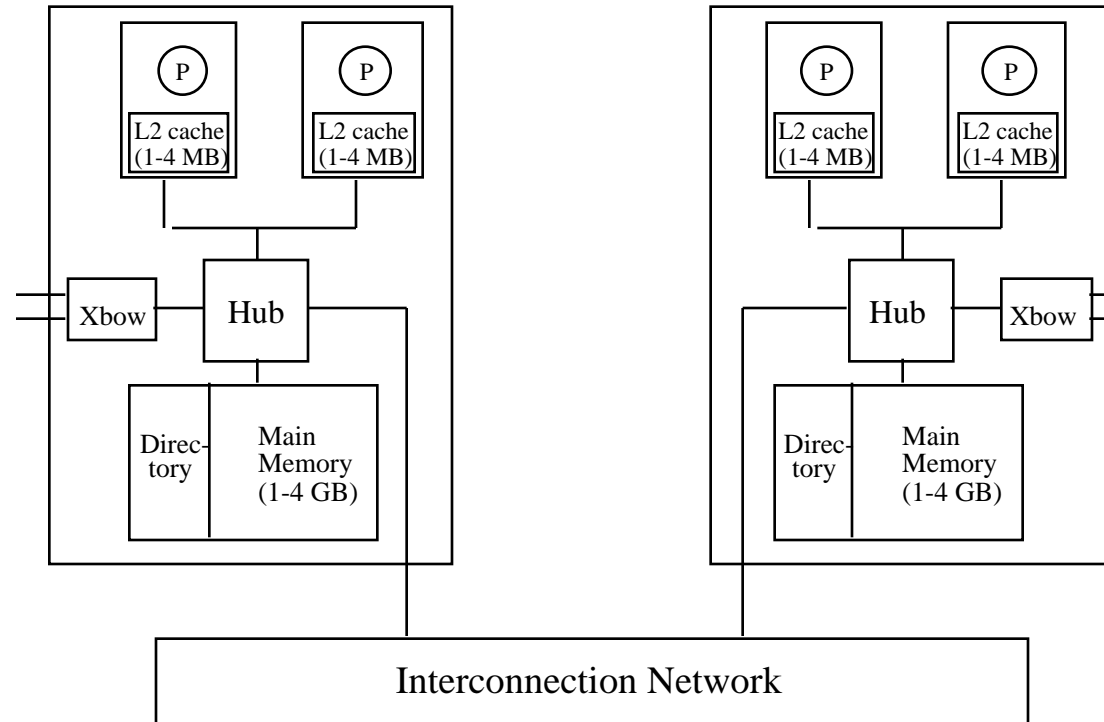
# 90 Scalable, Cache Coherent Multiprocessors

---



# SGI Origin 2000

---



## Where are things going

---

### ◦ High-end

- collections of almost complete workstations/SMP on high-speed network
- with specialized communication assist integrated with memory system to provide global access to shared data

### ◦ Mid-end

- almost all servers are bus-based CC SMPs
- high-end servers are replacing the bus with a network
  - Sun Enterprise 10000, IBM J90, HP/Convex SPP
- volume approach is Ppro quadpack + SCI ring
  - Sequent, Data General

### ◦ Low-end

- SMP desktop is here

### ◦ Major change ahead

- SMP on a chip as a building block

# Data Parallel Architectures

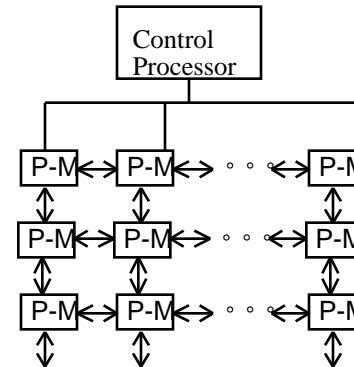
---

- **Programming model**

- operations are performed on each element of a large (regular) data structure in a single step
- arithmetic, global data transfer

- **processor is logically associated with each data element, general communication, and cheap global synchronization.**

- driven originally by simple O.D.E.



# Application

---

## ◦ Example:

Each PE contains an employee record with his/her salary

Code: `if salary > 100K then`

`salary = salary *1.05`

`else`

`salary = salary *1.10`

Logically, the whole operation is a single step

Notion of enabled processors.

## ◦ Other examples:

- Finite differences, linear algebra, ...
- Document searching, graphics, image processing, ...

## ◦ Recent Machines:

- Thinking Machines CM-1, CM-2 and CM-5
- Maspar MP-1 and MP-2,
- Wavetracer

## Evolution and Convergence

---

- **Rigid control structure (SIMD in Flynn's Taxonomy)**
  - SISD = uniprocessor, MIMD = multiprocessor
- **Cost savings in centralized instruction sequencer**
  - 60's when CPU was a cabinet of equipment
  - mid 80's when 32-bit slices of datapath would just fit on a chip
- **Simple, regular calculations usually have good locality**
  - realize on SM or MP machine with decent compiler
  - may still require fast global synchronization
- **Programming model converges with SPMD**
  - single program multiple data
  - compiled either to shared address or message passing
  - benefits from hardware support for fast barrier

# Convergence

- Diverse spectrum of parallel machines designed to implement a particular programming model directly
- Technological convergence on collections of microprocessors on a scalable interconnection network
- Map any programming model to simple hardware
  - with some specialization

