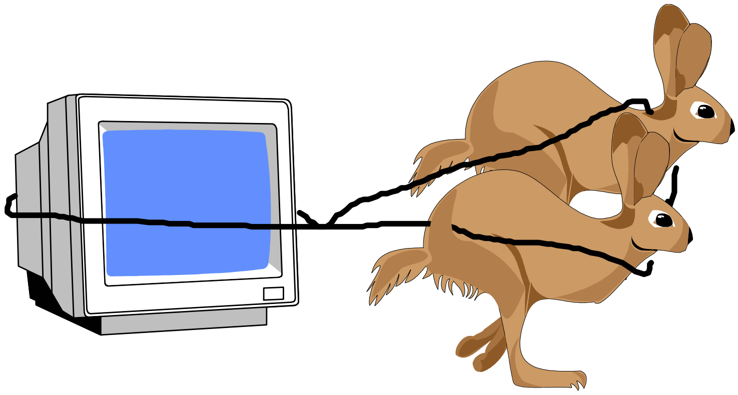
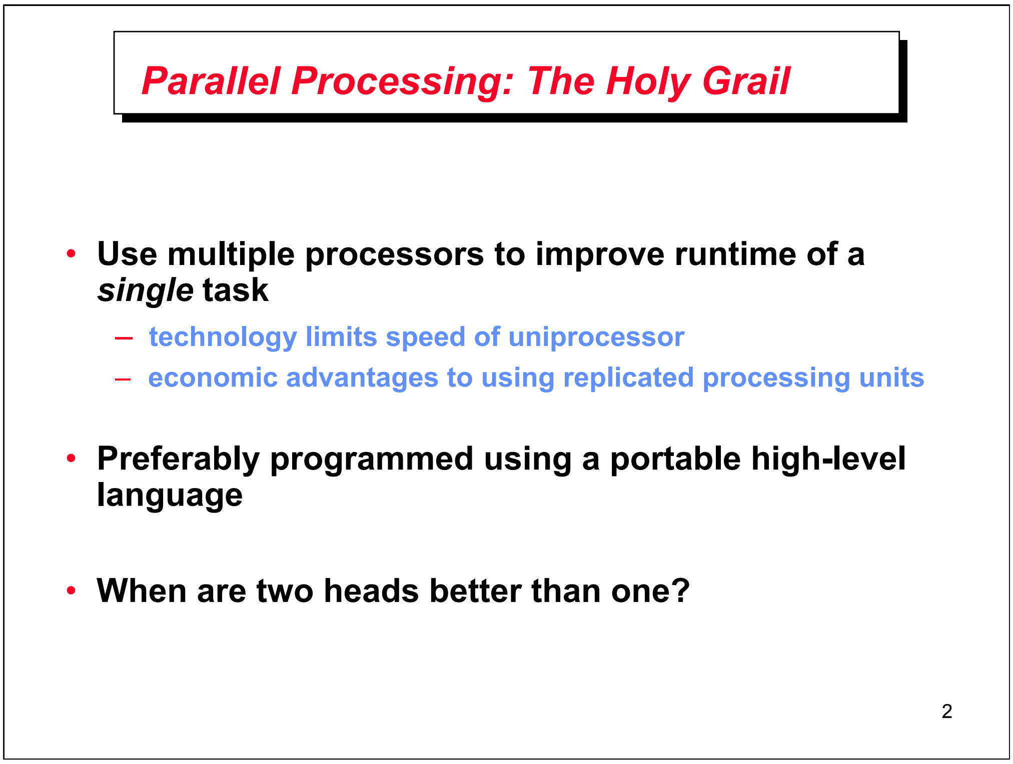
Parallel Processors

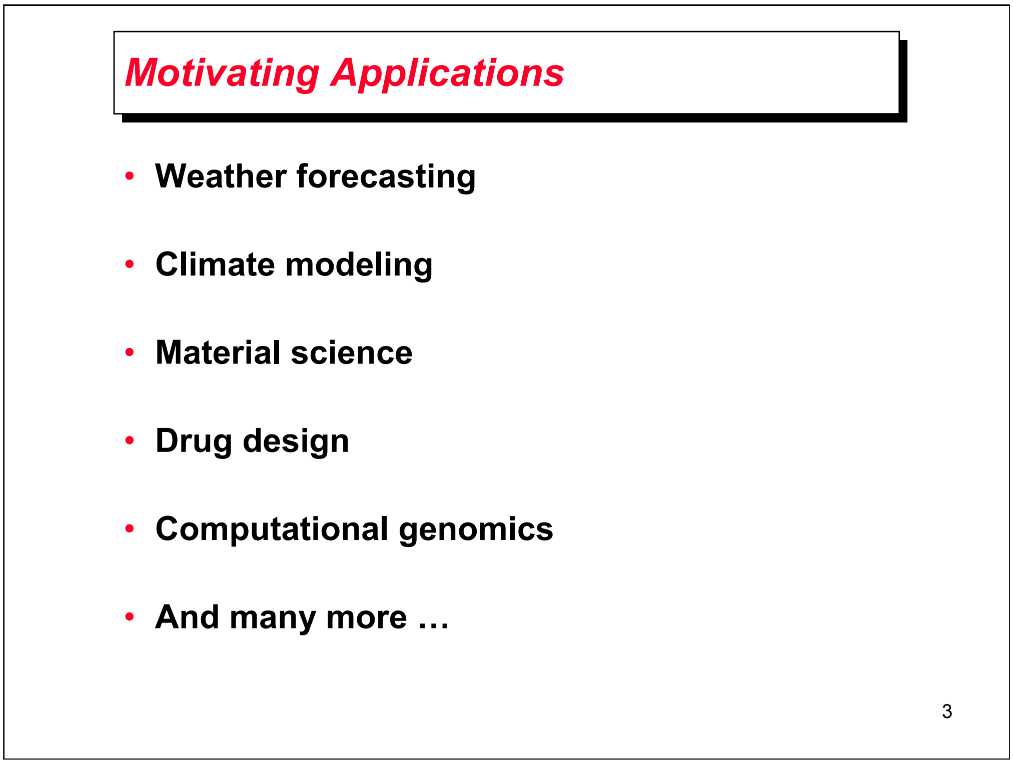




Parallel Processing: The Holy Grail

- Use multiple processors to improve runtime of a *single* task
  - technology limits speed of uniprocessor
  - economic advantages to using replicated processing units
- Preferably programmed using a portable high-level language
- When are two heads better than one?

Jigsaw puzzle analogy



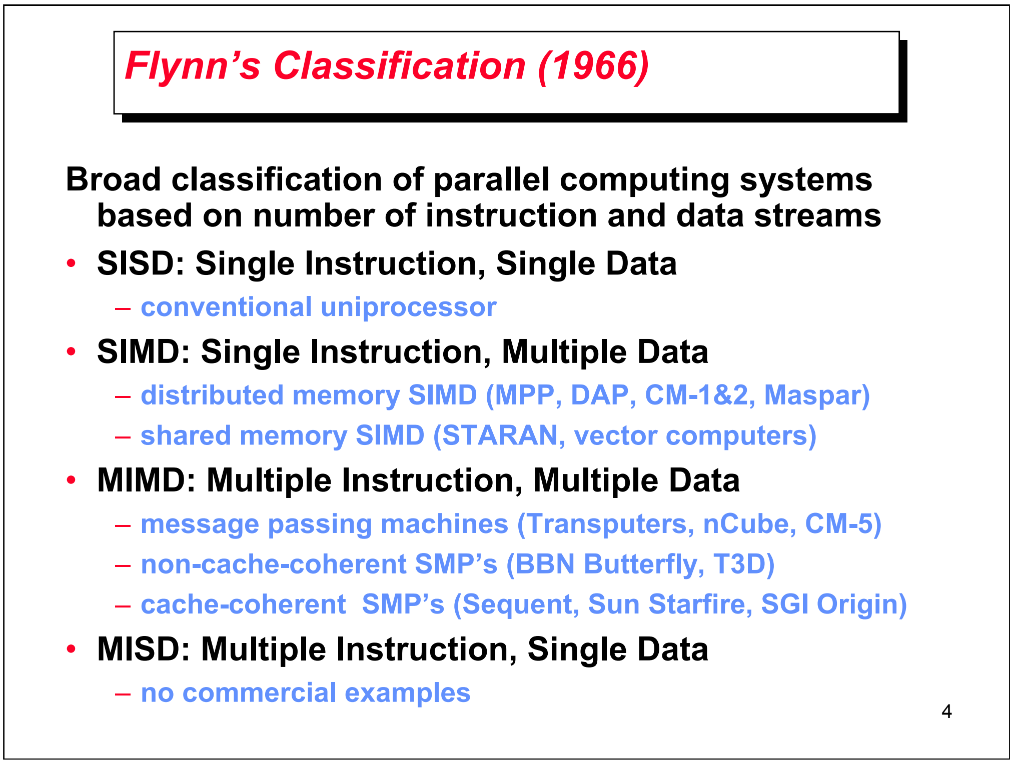
Motivating Applications

- **Weather forecasting**
- **Climate modeling**
- **Material science**
- **Drug design**
- **Computational genomics**
- **And many more …**

Number crunching apps

Claim that weather forecasts are accurate over 5 days, and would like to improve that.

Nonlinear phenomena.



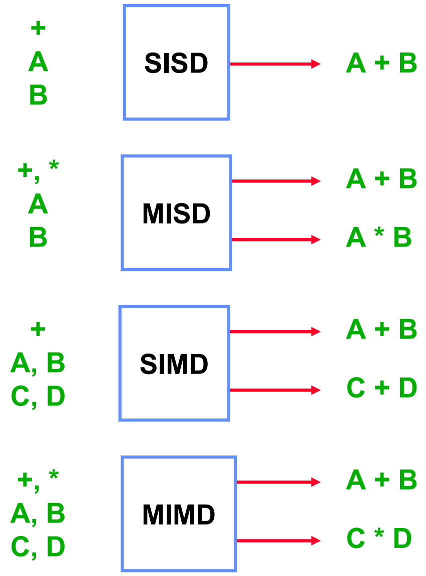
Flynn’s Classification (1966)

Broad classification of parallel computing systems based on number of instruction and data streams

- **SISD: Single Instruction, Single Data**
  - conventional uniprocessor
- **SIMD: Single Instruction, Multiple Data**
  - distributed memory SIMD (MPP, DAP, CM-1&2, Maspar)
  - shared memory SIMD (STARAN, vector computers)
- **MIMD: Multiple Instruction, Multiple Data**
  - message passing machines (Transputers, nCube, CM-5)
  - non-cache-coherent SMP’s (BBN Butterfly, T3D)
  - cache-coherent SMP’s (Sequent, Sun Starfire, SGI Origin)
- **MISD: Multiple Instruction, Single Data**
  - no commercial examples

MISD, check that a number is prime, do a divide on the same number with Processor number I.

Potential of the four classes



+
A
B
SISD
A + B

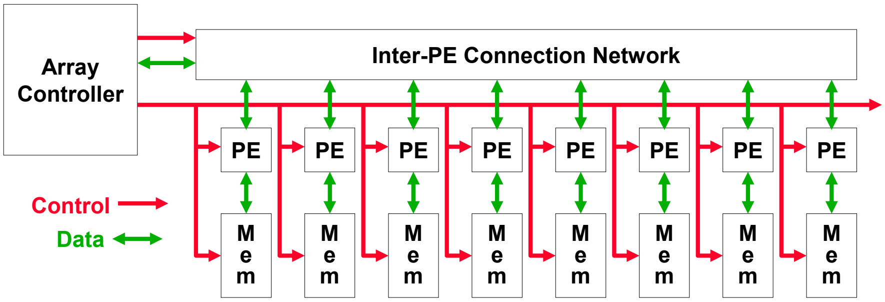
+, *
A
B
MISD
A + B
A * B

+
A, B
C, D
SIMD
A + B
C + D

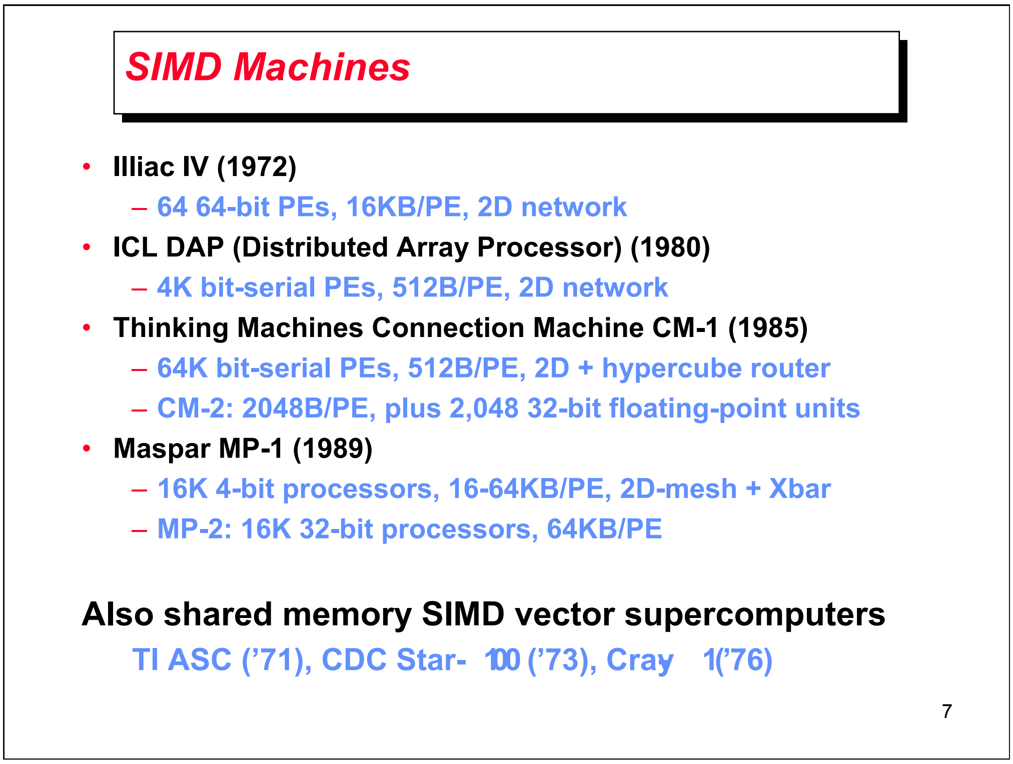
+, *
A, B
C, D
MIMD
A + B
C * D

SIMD Architecture

- Central controller broadcasts instructions to multiple processing elements (PEs)



- Only requires one controller for whole array
- Only requires storage for one copy of program
- All computations fully synchronized



SIMD Machines

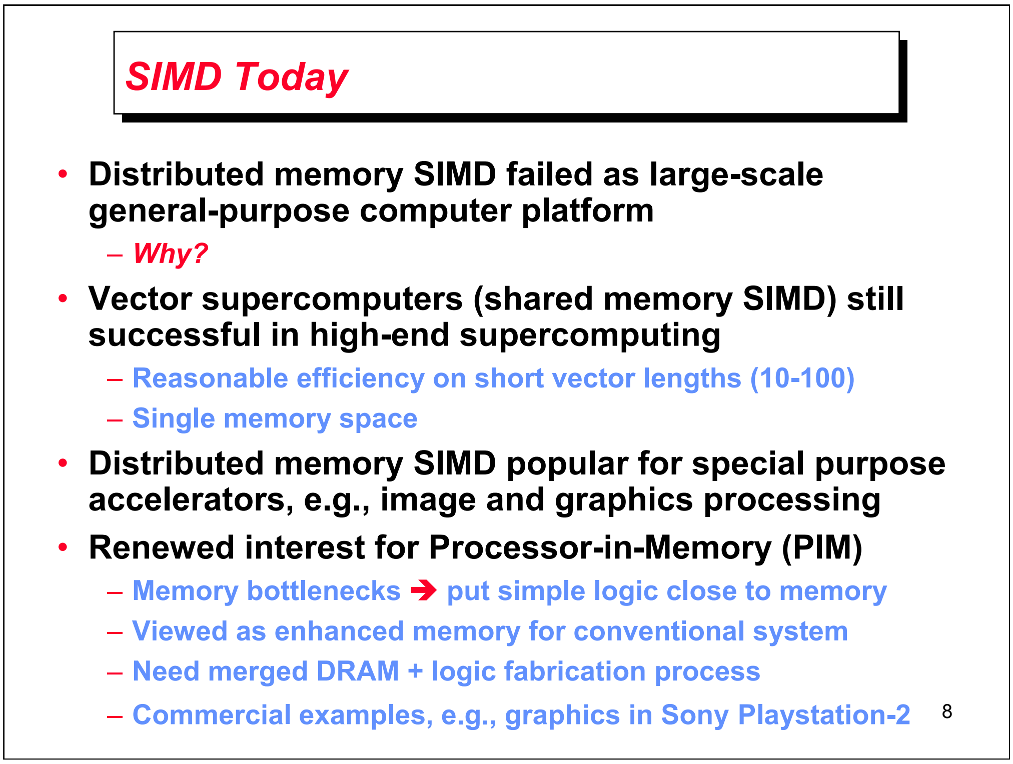
- Illiac IV (1972)
  - 64 64-bit PEs, 16KB/PE, 2D network
- ICL DAP (Distributed Array Processor) (1980)
  - 4K bit-serial PEs, 512B/PE, 2D network
- Thinking Machines Connection Machine CM-1 (1985)
  - 64K bit-serial PEs, 512B/PE, 2D + hypercube router
  - CM-2: 2048B/PE, plus 2,048 32-bit floating-point units
- Maspar MP-1 (1989)
  - 16K 4-bit processors, 16-64KB/PE, 2D-mesh + Xbar
  - MP-2: 16K 32-bit processors, 64KB/PE

Also shared memory SIMD vector supercomputers

TI ASC (’71), CDC Star-100 (’73), Cray-1(’76)

Illiac IV: $31M instead of $8M paid by DARPA, Illiaction

Glypnir language: Norse mythological rope that controls a vicious animal

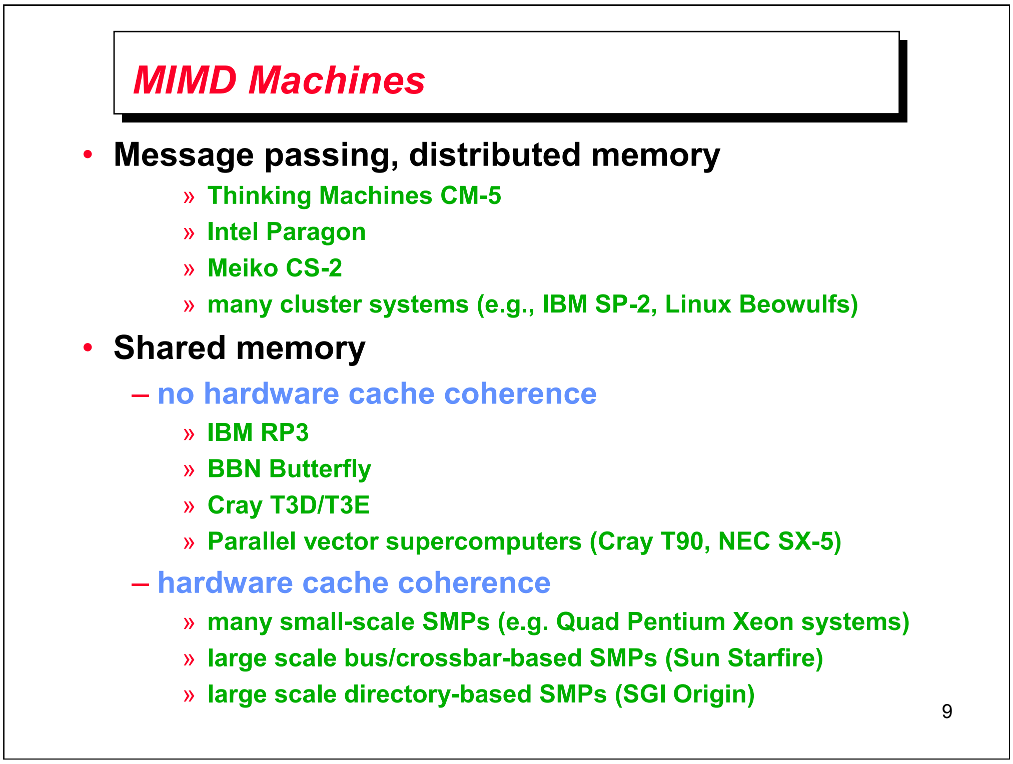


SIMD Today

- Distributed memory SIMD failed as large-scale general-purpose computer platform
  - *Why?*
- Vector supercomputers (shared memory SIMD) still successful in high-end supercomputing
  - Reasonable efficiency on short vector lengths (10-100)
  - Single memory space
- Distributed memory SIMD popular for special purpose accelerators, e.g., image and graphics processing
- Renewed interest for Processor-in-Memory (PIM)
  - Memory bottlenecks ➔ put simple logic close to memory
  - Viewed as enhanced memory for conventional system
  - Need merged DRAM + logic fabrication process
  - Commercial examples, e.g., graphics in Sony Playstation-2

Required huge quantities of data parallelism (> 10000 elements)

Required programmer-controlled distributed data layout.



MIMD Machines

- Message passing, distributed memory
    - » Thinking Machines CM-5
    - » Intel Paragon
    - » Meiko CS-2
    - » many cluster systems (e.g., IBM SP-2, Linux Beowulfs)
- Shared memory
    - – no hardware cache coherence
        - » IBM RP3
        - » BBN Butterfly
        - » Cray T3D/T3E
        - » Parallel vector supercomputers (Cray T90, NEC SX-5)
    - – hardware cache coherence
        - » many small-scale SMPs (e.g. Quad Pentium Xeon systems)
        - » large scale bus/crossbar-based SMPs (Sun Starfire)
        - » large scale directory-based SMPs (SGI Origin)

Thinking machines, interconnect network a FAT-TREE.

Rp3: research parallel processor prototype

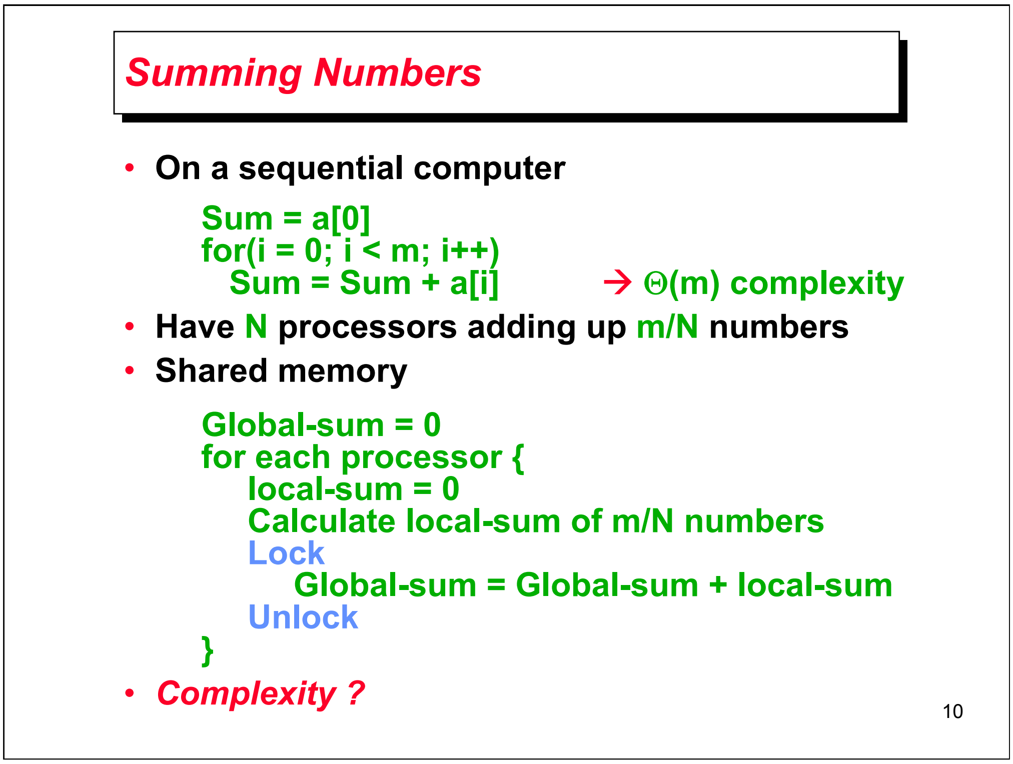
BBN”: Bolt Beranek and Newman, A spiffy interconnect will not by itself

Allow the construction of a supercomputer out of common household appliances.

Problem was the commodity processors did not support multiple outstanding

Memory requests. So the remote memory access killed off performance, unless

Excellent data layout was achieved. So it lost to message-passing networks.



Summing Numbers

- On a sequential computer

```
Sum = a[0]
for(i = 0; i < m; i++)
   Sum = Sum + a[i]          → Θ(m) complexity
```

- Have N processors adding up m/N numbers
- Shared memory

```
Global-sum = 0
for each processor {
     local-sum = 0
     Calculate local-sum of m/N numbers
     Lock
          Global-sum = Global-sum + local-sum
     Unlock
}
```

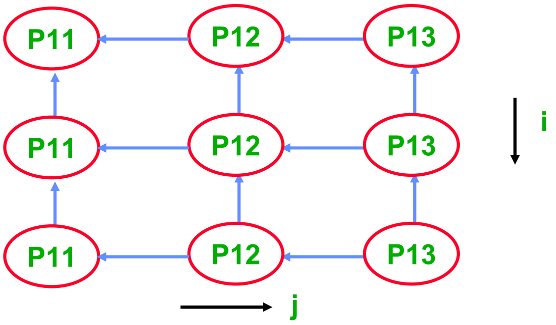
- ***Complexity ?***

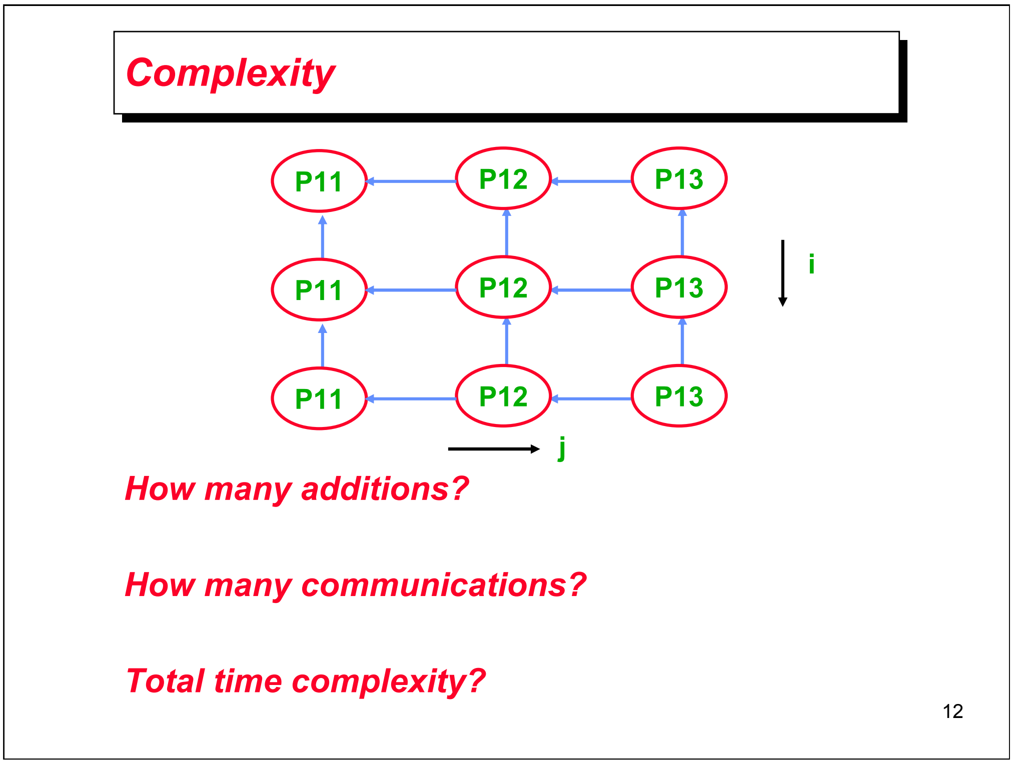
Theta(m/N) + Theta(N)

Theta(N) comes from summing N numbers in series.

Summing Numbers – Distributed Memory

- Assume a square mesh of processors
  - Each processor computes the local sum of its m/N numbers
  - Each processor passes its local sum to another processor in a coordinated way
  - The global sum in finally in processor P11





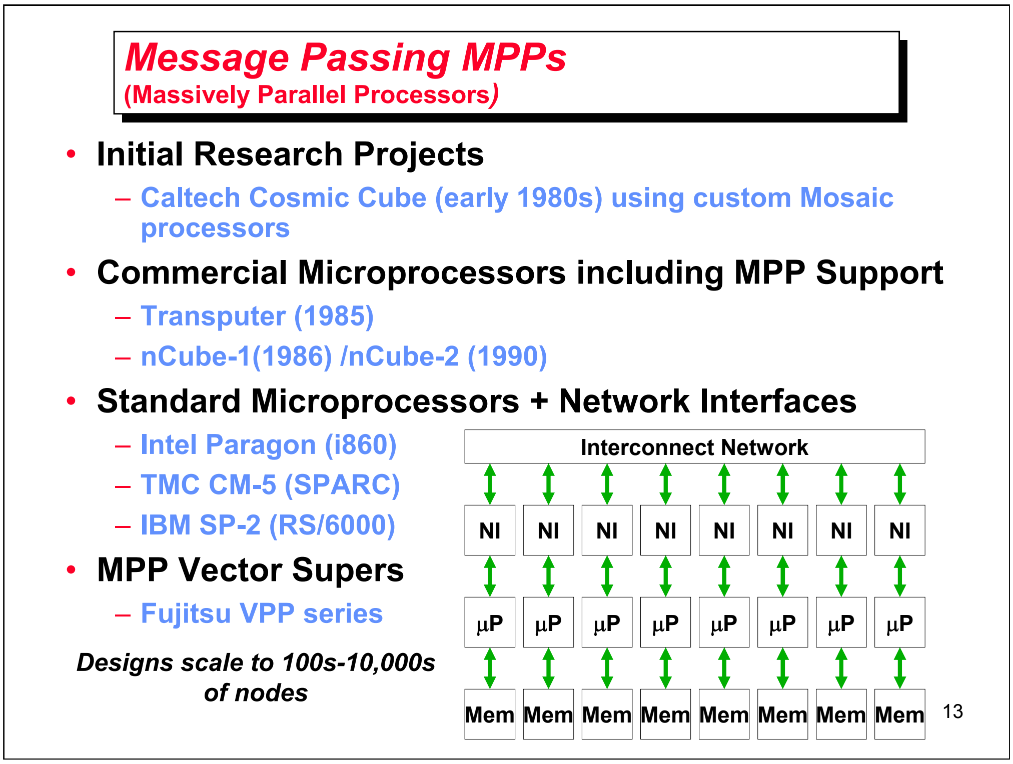
Complexity

How many additions?

How many communications?

Total time complexity?

Sqrt(N) – 1 additions, sqrt(N) – 1 communications in each direction
Theta(m/N) + Theta(sqrt(N)) + C, C is cost of communication

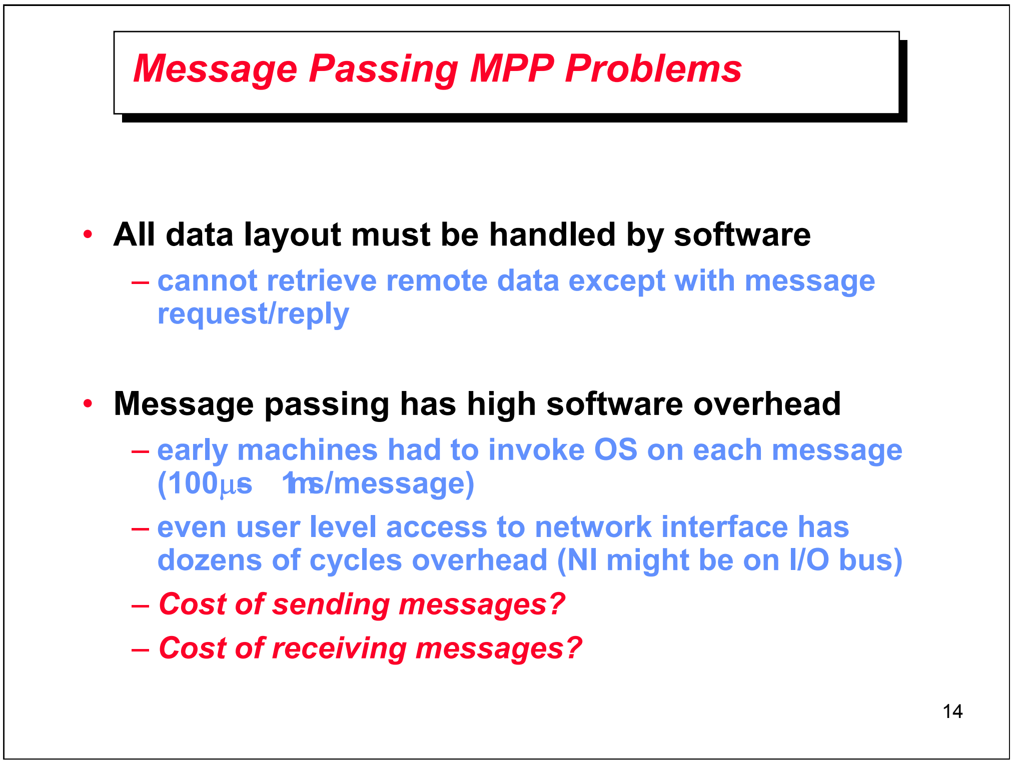


Message Passing MPPs
(Massively Parallel Processors)

- Initial Research Projects
  - Caltech Cosmic Cube (early 1980s) using custom Mosaic processors
- Commercial Microprocessors including MPP Support
  - Transputer (1985)
  - nCube-1(1986) /nCube-2 (1990)
- Standard Microprocessors + Network Interfaces
  - Intel Paragon (i860)
  - TMC CM-5 (SPARC)
  - IBM SP-2 (RS/6000)
- MPP Vector Supers
  - Fujitsu VPP series

Designs scale to 100s-10,000s of nodes

Read up on Cosmic Cube, transputer, CM5 other machines.



Message Passing MPP Problems

- All data layout must be handled by software
  - cannot retrieve remote data except with message request/reply
- Message passing has high software overhead
  - early machines had to invoke OS on each message (100µs 1ms/message)
  - even user level access to network interface has dozens of cycles overhead (NI might be on I/O bus)
  - *Cost of sending messages?*
  - *Cost of receiving messages?*

14

Sending can be cheap (like stores)

Receiving is expensive, need to poll or interrupt.

Shared Memory Machines

- Two main categories with respect to caches
  - non cache coherent
  - hardware cache coherent
- Will work with any data placement (but might be slow)
  - can choose to optimize only critical portions of code
- Load and store instructions used to communicate data between processes
  - no OS involvement
  - low software overhead
- Usually some special synchronization primitives
  - fetch&op
  - load linked/store conditional
- In large scale systems, logically shared memory is implemented as physically distributed modules

Shared Memory Machines

- Exclusive Read, Exclusive Write (EREW)
- Concurrent Read, Exclusive Write (CREW)
- Exclusive Read, Concurrent Write (ERCW)
- Concurrent Read, Concurrent Write (CRCW)

- Need to deterministically specify the contents of a memory location after concurrent write
  - Assign priorities to processors and store value from processor with highest priority
  - Allow concurrent writes if the values being written are the same
  - Max, min, average, or sum of values is stored (numeric applications)

Searching

- N processors search a list $S = \{L_1, L_2, \ldots, L_m\}$ for the index of an item x
  - Assume x can appear many times and any index will do

- Step 1:
```
for i = 1 to N in parallel
    read x
```

- Step 2: Sequentially search through sub-list

  S_i with m/N numbers

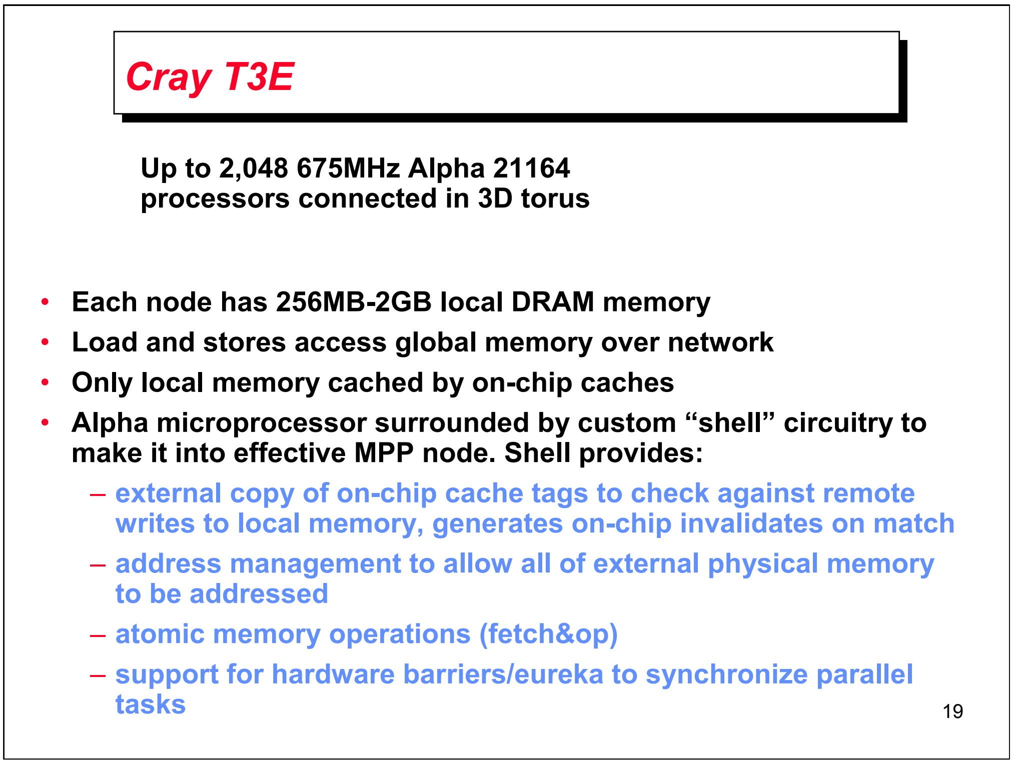
  Return $K_i = -1$ if x not in S_i, else index

- Step 3:
```
for i = 1 to N in parallel
    if  ($K_i > 0$)    k = $K_i$
```

Complexity

- What is the complexity for the different shared memory schemes?

	Step 1	Step 2	Step 3	Total
• EREW:	O(N)	O(m/N)	O(N)	O(N) + O(m/N)
• CREW:				
• ERCW:				
• CRCW:				



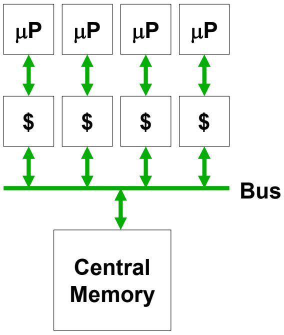
Cray T3E

Up to 2,048 675MHz Alpha 21164
processors connected in 3D torus

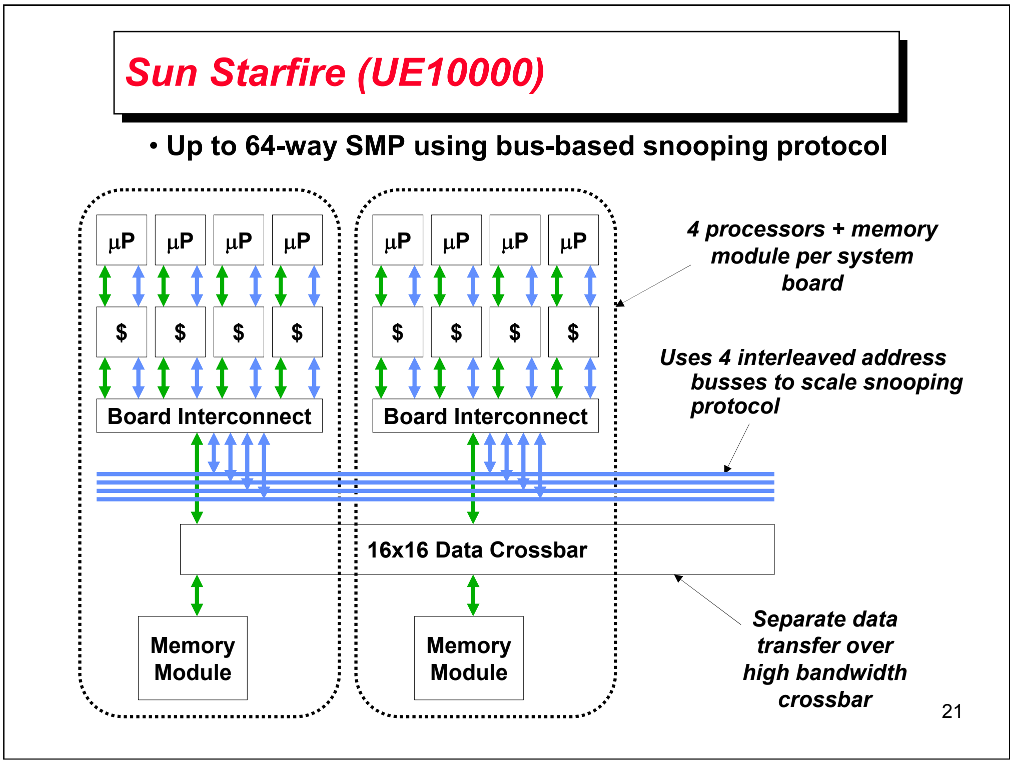
- Each node has 256MB-2GB local DRAM memory
- Load and stores access global memory over network
- Only local memory cached by on-chip caches
- Alpha microprocessor surrounded by custom “shell” circuitry to make it into effective MPP node. Shell provides:
  - external copy of on-chip cache tags to check against remote writes to local memory, generates on-chip invalidates on match
  - address management to allow all of external physical memory to be addressed
  - atomic memory operations (fetch&op)
  - support for hardware barriers/eureka to synchronize parallel tasks

Read up on no hardware coherence for the Cray T3E, eureka?

Bus-Based Cache-Coherent SMPs



- Small scale (<= 4 processors) bus-based SMPs by far the most common parallel processing platform today
- Bus provides broadcast and serialization point for simple snooping cache coherence protocol
- Modern microprocessors integrate support for this protocol



Sun Starfire (UE10000)

- Up to 64-way SMP using bus-based snooping protocol

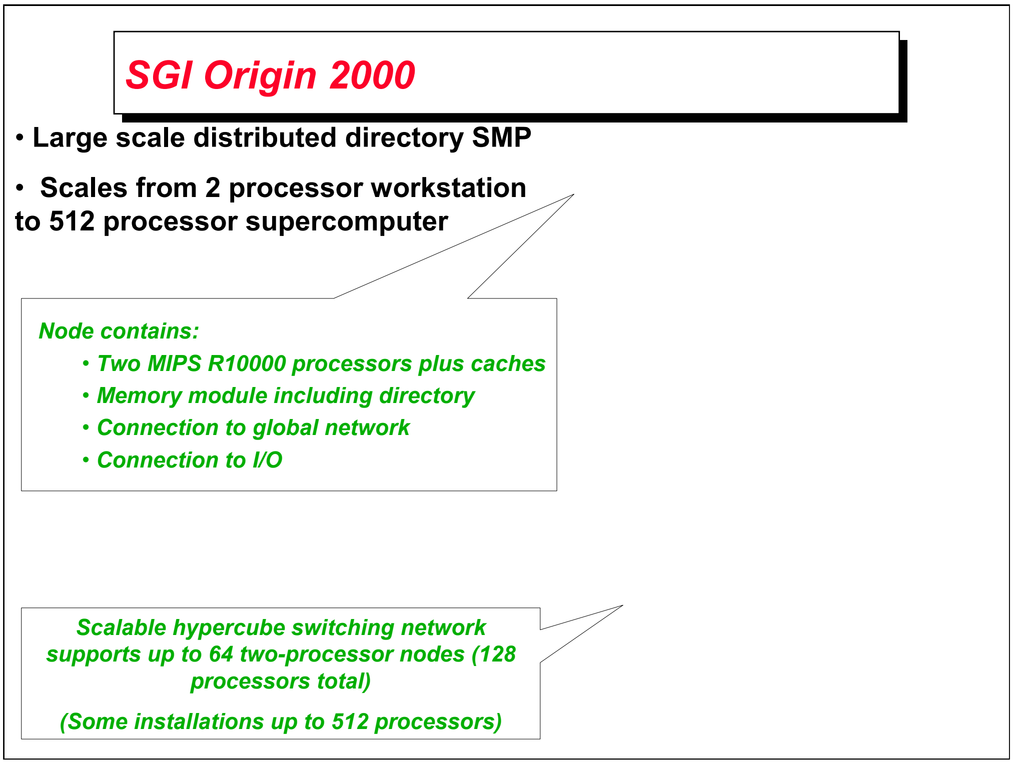
4 processors + memory module per system board

Uses 4 interleaved address busses to scale snooping protocol

Separate data transfer over high bandwidth crossbar

Interleaved multiple address busses. Hardest part of scaling up an SMP system is its coherence bandwidth.

Read Starmicro paper.



SGI Origin 2000

- Large scale distributed directory SMP
- Scales from 2 processor workstation to 512 processor supercomputer

Node contains:

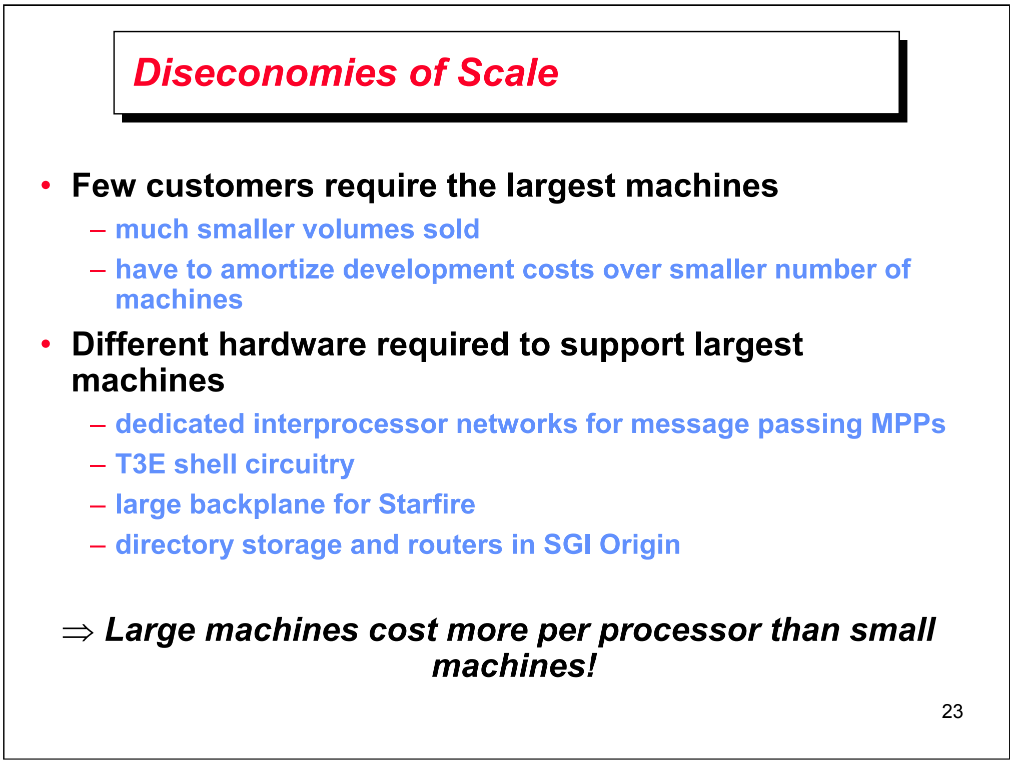
- *Two MIPS R10000 processors plus caches*
- *Memory module including directory*
- *Connection to global network*
- *Connection to I/O*

Scalable hypercube switching network supports up to 64 two-processor nodes (128 processors total)

(Some installations up to 512 processors)

Read up on directories, and compare to snoopy protocols.

SGI origin paper.



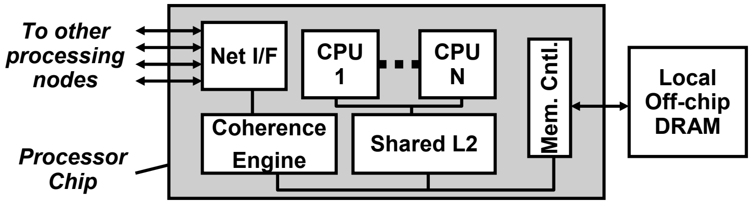
Diseconomies of Scale

- Few customers require the largest machines
  - much smaller volumes sold
  - have to amortize development costs over smaller number of machines
- Different hardware required to support largest machines
  - dedicated interprocessor networks for message passing MPPs
  - T3E shell circuitry
  - large backplane for Starfire
  - directory storage and routers in SGI Origin

⇒ ***Large machines cost more per processor than small machines!***

Trends in High-End Server CPUs

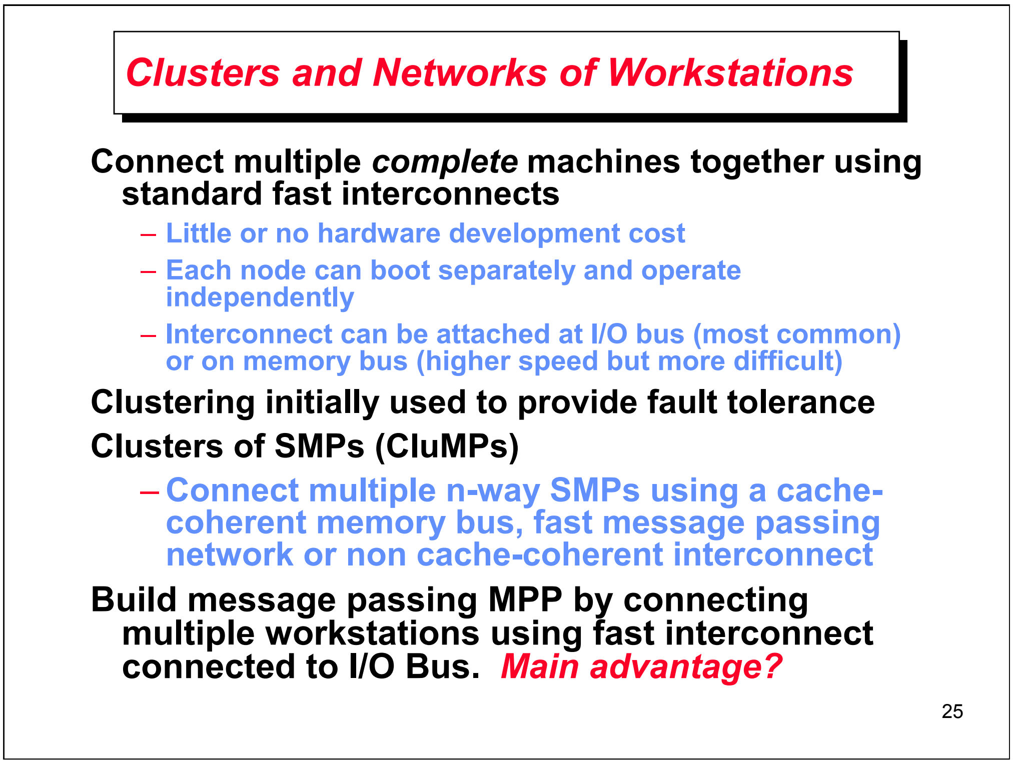
- DRAM controllers on chip (UltraSPARC-III, Alpha 21364)
  - – reduce main memory latency
  - – increase memory bandwidth
- On-chip network routers (Alpha 21364, Power-4)
  - – Cheaper/faster connectivity for cache coherence traffic
- Multiple processors on one chip
  - – Chip multiprocessor systems (IBM Power-4)
  - – Simultaneous multithreading (Pentium-4 Xeon)



24

INTEGRATION in VLSI. DRAM controllers on chip to reduce main memory latency (250ns to 170 ns in Ultrasparc III) and

Increase memory bandwidth – 12 GB/s over 8 Rambus channels in 21364.



Clusters and Networks of Workstations

- Connect multiple *complete* machines together using standard fast interconnects
  - Little or no hardware development cost
  - Each node can boot separately and operate independently
  - Interconnect can be attached at I/O bus (most common) or on memory bus (higher speed but more difficult)
- Clustering initially used to provide fault tolerance
- Clusters of SMPs (CluMPs)
  - Connect multiple n-way SMPs using a cache-coherent memory bus, fast message passing network or non cache-coherent interconnect
- Build message passing MPP by connecting multiple workstations using fast interconnect connected to I/O Bus. *Main advantage?*

25

HP Exemplar cache coherent, SGI Power Challenge Array, Vector supercomputers NEC SX-5

Ucberkeley NOW, 100 Sun workstations connected by a Myrinet network. What is Myrinet?

Sold commercially by IBM (SP-2), and Cray selling networks of Alpha’s.

The Future?