## Parallel Programming Platforms

Ananth Grama
Computing Research Institute and
Department of Computer Sciences,
Purdue University.

ayg@cs.purdue.edu
http://www.cs.purdue.edu/people/ayg

Reference:

Introduction to Parallel Computing, Ananth Grama, Anshul Gupta, Vipin Kumar, George Karypis, Addison Wesley, ISBN: 0-201-64865-2, 2003.



## Motivating Parallelism

- The Computational Speed Argument: For some applications, this is the only means of achieving needed performance.
- The Memory/Disk Speed Argument: For some other applications, the needed I/O throughput can be provided only by a collection of nodes.
- The Data Communication Argument: In yet other applications, the distributed nature of data implies that it is unreasonable to collect data to process it at a single location.



## Implicit Parallelism: Trends in Microprocessor Architectures

All microprocessors currently available rely on parallelism to varying degrees. These are generally hidden from the programmer.

- Pipelining and Superscalar Execution
- Very Long Instruction Word Processors

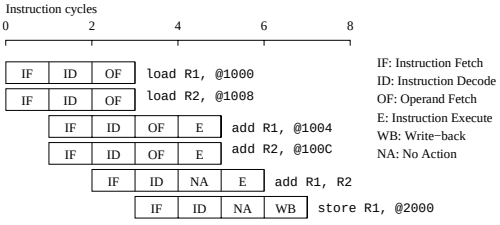


## Superscalar Execution

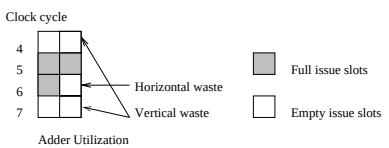
```
1. load R1, @1000     1. load R1, @1000     1. load R1, @1000
2. load R2, @1008     2. add R1, @1004      2. add R1, @1004
3. add R1, @1004      3. add R1, @1008      3. load R2, @1008
4. add R2, @100C      4. add R1, @100C      4. add R2, @100C
5. add R1, R2         5. store R1, @2000    5. add R1, R2
6. store R1, @2000                          6. store R1, @2000
```

(i) (ii) (iii)

(a) Three different code fragments for adding a list of four numbers.




(b) Execution schedule for code fragment (i) above.




(c) Hardware utilization trace for schedule in (b).

## Limitations of Memory System Performance

Often, the primary bottleneck to performance is not the CPU, rather, it is the memory system. Typical computations only run at 10-50% of peak CPU utilization because of memory bottlenecks. The key question here is how to connect a 50 ns latency memory to a processor that runs a 0.5 ns clock!

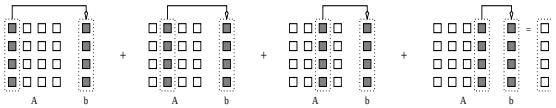
- Improving Effective Memory Latency Using Caches: A hierarchy of small, fast stores bridge the gap between processor and memory. These stores rely on repeated accesses to data to deliver higher aggregate performance.
- Improving Memory System Performance by Threading: Find something else to do while you are waiting for data to arrive from memory.



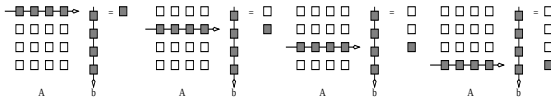
## Impact of Strided Access on Program Performance

```
// Fragment 1: Summing columns of a matrix.
for (i = 0; i < 1000; i++)
        column_sum[i] = 0.0;
        for (j = 0; j < 1000; j++)
                column_sum[i] += b[j][i];

// Fragment 2: Fragment 1, rewritten.
for (i = 0; i < 1000; i++)
        column_sum[i] = 0.0;
for (j = 0; j < 1000; j++)
        for (i = 0; i < 1000; i++)
                column_sum[i] += b[j][i];
```




(a) Column major data access




(b) Row major data access.



## Dichotomy of Parallel Computing Platforms

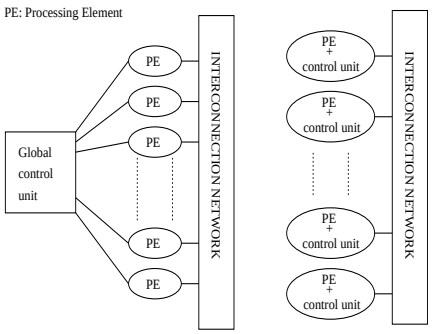
Control Structure of Parallel Platforms: What is the nature of concurrent tasks?

Communication Model of Parallel Platforms: How do multiple tasks cooperate with each other?

- Message Passing Platforms
- Shared-Address-Space Platforms



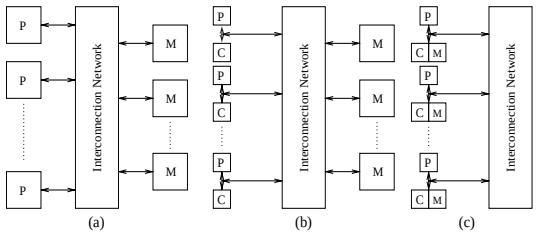
## Control Structure of Parallel Machines:




(a) (b)

In a Single Instruction Multiple Data (SIMD) paradigm (a), all processing elements execute the same instruction. In a Multiple Instruction Multiple Data paradigm (b), all processing elements execute possibly different instructions, independently. An intermediate paradigm, called Single Program Multiple Data (SPMD) is the most popular programming paradigm.

## Communication Model of Parallel Platforms

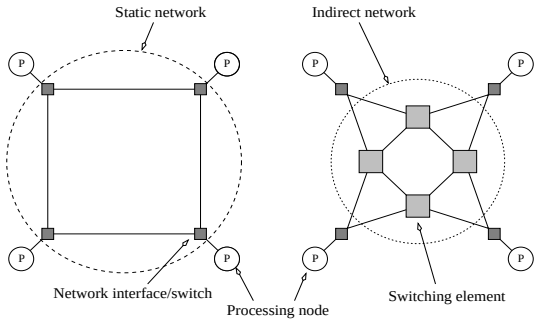



In a uniform memory access (UMA) shared memory model (a), all processors access memory through an interconnect. In (b), we show a UMA shared memory machine with caches, and in (c), we show a non-uniform memory access (NUMA) shared memory machine with private memories only. The logical name for all of these platforms is a shared address space machine.



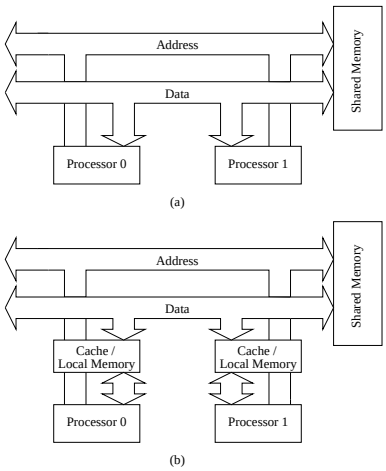
## Physical Organization of Parallel Platforms

The characterizing feature of parallel platforms is the underlying interconnection network. These networks can be static (a) or dynamic (b).





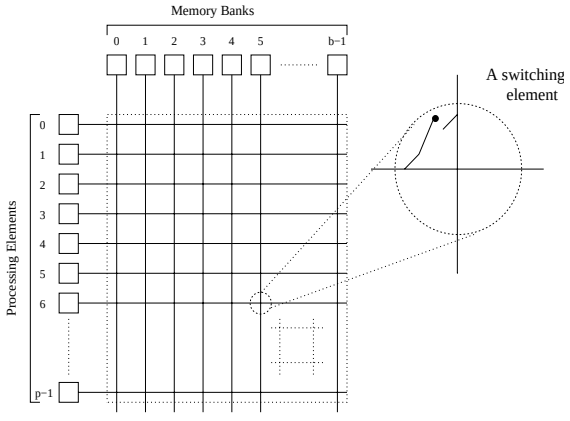

## Direct Interconnection Networks:




Bus-based interconnects (without (a) and with caches (b)) were the first networks in early commercially available platforms (Sequent Symmetry/Balance).

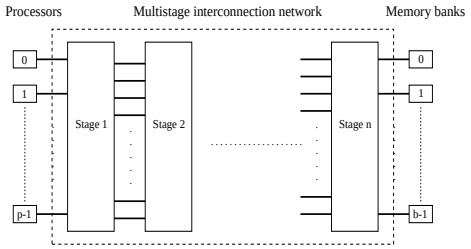


## Direct Interconnection Networks:




The other extreme in terms of performance and cost compared to buses, is the crossbar network.

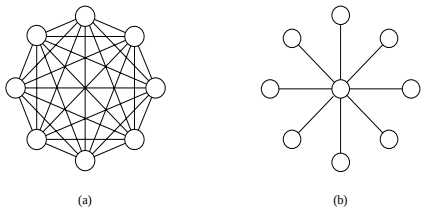
## Direct Interconnection Networks



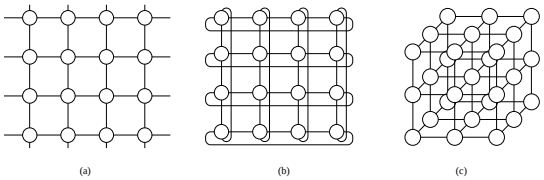

Multistage networks such as the Omega network fall between buses and crossbars in terms of cost and performance.



## Static Interconnection Networks:




A completely connected network (a) is the strongest model for a static network. A star connected network (b) is the other extreme.

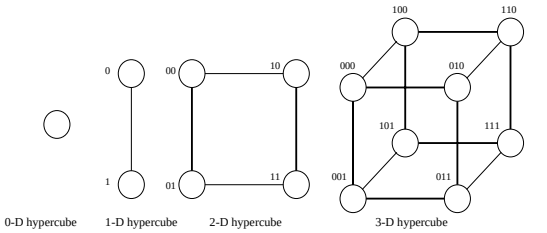


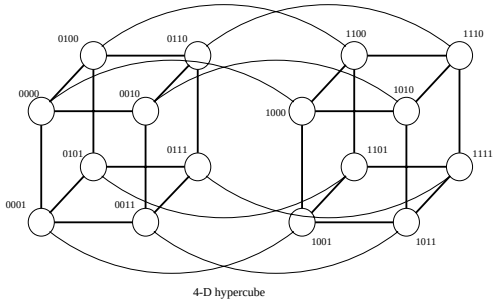

Meshes (2 and 3-D) are popular interconnects because of their desirable layout properties and performance for physical simulations.



## Static Interconnection Networks:

Hypercubes provide another popular interconnect.



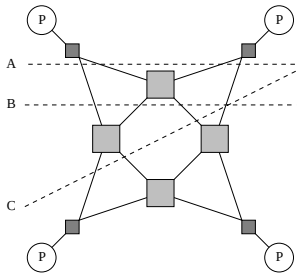






## Metrics for Interconnection Networks

Performance Metrics

- Link Bandwidth: how fat is each link?
- Arc Connectivity: how many links do I have to remove to separate a network.
- Bisection Bandwidth: how many pairs of people can have conversations at any given time, independently.

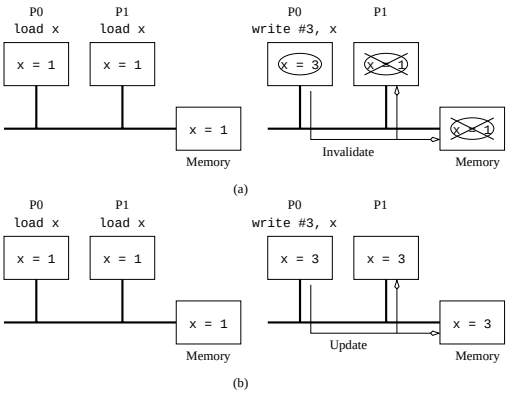



Cost Metrics

- Number of Links
- Layout Costs.

## Cache Coherence in Multiprocessor Systems:

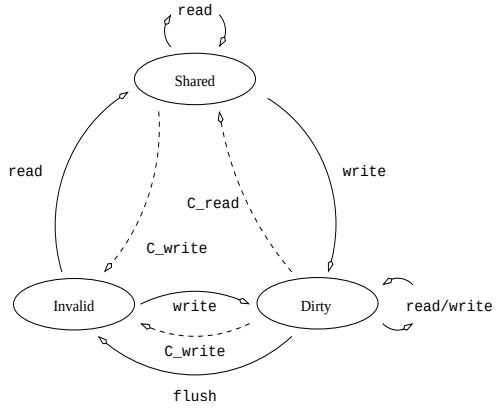
How do you deal with multiple copies of the same data item, being manipulated by different processors?




We can invalidate copies (a), or update them (b) when a processor changes a copy.



## Coherence Protocols:




A simple three-state protocol for implementing cache-coherence.



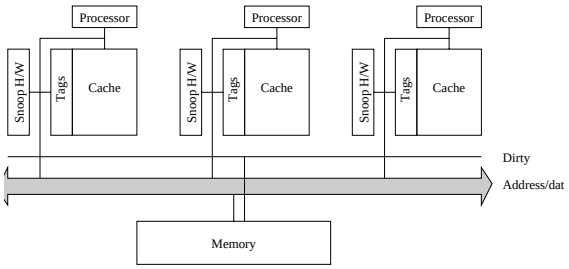
## Coherence Protocols:

An example of the coherence protocol.

| Time | Instruction at Processor 0 | Instruction at Processor 1 | Variables and their states at Processor 0 | Variables and their states at Processor 1 | Variables and their states in Global mem. |
|---|---|---|---|---|---|
| | | | | | x = 5, D<br>y = 12, D |
| | read x | | x = 5, S | | x = 5, S<br>y = 12, S |
| | | read y | | y = 12, S | |
| | x = x + 1 | | x = 6, D | | x = 5, I<br>y = 12, I |
| | | y = y + 1 | | y = 13, D | |
| | read y | | y = 13, S | y = 13, S | y = 13, S |
| | | read x | x = 6, S | x = 6, S | x = 6, S |
| | x = x + y | | x = 19, D | x = 6, I | x = 6, I |
| | | y = x + y | y = 13, I | y = 19, D | y = 13, I |
| | x = x + 1 | | x = 20, D | | x = 6, I |
| | | y = y + 1 | | y = 20, D | y = 13, I |



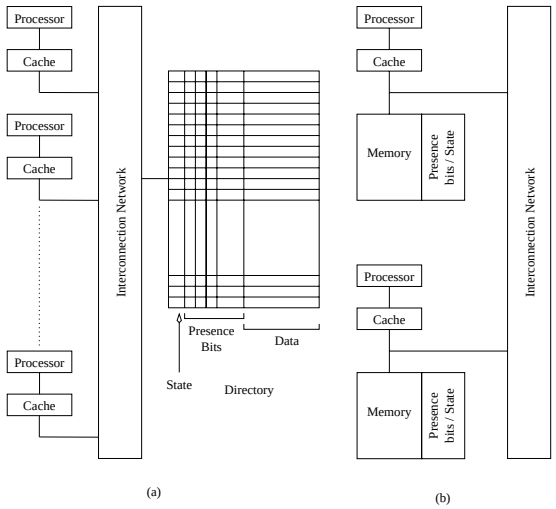
## Implementing Coherence Protocols:




Using a snoopy bus to implement the coherence protocol. Snoopy buses do not scale to large configurations.

**Implementing Coherence Protocols:**

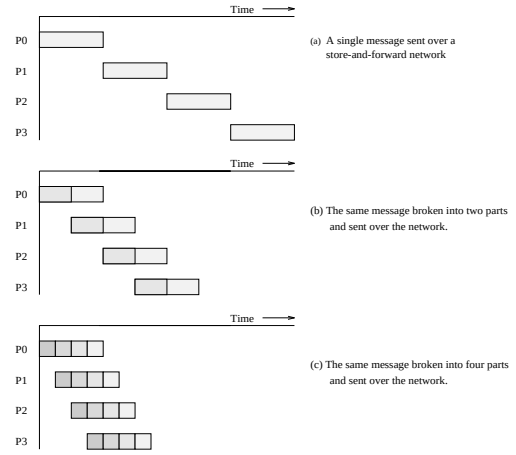
Directories provide a more scalable solution than snoopy buses.






**Routing Messages in Parallel Platforms:**

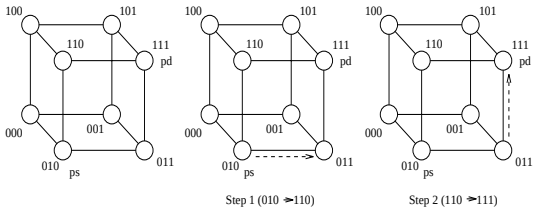
- Store-and-forward routing
- Packet Routing
- Cut-Through routing






**Routing in Parallel Platforms:**

- Dimension ordered / E-cube routing




- Hot-potato routing

- Randomized Routing



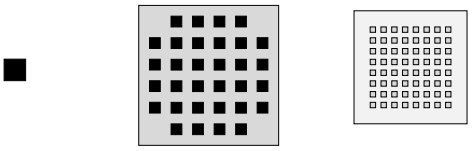
**Embeddings and Overhead:**

It is often possible to map a weaker architecture on to a stronger one with no performance overhead. Conversely, mapping a stronger architecture on to a weaker one results in performance penalties, depending on the program.

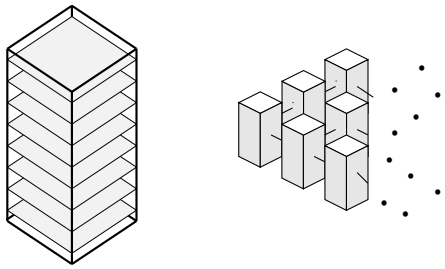
Algorithms for mapping between structured networks are well studied (see handout).

## Case Studies:

The IBM Blue Gene.



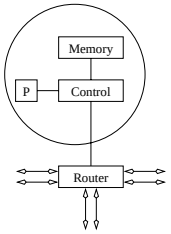
(a) CPU (1GF) (b) Chip (32 GF) (c) Board (2 TF)



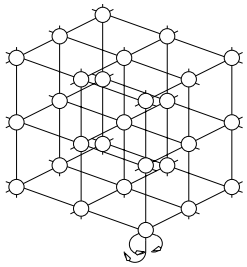
(d) Tower (16 TF) (e) Blue Gene (1 PF)



## The Cray T3E.



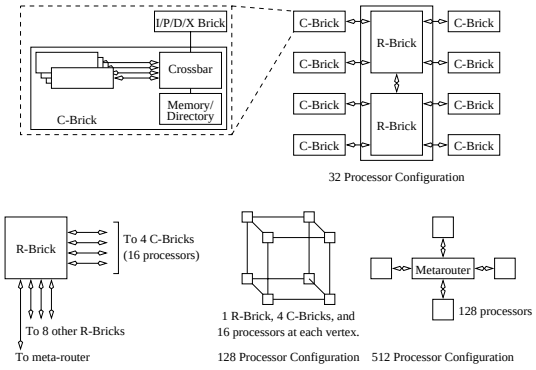

(a)



(b)

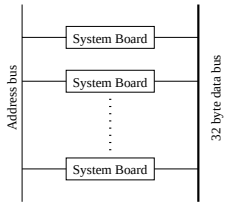


## SGI Origin 3000

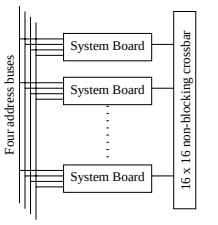





## Sun Enterprise Servers



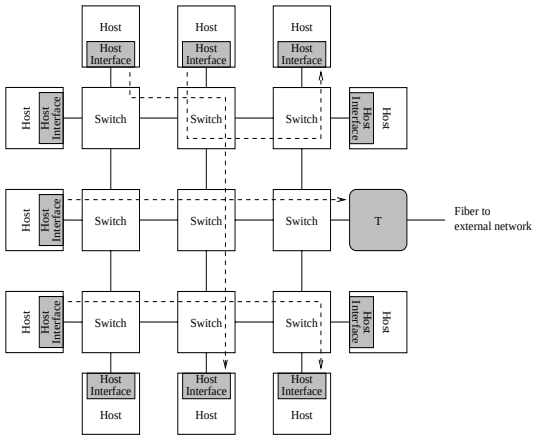

Sun Ultra 6000 (6 - 30 processors)




Starfire Ultra 1000 (up to 64 processors)

**Commonly used networks:**

Myrinet:




Other networks include Gigabit Ethernet, FiberChannel, HiPPI, etc.