Multiprocessors

Flynn’s Classification of multiple-processor machines:

\{SI, MI\} \times \{SD, MD\} = \{SISD, SIMD, MISD, MIMD\}

**SISD** = Single Instruction Single Data
Classical Von Neumann machines.

**SIMD** = Single Instruction Multiple Data
Also called Array Processors or Data Parallel machines.

**MISD** = Does not exist.

**MIMD** = Multiple Instruction Multiple Data
Control parallelism.
Grains of Parallelism

Example 1. Fine grained Parallelism (SIMD)

Example 2. Coarse-grained Parallelism (MIMD)

Also possible: medium grained parallelism
Synchronize after a few instructions
A Taxonomy of Parallel Computers

**Parallel Architecture**

- **SISD**
- **SIMD**
- **MISD**
- **MIMD**

**Multiprocessors**

- **UMA** (Uniform Memory Access)
- **COMA** (Cache Only Memory Access)
- **NUMA** (Non Uniform Memory Access)
- **CC-NUMA**
- **NC-NUMA**

**Multicomputers**

- **MPP** (Massively Parallel Processor)
- **COW** (Cluster Of Workstations)
- **Grid**
- **Cube**

**Shared memory**

**Message Passing**

- **UMA** Uniform Memory Access
- **NUMA** Non Uniform Memory Access
- **COMA** Cache Only Memory Access
- **MPP** Massively Parallel Processor
- **COW** Cluster Of Workstations
- **CC-NUMA** Cache Coherent NUMA
- **NC-NUMA** No Cache NUMA
Vector vs. Array Processing

Vector Processing

CRAY-1 defined the style of vector processing

Let \( n \) be the size of each vector.

Then, the time to compute \( f(V1, V2) = k + (n-1) \), where \( k \) is the length of the pipe \( f \).
Array Processing

In array processing, each of the operations \( f(v_{1j}, v_{2j}) \) with the components of the two vectors is carried out simultaneously, in one step.

In the early 60’s ILLIAC 4 defined the style of array processing. More recent efforts are Connection machine CM-2, CM-5, known as data-parallel machines. Now it is virtually dead, but SIMD concept is useful in MMX / SSE instructions.
**MIMD Machines (Multiprocessors & Multicomputers)**

**Multiprocessors**

Processors are connected with memory via the memory bus (which is fast), or switches.

*The address spaces of the processes overlap.*

*Interprocess communication via shared memory*
Multi-computers

Processes have private address space.

Interprocess communication via message passing only.

In clusters, computers are connected via LAN
Major Issues in MIMD

1. Processor–Memory Interconnection
2. Cache Coherence
3. Synchronization Support
4. Scalability Issue
5. Distributed Shared memory
What is a switch?

An Omega Network as a Switch
Blocking vs. Non-blocking Networks

Non-blocking Network
All possible 1-1 connections among PE’s feasible.

Blocking network
Some 1-1 mappings are not feasible.
Reminds us of telephone connections ....

Omega network is a blocking network. Why?
How many distinct mappings are possible?
Cross-bar switch

An \((N \times N)\) cross bar is a non-blocking network. Needs \(N^2\) switches. The cost is prohibitive.
UMA Bus-based Multiprocessors (SMP)

Central Shared Memory

Limited bus bandwidth is a bottleneck. So processors use private caches to store shared variables. This reduces the bus traffic. However, due to bus saturation, the architecture is not scalable beyond a certain limit (say 8 processors).
UMA Multiprocessors Using Cross-Bar Switch

Example: SUN Enterprise 10000

Uses a 16 x 16 cross-bar switch, called the Gigaplane-XB. Each P is a cluster of four 64-bit UltraSparc CPU’s. Each M is of size 4 GB. At this scale, there is no resource saturation problem, but the switch cost is $O(N^2)$. 
UMA Multiprocessors Using Multistage Interconnection Networks

Notice the self-routing property of the above network.

2x2 cross-bar
Eventually leads to an Omega or Cube like Network

![Diagram of a network with processors (P0 to P7) connected to memories (M0 to M7) via switches.]

This switch, or a variation of it, is used in connecting processors with memories.

**Self-routing network.** No central router is necessary. Switch complexity is \((N/2 \log_2 N)\). Scalable architecture. Blocking network leads to switch conflicts.
**NUMA Multiprocessors**

![Diagram of NUMA multiprocessors]

*Example of Cm* in CMU

Cm* is the oldest known example of NUMA architecture. Implemented a shared virtual address space. Processors have *faster access* to memory modules that are *closest*. Access to *remote* memory modules is *slower*.

Today’s large-scale NUMA machines use multi-stage interconnection networks (instead of bus) for better scalability.