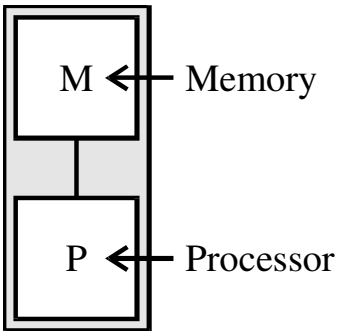
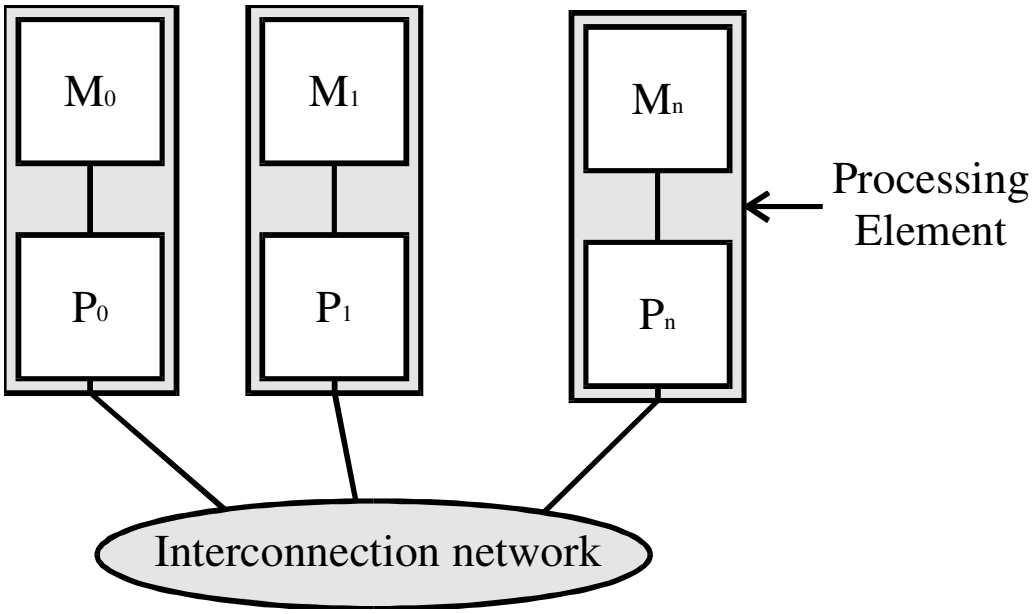
Introduction to MIMD Architectures

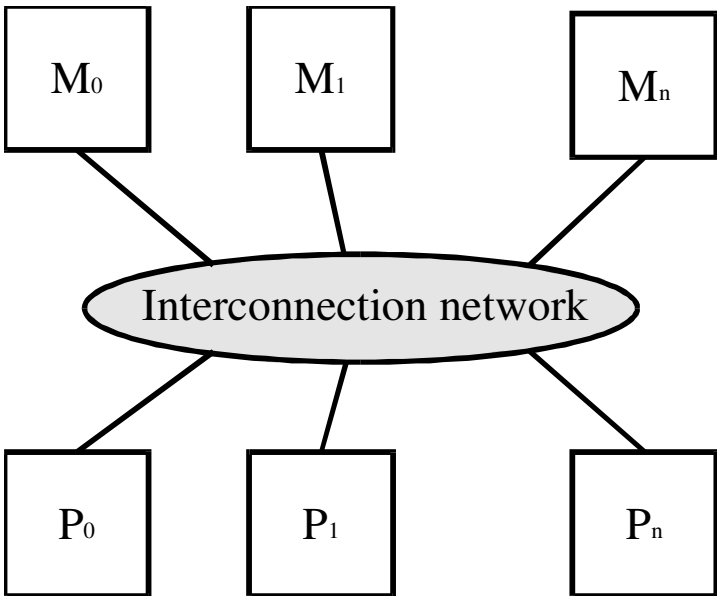


- MIMD architectures are a natural generalization of a von Neumann uniprocessor:

a) Distributed Memory MIMD Architectures



b) Shared Memory MIMD Architectures



Introduction to MIMD Architectures

Distributed Memory (Message Passing) MIMD systems

- Advantages

  1. Highly scalable (no memory contention)

  2. Message passing solves both communication and synchronisation

- Drawbacks

  1. Load balancing is a serious problem

  2. Message passing can lead to deadlocks

  3. Message passing requires the physical copy of data

Introduction to MIMD Architectures

Shared Memory MIMD sytems

- Advantages

  1. No need to partition data or program, uniprocessor programming techniques can be adapted

  2. No need to physically copy data between processes

- Drawbacks

  1. Synchronising constructs (semaphores, conditional critical regions, monitors) result in nondeterministc behaviour which can lead programming errors that are difficult to discover

  2. Lack of scalabilty due to memory contention

Introduction to MIMD Architectures

Solving the low scalability problem in Shared memory systems

1. High perfomance interconnection network

2. Use of cache memories

3. The logically shared memory is implemened as a physically distributed memory, *Virtual Shared Memory* or *Distributed Shared Memory*

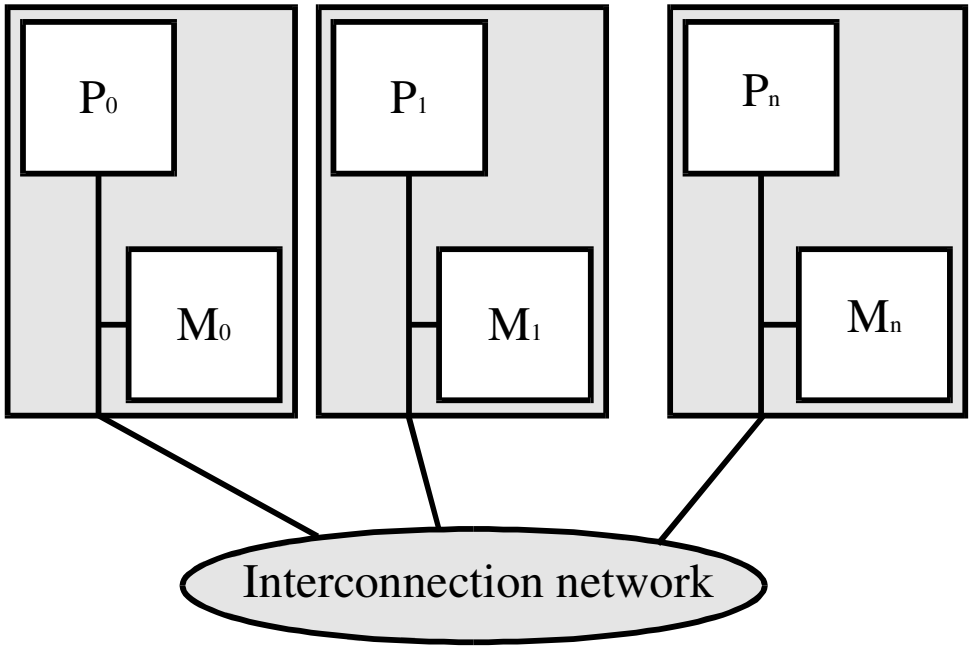
   a. Non-Uniform-Memory-Access (NUMA)

   b. Cache-Coherent Non-Uniform-Memory-Access (CC-NUMA)

   c. Cache-Only Memory Architecture (COMA)

Introduction to MIMD Architectures

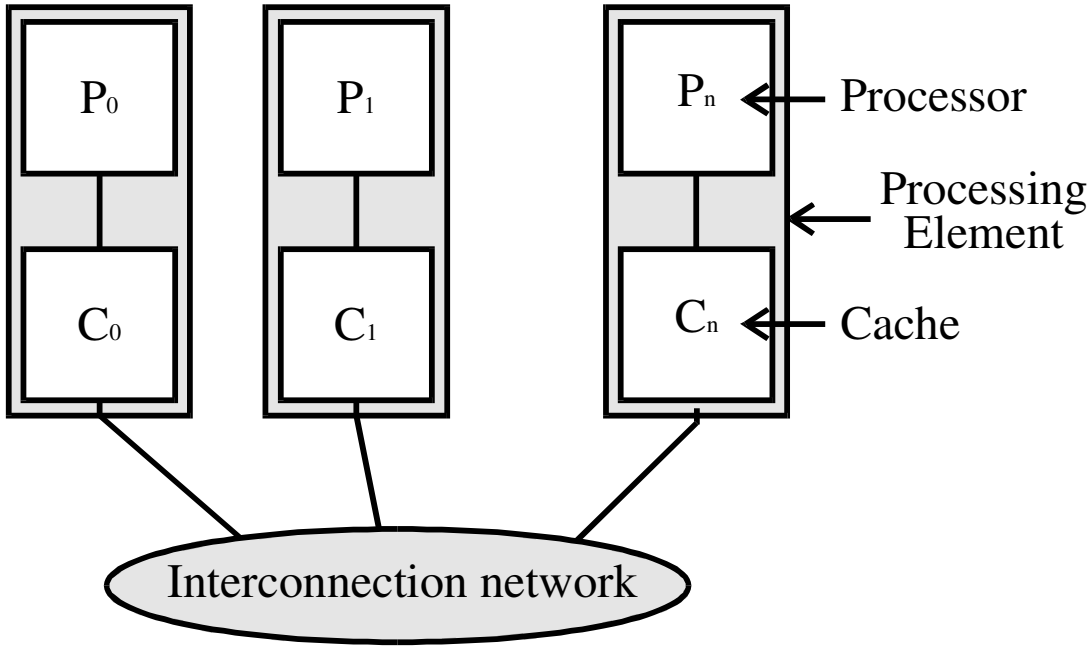
NUMA Architectures



- Non-uniform access: the acces to a local memory block is much faster
- Requires carefull data and program distribution
- Close to distributed memory systems, yet the programming paradigm is different
- Example: Cray T3D

Introduction to MIMD Architectures

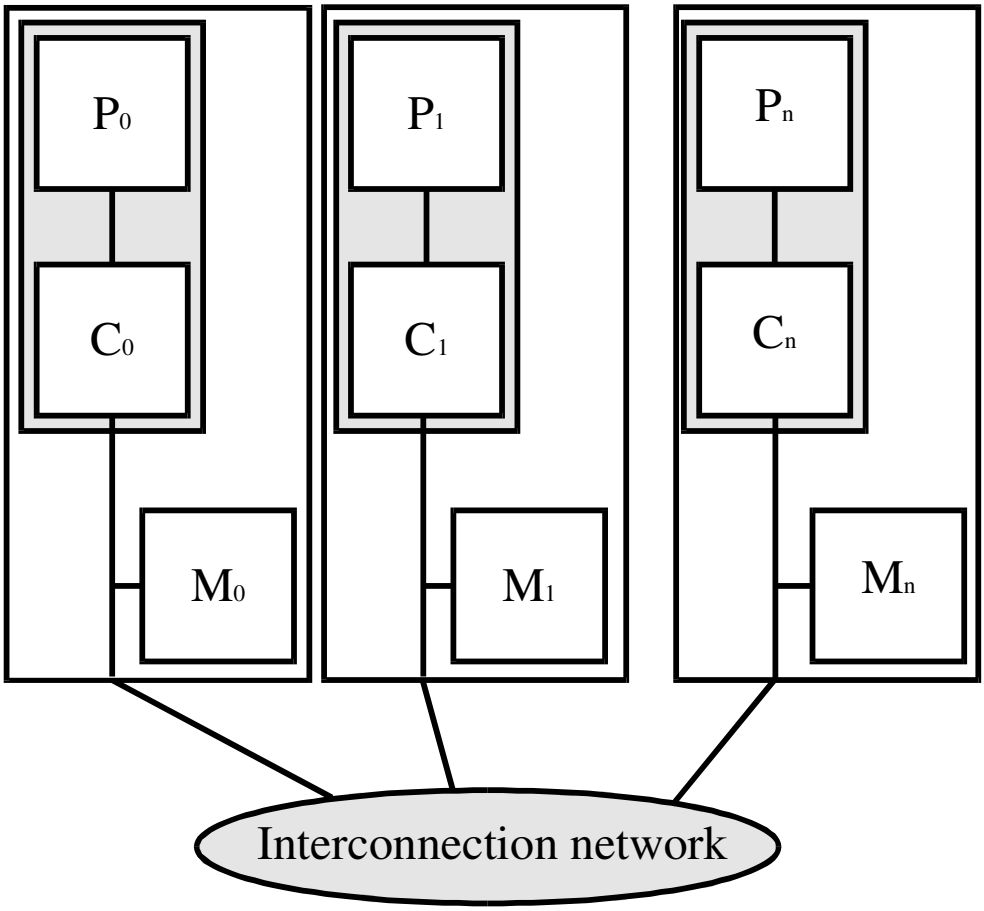
COMA Architectures



- Single address space and coherent caches
- Every memory block is a cache
- Continuous, dynamic migration of data
- Examples: KSR-1, DDM

Introduction to MIMD Architectures

CC-NUMA Architectures



- A compromise between NUMA and COMA architectures
- Initially static data distribution, then dynamic data migration
- Examples: Convex SPP1000, Stanford DASH, MIT Alewife

Introduction to MIMD Architectures

<table>
<tr><td colspan="8">MIMD computers</td></tr>
<tr><td colspan="5">Process-level architectures</td><td colspan="3">Thread-level architectures</td></tr>
<tr><td colspan="4">Single address space shared memory</td><td rowspan="2">Multiple address space distributed memory</td><td colspan="3" rowspan="2">Single address space shared memory</td></tr>
<tr><td colspan="4"></td></tr>
<tr><td>Physical shared memory</td><td colspan="3">Virtual (distributed) shared memory</td><td rowspan="2"></td><td>Physical shared memory</td><td colspan="2">Virtual (distributed) shared memory</td></tr>
<tr><td>UMA</td><td>NUMA</td><td>CC-NUMA</td><td>COMA</td><td>UMA</td><td>NUMA</td><td>CC-NUMA</td></tr>
</table>

Introduction to MIMD Architectures

- Problems of scalable computers
    1. Tolerate and hide latency of remote loads
    2. Tolerate and hide idling due to synchronisation
- Solutions:
    1. Cache memory
        - problem of cache coherence
    2. Prefetching
    3. Threads and fast context switching