International Journal of Scientific Engineering and Applied Science (IJSEAS) - Volume-1, Issue-9, December 2015
ISSN: 2395-3470
www.ijseas.com

MIMD Parallel Processing

Neha K. Shinde
Student, Department of Electronic Engineering,
J D College of Engineering and Management,
RTM Nagpur University, Maharashtra, India
nshinde73@gmail.com

Prof. Sunil R. Gupta
HOD, Department of Electronics engineering,
J D College of Engineering and Management,
RTM Nagpur University, Maharashtra, India
sungt_in@yahoo.com

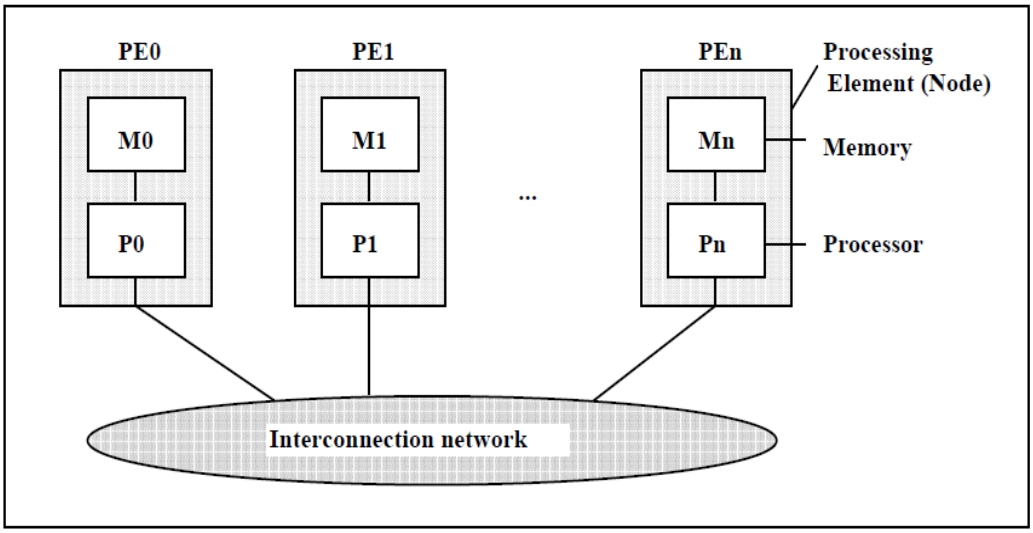
Abstract: Parallel architectures of Thread level and process-level are mostly realized by MIMD (Multiple Instruction Multiple Data) computers. This class of parallel computers is a general one because it allows autonomous operations on data sets from a set of processors without any restrictions on architectural. Instruction level data-parallel architectures should satisfy several constraints in order to build massively parallel systems.

Index terms: MIMD, **Distributed memory, shared memory.**

Introduction:

The MIMD architecture class represents a natural generalization of the uniprocessor von Neumann machine which in its simplest form consists of a single processor connected to a single memory module. If the goal is to extend this architecture to contain multiple processors and memory modules basically two alternative choices are available:

1) The first possible approach is to replicate the processor/memory pairs and to connect them via an interconnection network. The processor/memory pair is called **processing element (PE)** and they work more or less independently of each other. Whenever interaction is necessary among the PEs they send messages to each other. None of the PEs can ever access directly the memory module of another PE. This class of MIMD machines are called the **Distributed Memory MIMD Architectures** or **Message-Passing MIMD Architectures**.



The advantages of the distributed memory systems are:

1. Since processors work on their attached local memory module most of the time, the contention problem is not as severe as in the shared memory systems. As a result distributed memory multicomputers are highly scalable and good architectural candidates of building massively parallel computers.

2. Processes cannot communicate through shared data structures and hence sophisticated synchronization techniques like monitors are not needed. Message passing solves not only communication but synchronization as well.

Disadvantages of Distributed memory systems are:

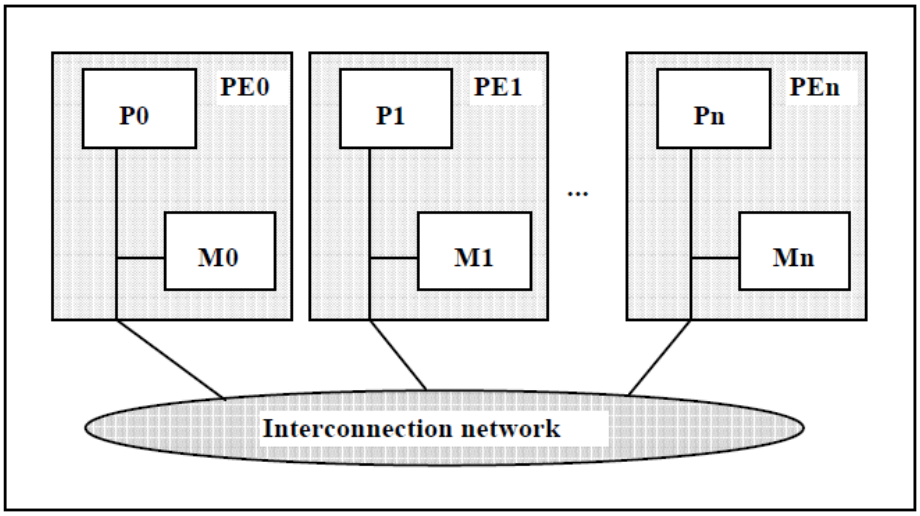
1. In order to achieve high performance in multicomputer special attention should be paid to load balancing.

2. Message-passing based communication and synchronization can lead to deadlock situations.

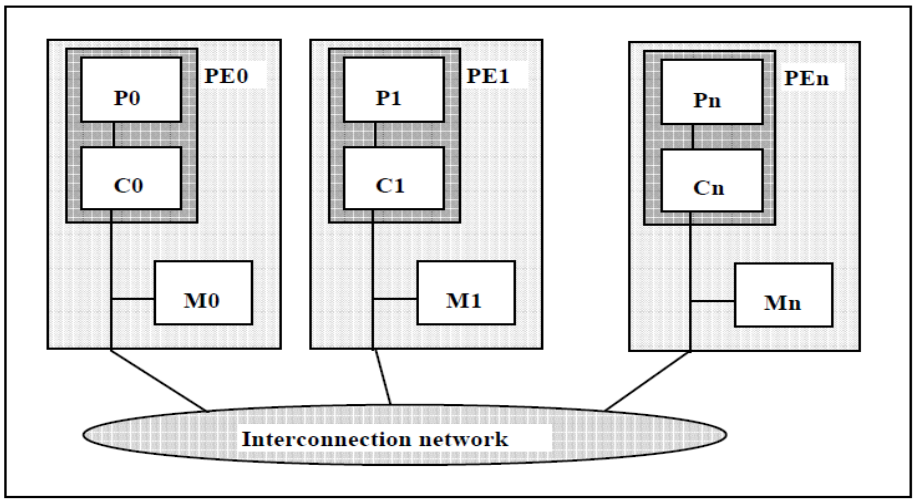
3. Though there is no architectural bottleneck in multicomputer, message-passing requires the physical copy of data structures among processes.

Distributed shared memory systems can be divided into three classes based on the access mechanism of the local memories:

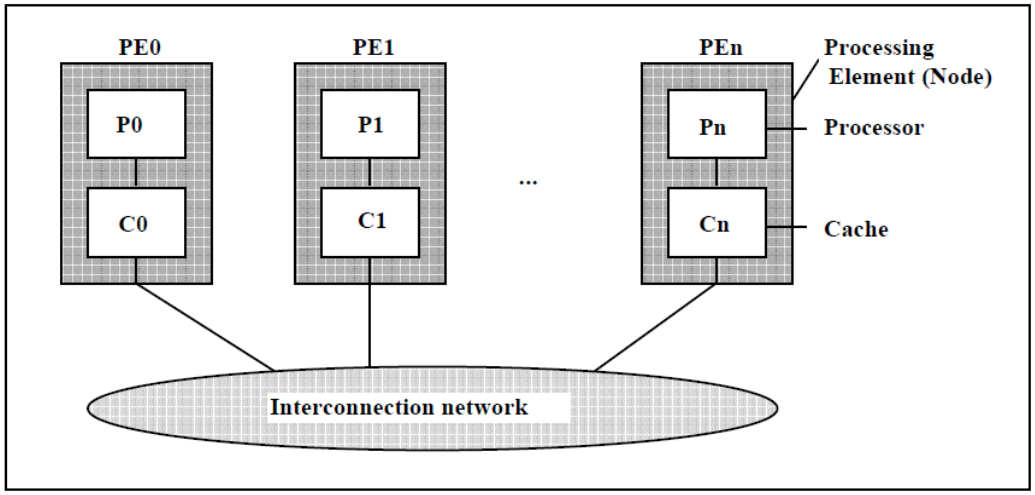
1. Non-Uniform-Memory-Access (**NUMA)** machines : In NUMA machines the shared memory is divided into as many blocks as many processors are in the system and each memory block is attached to a processor as a local memory with direct bus connection. As a result whenever a processor addresses the part of the shared memory that is connected as local memory, the access of that block is much faster than the access of the remote ones. This non-uniform access mechanism requires careful program and data distribution among the memory blocks in order to really exploit the potential high performance of these machines.



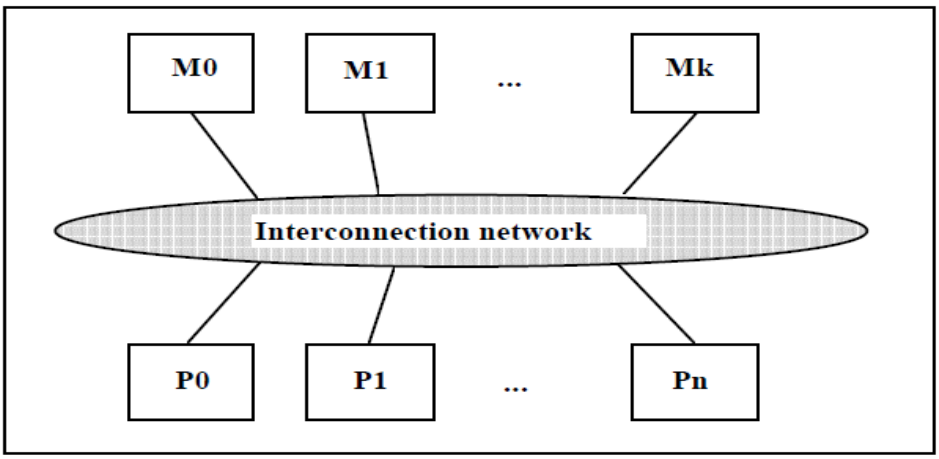
2. Cache-Coherent Non-Uniform-Memory-Architecture (**CC-NUMA**) machines: CC-NUMA machines represent a compromise between the NUMA and COMA machines. Like in the NUMA machines the shared memory is constructed as a set of local memory blocks. However, in order to reduce the traffic of the interconnection network each processor node is supplied with a large cache memory block. Though the initial data distribution is static like in the NUMA machines, dynamic load balancing is achieved by the cache coherence protocols like in the COMA machines.



2. Cache-Only Memory Architecture (**COMA**) machines : In COMA machines every memory block works as a cache memory. Based on the applied cache coherence scheme data dynamically and continuously migrate to the local caches of those processors where the data are most needed.



2. The second alternative approach is to create a set of processors and memory modules. Any processor can directly access any memory modules via an interconnection network as it is shown in Figure 2. The set of memory modules defines a global address space which is shared among the processors. The name of this kind of parallel machines is **Shared Memory MIMD Architectures** and this arrangement of processors and memory is called the **dance-hall** shared memory system. Distributed Memory MIMD Architectures are often simply called **multicomputer.** while Shared Memory MIMD Architectures are shortly referred as **multiprocessors**.



Advantages of shared memory systems:

1. There is no need to partition either the code or the data, therefore programming techniques applied for uniprocessors can easily be adapted in the multiprocessor environment. Neither new programming languages nor sophisticated compilers are needed to exploit shared memory systems.

2. There is no need to physically move data when two or more processes communicate. The consumer process can access the data on the same place where the producer composed it. As a result communication among processes are very efficient.

Disadvantages of shared memory systems:

1. Although programming shared memory systems is generally easier than programming multicomputer the synchronized access of shared data structures requires special synchronizing constructs like semaphores, conditional critical regions, monitors.

2. The main disadvantage of shared memory systems is the lack of scalability due to the contention problem.

6) Conclusion:

In computing, MIMD technique employs task of parallelism. Machines that are using MIMD techniques have a number of processors which functions asynchronously and without any dependence. At any interval of time, a different processor executes different instructions on different slots of data. MIMD architectures may be used in a number of application areas such as computer-aided design/computer-aided manufacturing, simulation, modeling, and as communication switches. MIMD machines can be of either shared memory or distributed memory categories. These classifications are based on how MIMD processors access memory.

7) References:

[1] Taxonomy of Supercomputers: http://www.cosc.brocku.ca/Offerings/3P93/notes/2-Taxonomy.pdf
[2] T. Lang and J.D. Bruguera, “Floating-Point Multiply-Add-Fused with Reduced Latency,” IEEE Trans. Computers, vol. 53, no. 8, pp. 988-1003, Aug. 2004.
[3] D. Sima, T. J. Fountain, P. Kacsuk, “Advanced Computer Architectures”.