LECTURE NOTE 7 [CSC 421]

INTRODUCTION TO PARALLEL SYSTEMS

7.1 Parallel Processing Systems

Parallel Processing Systems are designed to speed up the execution of programs by dividing the program into multiple fragments and processing these fragments simultaneously. Such systems are multiprocessor systems also known as *tightly coupled systems*. Parallel systems deal with the simultaneous use of multiple computer resources, including a single computer with multiple processors, several computers connected by a network to form a parallel processing cluster, or a combination of both.

Parallel computing is an evolution of serial computing where the jobs are broken into discrete parts that can be executed concurrently. Each part is further broken down into a series of instructions. Instructions from each part are executed simultaneously on different CPUs. Parallel systems are more difficult to program than computers with a single processor ***because the architecture of parallel computers varies accordingly and the processes of multiple CPUs must be coordinated and synchronized.***

Several models for connecting processors and memory modules exist, and each topology requires a different programming model.

The three models most commonly used in building parallel computers include ***synchronous processors each with its memory, asynchronous processors each with its memory, and asynchronous processors with a common, shared memory.***

7.2 Flynn's Classification of Parallel Systems

Michael Flynn in 1966 classified the computer systems based on parallelism in the instructions and the data streams. These are:

1. **Single Instruction, Single Data stream (SISD):** An SISD computing system is a uniprocessor machine capable of executing a single instruction, which operates on a single data stream as seen in Figure 7.1 below. In SISD, machine instructions are processed sequentially; hence computers adopting this model are popularly called sequential computers. Most conventional computers are built using the SISD model.
   All the instructions and data to be processed have to be stored in primary memory. The speed of the processing element in the SISD model is limited by the rate at which the computer can transfer information internally. Dominant representative SISD systems are IBM PCs and workstations.

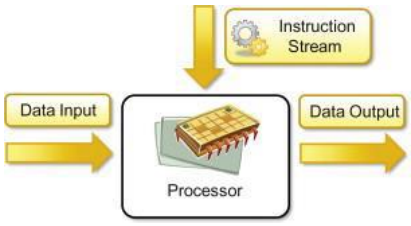

Figure 7.1: Single Instruction, Single Data stream (SISD)

2. **Single Instruction, Multiple Data stream (SIMD):** SIMD represents single-instruction multiple-data streams. The SIMD model of parallel computing includes two parts such as a front-end computer of the usual von Neumann style, and a processor array as displayed in Figure 7.2 below.
   The processor array is a collection of identical synchronized processing elements adequate for simultaneously implementing the same operation on various data. Each processor in the array has a small amount of local memory where the distributed data resides while it is being processed in parallel. The processor array is linked to the memory bus of the front end so that the front end can randomly create the local processor memories as if it were another memory.

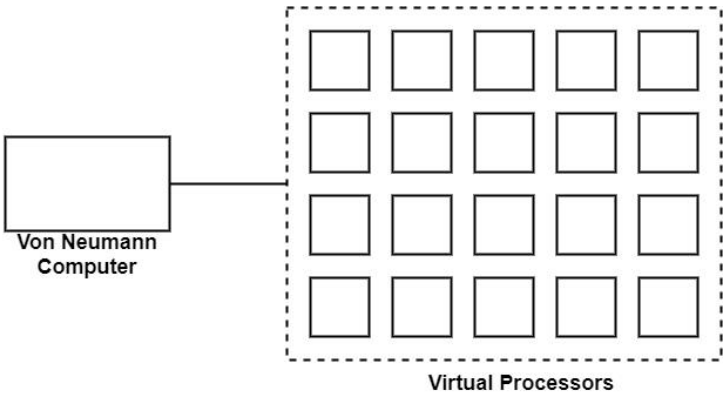

Figure 7.2: Single Instruction, Multiple Data stream (SIMD)

A program can be developed and performed on the front end using a traditional serial programming language. The application program is performed by the front end in the usual serial method but sends commands to the processor array to carry out SIMD operations in parallel.

The similarity between serial and data-parallel programming is one of the valid points of data parallelism. Synchronization is created irrelevant by the lock-step synchronization of the processors. Processors either do nothing or similar operations at the same time.

In SIMD architecture, parallelism is exploited by using simultaneous operations across huge sets of data. This paradigm is most beneficial for solving issues that have several data that require to be upgraded on a wholesale basis. It is dynamically powerful in many regular scientific calculations.

Two main configurations have been applied in SIMD machines. In the first scheme, each processor has its local memory. Processors can interact with each other through the interconnection network. If the interconnection network does not support a direct connection between given groups of processors, then this group can exchange information via an intermediate processor.

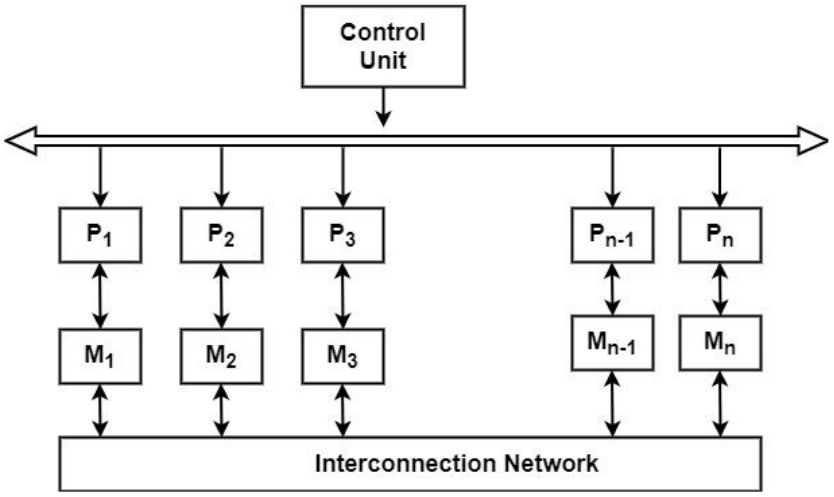

Figure 7.3: Single Instruction, Multiple Data Stream (SIMD) Scheme1

In the second SIMD scheme, processors and memory modules communicate with each other via the interconnection network. Two processors can send information to each other via intermediate memory module(s) or possibly via intermediate processor(s). The BSP (Burroughs' Scientific Processor) used the second SIMD scheme.

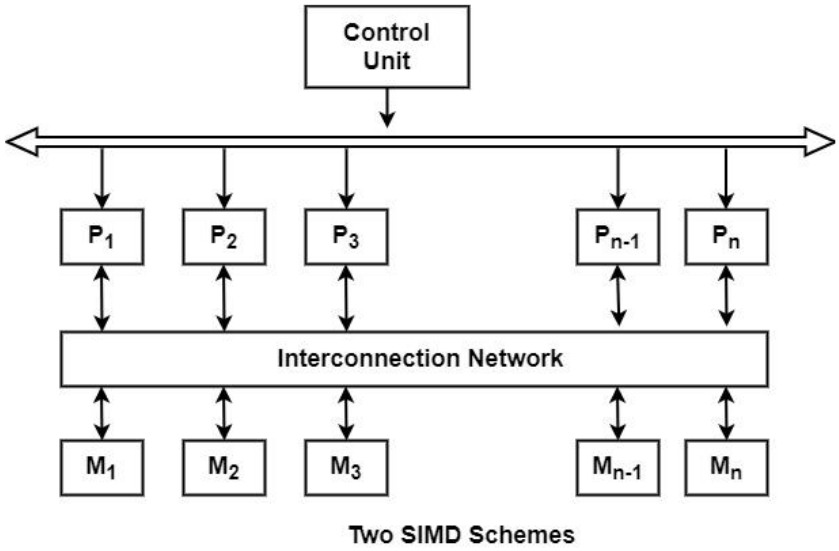

Figure 7.4: Single Instruction, Multiple Data Stream (SIMD) Scheme2

3. **Multiple Instruction, Single Data Stream (MISD):** In this classification, multiple processing elements are structured under the control of multiple control units. Each control unit handles a single instruction stream and is processed through its corresponding processing element. However, a single processing element is processing only one data stream at a time.
   Hence, for handling a single data stream and multiple instruction streams, multiple processing elements and multiple control units are organized in this classification. All processing elements are related to the common shared memory for the organization of one data stream as shown in Figure 7.5. The only identified instance of a computer capable of MISD operation is the C.mmp built by Carnegie-Mellon University.

   This type of computer organization is denoted as:

   Is > 1

   Ds = 1

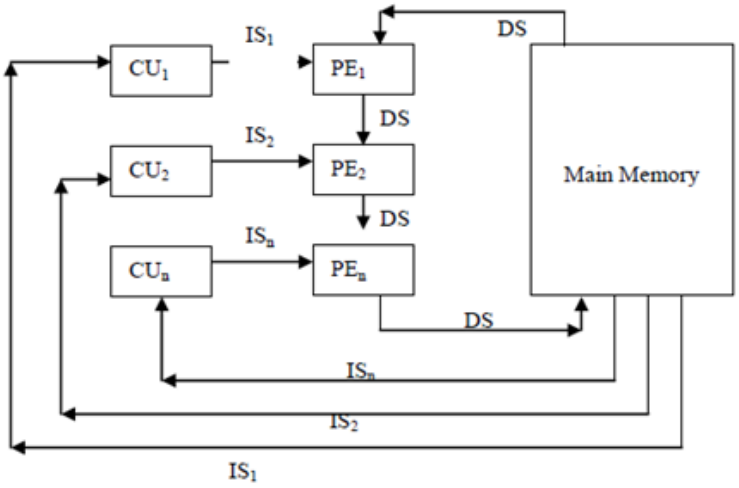

Figure 7.5: Multiple Instruction, Single Data Stream (MISD)

This classification is not popular in commercial machines as the thought of single data streams implemented on multiple processors is rarely functional. But for particular applications, MISD organization can be very useful.

For example, Real-time computers need to be fault tolerant where several processors implement the same data for producing the redundant data. This is also called as N-version programming.

All these redundant data are measured as results that should be similar; otherwise, a faulty unit is returned. Thus, MISD machines can be useful to fault-tolerant real-time computers.

4. **Multiple Instruction, Multiple Data Stream (MIMD):** In this classification,

MIMD stands for Multiple-instruction multiple-data streams. It includes parallel architectures made of multiple processors and multiple memory modules linked via some interconnection network. They fall into two broad types including shared memory or message passing.

A shared memory system generally accomplishes inter-processor coordination through a global memory shared by all processors. These are frequently server systems that communicate through a bus and cache memory controller.

The bus/cache architecture alleviates the need for expensive multi-ported memories and interface circuitry as well as the need to adopt a message-passing paradigm when developing application software. Because access to shared memory is balanced, these systems are also called ***SMP (symmetric multiprocessor) systems***. Each processor has an equal opportunity to read/write to memory, including equal access speed.

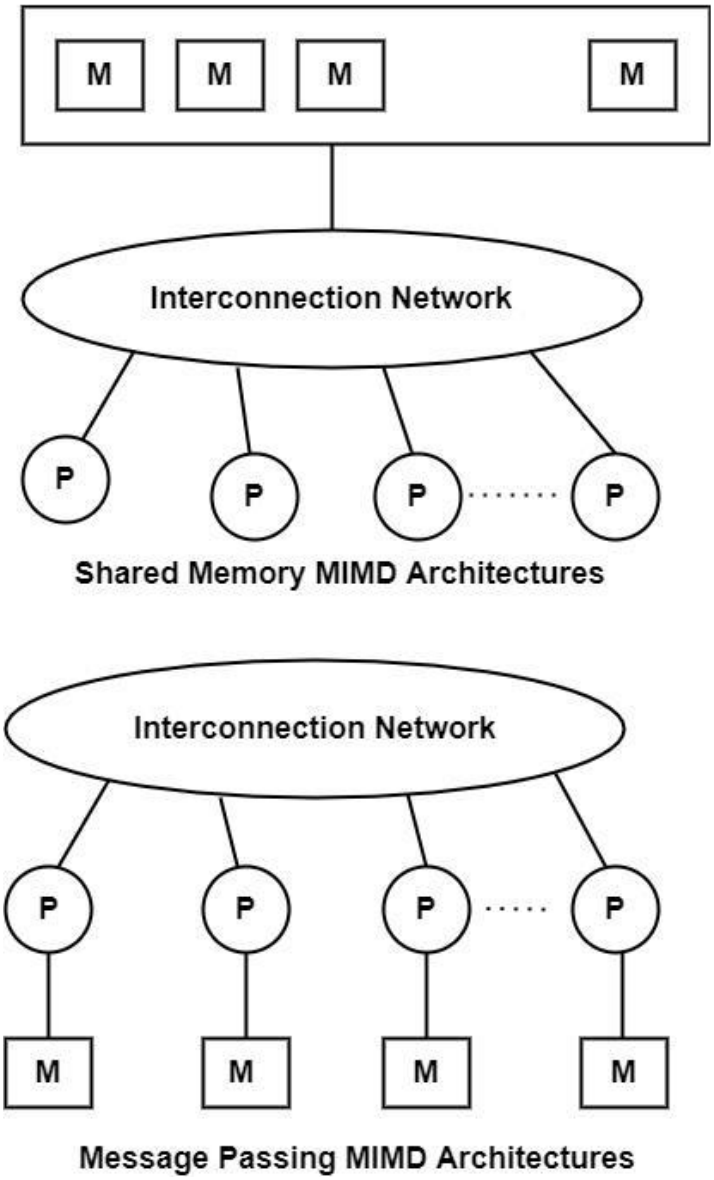

Figure 7.6: Multiple Instruction, Multiple Data stream (MIMD)

The above classification of parallel computing systems is focused in terms of two independent factors: *the number of data streams that can be simultaneously processed*, and *the number of instruction streams that can be simultaneously processed*.

Here, instruction stream means an algorithm that instructs the computer what to do whereas data stream (i.e., input to an algorithm) means the data that are being operated upon.

7.3 **Relevance of Flynn's Classification to Parallel Systems**

Even though Flynn has classified computer systems into four types based on parallelism only two of them are relevant to parallel computers. These are ***SIMD and MIMD computers***.

SIMD computers consist of 'n' processing units receiving a single stream of instruction from a central control unit and each processing unit operates on a different piece of data. Most SIMD computers operate synchronously using a single global clock. The block diagram of the SIMD computer is shown below:

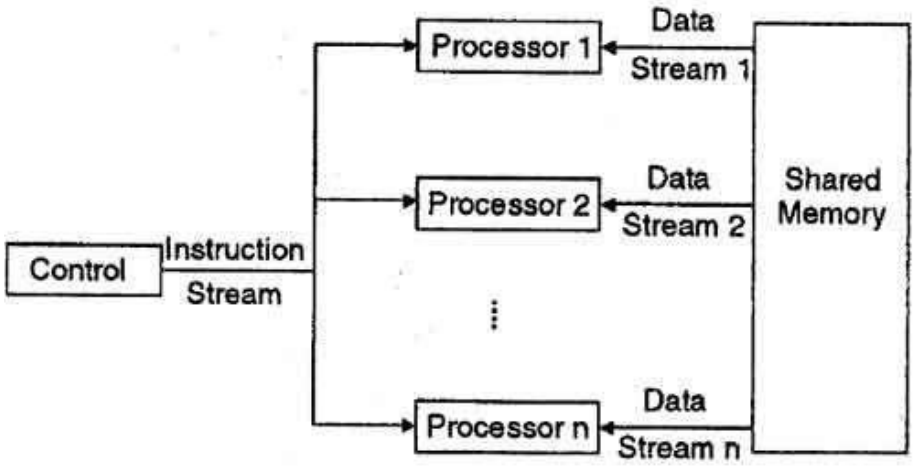

Figure 7.7: Single Instruction, Multiple Data Stream (SIMD) Block Diagram

MIMD computers consist of 'n' processing units; each with its stream of instruction and each processing unit operates on a different piece of data. MIMD is the most powerful computer system that covers a range of multiprocessor systems. The block diagram of the MIMD computer is shown below.

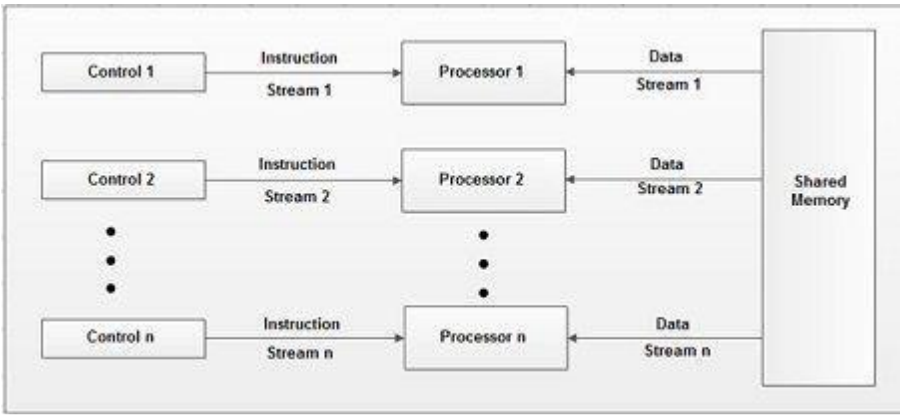

Figure 7.8: Multiple Instruction, Multiple Data Stream (MIMD) Block Diagram

The SIMD systems are easier to program because they deal with a single thread of execution. On the other hand, the MIMD machines are more efficient because you can utilize the full machine power.

7.4 **Parallel Computers and Applications**

Parallel operating systems are primarily concerned with managing the resources of parallel machines. A parallel computer is a set of processors that can work cooperatively to solve a computational problem. So, a parallel computer may be a supercomputer with hundreds or thousands of processors or maybe a network of workstations.

A few years ago, parallel computers could be found only in research laboratories and they were used mainly for computation-intensive applications like numerical simulations of complex systems. Today, there are a lot of parallel computers available in the market; used to execute both data-intensive applications in commerce and computation-intensive applications in science and engineering.

Today, new applications arise and demand faster computers. Commercial applications are the most used on parallel computers. A computer that runs such an application should be able to process large amounts of data in sophisticated ways. These applications include ***graphics, virtual reality, decision support, parallel databases, medicine diagnosis,*** and so on. Without any doubt, commercial applications will define future parallel computer architecture but scientific applications will remain important users of parallel computing technology.

Concurrency has become a fundamental requirement for algorithms and programs. A program has to be able to use a variable number of processors and also has to be able to run on multiple processors' computer architecture.

A distributed system can be described as a set of independent computers that appear to the user like a single one. So, the computers have to be independent and the software has to hide individual computers from the users. ***MIMD computers and workstations connected through LAN and WAN are examples of distributed systems***.

The main difference between parallel systems and distributed systems is how these systems are used. A parallel system uses a set of processing units to solve a single problem while a distributed system is used by many users together.