**CS546 “Parallel and Distributed Processing”**

**Homework 1**

1. **What are the two main styles of parallelism? Explain.**

Ans: Styles of parallelism-

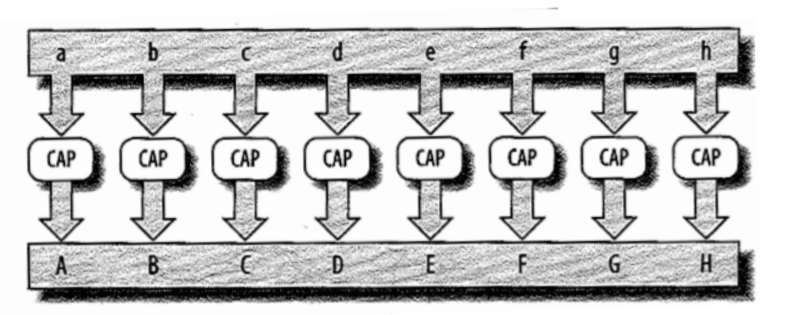
1. **Data Parallelism**
2. **Task Parallelism**
3. **Data Parallelism**

This form of parallelism includes distribution of data sets across the multiple processors. In this form, same operations are performed on different parallel computing processors on the distributed data sets.

It is the concept that you have a lot of data that you want to process like a lot of pixels in an image or you have a whole lot of payroll cheques to update. In short, data parallelism is taking that data and dividing it among multiple processors.

**Example-**

Let us consider example of converting all characters in an array to upper-case.



Provided that no dependencies between task causes their results to be ordered.

If we have two parallel computing processors, they will divide this task into two task and will run in parallel.

Processor 1 will convert letters a to d uppercase and processor 2 will do the same task in parallel on letters e to h so that resulting computing time required will be less than that of with only one processor.

1. **Task Parallelism-**

This form of parallelism covers the execution of computer programs across multiple processors on same or multiple machines. It focuses on executing different operations in parallel to fully utilize the available computing resources in form of processors and memory.

Instead of dividing up the data and doing the same work on different processors, in task parallelism task will be divided in parts to work on same data sets.

**Example**-

One example of task parallelism would be creating threads for doing parallel processing where each thread is responsible for performing a different task. Pseudo code for task parallelism -

FOR each processor in parallel computing environment

Retrieve next task from task queue

Create a thread and assign it with the retrieved task

Start the created thread

END FOR

1. What are the three parallel programming paradigms? Explain.

Ans:

Parallel Programming Models-

1. **Shared Memory**
2. **Message Passing**
3. **Data Parallel**
4. **Shared Memory**

Shared Memory programming model consist of f a number of processors or cores, a shared physical memory (global memory) and an interconnection network to connect the processors with the memory. The shared memory can be implemented as a set of memory modules. Data can be exchanged between processors via the global memory by reading or writing shared variables.

, the global memory usually consists of separate memory modules providing a common address space which can be accessed by all processors

Fig. 2.4 from pdf ‘shared memory’

Parallel programs in Shared memory model are often based on the execution of threads. A thread is a separate control flow which shares data with other threads via a global address space

Communication and cooperation between the processors is organized by writing and reading shared variables like static variables, shared common blocks that are stored in the global memory.

There are two machine models in Shared Memory viz **Symmetric Multiprocessors (SMPs) and Distributed Shared Memory**

**Symmetric Multiprocessors (SMPs)**

SMPs have a single shared memory which provides a uniform access time from any processor for all memory locations. SMPs usually have a small number of processors. Each processor has a private cache hierarchy. As usual, access to a local cache is faster than access to the global memory.

This architecture is generally called as Unified Memory Access (UMA)

**Distributed Shared Memory**

This architecture is same as Symmetric Multiprocessors except memory is logically shared but physically distributed among the processors. One processors can access memory location from another processor’s memory. These systems are therefore also called nonuniform memory access (NUMA). In this, cache coherence must be insured so that any memory access returns most recently written value regardless of physical location of value.

1. **Message Passing**

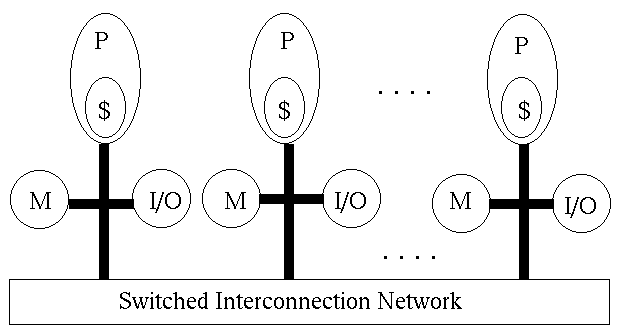
Message passing consist of several processors, each with its own memory space, and writes a program to run on each processor. The main point of the message-passing paradigm is that the processes communicate by sending each other messages. Thus, the message-passing model has no concept of a shared memory space or of processors accessing each other's memory directly.

Explain Diagram from ppt.

**2a. Distributed Memory-**

In Distributed Memory Programming model, each processor has its own memory and cache instead of one shared memory. One processor cannot directly access another processor’s memory. Every processor has its own Network Interface (NI) . Processor communicate with other using this network interface.

A schematic view of the distributed memory approach is shown in the figure below, where each processor has local memory and processors (each denoted by P) communicate through an interconnection network. Also, each processor has its own cache (denoted by $)



**2b. Internet/Grids/Cloud**

SETI@HOME

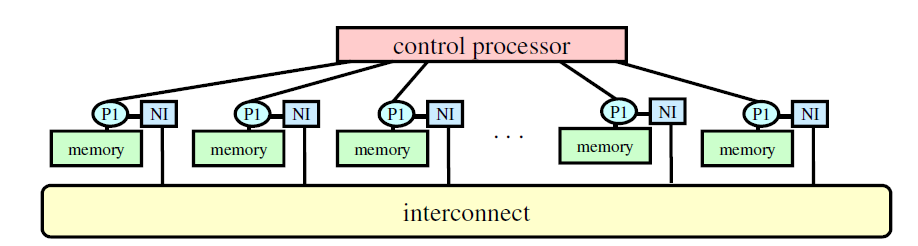
1. **Data Parallel**

the data-parallel approach is simply to assign an individual data element to a separate logical core for processing. Instead of breaking code down by subsystems.

there may be thousands to millions of data elements, enabling assignment to thousands of cores. For example, a modern GPU can support hundreds of ALUs (arithmetic logic units) with hundreds of threads per ALU for nearly 10,000 data elements on the die at once.

3a. SIMD Systems

 A type of [parallel](http://www.webopedia.com/TERM/P/parallel.html)computing [architecture](http://www.webopedia.com/TERM/A/architecture.html) that is classified under [Flynn's taxonomy](http://www.webopedia.com/TERM/F/Flynns_taxonomy.html). A single computer [instruction](http://www.webopedia.com/TERM/I/instruction.html) perform the same identical action (retrieve, calculate, or store) simultaneously on two or more pieces of data. Typically this consists of many simple [processors](http://www.webopedia.com/TERM/P/processor.html), each with a local [memory](http://www.webopedia.com/TERM/M/memory.html) in which it keeps the data which it will work on. Each processor simultaneously performs the same instruction on its local data progressing through the instructions, with the instructions issued by the controller processor.



**3b. Vector Machine**

**3. Discuss the difference between shared address space machines and distributed address space machines. And discuss the advantages and disadvantages of both architectures.**

Ans:

1. Shared Address Space Machine
   1. Provides hardware support for read/write to a shared address space that is accessible to all processors. Machines built this way are often called multiprocessors. Processors interact by modifying data objects stored in this shared address space. Memory in shared address space can be local (exclusive to processor) or global (Common to all processor).
      1. (1) A ***shared memory***machine has a single address space shared by all processors (**UMA, for Uniform Memory Access**). The time taken by a processor to access any memory word in the system is identical.
      2. (2) A ***distributed shared memory***system is a hybrid between the two previous ones. A **global address space is shared** among the processors but is distributed among them.
   2. Advantage:
      1. Global address space provides a user-friendly programming environment to memory access
      2. Data sharing between tasks is both fast and uniform due to proximity of memory to CPUs
   3. Disadvantage:
      1. Lack of scalability between memory and CPUs: Adding processors can geometrically increase traffic on the shared memory-CPU path and for cache coherence management
      2. Programmer’s responsibility for synchronization constructs (correct access to memory)
      3. Expensive to design shared memory computers with increasing numbers of processors
2. Distributed Address Space Machine
   1. In a *distributed memory machine* each processor has its own memory. Each processor can access its own memory faster than it can access the memory of a remote processor (**NUMA for Non-Uniform Memory Access**). This architecture is also known as message-passing architecture and such machines are commonly referred to as **multicomputers**. Interaction between the processes running on different nodes must be accomplished using messages hence the name message passing. This exchange of messages is used to transfer data, work and to synchronize actions among the processes.
   2. Advantage:
      1. Message passing requires little hardware support, other than a network.
      2. The programmer has an explicit control of data locality.
      3. Memory is scalable with the number of processors
      4. Increase the number of processors, the size of memory increases proportionally
      5. Each processor can rapidly access its own memory without interference and without the overhead incurred with trying to maintain cache coherence
      6. Cost effectiveness: can use commodity, off-the shelf processors and networking
   3. Disadvantage:
      1. Difficult to program: Programmer has to handle data communication between processors
      2. Nonuniform memory access (NUMA) times
      3. It may be difficult to map existing data structures, based on global memory, to distributed memory organization