Week 01

Introduction to Parallel & Distributed Computation

Overview

- The simultaneous growth in availability of big data and in the number of simultaneous users on the Internet places particular pressure on the need to carry out computing tasks “in parallel,” or simultaneously. Parallel and distributed computing occurs across many different topic areas in computer science, including algorithms, computer architecture, networks, operating systems, and software engineering. During the early 21st century there was explosive growth in multiprocessor design and other strategies for complex applications to run faster. Parallel and distributed computing builds on fundamental systems concepts, such as concurrency, mutual exclusion, consistency in state/memory manipulation, message-passing, and shared-memory models.

Distributed Computing:

- In distributed computing we have multiple autonomous computers which seems to the user as single system. In distributed systems there is no shared memory and computers communicate with each other through message passing. In distributed computing a single task is divided among different computers.

Parallel Computing

- In parallel computing multiple processors performs multiple tasks assigned to them simultaneously. Memory in parallel systems can either be shared or distributed. Parallel computing provides concurrency and saves time and money.

Difference between Parallel Computing and Distributed Computing

S.NO	Parallel Computing	Distributed Computing
1.	Many operations are performed simultaneously	System components are located at different locations
2.	Single computer is required	Uses multiple computers
3.	Multiple processors perform multiple operations	Multiple computers perform multiple operations
4.	It may have shared or distributed memory	It have only distributed memory
5.	Processors communicate with each other through bus	Computer communicate with each other through message passing.
6.	Improves the system performance	Improves system scalability, fault tolerance and resource sharing capabilities

Parallel Computing Advantages

- It is the use of multiple processing elements simultaneously for solving any problem. Problems are broken down into instructions and are solved concurrently as each resource which has been applied to work is working at the same time.
- Advantages of Parallel Computing over Serial Computing are as follows:
- 1. It saves time and money as many resources working together will reduce the time and cut potential costs.
- 2. It can be impractical to solve larger problems on Serial Computing.
- 3. It can take advantage of non-local resources when the local resources are finite.
- 4. Serial Computing ‘wastes’ the potential computing power, thus Parallel Computing makes better work of hardware.

Types of Parallelism

1. Bit-level parallelism: It is the form of parallel computing which is based on the increasing processor's size. It reduces the number of instructions that the system must execute in order to perform a task on large-sized data. Example: Consider a scenario where an 8-bit processor must compute the sum of two 16-bit integers. It must first sum up the 8 lower-order bits, then add the 8 higher-order bits, thus requiring two instructions to perform the operation. A 16-bit processor can perform the operation with just one instruction.
2. Instruction-level parallelism: A processor can only address less than one instruction for each clock cycle phase. These instructions can be re-ordered and grouped which are later on executed concurrently without affecting the result of the program. This is called instruction-level parallelism.
3. Task Parallelism: Task parallelism employs the decomposition of a task into subtasks and then allocating each of the subtasks for execution. The processors perform execution of sub tasks concurrently

Why parallel computing?

- The whole real world runs in dynamic nature i.e. many things happen at a certain time but at different places concurrently. This data is extensively huge to manage.
- Real world data needs more dynamic simulation and modeling, and for achieving the same, parallel computing is the key.
- Parallel computing provides concurrency and saves time and money.
- Complex, large datasets, and their management can be organized only and only using parallel computing’s approach.
- Ensures the effective utilization of the resources. The hardware is guaranteed to be used effectively whereas in serial computation only some part of hardware was used and the rest rendered idle.
- Also, it is impractical to implement real-time systems using serial computing.

Applications of Parallel Computing

- Data bases and Data mining.
- Real time simulation of systems.
- Science and Engineering.
- Advanced graphics, augmented reality and virtual reality.

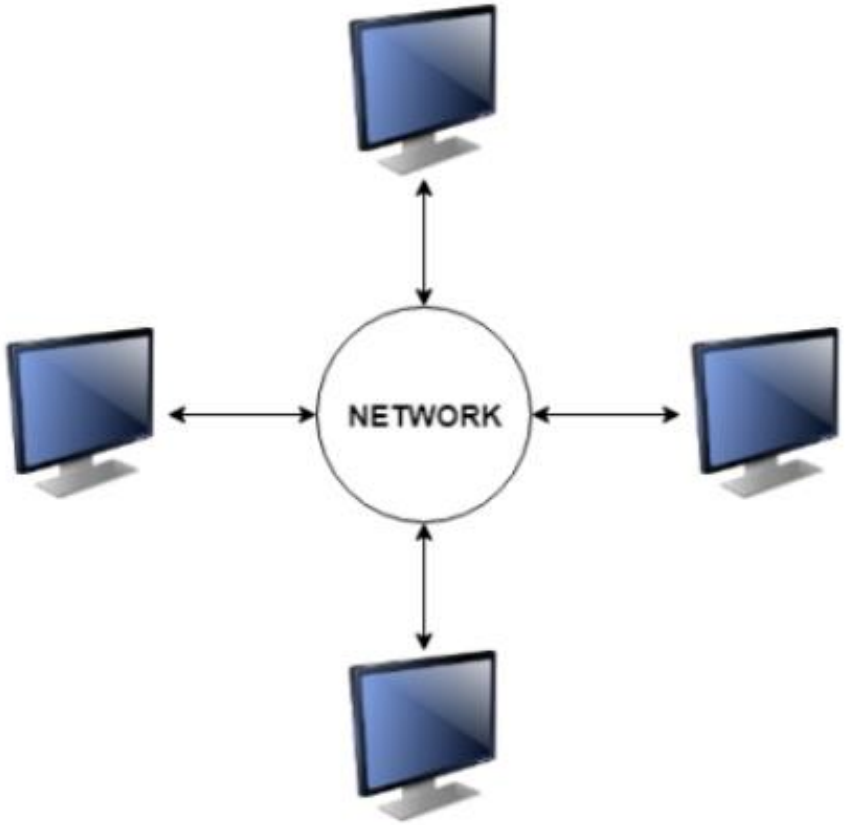
Future of Parallel Computing

- The computational graph has undergone a great transition from serial computing to parallel computing. Tech giant such as Intel has already taken a step towards parallel computing by employing multicore processors. Parallel computation will revolutionize the way computers work in the future, for the
- better good. With all the world connecting to each other even more than before, Parallel Computing does a better role in helping us stay that way. With faster networks, distributed systems, and multi-processor computers, it becomes even more necessary

Distributed System

- A distributed system contains multiple nodes that are physically separate but linked together using the network. All the nodes in this system communicate with each other and handle processes in tandem. Each of these nodes contains a small part of the distributed operating system software. A diagram to better explain the distributed system is −

A diagram to better explain the distributed system is



DISTRIBUTED OPERATING SYSTEM

Types of Distributed Systems

- The nodes in the distributed systems can be arranged in the form of client/server systems or peer to peer systems. Details about these are as follows.
- **Client/Server Systems**
- In client server systems, the client requests a resource and the server provides that resource. A server may serve multiple clients at the same time while a client is in contact with only one server. Both the client and server usually communicate via a computer network and so they are a part of distributed systems.
- **Peer to Peer Systems**
- The peer to peer systems contains nodes that are equal participants in data sharing. All the tasks are equally divided between all the nodes. The nodes interact with each other as required as share resources. This is done with the help of a network.

Advantages of Distributed Systems

- Some advantages of Distributed Systems are as follows −
- All the nodes in the distributed system are connected to each other. So nodes can easily share data with other nodes.
- More nodes can easily be added to the distributed system i.e. it can be scaled as required.
- Failure of one node does not lead to the failure of the entire distributed system. Other nodes can still communicate with each other.
- Resources like printers can be shared with multiple nodes rather than being restricted to just one.

Disadvantages of Distributed Systems

- Some disadvantages of Distributed Systems are as follows −
- It is difficult to provide adequate security in distributed systems because the nodes as well as the connections need to be secured.
- Some messages and data can be lost in the network while moving from one node to another.
- The database connected to the distributed systems is quite complicated and difficult to handle as compared to a single user system.
- Overloading may occur in the network if all the nodes of the distributed system try to send data at once.

Concurrency

- Consider multiple tasks to be executed in a computer
- Tasks are concurrent with respect to each if
  - They *can* execute at the same time (*concurrent execution*)
  - Implies that there are no dependencies between the tasks
- Dependencies
  - If a task requires results produced by other tasks in order to execute correctly, the task's execution is *dependent*
  - If two tasks are dependent, they are not concurrent
  - Some form of synchronization must be used to enforce (satisfy) dependencies
- Concurrency is fundamental to computer science
  - Operating systems, databases, networking, …

Concurrency and Parallelism

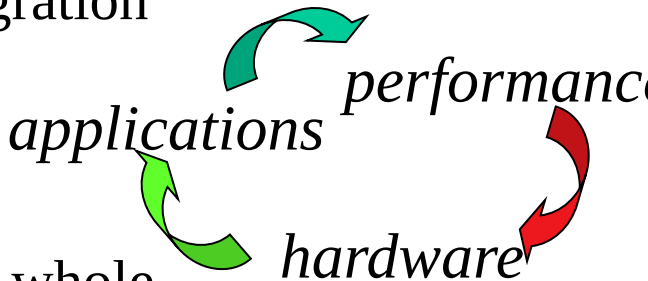
- Concurrent is not the same as parallel! Why?
- Parallel execution
  - Concurrent tasks *actually* execute at the same time
  - Multiple (processing) resources have to be available
- **Parallelism = concurrency + "parallel" hardware**
  - Both are required
  - Find concurrent execution opportunities
  - Develop application to execute in parallel
  - Run application on parallel hardware

Inevitability of Parallel Computing

- Application demands
  - Insatiable need for computing cycles
- Technology trends
  - Processor and memory
- Architecture trends
- Economics
- Current trends:
  - Today's microprocessors have multiprocessor support
  - Servers and workstations available as multiprocessors
  - Tomorrow's microprocessors are multiprocessors
  - Multi-core is here to stay and #cores/processor is growing
  - Accelerators (GPUs, gaming systems)

Application Characteristics

- Application performance demands hardware advances
- Hardware advances generate new applications
- New applications have greater performance demands
  - Exponential increase in microprocessor performance
  - Innovations in parallel architecture and integration



- Range of performance requirements
  - System performance must also improve as a whole
  - Performance requirements require computer engineering
  - Costs addressed through technology advancements

Broad Parallel Architecture Issues

- Resource allocation
  - How many processing elements?
  - How powerful are the elements?
  - How much memory?
- Data access, communication, and synchronization
  - How do the elements cooperate and communicate?
  - How are data transmitted between processors?
  - What are the abstractions and primitives for cooperation?
- Performance and scalability
  - How does it all translate into performance?
  - How does it scale?

Leveraging Moore's Law

- More transistors = more parallelism opportunities
- Microprocessors
  - Implicit parallelism
    - pipelining
    - multiple functional units
    - superscalar
  - Explicit parallelism
    - SIMD instructions
    - long instruction works