Unit 1: Introduction to parallel processing and trends

Uniprocessor is a computer system type that has only one central processing unit which means that only one instruction will be executed at a time All tasks and operations are handled by a single processor. This type of processor is found only on personal computers mobile devices and small embedded systems. These systems have limited computing power as they can execute only one instruction at a time. These types of processors are suitable for various common computing tasks such as web browsing, word processing, and basic gaming. But multiprocessor systems it has multiple processors that execute instructions simultaneously by providing greater computing power and faster processing speeds.

Parallelism is the process performed by the computer system to perform multiple instructions simultaneously by dividing each task into different processors. This ability is achieved in Uniprocessor by using many ways such as using multi-processor or cores within a single processor, diving a task into smaller subtasks that can be executed concurrently, or using specialized hardware or software to manage parallel processing.

Parallelism in Uniprocessor

Uniprocessor has only one processor but still, it is possible to achieve parallelism by using certain techniques such as pipelining and multitasking.

Pipelining is a technique that allows a processor to execute a set of instructions simultaneously by dividing the instructions execution process into several stages. Each stage works on a different instruction at the same time, so that when one instruction is being fetched in the memory another instruction is being executed. This increases the throughput of a processor and improves performance.

Multitasking is a technique where it enables a uniprocessor to execute multiple tasks simultaneously. This is achieved by dividing the processor's time into short time slots and switching tasks rapidly. Each task is given a specific time slot in which the needs to be executed. This gives an appearance of parallel execution even if the processor is only executing one task at a time.

These techniques improve the performance of a uniprocessor but as the number of tasks or instructions are executed simultaneously increases the performance eventually gets declined hence here we need a multiprocessor to improve performance.

Advantages of Uniprocessor

1. **Improves performance-** Improves the performance of a uniprocessor by allowing it to execute multiple tasks or instructions simultaneously. This is achieved by increasing throughput which reduces the time required to complete a particular task.
2. **Cost Effective-** A Parallelism in uniprocessor is cost-effective for applications that do not require the performance of a multiprocessing system. The cost of a uniprocessor with parallelism is often lower compared to a multiprocessing system.
3. **Low power consumption-** A uniprocessor consumes less power than a multiprocessor system which makes it suitable for mobile and battery powers devices.

Disadvantage of Uniprocessor

1. **Limited scalability–** Parallelism is achieved in a very limited way and as the number of tasks or instructions being executed simultaneously increases the performance decrease. This makes it unsuitable for applications that require high levels of parallelism.
2. **Limited processing power–** It has limited processing power as compared to a multiprocessing system hence it is not suitable for applications that require high computational power like scientific simulations and large-scale data processing.
3. **Complex design–** Implementing parallelism in a uniprocessor can be complex as it requires careful design and optimization to ensure that the system operates correctly and efficiently this increases the development and maintenance costs of the system.

Applications of Parallelism in Uniprocessor

1. **Multimedia applications–** In multimedia applications such as video and audio playback, image processing, and 3D graphics rendering it helps in increasing performance.
2. **Web servers–** Provides assistance to web servers by allowing them to handle multiple requests simultaneously which makes it more reliable.

3. **Artificial Intelligence and machine learning–** It improves performance in artificial intelligence and machine learning applications allowing them to process large amounts of data more quickly.
4. **Scientific simulations–** Parallelism performs scientific simulations such as weather forecasting, fluid dynamics, and molecular modeling.
5. **Database management systems–** Parallelism in uniprocessors is used to improve the performance of database management systems by allowing them to handle large volumes of data more efficiently.

Parallel Computing

Parallel computing is a computing architecture that divides a problem into smaller tasks and runs them concurrently. It has the ability to process multiple tasks simultaneously, making it significantly faster than a sequential computer. Parallel computing helps to solve large, complex problems in a much shorter time.

The concept of parallel computing is not new. It dates back to the mid-20th century when it was introduced to speed up numerical calculations. Today, thanks to technological advancements, parallel computing is used in a wide range of applications, including in big data analytics, artificial intelligence, weather forecasting, and scientific research. Modern parallel computing systems can scale up to millions of computers and perform operations on massive datasets in a fraction of a second.

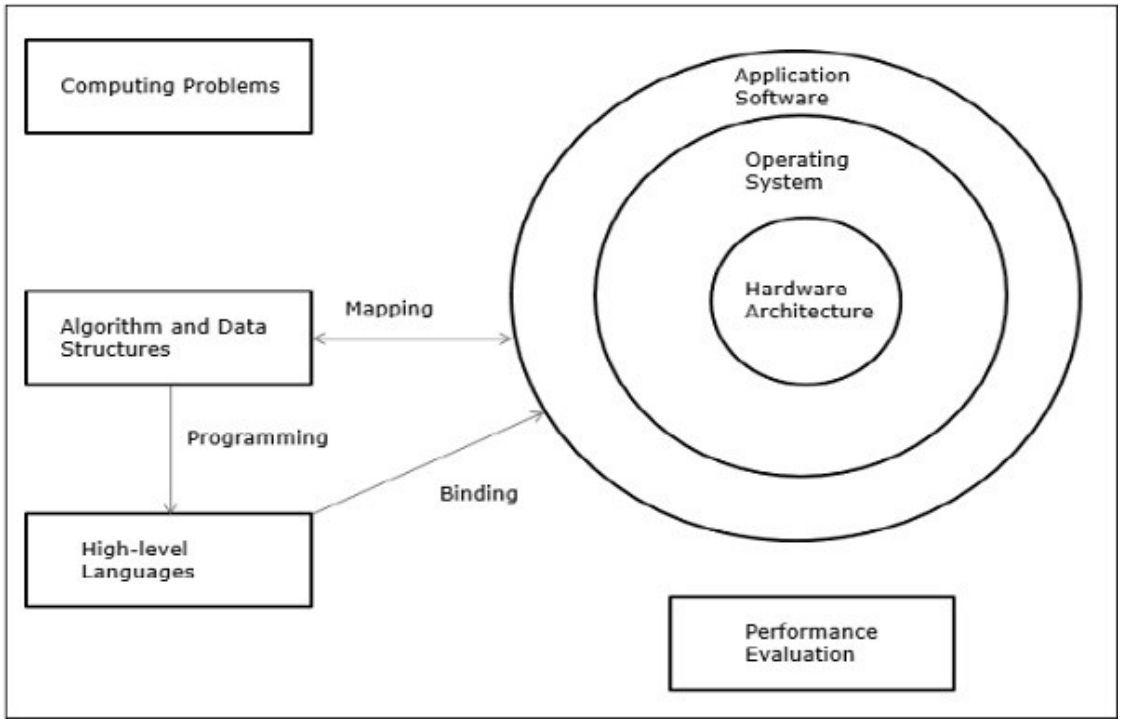
Parallel processing has been developed as an effective technology in modern computers to meet the demand for higher performance, lower cost and accurate results in real-life applications. Concurrent events are common in today's computers due to the practice of multiprogramming, multiprocessing, or multicompetent.

Modern computers have powerful and extensive software packages. To analyze the development of the performance of computers, first we have to understand the basic development of hardware and software.

Computer Development Milestones − There is two major stages of development of computer - mechanical or electromechanical parts. Modern computers evolved after the introduction of electronic components. High mobility electrons in electronic computers replaced the

operational parts in mechanical computers. For information transmission, electric signal which travels almost at the speed of a light replaced mechanical gears or levers.

Elements of Modern computers − A modern computer system consists of computer hardware, instruction sets, application programs, system software and user interface.



The computing problems are categorized as numerical computing, logical reasoning, and transaction processing. Some complex problems may need the combination of all the three processing modes.

Evolution of Computer Architecture − In last four decades, computer architecture has gone through revolutionary changes. We started with Von Neumann architecture and now we have multicomputers and multiprocessors.

Performance of a computer system − Performance of a computer system depends both on machine capability and program behavior. Machine capability can be improved with better hardware technology, advanced architectural features and efficient resource management. Program behavior is unpredictable as it is dependent on application and run-time conditions

Classes of Parallel Computers

Parallel computers are classified based on their structure and the way they handle tasks. Here are the main types:

1. **Multi-Core Computing:-** One of the most common forms of parallel computing is multi-core computing. This involves a single computing component with two or more independent processing units, known as cores. Each core can execute instructions independently of the others.

   Multi-core processors have become the norm in personal computers and servers, as they increase performance and energy efficiency. They are particularly useful in multitasking environments where several programs run concurrently.

2. **Symmetric Multiprocessing:-** Symmetric multiprocessing (SMP) is a class of parallel computing architecture where two or more identical processors are connected to a single shared main memory. Most SMP systems use a uniform memory access (UMA) architecture, in which all processors share the physical memory.

   SMP systems are highly efficient when running multiple tasks that require frequent inter-processor communication. They are commonly used in servers, where many tasks need to be executed simultaneously. The primary advantage of SMP systems is their ability to increase computational speed while maintaining the simplicity of a single processor system.

3. **Distributed Computing:-** In distributed computing, a single task is divided into many smaller subtasks that are distributed across multiple computers. These computers may be located in the same physical location, or they may be spread out across different geographical locations.

   Distributed computing systems are highly scalable, as more computers can be added to the network to increase computational power. They are used for tasks that require massive amounts of data and computational resources, such as processing of large databases, scientific simulations, and large-scale web applications.

4. **Cluster Computing:-** Cluster computing is a type of parallel computing where a group of computers are linked together to form a single, unified computing resource. These

computers, known as nodes, work together to execute tasks more quickly than a single computer could.

Cluster computing is useful for tasks that require high performance, reliability, and availability. By distributing tasks across multiple nodes, cluster computing reduces the risk of system failure, as even if one node fails, the remaining nodes can continue processing.

5. **Massively Parallel Computing:-** Massively parallel computing is a type of parallel computing where hundreds or thousands of processors are used to perform a set of coordinated computations simultaneously. This type of computing is used for tasks that require high computational power, such as genetic sequencing, climate modeling, and fluid dynamics simulations.

   Massively parallel computers use a distributed memory architecture, where each processor has its own private memory. Communication between processors is achieved through a variety of methods, including messaging systems and shared memory.

6. **Grid Computing:-** Grid computing is a form of distributed computing where a virtual supercomputer is composed of networked, loosely coupled computers, which are used to perform large tasks.

   Grid computing is used for tasks that require a large amount of computational resources that can't be fulfilled by a single computer but don't require the high performance of a supercomputer. It's commonly used in scientific, mathematical, and academic research, as well as in large enterprises for resource-intensive tasks.

Types of Parallel Computing Architectures

1. **Shared Memory Systems:-** In shared memory systems, multiple processors access the same physical memory. This allows for efficient communication between processors because they directly read from and write to a common memory space. Shared memory systems are typically easier to program than distributed memory systems due to the simplicity of memory access.

   However, shared memory systems can face challenges with scalability and memory contention. As the number of processors increases, the demand for memory access can lead to bottlenecks, where processors are waiting for access to shared memory.

Common examples of shared memory systems include symmetric multiprocessors (SMP) and multicore processors found in modern desktop computers and servers. These systems are well-suited for applications that require tight coupling and frequent communication between processors, such as real-time data processing and complex simulations.

2. **Distributed Memory Systems:-** Distributed memory systems consist of multiple processors, each with its own private memory. Processors communicate by passing messages over a network. This design can scale more effectively than shared memory systems, as each processor operates independently, and the network can handle communication between them.
   The primary challenge in distributed memory systems is the complexity of communication and synchronization. Programmers need to explicitly manage data distribution and message passing, often using libraries such as MPI (Message Passing Interface). The latency and bandwidth of the network can also impact performance.

   Distributed memory systems are commonly used in high-performance computing (HPC) environments, such as supercomputers and large-scale clusters. They are suitable for applications that can be decomposed into independent tasks with minimal inter-process communication, like large-scale simulations and data analysis.

3. **Hybrid Systems:-** Hybrid systems combine elements of shared and distributed memory architectures. They typically feature nodes that use shared memory, interconnected by a distributed memory network. Each node operates as a shared memory system, while communication between nodes follows a distributed memory model.
   Within a node, tasks can communicate quickly using shared memory, while inter-node communication uses message passing.

   One common use case for hybrid systems is in large-scale scientific computing, where computations are divided into smaller tasks within nodes and coordinated across a larger network. Hybrid systems can efficiently handle complex workloads that require both high-speed local computation and distributed processing across a large number of processors.

Parallel Computing Techniques

Here are the primary techniques used to parallelize tasks on computing systems:

1. **Bit-Level Parallelism**

Bit-level parallelism is a type of parallel computing that seeks to increase the number of bits processed in a single instruction. This form of parallelism dates back to the era of early computers, where it was discovered that using larger word sizes could significantly speed up computation.

In bit-level parallelism, the focus is primarily on the size of the processor's registers. These registers hold the data being processed. By increasing the register size, more bits can be handled simultaneously, thus increasing computational speed. The shift from 32-bit to 64-bit computing in the early 2000s is a prime example of bit-level parallelism.

While the implementation of bit-level parallelism is largely hardware-based, it's crucial to understand its implications. For programmers, understanding bit-level parallelism can help design more efficient algorithms, especially for tasks that involve heavy numerical computation.

2. **Instruction-Level Parallelism**

Instruction-level parallelism (ILP) is another form of parallel computing that focuses on executing multiple instructions simultaneously. Unlike bit-level parallelism, which focuses on data, ILP is all about instructions.

The idea behind ILP is simple: instead of waiting for one instruction to complete before the next starts, a system can start executing the next instruction even before the first one has completed. This approach, known as pipelining, allows for the simultaneous execution of instructions and thus increases the speed of computation.

However, not all instructions can be effectively pipelined. Dependencies between instructions can limit the effectiveness of ILP. For instance, if one instruction depends on the result of another, it cannot be started until the first instruction completes.

3. Super word Level Parallelism

Superword Level Parallelism (SLP) is a type of parallel computing that focuses on vectorizing operations on data stored in short vector registers. It is a form of data parallelism that operates on arrays or vectors of data.

In superword level parallelism, single instruction, multiple data (SIMD) operations are performed, where one instruction is applied to multiple pieces of data simultaneously. This technique is particularly effective in applications that require the same operation to be performed on large datasets, such as in image and signal processing.

SLP requires both hardware support in the form of vector registers and compiler support to identify opportunities for vectorization. As such, effectively leveraging SLP can be challenging, but the potential performance gains make it a valuable tool in the parallel computing toolbox.

4. Task Parallelism

While bit-level and instruction-level parallelism focus on data and instructions, in task parallelism, the focus is on distributing tasks across different processors.
A task, in this context, is a unit of work performed by a process. It could be anything from a simple arithmetic operation to a complex computational procedure. The key idea behind task parallelism is that by distributing tasks among multiple processors, we can get more work done in less time.

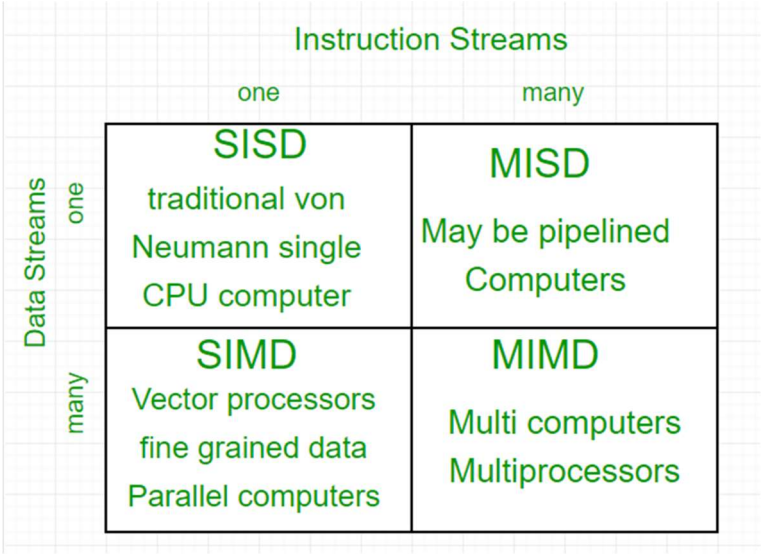
This form of parallelism requires careful planning and coordination. Tasks need to be divided in such a way that they can be executed independently. Furthermore, tasks may need to communicate with each other, which requires additional coordination.

Computer Architecture | Flynn's taxonomy

Parallel computing is a computing where the jobs are broken into discrete parts that can be executed concurrently. Each part is further broken down to a series of instructions. Instructions from each part execute simultaneously on different CPUs. Parallel systems deal with the simultaneous use of multiple computer resources that can include a single computer with multiple processors, a number of computers connected by a network to form a parallel processing cluster or a combination of both.

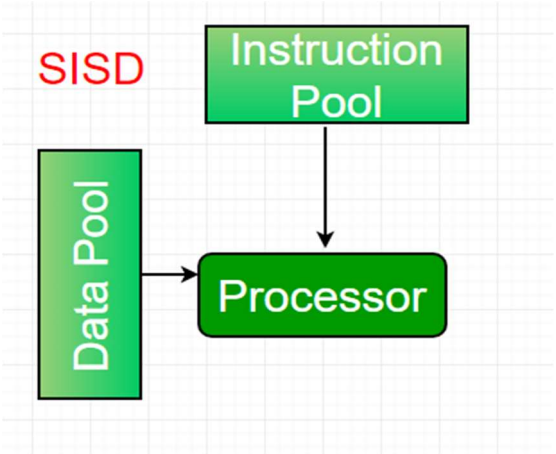
Parallel systems are more difficult to program than computers with a single processor because the architecture of parallel computers varies accordingly and the processes of multiple CPUs must be coordinated and synchronized.

The crux of parallel processing are CPUs. Based on the number of instruction and data streams that can be processed simultaneously, computing systems are classified into four major categories:



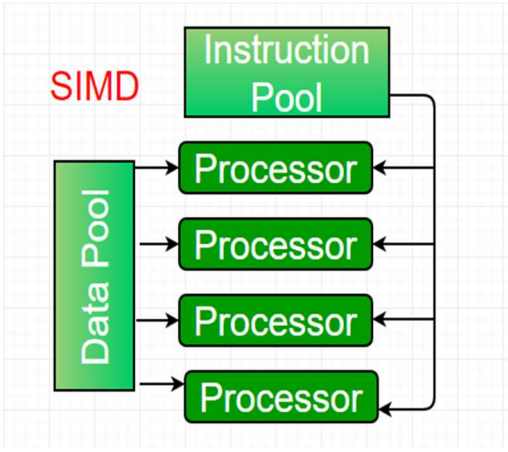
Flynn's classification –

Single-instruction, single-data (SISD) systems:- An SISD computing system is a uniprocessor machine which is capable of executing a single instruction, operating on a single data stream. In SISD, machine instructions are processed in a sequential manner and computers adopting this model are popularly called sequential computers. Most conventional computers have SISD architecture. All the instructions and data to be processed have to be stored in primary memory.



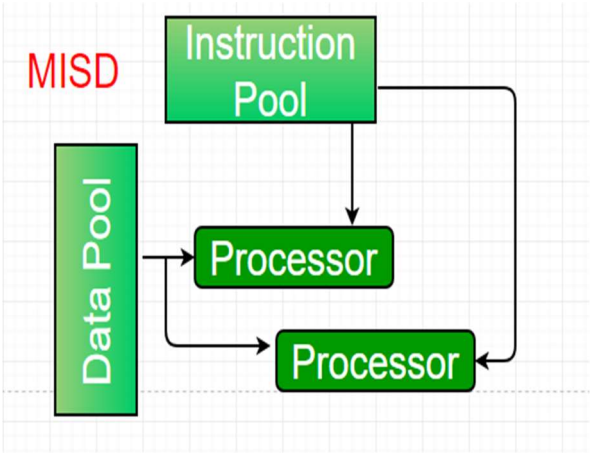
The speed of the processing element in the SISD model is limited(dependent) by the rate at which the computer can transfer information internally. Dominant representative SISD systems are IBM PC, workstations.

Single-instruction, multiple-data (SIMD) systems:- An SIMD system is a multiprocessor machine capable of executing the same instruction on all the CPUs but operating on different data streams. Machines based on an SIMD model are well suited to scientific computing since they involve lots of vector and matrix operations. So that the information can be passed to all the processing elements (PEs) organized data elements of vectors can be divided into multiple sets(N-sets for N PE systems) and each PE can process one data set.



Dominant representative SIMD systems is Cray's vector processing machine.

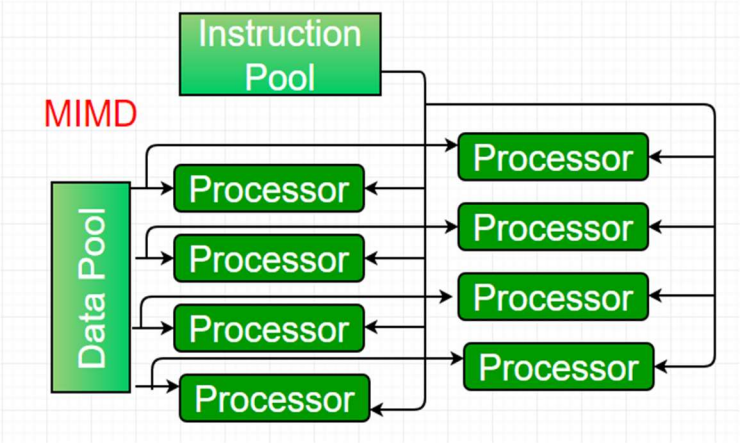
Multiple-instruction, single-data (MISD) systems:- An MISD computing system is a multiprocessor machine capable of executing different instructions on different PEs but all of them operating on the same dataset .



Example Z = sin(x)+cos(x)+tan(x)

The system performs different operations on the same data set. Machines built using the MISD model are not useful in most of the application, a few machines are built, but none of them are available commercially.

Multiple-instruction, multiple-data (MIMD) systems:- An MIMD system is a multiprocessor machine which is capable of executing multiple instructions on multiple data sets. Each PE in the MIMD model has separate instruction and data streams; therefore machines built using this model are capable to any kind of application. Unlike SIMD and MISD machines, PEs in MIMD machines work asynchronously.



MIMD machines are broadly categorized into shared-memory MIMD and distributed-memory MIMD based on the way PEs are coupled to the main memory.

In the shared memory MIMD model (tightly coupled multiprocessor systems), all the PEs are connected to a single global memory and they all have access to it. The communication between PEs in this model takes place through the shared memory, modification of the data stored in the global memory by one PE is visible to all other PEs. Dominant representative shared memory MIMD systems are Silicon Graphics machines and Sun/IBM's SMP (Symmetric Multi-Processing).

In Distributed memory MIMD machines (loosely coupled multiprocessor systems) all PEs have a local memory. The communication between PEs in this model takes place through the interconnection network (the inter process communication channel, or IPC). The network connecting PEs can be configured to tree, mesh or in accordance with the requirement. The shared-memory MIMD architecture is easier to program but is less tolerant to failures and harder to extend with respect to the distributed memory MIMD model. Failures in a shared-

memory MIMD affect the entire system, whereas this is not the case of the distributed model, in which each of the PEs can be easily isolated. Moreover, shared memory MIMD architectures are less likely to scale because the addition of more PEs leads to memory contention. This is a situation that does not happen in the case of distributed memory, in which each PE has its own memory. As a result of practical outcomes and user's requirement, distributed memory MIMD architecture is superior to the other existing models.