International Journal of Scientific Research in Engineering and Management (IJSREM)
Volume: 08 Issue: 10 | Oct - 2024 SJIF Rating: 8.448 ISSN: 2582-3930

PARALLEL COMPUTING

Mrs. Anush Sharma
(Department of Computer Science and Engineering) HIET Shahpur, Kangra

Aniket, Rishav Sharma, Raman Kumar
(Student of Computer Science Department)
HIET Shahpur, Kangra

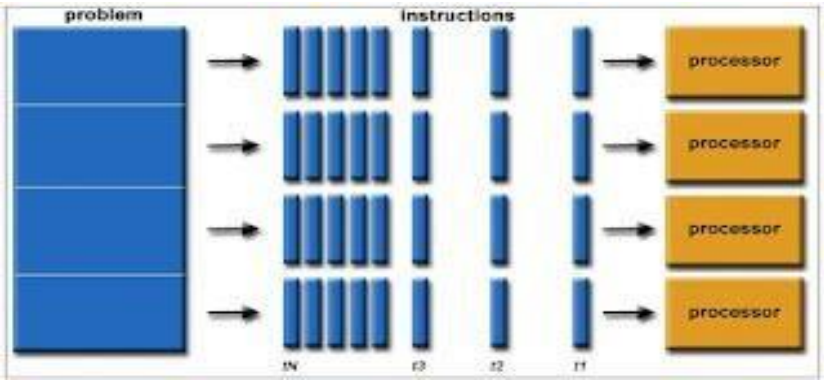
Abstract

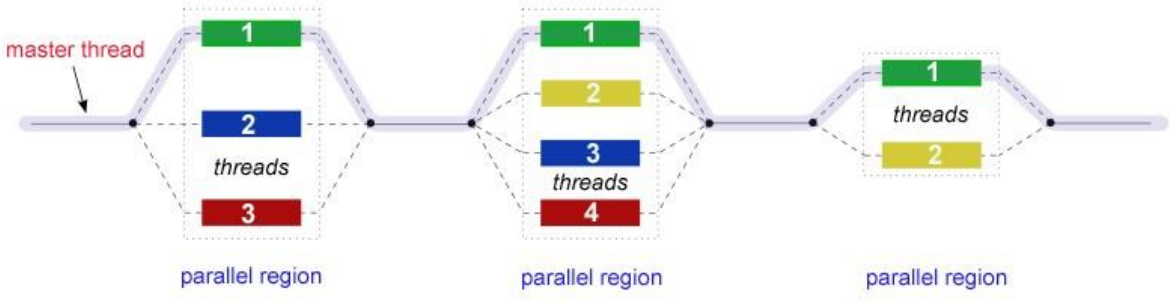
Parallel computing is a computational technique in which two or more tasks or instructions can be processed concurrently by means of several processors resulting into enhanced computational efficiency as well as decreased time for solving complex problems. As an approach to solve a particular problem, parallel computing is an attempt that decomposes a problem into constituent subproblems to be solved concurrently; it uses multiple cores, hardware systems to gain advantage in speed The use of parallel computing is prominent in sciences that call for massive computations such as simulations, data mining, artificial intelligence and HPC.

Key Words- Parallel computing, Concurrent processing, Computational efficiency, Problem decomposition, Subproblems, Simulations, Data mining, Artificial intelligence, High-performance computing (HPC), Multiple processors, Multiple cores, Hardware systems

Introduction to Parallel Computing

Parallel computing is a method where several processing units or cores join forces to tackle complex. It helps make things quicker. This technique breaks down large tasks into smaller, separate pieces that can be handled at the same time by different processors or computing units. So, it boosts how fast applications work & how quickly tasks get done by making more computing power available in a system Supercomputers widely use the principle of parallel computing to get their job done. In situations that demand a lot of processing power, you'll often find parallel processing being used.





Objective

Parallel computing aims to increase the computational rate by splitting a larger problems into finer and smaller ones so that can be runs parallel with each other. Parallel computing makes use of multiple processors or cores to work simultaneously to cut down execution time, process larger datasets and solve complex problems that otherwise would not be possible for a single processor.

- **Speedup and Performance Improvement:**
  Decrease Execution Time: The same problem can be solved on the order of 1000 times faster using the parallel approach to distributing tasks over multiple processors instead of sequentially computing them.

- **Scalability**:
  We are only going to use a small dataset but many times this is the best way to test your parallel code on larger datasets that you can't run in a single machine reduces when dealing with multiple nodes.

- **Resource Utilization**:
  Optimal Use of Hardware Resources: Parallel computing aims to make the best use of available hardware by distributing tasks efficiently across multiple processors, cores, or machines. This avoids underutilization of computing resources, especially in multi-core processors, clusters, and supercomputers.

- **Solving Complex Problems**:
  High-Performance Computing (HPC) — parallel computing, e.g. for supercomputing applications (e.g. simulations of nuclear processes, quantum computers or bio informatics research where without the parallel approach we just would not be able to do it in any reasonable time with serial methods)

- **Cost-Effectiveness**
  : Energy Efficiency: Because parallel computing allows for work to be spread across systems, it decreases the energy required to perform large tasks as more activity is being done in parallel.

- **Data Parallelism and Task Parallelism**:
  Task Parallelism: Different independent tasks or parts of an algorithm can be executed in parallel, improving efficiency by overlapping tasks.

Parallel computing Types

There are generally three types of parallel

1. **Bit-level parallelism**: The form of parallel computing in which every task is dependent on processor word size. In terms of performing a task on large-sized data, it reduces the number of instructions the processor must execute. There is a need to split the operation into series of instructions. For example, there is an 8-bitprocessor, and you want to do an operation on 16bit numbers. First, it must operate the 8 lower-order bitsand then the 8 higher-order bits. Therefore, two instructions are needed to execute the operation. The operation can be performed with one instruction by a 16-bit processor.
   Example:- A 64-bit processor can perform operations on 64 bits of data at once, as opposed to a 32-bit processor, which can handle only 32 bits at a time.

2. **Instruction-level parallelism**: In a single CPU clock cycle, the processor decides in instruction level parallelism how many instructions are implemented at the same time. For each clock cycle phase, aprocessor in instruction-level parallelism can have the ability to address that is less than one instruction. The software approach in instruction-level parallelism functions on static parallelism, where the computer decides which instructions to execute simultaneously.
   Example:- Modern CPUs can fetch, decode, and execute multiple instructions in parallel, improving performance without increasing clock speed.

3. **Task Parallelism**: Task parallelism is the form of parallelism in which the tasks are decomposed into subtasks. Then, each subtask is allocated for execution. And, the execution of subtasks is performed concurrently by processors.
   Example:- One thread might be handle file I/O while another thread processor data in memory

Applications of Parallel Computing

1. Sciences and engineering
   - Quantum mechanics and material science
   - Computational fluid dynamics
   - Optimization and design space exploration

2. Graphics game Development
   - Physics Simulation
   - Game Development
   - Multi-threading
   - Dynamic Simulation

3. High Perfomance Computing
    - Cryptography and cybersecurity
    - Data analytics
    - Climate modeling
    - Astrophysics

4. Network and Distributed Systems:
    - Cloud computing
    - Network function virtualization
    - Content delivery networks
    - Client-server architecture

How parallel processing work?

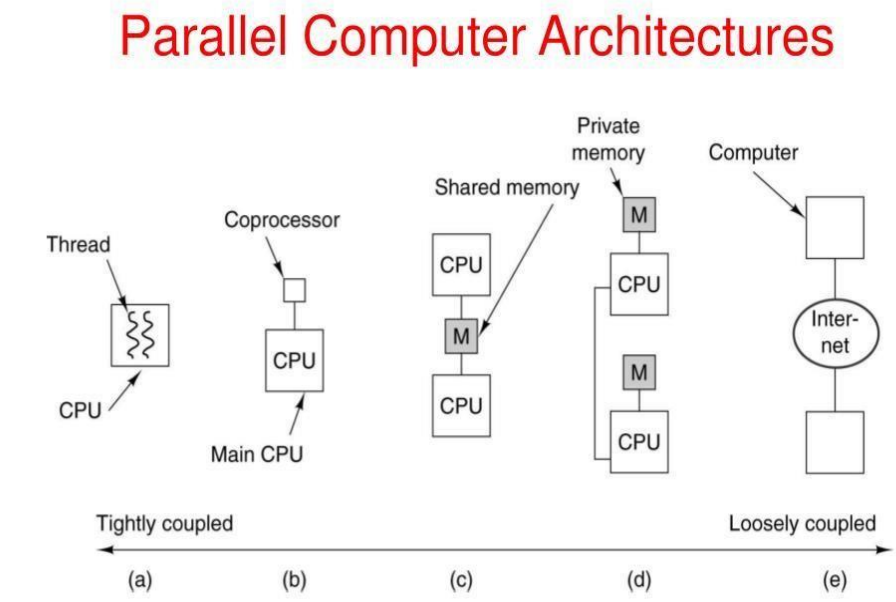
Parallel processing is a method in computing where multiple processors or cores are used to perform tasks simultaneously. The main task is broken down into smaller sub-tasks that can be executed independently. And To ensure correct results, parallel processes often need to coordinate and synchronize their operations, especially when they share resources or data. The Problems like race conditions occur when multiple processes try to access shared resources simultaneously.

Each processor solves its part of the overall computational problem. The software reassembles the data to reach the end conclusion of the original complex problem. It's a high-tech way of saying that it's easier to get work done if you can share the load. There are several ways to achieve the same goal.

Parallel Computer Architectures

Parallel computer architectures can be classified into different types

- Multi-Core Computing: Uses multiple cores within a single processor.
- Distributed Computing: Utilizes multiple computers connected by a network.
- Massively Parallel Computing: Involves a large number of processors.
- Cluster Computing: Combines multiple computers to work on the same task.



(a) On-chip parallelism. (b) A coprocessor. (c) A multiprocessor. (d) A multicomputer. (e) A grid.

❖ **On-chip parallelism**

- Parallelism In Chip Scale Parallelism refers to a single processor (chip) can carry out more than one task or operation at the same time with parallel execution of tasks or operations by using hardware capabilities like multiple cores as well as other features. This allows the chip to process more data all at once, resulting in increased performance. In modern computing, on-chip parallelism is important for achieving higher performance; it provides a means to spread the workload across multiple parallel units allowing processors to process different tasks more efficiently.

❖ **A coprocessor**

- In the design of parallel computer architecture, a coprocessor is a processor which supplements that processing coordinated with the main CPU in executing some jobs. Moreover, as the image above shows (Step 5), coprocessors help in parallel execution of operations (although with overheads due to communication or control operation) that enhances system performance, especially for computation-intensive applications by offloading few tasks from the main CPU.

❖ **Multiprocessor**

- Multiprocessor: A Computer system having more than one CPU that repeatedly works together to bring it about collective computation. These CPUs are usually integrated in the same system and hence they share resources like memory, input/output devices and buses. A multiprocessing system is a system where many CPUs copy out of control over various process frames or execution pieces, which could pressburdened in parallel.

❖ **Multicomputer**

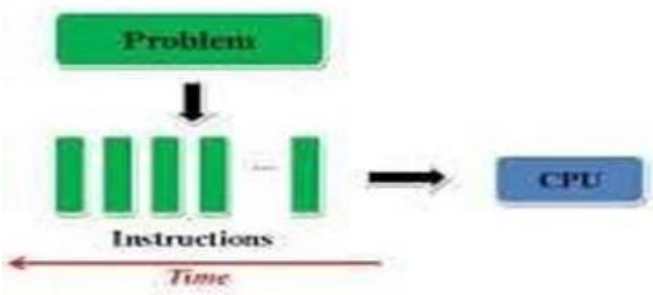
- A multicomputer is a parallel computing architecture where multiple independent computers (or nodes) work together to solve a common problem. Unlike multiprocessor systems, where multiple processors share a single memory space, each computer in a multicomputer system has its own private memory and operates independently. These individual computers communicate through a network to collaborate on a large, distributed task.

❖ **Grid**

- Grid in computing, and more particularly in Grid Computing, is a type of parallel and distributed system that enables the sharing, selection, and aggregation of resources distributed across multiple administrative domains based on their availability, capacity, performance, cost and/or users' quality-of-service requirements. Resources such as processing power, storage and data are shared in a grid between multiple independent organizations or locations, with the work being processed concurrently on different nodes (computers).

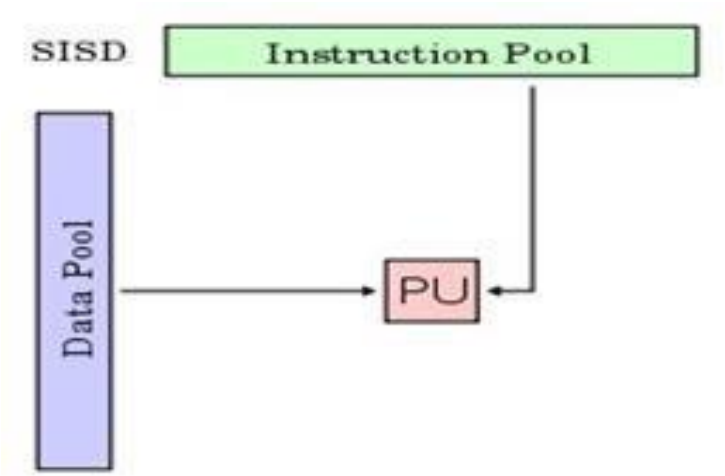
Serial computation

Traditionally software has been written for serial computation:- run on a single computer-instructions are run one after another-only oneinstruction executed at a time.

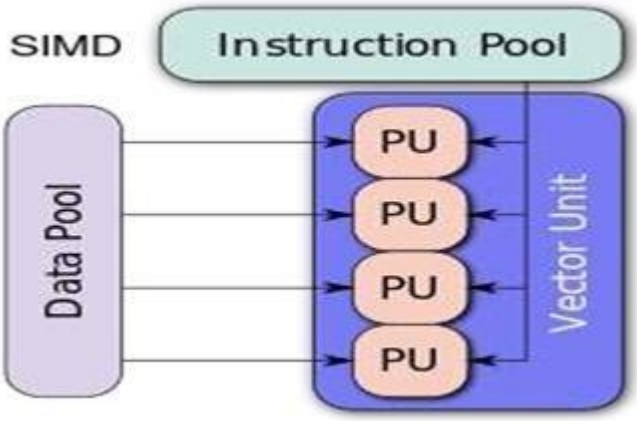


- **Concepts of Parallel Processing**

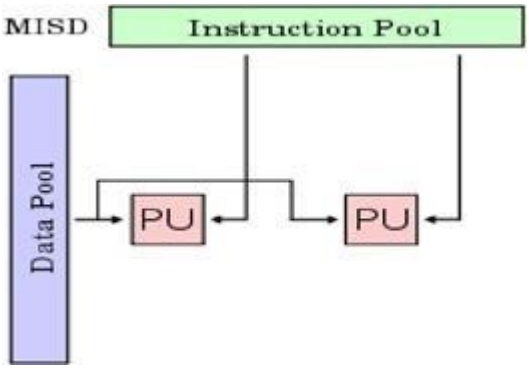
  SISD stands for Single Instruction Stream, Single Data Stream, this classification is part of Flynn's taxonomy and it describes a system where a single processor executes a single stream of instructions on a single stream of data. This is the most classical and traditional type of processing architecture, which is also known in other words as sequential computing system: one instruction to process after another, working on a single piece of data at a time**..**



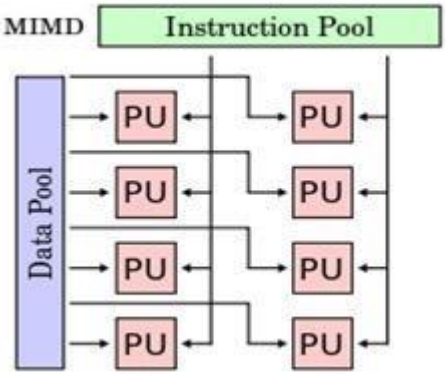
- **Single instruction stream, multiple data stream (SIMD).**SIMD as the name indicates is an architecture that splits like a tree from one instruction but have lots of inputs to it which gets parallely executed. Many data streams are processed under the control of a single instruction stream in this architecture. SIMD refers to a technique where the same operation is executed by a processor on multiple pieces of data concurrently, as you can imagine it being very efficient for operations on large amounts of concurrent data.



- **Multiple instruction, single data (MISD) Multiple** Instruction, Single Data (MISD) is another type of parallel computing architecture under Flynn's taxonomy. In this model, distinct processing units perform dissimilar operations on the same data stream. This type of code pattern, however, is quite rare and not particularly useful in the real world because most practical applications do not require more than a single instruction working on any given piece of data.



- **Multiple instruction, multiple data (MIMD)** Multiple Instruction, Single Data (MISD) is another type of parallel computing architecture under Flynn's taxonomy. In this model, distinct processing units perform dissimilar operations on the same data stream. This type of code pattern, however, is quite rare and not particularly useful in the real world because most practical applications do not require more than a single instruction working on any given piece of data.



Principles of Parallel Algorithm Design: he tenets of parallel algorithm design
lie in creating algorithms that can be performed on numerous processors or computational units at the same time. The goal of designing a parallel algorithm is to divide the workload among several processors so that performance scales with the number of processors and communication overhead is minimized.

well, there is an one important principle

1. Decomposition:

Breaking down the problem: The very first step in designing a parallel algorithm is to divide up the problem into smaller parts, or subproblems, each of which should be independent and (ideally) can be solved at the same time.

Types of Decomposition:

Decomposition: Divides the problem according to data. Various processors parallelly process different subsets of data. For example, matrix multiplication and sorting.

Task Decomposition: This occurs when the problem is broken down into separate tasks that can be performed individually. This is appropriate for applications with multiple tasks different (e.g., in pipelined processing).

Example: Output Data Partitioning:

Example: Matrix multiplication (A × B = C)

Partition the output matrix C into four tasks: c11, c12, c21, and c22

Each task computes a portion of the output matrix independently **2 Concurrency Identification:**

Concurrent Independent Operation: Operations can be executed in parallel without interfering with each other. It seems clear that we can run some sections of the algorithm concurrently to exploit this kind of parallelism.

Maximize Concurrency: the idea is to run as many operations as possible concurrently for performance benefits.

3 **Task Assignment:**

The Problem Decomposition: As mentioned above, work must be partitioned across the processors and You have to do it in a way so that every processor has an equally balanced load. It is all about reducing any processor to spend idle time, and no bottleneck should be there.

Load balancing: For overall good performance, the work should be distributed evenly so that no processor is too slow and therefore no other processor can do more.

Static Load Balancing: The division of work is done up front and does not change.

Dynamic Load Balancing: Striving to assign new tasks dynamically as processor finish their current work.

4 ***Minimizing Communication:***

Overhead: Example overhead in parallel computing is where processors may need to communicate with each other to exchange intermediate results. For our paralleldatagen() applications, we need to avoid over-communicating, as this can cause delays between processors.

Data Locality: Place tasks that use the same data on a single processor when feasible to limit interprocessor communication

5 **Data Dependencies:**

Dependency Management: For a tent to be assembled, there are certain tasks which need to be dependent on the completion of others. This limits the parallelism available for those dependencies. That algorithm is also built to minimize or control dependencies.

Types of Dependencies:

Data Dependency: I f one task needs the output that is produced by another task.

Control Dependency: The situation when the execution of a task are controlled by the outcome of the previous one.

➢ **Scope of Parallelism**

• Traditional architectures crudely consist of processor, data path and memory system.

All of these are performance bottlenecks in and of themselves.

• The corresponding manifestation of parallelism in these 4 areas is content addressed at important length by Parallelism.

• Different applications utilize different levels of parallelism – E.g., data intensive applications use high

server application high aggregate network bandwidth, scientific application high point-to-point stream uniform level of service and other types of applications. often use high processing and memory system performance.

CONCLUSION

Parallel computation is a very powerful technique that enables the execution of several functions or instructions at the same time, providing great improvements in computational efficiency and speed. This review paper provides overview principles, architectures, and applications of parallel computing which include Flynn's taxonomy, SIMD, MISD, and MIMD architectures together with parallel algorithm design principles. This paper extends towards the topic of why parallel computing is so important across various domains of sciences, engineering, graphics, and high-performance computing.

The examples of references provided herein are not all of them. However, the references that could be used in the review paper would depend on the content and focus that could be used in the paper. It could be a good starting point for further research and discovery of the topic.

Reference

- Flynn's Taxonomy: M. J. Flynn, "Some Computer Organizations and Their Effectiveness," IEEE Transactions on Computers, vol. C-21, no. 9, pp. 948-960, September 1972.
- SIMD, MISD, and MIMD architectures: H. S. Stone, "Introduction to Computer Organization and Data Structures," McGraw-Hill, 1972.
- Parallel algorithm design principles: A. Grama, A. Gupta, G. Karypis, and V. Kumar, "Introduction to Parallel Computing," Addison-Wesley, 2003.
- "A Survey of Parallel Computing" by M. J. Flynn (1972)
- “The Impact of Robot and AI on Human Workforce
- "Parallel Computing: A Survey" by D. J. Evans (1982)
- "Parallel Algorithms for Scientific Computing" by G. A. Geist and V. S. Sunderam (1992)