Introduction to HPC

- Processor (CPU) is the active part of computer, which does all work of data manipulation and decision making.
- Datapath is the hardware that performs all the required data processing operations for eg: ALU, buses etc.
- Control is the hardware that tells datapath what to do in terms of switching, operation selection, data movement between ALU components etc.

Motivating Parallelism.

→ Developing parallel hardware & software is more time & effort intensive.

→ Standardized parallel programming environments, solutions and libraries have reduced the time to achieve parallelism.

→ Speed of an application is influenced by factors beyond processor, memory speed, disk speed and network speed.

→ Parallel computing offers Scalability, allowing system to handle larger workloads by adding more processors or nodes. (Massive datasets and complex computations).

Need.

-> Performance Enhancement:.
- Significantly ~~increases~~ improves the performance and execution speed of application by dividing tasks into smaller, manageable chunks and processing them simultaneously.

-> Handling Complex tasks:
- AI/ML algos involve complex computations and massive dataset

-> future-proofing:
-> Embracing parallel computing prepares system for the future, where the focus is shifting from increasing clock speeds to optimizing parallelism.

-> Overcoming physical limitations:.
- Uniprocessor systems face physical limits in terms of clock speeds and power consumption. Parallel computing provides a way to achieve higher performance without encountering these constraints.

-> Energy efficiency. across multiple processors
- By distributing workloads, energy is utilised efficiently.

-> Real-time processing:
- Video processing, financial trading and robotics require real time responses. Parallel computing enables the swift processing of large amounts of data in real-time, ensuring timely decision-making and actions.

Moore's Law - 1975.

-> Moore's Law states that the computer chip complexity doubles approximately every year.

-> Computer chips gets more complex & powerful.

-> Components like transistors double in each year.

-> It has lead to faster, smaller and more powerful computers.

Short term Prediction: Doubling components per chip will continue.

Long term Prediction: Uncertain but remains constant for 10 years.

Two year Doubling: "No of transistor on a micro chip doubles every two years".

Moore shifted his focus due to:

- Physical Limits reached by 2020.
- Cooling transistors became challenging.
- Shift to bigger dies and finer dimensions.

* Computational Law Argument.

"Bigger dies and finer dimensions".

- New doubling will happen after 18 months after 1975

-

* Memory / Disk Speed Arguments

- Processor clock speed = 40% per year.
- DRAM access time = 10% per year.
- Process clock rates increased faster than DRAM access times, causing performance bottlenecks.

Sol
- Principle of locality: Locality guides parallel also design.

* Data Communication Argument.

- Internet is seen as a large computing platform.
- used in applications like SETI@Home (Extraterrestrial Intelligence) and Folding@Home (Disease research)
- In some tasks including databases and data mining, data volume is too large.

Pipelining.

→ Pipelining is a technique used in advanced microprocessor to increase instruction execution efficiency by overlapping the execution of multiple instructions.

→ Concurrent execution: In pipelining, the microprocessor starts executing a second instruction before the first one has been completed. This allows several instructions to be in different stages of processing simultaneously.

Pipelining Structure

Ins-1	IF	ID	OF	OE	OS
Ins-2	IF	ID	OF	OE	OS
Ins-3	IF	ID	OF	OE	OS

IF - Instruction fetch
ID - Instruction Decode.
OF - Operation fetch.
OE - Operation Execute.
OS - Operation store.

* Pipelining has some limitations:

1) Speed of Pipeline is eventually limited by the slowest stage.

2) Conditional jump occurs in every 5th-6th instruction.

3) Penalty of misprediction grows with the depth of a pipeline.

* One solution is to use multiple pipelines *.

* Superscalar Execution.

1) True Data Dependency: The result of one operation is an input to next.

2) Resource dependency: Two operations require the same resource.

3) Branch Dependency: Scheduling instructions across conditional branch statements cannot be done deterministically a-priori.

Issue Mechanism.

(1) In simpler model, instructions models can be issued only in the order in which they are encountered. That is if the second instruction cannot be issued because it has a data dependency with the first, only one instruction is issued in the cycle. This is called in-order issue.

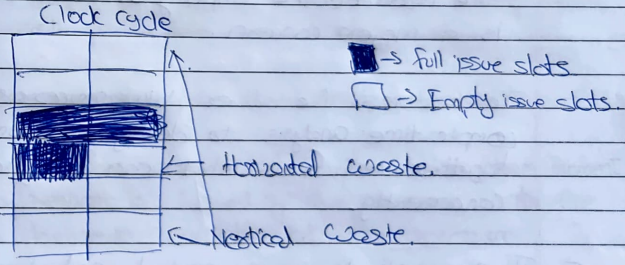
(2) In a more aggressive model, instructions can be issued out of order. In this case, if the second instruction has data dependencies with the first, but the third instruction does not, the first and third instructions can be co-scheduled. This is also called dynamic issue.

* Efficiency Considerations.

- Not all functional units can be kept busy all the times.

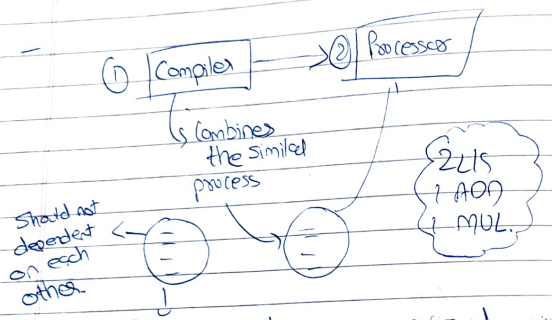
- If during a cycle, no functional units are utilised, this is referred to as vertical waste.

- If during a cycle, only some of the functional units are utilised, this is referred to as horizontal waste.



VLIW Architecture.

- Very Long Instruction Word.



These operations are passed as a word.

- The hardware cost & complexity of Superscalar schedules is a major concern.

- To address the issues, VLIW processors rely on compile time analysis to identify and bundle together instructions that can be executed concurrently.

- These instructions are packed and dispatched together, thus the name.

- 4way - 8way Parallelism.

✗ Advantages.

- Simplifies Dependency.
- Reduces Hardware complexity.

✗ Limitations

- Increases software/Compiler complexity.
- Cannot work in dynamic environment.
- Depended on functional units provided by processor.

Eg: Multiflow Trace, Intel IA64.

✗ Limitation of memory system performance.

- Bottleneck: for many applications the memory system, not the processor speed, becomes the limiting factor in overall performance. Data fetching is slow.

- Memory Performance Parameters: Latency & Bandwidth.

- Latency: Latency refers to the time taken from moment memory request is issued by the processor until the data becomes available at the processor.

- Bandwidth: Bandwidth is the rate at which data can be transferred or pumped from memory system to the processor.

Impact on performance: Memory stalls arise, processor must wait for data, causing performance slowdown.

* Flynn's Classification. *

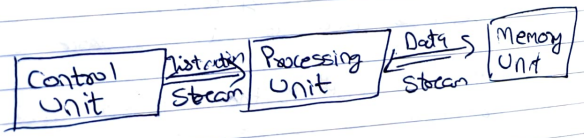
- It is a way of organising multiprocessor system.
- It's based on the notion of stream of information.
- Two types of information flow into a processor : Instruction and Data.
- They can be either single or multiple.

* Computer Architectures can be categorised into 4 categories.

SISD - Single Instruction Single Data stream.
SIMD - Single Instruction multiple Data stream.
MISD - Multiple Instruction Single Data stream.
MIMD - Multiple Instruction Multiple Data Stream.

* SISD

- In SISD, one instruction runs on one processor on the data stored on single memory.
- Uniprocessor comes in this category.

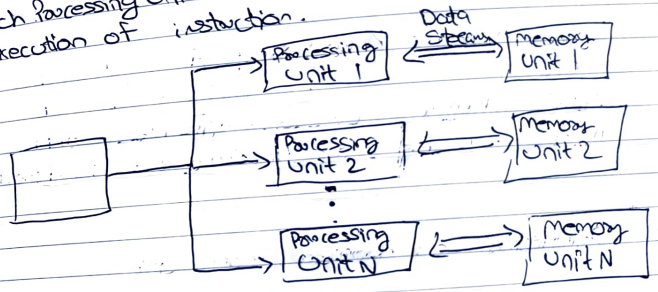


- Instruction are executed in sequential order.
- This architecture is widely used in sequential computers.
- Many conventional computers follow this Architecture.
- Clock defines the speed of SISD.

* SIMD

In SIMD, one instruction control multiple simultaneous execution of no. of processing elements on lock step basis.

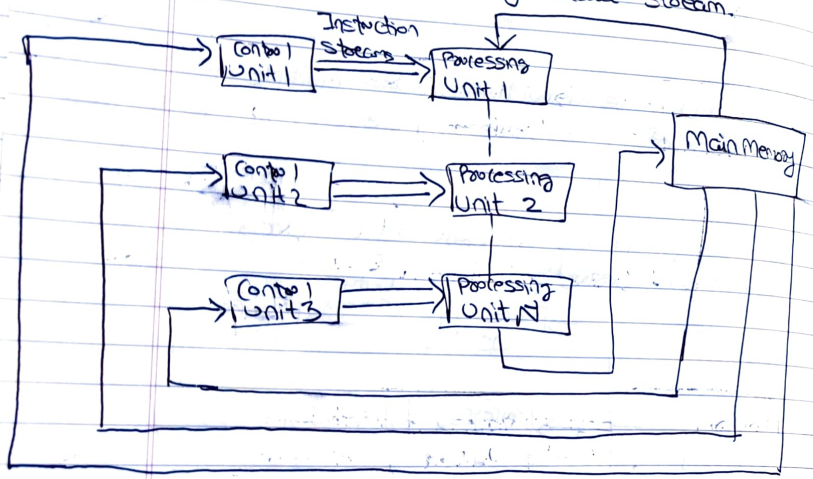
- Each Processing unit has associate data memory for execution of instruction.



- Scientific computing based on SIMD Architecture.
- Example of this model are Vectors and Arrays.

* MISO

- In MISO, Here multiple processor executes different instructions with single data stream.

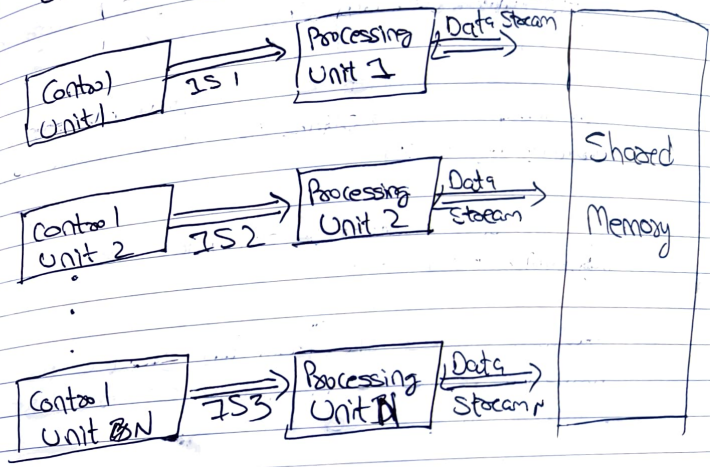


- Architecture not completed commercially.

A = Sin A
B = Cos A
C = Tan A.

* MIMD

In MIMD, Here multiple processor executes different instructions with multiple data streams.



- Shared memory in MIMD system is Symmetric multiprocessor system.

- Different processor takes instructions and data from common shared memory.

- With the use of buses, multiple processor executes program along with the shared memory.

Analytical Modeling

- A sequential algorithm is evaluated by its runtime.
- The asymptotic runtime of a sequential program is identical on any serial platform.
- The parallel runtime of a program depends upon input size, No. of processes & communication parameters.
- ∴ An algorithm must be evaluated using its underlying platform/Architecture.
- Wall clock Time = the time from the start of the first processor to the stopping time of the last processor in a parallel ensemble.

Overhead Problems in Parallel Programs.

- Interprocess interactions = Processes working on any non-trivial parallel problem will need to talk to each other.
- Idling = Processes may become idle because of load imbalance, Synchronization or serial components.
- Excess computation - This is computation not performed by serial version. This might be because serial algorithm is difficult to parallelize.

Let T_s be execution time of serial program.

Let T_p be execution time of parallel program.

Let T_{all} be total time collectively spent by all the processing elements.

→ We observe that $T_{all} - T_s$ is the total time spend by all processes combined in non-useful work. This is called total overhead.

→ Conversely, the total time spent collectively by all processing elements cost.

$T_{all} = p T_p$ (p → No. of Processes)

∴ The overhead function is therefore given by.

$$T_o = pT_p - T_s$$

- SpeedUp :- It is ratio of Is/Tp.

→ S ratio of time required of executing a problem by a single processor to the time for executing same problem by a parallel system comprising of p components

Efficiency :- It is a measure of fraction of time for which a processing element is usefully employed

$$E = \frac{S}{P}.$$

- following the bounds on speedup, efficiency can be as low as 0 and as high as 1.

HPC Unit-2:
Designing Parallel Algorithms

* Parallel Algorithm Development.

- Algorithm development is a critical component of problem solving using computers.
- A sequential algorithm is essentially a recipe of basic steps for solving a given problem using a serial computer.
- Similarly a parallel algorithm is a recipe that tells us how to solve a given problem using multiple processors.

* Designing Steps that are most :-

- Identifying portions of work that can be performed concurrently
- Mapping concurrent pieces of work onto multiple processors.
- Distributing input, output and intermediate data associated with the program.
- Managing access to data shared by multiple processors
- Synchronising the processes at various stages of the parallel program execution.

Preliminaries:

1. Decomposition.

→ The process of dividing a computation into smaller parts, some or all of which may potentially be executed in parallel is called decomposition.

2. Task.

→ Tasks are programmer-defined units of computation into which the main computation is subdivided by means of decomposition.

→ Tasks may be of same or different or even indeterminate sizes.

3. Dependency Graph.

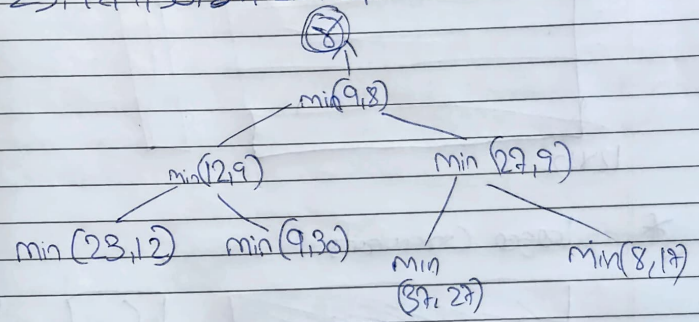
→ Tasks may use data produced by other tasks and thus may need to wait for these tasks to finish execution. An abstraction used to express such dependencies among tasks is known as task-dependency graph.

- It is a directed acyclic graph in which Node represents task and the edges represents the dependencies among tasks.

- The task corresponding to a node can be executed when all tasks connected to this node by incoming edges have completed.

Eg:- Task dependency Graph.

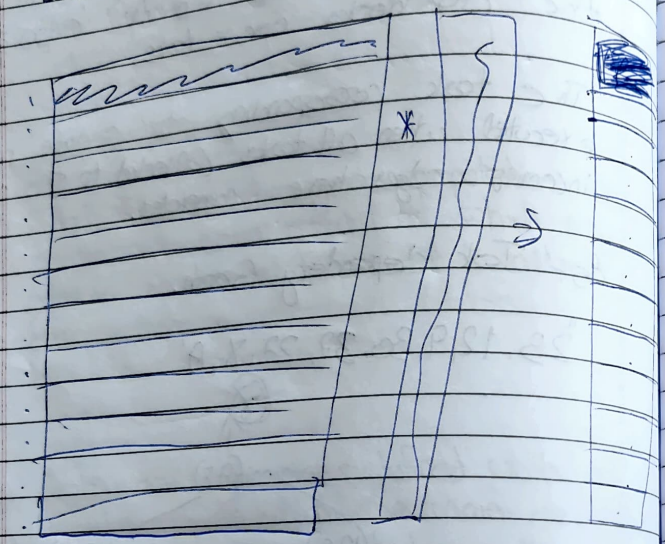
"23, 12, 9, 30, 37, 27, 8, 17"



4. Granularity.

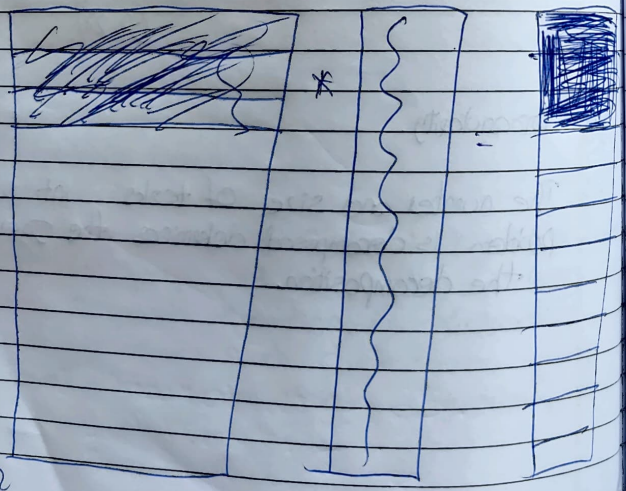
→ The number and size of tasks into which a problem is decomposed determines the granularity of the decomposition.

* Smaller Granularity.



12X12

* Larger Granularity.



12X12

fine-grained decomposition: Large no. of small tasks.

Coarse-grained decomposition: Small no. of large tasks.

* Concurrency: Executing multiple tasks simultaneously is called Concurrency.

Max. Degree of Concurrency:

→ The max number of tasks that can be executed simultaneously in a parallel program at any given time is known as maximum degree of Concurrency.

In most cases, max degree of concurrency is less than total no of tasks due to dependencies among the tasks.

→ Avg. Degree of Concurrency

It is the average number of tasks that can run concurrently over the entire duration of execution of program.

Note: Both the maximum and average degrees of Concurrency usually increase as the granularity become finer (smaller).

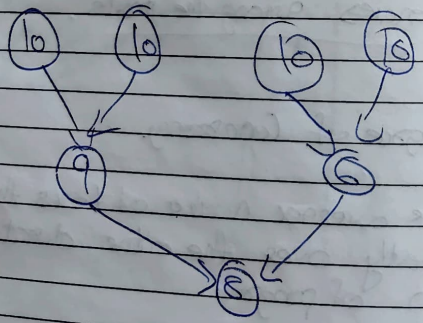
Critical path:- The longest directed path between any pair of start and finish nodes is known as Critical Path.

Critical Path length:- The sum of weights of nodes along Critical path is known as Critical path length, where the weight of a node is size of amount of work associated with the corresponding tasks.

Avg. Deg of Concurrency =

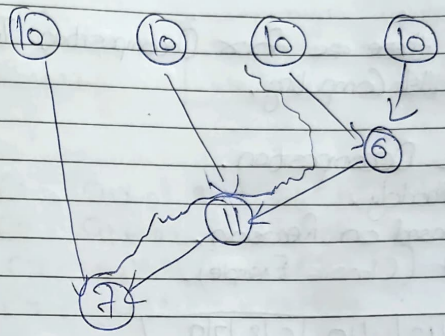
Total Amount of work

Critical Path Length.



Total Amount of work = 63
Critical Path Length :- 27
Avg. Degree of Concurrency :- 2.33

(Max weighted)



Total Amount of work = 64.
Critical Path length = 34
Avg. Deg of Concurrency = 64/34 = 1.88.

*. Task interaction Graphs

The graph of tasks (nodes) and their interactions (dependencies) (edges) is referred to as task interaction Graph.

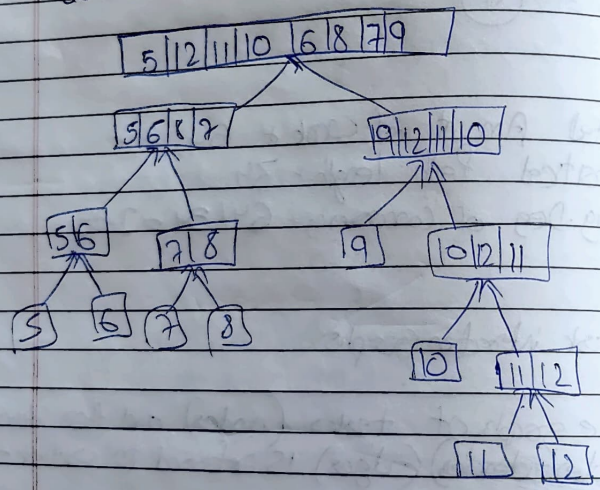
→ Represent Data dependencies

Task dependency represent control dependencies.

Decomposition Techniques.

→ Broadly there are four Decomposition Techniques in Parallel Computing.

① Recursive Decomposition.
- DnC Methoddy.
- It is based on Recursion.
- Quick Sort (Classic Example).



② Data decomposition.

- Identify data on which computations are performed
- Partition this data across various tasks.
- This partitioning induces a decomposition of the

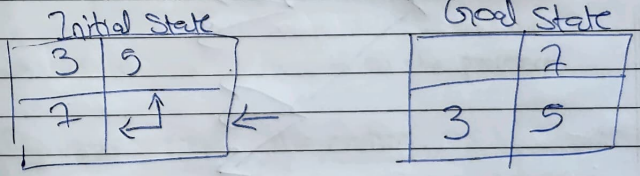
a_1	a_2	*	b_1	b_2	=	c_1	c_2
a_3	a_4		b_3	b_4		c_3	c_4

$C_1 = a_1 * b_1 + a_2 * b_3$.
$C_2 = a_1 * b_2 + a_2 * b_4$.
$C_3 = a_3 * b_1 + a_4 * b_3$
$C_4 = a_3 * b_2 + a_4 * b_4$.

③ Exploratory Decomposition.

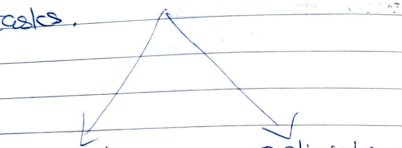
- In many cases, the decomposition of Problem goes hand-in-hand with execution. These problems typically involve the search (exploration) of a state space of solutions.

- Eg:- Theorem proving, Game playing etc.

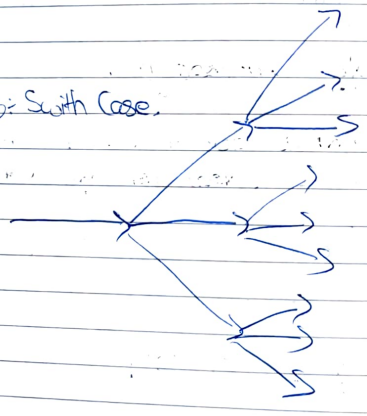


4) Speculative Decomposition.

- In some application, task dependencies are not known a-priori.
- For such applications, it is impossible to identify tasks.



Eg:- Swith Case.



Characteristic of Tasks.

→ Once a problem has been decomposed into dependent tasks, the characteristics of these tasks critically impact choice and performance of parallel algorithms. Relevant task characteristics include:

- Task generation.
- Task Sizes.
- Size of data associated with tasks.

X Task generation.

- Static task generation.

→ Concurrent tasks are identified a-priori. Typical matrix operations, graph algorithms, image processing applications fall in this class.

→ Which task to be mapped to the processor.

- Dynamic task generation.

→ When execution of algorithm is going on that time we get to know what task is to be mapped to the processor.

→ Ese:- Game-playing.

* Task Sizes

- Task sizes may be uniform or Non-uniform

* Size of data associated with the task.

- May be small or large when viewed in the context of the size of the task.

- A small context of a task implies that an algorithm can easily communicate this task to other processes dynamically (eg: 15puzzle).

- A large context ties the task to a process/ may attempt to reconstruct the context of another processes as opposed to communicating the context of the task.

* Characteristics of Inter-Task interaction.

1. Static vs dynamic. (Is during execution.) (At Particular Time-Interval)
2. Regular vs Irregular (Practical pattern / Not Practical Pattern)
3. Read only vs Read-write.
4. One-way vs Two-way.

Data
Work
Synchronisation information

* Mapping Techniques can be broadly be categorised into two categories:

1. Static mapping: Static mapping techniques distribute the tasks among processes a-priori to the execution of program.

2. Dynamic mapping: Dynamic mapping techniques distribute the work among processes during the execution of the algorithm.

Schemes for Static Mapping:-

3) Mapping Based on Data Partitioning.

1. Block distribution Schemes.
2. Cyclic and Block Cyclic Distribution.
3. Randomised Block
- Graph

2 Mappings Based on Task Partitioning.

3 Hierarchical Mappings.

Schemes for Dynamic Mapping.

- Master Process
- Slave Processes