HPC

Assignment No: 02

Q1) Explain any Three data decomposition techniques with examples.

→ Data decomposition techniques are used in parallel computing to divide a computational task into smaller parts that can be processed concurrently by multiple processors or nodes.

Here are three common data decomposition techniques.

① Domain Decomposition:

Domain decomposition involves partioning the computational domain into smaller subdomains, with each subdomain assigned to a processing unit.

Example:
In a finite element simulation of a structure analysis problem, the domain can be divided into smaller elements.

② Block Decomposition:

Block decomposition involves dividing the data into fixed-size blocks or chunks, with each block processed by separate processing unit.

Example:

In image processing tasks, such as applying filters or performing convolution operations, the image can be divide into blocks of pixels.

③ wavefront Decomposition:

wavefront decomposition is often used in problems where data dependencies exist along specific patterns, such as in certain types of iterative alogrithms or dynamic programming problems.

Example:

Consider the dynamic programming approach to solve the shortest path problem in a gricl graph.

Q2) Explain different characteristics of tasks.

→ in the context of task-based parallel computing, task refer to individual units of works that can be executed concurrently by multiple processing units, such as cpu or GPUs.

① Granularity:

Granularity refers to the size or level of detail of task. Task can vary in granularity from fine-grained task, which are small and execute quickly.

② Data Dependencies:

Task may have dependencies on input data or the output of other tasks. These dependencies determine the order in which tasks much be executed and may restrict the degree of parallelism.

③ computational intensity:

computational intensity refers to the ratio of computation to data movement or communication. Task with high computational intensity require relatively more computation compared to data movement.

④ Task lifespan:

The lifespan of a task refers to the duration for which it remains active or relevant during program execution. short lived task are created and completed quickly, often within a few iterations of small section of code.

Q3) Explain classification of Dynamic mapping techniques.

→ Dynamic mapping techniques, also known as dynamic task scheduling or load balancing techniques, are used in parallel computing to efficiently assign task to processing units at runtime.

Here are some common classifications.

① work stealing:
work stealing is a popular dynamic mapping technique used in system with a shared memory architecture, such as multi-core processor or distributed memory system.

② Task queuing:
Task queuing techniques involves maintaining a global or distributed task queue from with processing units retrieve tasks for execution.

③ Dynamic Task Decomposition:
Dynamic task decomposition techniques involve dynamically decomposing computational tasks into smaller subtasks based on workload distribution, system

resources, or other runtime factors.

⑤ feedback-based techniques

feedback-based techniques use feedback mechanism to adaptively adjust task assignment based on performance metrics, system utilization, or workload characteristics.

(Qn) what are mapping & techniques for load balancing? Explain at least two mapping techniques.

→ Load balancing techniques aim to distribute computational workload evenly among processing units to maximize resources utilization and minimize execution time.

mapping techniques are a subset of load balancing techniques that focus on assigning tasks to processing units.

Here are two common mapping techniques for load balancing.

① Static mapping:

static mapping involve pre-assigning task to processing units before the execute of parallel program begins.

Task assignment are typically based on static information such as task characteristics.

② Dynamic mapping :
Dynamic mapping techniques adjust task assignments at runtime based on the current system state, workload distribution and resource availability.

Q5) Explain any three parallel algorithm models with suitable examples.

→ Parallel algorithm models provide frameworks for designing and analyzing algorithms that exploit parallelism to achieve improved performance on parallel computing architecture. Here are explanations of three common parallel algorithm models along with suitable examples.

① Data parallelism :
In data parallelism, the computational task is divided into multiple independent units of work, and each unit operate on different portions of the input data concurrently.

Example:

Matrix multiplication is a classic example of a problem that can be solved using data parallilism.

② Task parallelism:

In task parallism, the computation task is divided into multiple distinct subtask, and each subtask is executed concurrently by different processing units.

Example:

Merge sort is a sorting algorithm that can be parallelized using task parallel. In merge sort, the input array is recursively divided into smaller subarrays.

③ pipeline parallelism:

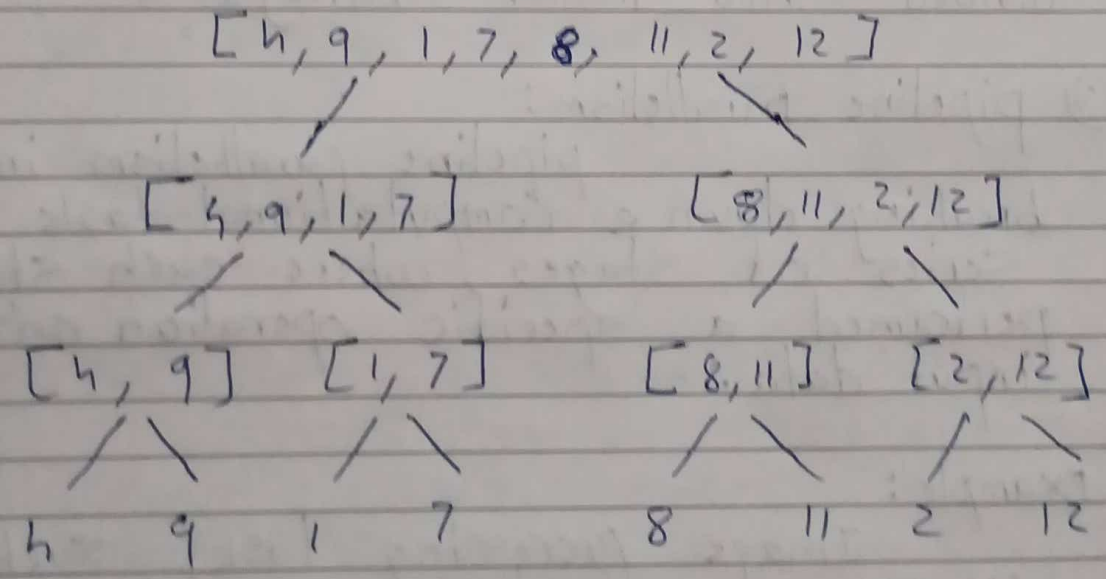
pipeline parallelism involves breaking down a computational task into a series of stages, where each stage performed a specific operation on the input data.

Example:

Image processing tasks, such as filtering or feature extraction, can be paralletized using pipeline parallelism.

Q6) Draw the task-dependency graph for finding the minimum number in the sequence {4, 9, 1, 7, 8, 11, 2, 12} where each node in the tree represents the task of finding the minimum of a pair of numbers. Compare this with serial version of finding minimum number of an array.

→ To draw the task-dependency graphy for finding the minimum number in the sequence {4, 9, 1, 7, 8, 11, 2, 12}, we'll contruct a binary tree where each node represent the task of finding the minimum of pair of numbers.



Comparing the two approaches:

- The parallel approach divides the task of finding the minimum into smaller subtasks, allowing for potential parallel execution on multiple processor or cores.

- The serial approach iterates through the array sequentially, performing comparisons one after another, which can be least efficient on parallel architecture.

8(i) Explain the methods for containing iteraction overloads.

→ Interaction overheads refer to the costs associated with communication, synchronization and coordination between parallel tasks or ~~para~~ processing unit in a parallel computing system.

① minimize communication:
One of the most effective ways to reduce iteraction overheads is to minimize the amount of communication between processing units.

② Overlap communication with computation:

Overlapping communication with computation can help hide communication latency and overlap interaction overheads wi useful computation.

③ optimize synchronization:

Synchronization overheads arise from the the need coordinate the execution of parallel task to ensure correctness and consistency.

Q8) write a short note on anomalies in parallel algorithm.

→ Anomalies in parallel algorithms refer to unexpected or undesirable behaviors that arise due to the concurrent execution of multiple tasks or threads in parallel computating systems.

Here are some common types of anomalies in parallel algorithms.

① Race conditions :

Race condition occur when the outcome of a computation depends on the non-deterministic interleaving of execution paths of concurrent task.

② Deadlocks :

Deadlocks occur when two or more tasks are unable to proceed because each is waiting for the other to release a resource or complete an action.

③ Starvation :

Starvation occurs when a task is unable to make progress or complete its execution due to resource contention or scheduling policies favouring other tasks.

④ Data Race :

Data races occurs when multiple tasks access shared concurrently, and at least one of the accesses is a write operation, without proper synchronization.

Q9) Differentiate between sequential and parallel computational complexity!

→

Sequential computers	parallel computers
- Are uniprocessor system (1 CPU)	- Are multiprocessor system (many CPU)
- Can Execute 1 instruction at a time	- Can execute several instruction at a time.
- speed is limited	- No limitation on speed
- It is quite expensive to make single CPU faster	- less expensive if we use larger number of fast processors to achieve
- Area when it can be used (colleges, labs	- Area when it can be used in wide networks.
ex pentium pc	Ex Cray 1, Cray-xmp