**📚 Quick Oral Answers:**

1. **What is HPC?**

HPC is the use of supercomputers and parallel processing to solve large problems faster than a normal computer.

1. **What is parallel computing?**

Parallel computing means solving a problem by dividing it into smaller parts and executing them simultaneously using multiple processors.

1. **Difference between parallel and distributed computing?**

Parallel computing uses shared memory within one machine. Distributed computing uses separate machines connected over a network.

1. **What is OpenMP?**

OpenMP is an API that supports multi-threaded programming in shared memory systems.

1. **What is MPI?**

MPI stands for Message Passing Interface, used for communication between processes in distributed memory systems.

1. **What is Amdahl’s Law?**

Amdahl’s Law states that the speedup of a program is limited by the portion that cannot be parallelized.

1. **What is speedup?**

Speedup is the ratio of the time taken by one processor to the time taken by multiple processors.

1. **What is efficiency?**

Efficiency is the speedup divided by the number of processors, showing how well resources are utilized.

1. **What is shared memory?**

In shared memory, all processors access the same physical memory space.

1. **What is distributed memory?**

In distributed memory, each processor has its own private memory and processors communicate via messages.

1. **Why is synchronization needed?**

Synchronization prevents conflicts when multiple processors access or modify shared data.

1. **What are race conditions?**

Race conditions happen when the output depends on the timing of threads accessing shared resources.

1. **What is load balancing?**

Load balancing ensures all processors have roughly equal work, avoiding idle processors.

1. **Static vs Dynamic Scheduling in OpenMP?**

Static assigns tasks at compile time; dynamic assigns tasks during runtime depending on availability.

1. **What is scalability?**

Scalability measures how well a parallel system's performance improves as more processors are added.

1. **What are SIMD and MIMD?**

SIMD: Single Instruction Multiple Data — same operation on multiple data points.  
MIMD: Multiple Instruction Multiple Data — different processors can perform different operations.

1. **What does #pragma omp parallel do?**

It tells the compiler to run the following block of code in parallel using multiple threads.

1. **What is a critical section in OpenMP?**

A part of the code that only one thread can execute at a time to avoid conflicts, declared with #pragma omp critical.

1. **What is a thread?**

A thread is the smallest unit of execution in a program that can run independently.

1. **What is a process?**

A process is an independent program running in its own memory space, possibly containing multiple threads.

1. **Difference between thread and process?**

Threads share memory within a process; processes have separate memory.

1. **What is multithreading?**

Multithreading is the ability of a CPU to execute multiple threads concurrently.

1. **What are fork and join in OpenMP?**

Fork: Main thread creates new threads.  
Join: All threads complete and join back to the main thread.

1. **What is deadlock?**

Deadlock occurs when two or more threads are waiting for each other to release resources, causing a standstill.

1. **What is barrier in OpenMP?**

A barrier forces all threads to wait until each has reached the barrier point before continuing.

1. **What is critical path in parallel computing?**

The longest sequence of dependent tasks that determines the minimum execution time.

1. **What is cache coherence?**

Cache coherence ensures that all processors see a consistent view of shared memory.

1. **What are parallel directives in OpenMP?**

Instructions like #pragma omp parallel, #pragma omp for, #pragma omp sections to control parallel behavior.

1. **What is fine-grained vs coarse-grained parallelism?**

Fine-grained: Tasks are small and frequent communication.  
Coarse-grained: Tasks are larger, less communication needed.

1. **What is load imbalance?**

When some processors have more work than others, causing inefficiency.

1. **Explain reduction operation in OpenMP.**

Combining results from multiple threads safely into a single result, using reduction clause.

1. **What is task parallelism vs data parallelism?**

Task parallelism: Different tasks run in parallel.  
Data parallelism: Same task runs on different chunks of data.

1. **What is hyperthreading?**

Technology where a single physical CPU core acts like two logical cores, running multiple threads.

1. **What is latency vs bandwidth?**

Latency: Time to start communication.  
Bandwidth: Amount of data transferred per unit time.

1. **Explain false sharing.**

When multiple threads access different data in the same cache line, causing unnecessary cache invalidations.

1. **What is profiling in HPC?**

Profiling is measuring the performance of a program to find bottlenecks and optimize.

1. **What is heterogenous computing?**

Using different types of processors (CPU + GPU) together for faster computation.

1. **What is SIMD vectorization?**

Performing a single instruction on multiple data elements at once for better performance.

1. **What is NUMA?**

Non-Uniform Memory Access — memory access time depends on memory location relative to a processor.

1. **What is CUDA?**

CUDA is a parallel computing platform by NVIDIA for using GPUs for general-purpose computing.

**Commands to Install OpenMP in Ubuntu**

1. First, update your system :- sudo apt update

2. Install GCC and G++ (they support OpenMP): sudo apt install build-essential

3. Check if G++ is installed correctly: g++ --version

**How to Compile a C++ Program Using OpenMP**

1.When you compile a C++ program that uses OpenMP, add the -fopenmp flag:

🡪 g++ -fopenmp your\_program.cpp -o your\_program

Ex. g++ -fopenmp dfs.cpp -o dfs

2. Then run it:

🡪 ./your\_program

Ex. ./dfs

**What is OpenMP?**

**OpenMP** (Open Multi-Processing) is an **API** (Application Programming Interface) that allows you to write programs that can **run on multiple CPU cores at the same time** — this is called **parallel programming**.

* It is mainly used with **C**, **C++**, and **Fortran** languages.
* OpenMP makes it **easy** to write **multi-threaded** programs using **simple #pragma commands** (special instructions for the compiler).