# Parallel Computing Questions and Answers

## 1. Explain the differences between serial and parallel code. [4 points]

Serial code executes instructions one after another on a single processor, whereas parallel code divides tasks into smaller sub-tasks that can be executed simultaneously across multiple processors or cores. Serial execution is simple but can be slow for large problems, while parallel execution improves speed and efficiency but requires synchronization and coordination.

## 2. What is the FLOPS and its usage in parallel computing? [2 points]

FLOPS stands for Floating Point Operations Per Second, a measure of computer performance. In parallel computing, FLOPS is used to evaluate the computational power of systems and compare efficiency across architectures.

## 3. Simply explain the computer architecture of von Neumann. [4 points]

The von Neumann architecture consists of a CPU, memory, input/output devices, and a bus system. It uses a stored-program concept where both data and instructions are stored in the same memory. The CPU fetches instructions sequentially, decodes, and executes them.

## 4. Simply explain: what is shared memory architecture and what is distributed memory architecture? [4 points]

Shared memory architecture allows multiple processors to access the same physical memory, enabling easy communication but requiring synchronization mechanisms. Distributed memory architecture assigns each processor its own local memory, and processors communicate by passing messages, which scales better but is more complex to program.

## 5. Please give the parallel programming models in common use and simply explain. [4 points]

Common parallel programming models include:  
- Shared Memory (e.g., OpenMP): Threads share memory and synchronize access.  
- Distributed Memory (e.g., MPI): Processes communicate via message passing.  
- Hybrid: Combines shared and distributed memory approaches.  
- Data Parallelism: Operations are applied simultaneously to elements of large datasets.

## 6. According to the lectures, how to evaluate your parallel performance practically? List the calculating formulas and explain. [5 points]

Parallel performance is evaluated by:  
- Speedup (S): S = T\_serial / T\_parallel  
- Efficiency (E): E = S / P, where P is the number of processors  
- Scalability: How performance changes as the problem size and number of processors increase  
Speedup shows performance gain, while efficiency shows how well processors are utilized.

## 7. Briefly explain the hybrid parallel programming model on current supercomputers (HPC clusters). [4 points]

Hybrid programming combines MPI (for inter-node communication in distributed memory) with OpenMP (for intra-node parallelism in shared memory). This model leverages both cluster-level and core-level parallelism, making it common in supercomputers.

## 8. Explain the elements of Flynn's taxonomy. [4 points]

Flynn's taxonomy classifies computer architectures:  
- SISD: Single Instruction, Single Data (traditional serial computing)  
- SIMD: Single Instruction, Multiple Data (vector processors, GPUs)  
- MISD: Multiple Instruction, Single Data (rare, fault-tolerant systems)  
- MIMD: Multiple Instruction, Multiple Data (common in multicore CPUs and clusters)

## 9. Explain the terms of nodes and cluster. [4 points]

A node is a single computing unit containing processors, memory, and storage. A cluster is a collection of interconnected nodes that work together as a single parallel computing resource.

## 10. Please list the three primary API components in OpenMP and give simple examples. [4 points]

The three primary components are:  
- Compiler Directives: e.g., #pragma omp parallel  
- Library Routines: e.g., omp\_get\_num\_threads()  
- Environment Variables: e.g., OMP\_NUM\_THREADS=4

## 11. Please write the steps to use gcc compiler to compile a C file “hello\_omp.c” with OpenMP flag and get the executable file “hello\_omp”. [3 points]

Commands on Linux bash shell:  
gcc -fopenmp hello\_omp.c -o hello\_omp  
export OMP\_NUM\_THREADS=4  
./hello\_omp

## 12. In HPC, there are two main ways of scaling a program. Please explain them. [4 points]

- Strong Scaling: Reducing execution time by adding processors while keeping the problem size fixed.  
- Weak Scaling: Increasing problem size proportionally with the number of processors to maintain efficiency.

## 13. When you run your serial code on multiple cores (CPUs), you can employ two different approaches. [2 points]

Two approaches:  
- Data Parallelism: Dividing data across cores so each core processes part of the data.  
- Task Parallelism: Dividing tasks or functions across cores.  
  
Two laws:  
- Amdahl’s Law: Speedup is limited by the serial portion of the code.  
- Gustafson’s Law: Increasing problem size allows near-linear speedup with more processors.