Unit 5

Question and Answer

1)Explain MultiCore architecture.

Multicore refers to an architecture in which a single physical processor incorporates the core logic of more than one processor. A single integrated circuit is used to package or hold these processors. These single integrated circuits are known as a die. Multicore architecture places multiple processor cores and bundles them as a single physical processor. The objective is to create a system that can complete more tasks at the same time, thereby gaining better overall system performance.

This technology is most commonly used in multicore processors, where two or more processor chips or cores run concurrently as a single system. Multicore-based processors are used in mobile devices, desktops, workstations and servers.

2)What is parallel computing?

Parallel computing refers to the process of breaking down larger problems into smaller, independent, often similar parts that can be executed simultaneously by multiple processors communicating via shared memory, the results of which are combined upon completion as part of an overall algorithm. The primary goal of parallel computing is to increase available computation power for faster application processing and problem solving.

Parallel computing infrastructure is typically housed within a single datacenter where several processors are installed in a server rack; computation requests are distributed in small chunks by the application server that are then executed simultaneously on each server.

There are generally four types of parallel computing, available from both proprietary and open source parallel computing vendors -- bit-level parallelism, instruction-level parallelism, task parallelism, or superword-level parallelism:

- Bit-level parallelism: increases processor word size, which reduces the quantity of instructions the processor must execute in order to perform an operation on variables greater than the length of the word.
- Instruction-level parallelism: the hardware approach works upon dynamic parallelism, in which the processor decides at run-time which instructions to execute in parallel; the software approach works upon static parallelism, in which the compiler decides which instructions to execute in parallel

- Task parallelism: a form of parallelization of computer code across multiple processors that runs several different tasks at the same time on the same data
- Superword-level parallelism: a vectorization technique that can exploit parallelism of inline code

Parallel applications are typically classified as either fine-grained parallelism, in which subtasks will communicate several times per second; coarse-grained parallelism, in which subtasks do not communicate several times per second; or embarrassing parallelism, in which subtasks rarely or never communicate. Mapping in parallel computing is used to solve embarrassingly parallel problems by applying a simple operation to all elements of a sequence without requiring communication between the subtasks.

The popularization and evolution of parallel computing in the 21st century came in response to processor frequency scaling hitting the power wall. Increases in frequency increase the amount of power used in a processor, and scaling the processor frequency is no longer feasible after a certain point; therefore, programmers and manufacturers began designing parallel system software and producing power efficient processors with multiple cores in order to address the issue of power consumption and overheating central processing units.

The importance of parallel computing continues to grow with the increasing usage of multicore processors and GPUs. GPUs work together with CPUs to increase the throughput of data and the number of concurrent calculations within an application. Using the power of parallelism, a GPU can complete more work than a CPU in a given amount of time.

3)Explain different types of parallelism

1. Bit-level parallelism –
   It is the form of parallel computing which is based on the increasing processor's size. It reduces the number of instructions that the system must execute in order to perform a task on large-sized data.
   Example: Consider a scenario where an 8-bit processor must compute the sum of two 16-bit integers. It must first sum up the 8 lower-order bits, then add the 8 higher-order bits, thus requiring two instructions to perform the operation. A 16-bit processor can perform the operation with just one instruction.
2. Instruction-level parallelism –
   A processor can only address less than one instruction for each clock cycle phase. These instructions can be re-ordered and grouped which are later on executed concurrently without affecting the result of the program. This is called instruction-level parallelism.
3. Task Parallelism –
   Task parallelism employs the decomposition of a task into subtasks and then

allocating each of the subtasks for execution. The processors perform the execution of sub-tasks concurrently.

4. Data-level parallelism (DLP) –

Instructions from a single stream operate concurrently on several data – Limited by non-regular data manipulation patterns and by memory bandwidth

4)Explain Amdahl's Law

It is often used in parallel computing to predict the theoretical speedup when using multiple processors.

Speedup-

Speedup is defined as the ratio of performance for the entire task using the enhancement and performance for the entire task without using the enhancement or speedup can be defined as the ratio of execution time for the entire task without using the enhancement and execution time for the entire task using the enhancement.

If Pe is the performance for entire task using the enhancement when possible, Pw is the performance for entire task without using the enhancement, Ew is the execution time for entire task without using the enhancement and Ee is the execution time for entire task using the enhancement when possible then,

Speedup = Pe/Pw

or

Speedup = Ew/Ee

Amdahl's law uses two factors to find speedup from some enhancement –

Fraction enhanced – The fraction of the computation time in the original computer that can be converted to take advantage of the enhancement. For example- if 10 seconds of the execution time of a program that takes 40 seconds in total can use an enhancement , the fraction is 10/40. This obtained value is Fraction Enhanced. Fraction enhanced is always less than 1.
Speedup enhanced – The improvement gained by the enhanced execution mode; that is, how much faster the task would run if the enhanced mode were used for the entire program. For example – If the enhanced mode takes, say 3 seconds for a portion of the program, while it is 6 seconds in the original mode, the improvement is 6/3. This value is Speedup enhanced. Speedup Enhanced is always greater than 1.

The overall Speedup is the ratio of the execution time:-

$$\text{Overall Speedup} = \frac{\text{Old execution time}}{\text{New execution time}}$$

$$= \frac{1}{\left((1 - \text{Fraction}_{\text{enhanced}}) + \dfrac{\text{Fraction}_{\text{enhanced}}}{\text{Speedup}_{\text{enhanced}}} \right)}$$