**Part I:**

**1.**

(1) Separating the work into discrete parts helps to solve the problem at the same time.

(2) Execute multiple program instructions at any time and in time

(3) It takes less time to solve the problem with multiple computing resources than with a single computing resource

**2.**

There are two models: shared memory model and distributed memory model.

Shared memory model: In this programming model, all tasks share a memory space, and the read and write of shared resources are asynchronous.

Distributed memory model: In this model, a set of tasks that use their own local memory during computation. Multiple tasks can reside on the same physical machine and/or across an arbitrary number of machines. Tasks exchange data through communications by sending and receiving messages. Data transfer usually requires cooperative operations to be performed by each process.

**3.**

Flynn classification is a kind of classification method with high efficiency and computer. In 1972, Michael J. Flynn was divided into instruction and data according to information stream. It can be divided into four types: Single instruction stream single data stream computer (SISD), Single instruction stream multi data stream computer (SIMD), Multiple instruction stream single data stream computer (MISD), Multi instruction stream multi data stream computer (MIMD)

**4.**

SIMD and MIMD. Because the GPU core is light, it uses simple control logic to optimize data parallel tasks, so it pays much attention to the throughput of parallel programs. SIMD and MIMD both process multiple data streams, but the former has only one instruction for all cores at any time point to process different data streams, while the latter uses multiple instruction streams to process multiple data streams asynchronously.

**5.**

An embarrassingly parallel task can be considered a trivial case - little or no manipulation is needed to separate the problem into a number of parallel tasks. This is often the case where there is little or no dependency or need for communication between those parallel tasks, or for results between them.

**6.**

Speedup is the ratio of the time consumed by the same task in a single-processor system and a parallel processor system. It is used to measure the performance and effect of parallel systems or program parallelization. Assuming that the time required to solve a certain problem with a single processing unit is Ts, and the time to solve this problem with p identical processing units is Tp, then the speedup ratio S=Ts/Tp. If S=p, the speed-up ratio is linear, which means that the execution speed increases as the number of processors increases. This is just an ideal state. When Ts is the execution time of the best serial algorithm, the speedup is absolute, and when Ts is the execution time of the parallel algorithm on a single processor, then the speedup is relative.

S=p is the linear acceleration ratio (ideal acceleration ratio).

S<p is the real speedup

S>p is the super linear speedup ratio

**7.**

ATLAS, OpenBLAS, LAPACK

**8.**

MPI\_Barrier, MPI\_Bcast, MPI\_Scatter, MPI\_Gather, MPI\_Allgather, MPI\_Redue

**9.**

Strong scaling: speedup=1/(s+p/N)1/(s+p/N), s is the proportion of the time spent by the parts that can not be parallelized, P is the proportion of the time spent in the parts that can be paralleled, and N is the number of processors. This law describes the scaling ability of increasing computing resources when the problem scale is fixed.

Weak scaling: speedup=s+p∗Ns+p∗N, parameters are the same as the strong scaling. This law describes the scaling capability when the problem size and computing resources increase simultaneously.

**10.**

There is some communication delay between different nodes, but it’s much lower when it comes to inter-core communication. So the computation time on multi-CPU on multiple nodes is slower than that of multi-CPU on a single node.

**11.**

(1) Allocate memory on both Host and Device

(2) Initialize variables on Host

(3) Copy data from Host and Device

(4) Perform parallel computation on Device

(5) Copy data back form device to Host

(6) Free memory on both Host and Device

**12.**

The GPU off-chip memory space consists of texture memory, constant memory, local memory, and global memory.

The GPU on-chip memory space consists of register file and shared memory.