

# SPPU-BE-COMP-CONTENT - KSKA Git

Q1) How is CUDA programming useful to study parallel Algorithm?

ANS.

CUDA (Compute Unified Device Architecture) is a parallel computing platform by NVIDIA that allows developers to use GPU Power for general purpose computing.

→ Why CUDA is Useful:

- Massive parallelism

GPUs have thousands of cores → ideal for parallel algorithms

- Hands On Learning.

Helps understand:

- Thread Level parallelism
- Memory hierarchy (global, shared, local)
- Synchronization.

- Performance Optimization

Student Learns:

- Memory coalescing

- Load Balancing

- Minimizing Latency.

- Real World Application

- AI/ML

- Image Processing.

- Scientific simulation.

CUDA provides a practical platform to design, implement, and optimize parallel algorithm efficiently.

Q2) Discuss the importance of parallel reduction operations.

ANS.

Reduction:-

Reduction means combining elements of a dataset into a single value.

For Eg:-

- Sum of Array.

- Average.

- Maximum value

# SPPU-BE-COMP-CONTENT - KSKA Git

⇒ Importance:-

① Core Operation in Parallel Computing

Used in many algorithms:

- Sorting
- Machine Learning
- Graph processing.

② Improves Performance

- Instead of Sequential sum:
  - Parallel reduction divides work among threads.
  - Reduces time complexity.

③ Tree-based Computation.

- Uses Divide and Conquer.
- Reduces operations from  $O(n)$  to  $O(\log n)$  (parallel depth)

④ Efficient Resource Utilization.

- Uses shared memory for faster computation.

Q3.) Discuss the Operations on vectors and the approach for parallel algorithm design for the same.

ANS. Common Vector Operations:

- Vector addition.
- Scalar Multiplication.
- Dot Product.
- Elementwise Operation

Example:-

$$C[i] = A[i] + B[i]$$

→ Parallel Approach:-

Step I:- Data Partitioning

# SPPU-BE-COMP-CONTENT - KSKA Git

- Divide vector into chunks.
- Assign each chunk to threads.

## STEP II:- Independent Computation

- Each thread works on:

$$C[i] = A[i] + B[i]$$

## STEP III:- Synchronization (if needed)

- Required in operations like dot product.

## Example:-

$\text{int } i = \text{threadIdx.x} + \text{blockIdx.x} * \text{blockDim.x};$

$C[i] = A[i] + B[i];$

## Benefits:-

- High Speed for Large Vectors.

- No Dependency  $\rightarrow$  easy parallelization.

- Scalable.

Q4) How is Parallelism achieved in CUDA?

Ans.

CUDA achieves parallelism using a hierarchy:

## Key concepts:-

① Thread : Smallest Unit of Execution

② Block : Group of Threads.

③ Grid : Collection of Blocks.

## TYPES OF PARALLELISM:-

② Data Parallelism:

- Same Operation on Different data.

# SPPU-BE-COMP-CONTENT - KSKA Git

## ② Task Parallelism.

- Different task run simultaneously.

## - Execution Model:-

- Thousands of kernel execute same kernel function.
- Each thread handles different data.

## - Memory Hierarchy

- Registers
- Shared memory (within block)
- Global Memory (fastest)

Hence, CUDA Achieves parallelism by executing many threads simultaneously across GPU cores.

Q5) Explain Grid, Block, and Thread structure in relation with parallel reduction.

ANS.

Structure:-

Grid

└ Block

└ Threads

→ Role in Parallel Reduction:-

### Step 1: Thread-Level Work

- Each thread loads one element.
- Performs partial computation.

### Step 2: Block Level Reduction

- Threads in a Block
- stores value in a shared memory.
- Perform Reduction (sum, max)

# SPPU-BE-COMP-CONTENT - KSKA Git

## ◦ Example:-

Thread 0 + Thread 1

Thread 2 + Thread 3

→ Continue until one value per block.

## STEP III:-

- Each block gives one result.
- Final Reduction across blocks.

## ◦ Key concepts:-

- ① Shared Memory
  - Fast communication within block.
- ② Synchronization
  - syncthreads() ensure correctness.
- ③ Hierarchical Reduction.
  - Improves Efficiency.

## ◻ Example Flow:-

1. Load Data → threads.
2. Reduce inside block.
3. Combine block results.