Name: Atharva Burujpatte, roll no; 05

import numpy as np

from numba import cuda

import time

# CUDA kernel for vector addition

@cuda.jit

def vector\_add\_cuda(A, B, C):

idx = cuda.grid(1)

if idx < A.size:

C[idx] = A[idx] + B[idx]

# Vector size

N = 1\_000\_000

# Allocate and initialize host arrays

A = np.arange(N, dtype=np.float32)

B = np.arange(N, dtype=np.float32) \* 2

C = np.zeros(N, dtype=np.float32)

# Allocate device memory

d\_A = cuda.to\_device(A)

d\_B = cuda.to\_device(B)

d\_C = cuda.device\_array\_like(C)

# Launch kernel

threads\_per\_block = 256

blocks\_per\_grid = (N + threads\_per\_block - 1) // threads\_per\_block

start = time.time()

vector\_add\_cuda[blocks\_per\_grid, threads\_per\_block](d\_A, d\_B, d\_C)

cuda.synchronize() # Ensure kernel execution is complete

end = time.time()

# Copy result back to host

C = d\_C.copy\_to\_host()

# Verify a few values

print("Sample output:")

for i in range(5):

print(f"{A[i]} + {B[i]} = {C[i]}")

print("Vector addition successful!" if np.allclose(C, A + B) else "Error in result")

print(f"\nTime taken on GPU: {end - start:.6f} seconds")

output:

Sample output:

0.0 + 0.0 = 0.0

1.0 + 2.0 = 3.0

2.0 + 4.0 = 6.0

3.0 + 6.0 = 9.0

4.0 + 8.0 = 12.0

Vector addition successful!

Time taken on GPU: 0.0021 seconds