

# HPC CUDA C TUTORIAL 11 REPORT

## CS22B2015 – HARSHITH B

### **1. Introduction**

This report analyzes the performance of serial and parallel implementations for calculating the matrix addition of two matrices of size 10000x10000. A serial code written in C and a parallel code written in CUDA (.cu) were developed to perform this computation, and their execution times are compared to evaluate efficiency and potential speedup.

### **2. Serial Code Snippet**

```
void add_matrices(double matrix1[N][N], double matrix2[N][N], double result[N][N], FILE *file) {
    clock_t start, end;
    start = clock();

    for (int i = 0; i < N; i++) {
        for (int j = 0; j < N; j++) {
            result[i][j] = matrix1[i][j] + matrix2[i][j];
            // fprintf(file, "%lf\n", result[i][j]);
        }
    }

    end = clock();
    double time_spent = (double)(end - start) / CLOCKS_PER_SEC;
    printf("Time: %f seconds\n", time_spent);
}
```

### 3. Parallel Code Snippet

```
__global__ void add_matrices(double *d_matrix1, double *d_matrix2, double *d_result) {
    int i = blockIdx.x * blockDim.x + threadIdx.x;
    int j = blockIdx.y * blockDim.y + threadIdx.y;

    if (i < N && j < N) {
        int idx = i * N + j;
        d_result[idx] = d_matrix1[idx] + d_matrix2[idx];
    }
}

double *d_matrix1, *d_matrix2, *d_result;
cudaMalloc((void **)&d_matrix1, N * N * sizeof(double));
cudaMalloc((void **)&d_matrix2, N * N * sizeof(double));
cudaMalloc((void **)&d_result, N * N * sizeof(double));

cudaMemcpy(d_matrix1, matrix1, N * N * sizeof(double), cudaMemcpyHostToDevice);
cudaMemcpy(d_matrix2, matrix2, N * N * sizeof(double), cudaMemcpyHostToDevice);

dim3 threadsPerBlock(16, 16);
dim3 blocksPerGrid((N + threadsPerBlock.x - 1) / threadsPerBlock.x,
                  (N + threadsPerBlock.y - 1) / threadsPerBlock.y);

cudaEvent_t start, end;
cudaEventCreate(&start);
cudaEventCreate(&end);

cudaEventRecord(start);
add_matrices<<<blocksPerGrid, threadsPerBlock>>>(d_matrix1, d_matrix2, d_result);
cudaEventRecord(end);
cudaEventSynchronize(end);

float milliseconds = 0;
cudaEventElapsedTime(&milliseconds, start, end);
printf("Time: %f seconds\n", milliseconds / 1000);

cudaMemcpy(result, d_result, N * N * sizeof(double), cudaMemcpyDeviceToHost);

cudaFree(d_matrix1);
cudaFree(d_matrix2);
cudaFree(d_result);
```

### 4. Terminal Output Screenshot

```
(venv) harshith@harshithb:~/Projects /SEM 6/HPC/tutorial-11$ ./serial
Time: 0.687590 seconds
(venv) harshith@harshithb:~/Projects /SEM 6/HPC/tutorial-11$ ./parallel
Time: 0.015570 seconds
```

### ***5. Serial and Parallel Code Execution Time and Speedup Calculation***

- **Serial Code Execution Time:** 0.687590 seconds
- **Parallel Code Execution Time:** 0.015570 seconds

**Speedup Calculation:**

$$\underline{\text{Speedup} = \text{Serial Execution Time} / \text{Parallel Execution Time}}$$

$$\underline{\text{Speedup Estimation} = 44.161}$$