REPORT

UE22EC341B
High Performance Computing

AHP-2 ASSIGNMENT

MADE BY: SHARVI RANJAN
PES1UG22EC918
Section 6-F

Problem Statement:

Implement and run a C++ CUDA-based odd–even transposition sort on Google Colab and empirically compare its GPU execution time against a standard CPU bubble-sort for a 512-element array.

Code:

```
!nvidia-smi
code = r'''
#include "cuda_runtime.h"
#include "device_launch_parameters.h"
#include <iostream>
#include <chrono>
#include <algorithm>

#define SIZE 512
__device__ void swap(int& a, int& b) {
int tmp = a;
a = b;
b = tmp;}

__global__ void oddEvenSort(int* data, int n) {
int tid = threadIdx.x;

for (int i = 0; i < n; i++) {
int idx = 2 * tid + (i % 2);
if (idx + 1 < n && data[idx] > data[idx + 1]) {
swap(data[idx], data[idx + 1]);
}
__syncthreads(); // sync between odd/even phases
}
}

void cpuBubbleSort(int* arr, int n) {
for (int i = 0; i < n - 1; i++) {
for (int j = 0; j < n - i - 1; j++) {
if (arr[j] > arr[j + 1]) {
std::swap(arr[j], arr[j + 1]);
}
}
}
}
void printArray(const int* arr, int n, const char* label) {
std::cout << label << ": ";
for (int i = 0; i < n; ++i) {
std::cout << arr[i] << " ";
}
std::cout << "\n";
}

int main() {
int h_data[SIZE], h_cpu[SIZE];
```

```
for (int i = 0; i < SIZE; ++i)
h_data[i] = rand() % 100;

std::copy(h_data, h_data + SIZE, h_cpu);

std::cout << "Original Array:\n";
printArray(h_data, SIZE, "Original");

// CPU Timing
auto start_cpu = std::chrono::high_resolution_clock::now();
cpuBubbleSort(h_cpu, SIZE);
auto end_cpu = std::chrono::high_resolution_clock::now();
std::chrono::duration<double, std::milli> cpu_duration = end_cpu - start_cpu;
std::cout << "CPU Time: " << cpu_duration.count() << " ms\n";

std::cout << "CPU Sorted Array:\n";
printArray(h_cpu, SIZE, "CPU Sorted");

// GPU Setup
int* d_data;
cudaMalloc((void**)&d_data, SIZE * sizeof(int));
cudaMemcpy(d_data, h_data, SIZE * sizeof(int), cudaMemcpyHostToDevice);

// GPU Timing
cudaEvent_t start, stop;
cudaEventCreate(&start);
cudaEventCreate(&stop);
cudaEventRecord(start);

dim3 grid(1);
dim3 block(SIZE / 2); // Each thread compares one pair

oddEvenSort<<<grid, block>>>(d_data, SIZE);
cudaEventRecord(stop);
cudaEventSynchronize(stop);

float gpu_time = 0;
cudaEventElapsedTime(&gpu_time, start, stop);
std::cout << "GPU Time: " << gpu_time << " ms\n";

// Get result
int h_result[SIZE];
cudaMemcpy(h_result, d_data, SIZE * sizeof(int), cudaMemcpyDeviceToHost);

std::cout << "GPU Sorted Array:\n";
printArray(h_result, SIZE, "GPU Sorted");

// Validation (optional)
bool isValid = true;
for (int i = 0; i < SIZE; ++i) {
if (h_result[i] != h_cpu[i]) {
std::cerr << "Mismatch at index " << i << "!\n";
```

```
isValid = false;
break;
}
}

if (isValid) {
std::cout << "Validation Passed: CPU and GPU results match.\n";
} else {
std::cerr << "Validation Failed: CPU and GPU results do not match.\n";
}

std::cout << "Speedup (CPU/GPU): " << cpu_duration.count() / gpu_time << "x\n";

cudaFree(d_data);
cudaEventDestroy(start);
cudaEventDestroy(stop);
return 0;
}
'''
with open('bubblesort.cu', 'w') as f:
f.write(code)
!nvcc -arch=sm_75 bubblesort.cu -o bubblesort
!./bubblesort
```

0utput:

Original Array: 83 86 77 15 93 35 86 92 49 21 62 27 90 59 63 26 40 26 72 36 11 68 67 29 82 30 62 23 67 35 29 2 22 58 69 67 93 56 11 42 29 73 21 19 84 37 98 24 15 70 13 26 91 80 56 73 62 70 96 81 5 25 84 27 36 5 46 29 13 57 24 95 82 45 14 67 34 64 43 50 87 8 76 78 88 84 3 51 54 99 32 60 76 68 39 12 26 86 94 39 95 70 34 78 67 1 97 2 17 92 52 56 1 80 86 41 65 89 44 19 40 29 31 17 97 71 81 75 9 27 67 56 97 53 86 65 6 83 19 24 28 71 32 29 3 19 70 68 8 15 40 49 96 23 18 45 46 51 21 55 79 88 64 28 41 50 93 0 34 64 24 14 87 56 43 91 27 65 59 36 32 51 37 28 75 7 74 21 58 95 29 37 35 93 18 28 43 11 28 29 76 4 43 63 13 38 6 40 4 18 28 88 69 17 17 96 24 43 70 83 90 99 72 25 44 90 5 39 54 86 69 82 42 64 97 7 55 4 48 11 22 28 99 43 46 68 40 22 11 10 5 1 61 30 78 5 20 36 44 26 22 65 8 16 82 58 24 37 62 24 0 36 52 99 79 50 68 71 73 31 81 30 33 94 60 63 99 81 99 96 59 73 13 68 90 95 26 66 84 40 90 84 76 42 36 7 45 56 79 18 87 12 48 72 59 9 36 10 42 87 6 1 13 72 21 55 19 99 21 4 39 11 40 67 5 28 27 50 84 58 20 24 22 69 96 81 30 84 92 72 72 50 25 85 22 99 40 42 98 13 98 90 24 90 9 81 19 36 32 55 94 4 79 69 73 76 50 55 60 42 79 84 93 5 21 67 4 13 61 54 26 59 44 2 2 6 84 21 42 68 28 89 72 8 58 98 36 8 53 48 3 33 33 48 90 54 67 46 68 29 0 46 88 97 49 90 3 33 63 97 53 92 86 25 52 96 75 88 57 29 36 60 14 21 60 4 28 27 50 48 56 2 94 97 99 43 39 2 28 3 0 81 47 38 59 51 35 34 39 92 15 27 4 29 49 64 85 29 43 35 77 0 38 71 49 89 67 88 92 95 43 44 29 90 82 40 41 69 26 32 61 42 60 17 23 61 81 9 90 25 96 67

Sorted Array: 0 0 0 0 0 1 1 1 1 2 2 2 2 2 3 3 3 3 3 4 4 4 4 4 4 4 4 5 5 5 5 5 5 5 6 6 6 6 7 7 7 8 8 8 8 8 9 9 9 9 10 10 11 11 11 11 11 11 12 12 13 13 13 13 13 13 13 14 14 14 15 15 15 15 16 17 17 17 17 17 18 18 18 18 19 19 19 19 19 19 20 20 21 21 21 21 21 21 21 21 21 22 22 22 22 22 22 23 23 23 24 24 24 24 24 24 24 24 25 25 25 25 25 26 26 26 26 26 26 26 26 27 27 27 27 27 27 27 28 28 28 28 28 28 28 28 28 28 29 29 29 29 29 29 29 29 29 29 29 29 30 30 30 30 31 31 32 32 32 32 32 33 33 33 33 34 34 34 34 35 35 35 35 35 36 36 36 36 36 36 36 36 36 37 37 37 37 38 38 38 39 39 39 39 39 39 40 40 40 40 40 40 40 40 41 41 41 42 42 42 42 42 42 42 42 43 43 43 43 43 43 43 43 44 44 44 44 44 45 45 45 46 46 46 46 46 47 48 48 48 48 48 49 49 49 49 49 50 50 50 50 50 50 50 51 51 51 51 52 52 52 53 53 53 54 54 54 54 55 55 55 55 55 56 56 56 56 56 56 56 57 57 58 58 58 58 58 59 59 59 59 59 59 60 60 60 60 60 60 61 61 61 61 62 62

62 62 63 63 63 63 64 64 64 64 64 65 65 65 65 66 67 67 67 67 67 67 67 67 67 67 67 68 68 68 68 68 68 68
68 69 69 69 69 69 69 70 70 70 70 70 71 71 71 71 72 72 72 72 72 72 72 73 73 73 73 73 74 75 75 75 76 76
76 76 76 77 77 78 78 78 79 79 79 79 79 80 80 81 81 81 81 81 81 81 81 82 82 82 82 82 83 83 83 84 84 84
84 84 84 84 84 84 85 85 86 86 86 86 86 86 86 87 87 87 87 88 88 88 88 88 88 89 89 89 90 90 90 90 90 90
90 90 90 90 90 91 91 92 92 92 92 92 92 93 93 93 93 93 94 94 94 94 95 95 95 95 95 96 96 96 96 96 96 96
97 97 97 97 97 97 97 98 98 98 98 99 99 99 99 99 99 99 99 99

CPU Time: 1.31995 ms
GPU Time: 0.264224 ms
Validation Passed: CPU and GPU results match.
Speedup (CPU/GPU): 4.99557x

Inference:

- Overall, GPU odd–even sort outperforms CPU bubble sort by thread-level parallelizing up to 256 simultaneous compare-and-swap operations per phase, reducing kernel time.
- The GPU sorted the array in about 0.26 ms, while the CPU took around 1.32 ms—almost a 5× speed-up.