**Lab Exercise 9.4 – Performance Comparison – CPU vs GPU vs cuFFT**

**Objective:**

To measure and compare the performance of:

* CPU-based Naive DFT
* GPU-based custom FFT kernel (simplified)
* cuFFT (highly optimized NVIDIA CUDA FFT library)

**Files Required:**

You’ll write a single .cu file that contains:

1. CPU Naive DFT
2. GPU FFT kernel (simplified — not fully optimized Cooley-Tukey)
3. cuFFT-based transform

**fft\_comparison\_full.cu**

#include <iostream>

#include <cmath>

#include <chrono>

#include <cuda\_runtime.h>

#include <cufft.h>

#ifndef M\_PI

#define M\_PI 3.14159265358979323846

#endif

#define N 1024

// ---------------- CPU Naive DFT ----------------

void naiveDFT(const float\* real, const float\* imag, float\* out\_real, float\* out\_imag, int n) {

for (int k = 0; k < n; ++k) {

float sum\_real = 0.0f, sum\_imag = 0.0f;

for (int t = 0; t < n; ++t) {

float angle = 2.0f \* M\_PI \* t \* k / n;

sum\_real += real[t] \* cos(angle) + imag[t] \* sin(angle);

sum\_imag += -real[t] \* sin(angle) + imag[t] \* cos(angle);

}

out\_real[k] = sum\_real;

out\_imag[k] = sum\_imag;

}

}

// ---------------- GPU FFT Kernel (Naive) ----------------

\_\_global\_\_ void fftKernel(const float\* real, float\* out\_real, float\* out\_imag, int n) {

int k = blockIdx.x \* blockDim.x + threadIdx.x;

if (k < n) {

float sum\_real = 0.0f, sum\_imag = 0.0f;

for (int t = 0; t < n; ++t) {

float angle = 2.0f \* M\_PI \* t \* k / n;

sum\_real += real[t] \* cosf(angle);

sum\_imag += -real[t] \* sinf(angle);

}

out\_real[k] = sum\_real;

out\_imag[k] = sum\_imag;

}

}

int main() {

float h\_signal[N], h\_imag[N] = {0};

for (int i = 0; i < N; ++i)

h\_signal[i] = sin(2 \* M\_PI \* i / N);

// ---------------- CPU DFT ----------------

float cpu\_out\_real[N], cpu\_out\_imag[N];

auto start\_cpu = std::chrono::high\_resolution\_clock::now();

naiveDFT(h\_signal, h\_imag, cpu\_out\_real, cpu\_out\_imag, N);

auto end\_cpu = std::chrono::high\_resolution\_clock::now();

double cpu\_time = std::chrono::duration<double, std::milli>(end\_cpu - start\_cpu).count();

std::cout << "CPU DFT Time: " << cpu\_time << " ms" << std::endl;

// ---------------- GPU DFT Kernel ----------------

float \*d\_input, \*d\_out\_real, \*d\_out\_imag;

cudaMalloc(&d\_input, N \* sizeof(float));

cudaMalloc(&d\_out\_real, N \* sizeof(float));

cudaMalloc(&d\_out\_imag, N \* sizeof(float));

cudaMemcpy(d\_input, h\_signal, N \* sizeof(float), cudaMemcpyHostToDevice);

cudaEvent\_t start\_gpu, stop\_gpu;

cudaEventCreate(&start\_gpu);

cudaEventCreate(&stop\_gpu);

cudaEventRecord(start\_gpu);

fftKernel<<<(N + 255)/256, 256>>>(d\_input, d\_out\_real, d\_out\_imag, N);

cudaEventRecord(stop\_gpu);

cudaEventSynchronize(stop\_gpu);

float gpu\_kernel\_time = 0;

cudaEventElapsedTime(&gpu\_kernel\_time, start\_gpu, stop\_gpu);

std::cout << "GPU Naive FFT Kernel Time: " << gpu\_kernel\_time << " ms" << std::endl;

// ---------------- cuFFT ----------------

cufftComplex \*d\_cufft\_in, \*d\_cufft\_out;

cufftHandle plan;

cudaMalloc(&d\_cufft\_in, sizeof(cufftComplex) \* N);

cudaMalloc(&d\_cufft\_out, sizeof(cufftComplex) \* N);

cufftComplex h\_cufft\_in[N];

for (int i = 0; i < N; ++i) {

h\_cufft\_in[i].x = h\_signal[i];

h\_cufft\_in[i].y = 0.0f;

}

cudaMemcpy(d\_cufft\_in, h\_cufft\_in, sizeof(cufftComplex) \* N, cudaMemcpyHostToDevice);

cufftPlan1d(&plan, N, CUFFT\_C2C, 1);

cudaEvent\_t start\_cufft, stop\_cufft;

cudaEventCreate(&start\_cufft);

cudaEventCreate(&stop\_cufft);

cudaEventRecord(start\_cufft);

cufftExecC2C(plan, d\_cufft\_in, d\_cufft\_out, CUFFT\_FORWARD);

cudaEventRecord(stop\_cufft);

cudaEventSynchronize(stop\_cufft);

float cufft\_time = 0;

cudaEventElapsedTime(&cufft\_time, start\_cufft, stop\_cufft);

std::cout << "cuFFT Time: " << cufft\_time << " ms" << std::endl;

// Cleanup

cufftDestroy(plan);

cudaFree(d\_input);

cudaFree(d\_out\_real);

cudaFree(d\_out\_imag);

cudaFree(d\_cufft\_in);

cudaFree(d\_cufft\_out);

return 0;

}

**Compilation Command:**

nvcc fft\_comparison\_full.cu -lcufft -o fft\_comparison\_full

**Run the program:**

./fft\_comparison\_full

**Sample Output:**

CPU DFT Time: 280.57 ms

GPU Naive FFT Kernel Time: 4.31 ms

cuFFT Time: 0.37 ms

**Observation:**

| **Method** | **Time (ms)** | **Speed-up vs CPU** |
| --- | --- | --- |
| CPU DFT | ~280 ms | 1× |
| GPU Naive Kernel | ~4 ms | ~70× |
| cuFFT | ~0.3 ms | ~930× |