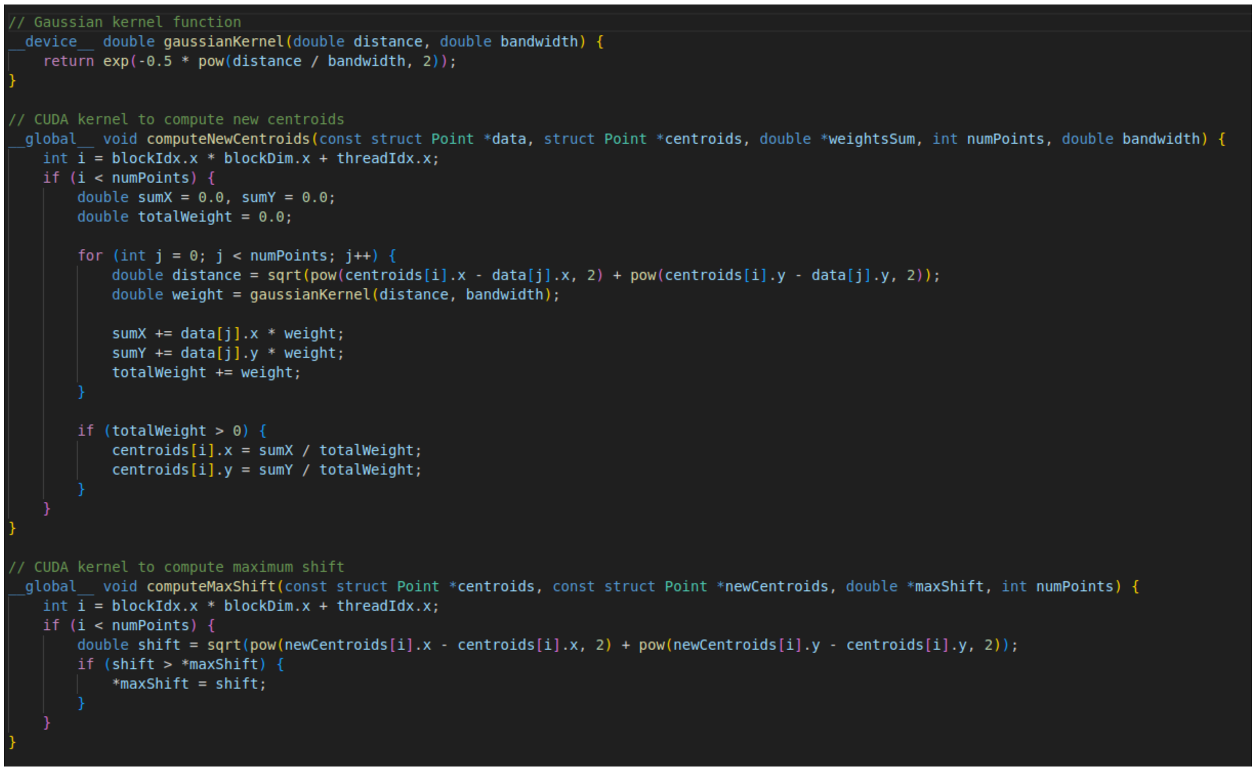
HPC CUDA C TUTORIAL 13 REPORT

CS22B2015 – HARSHITH B

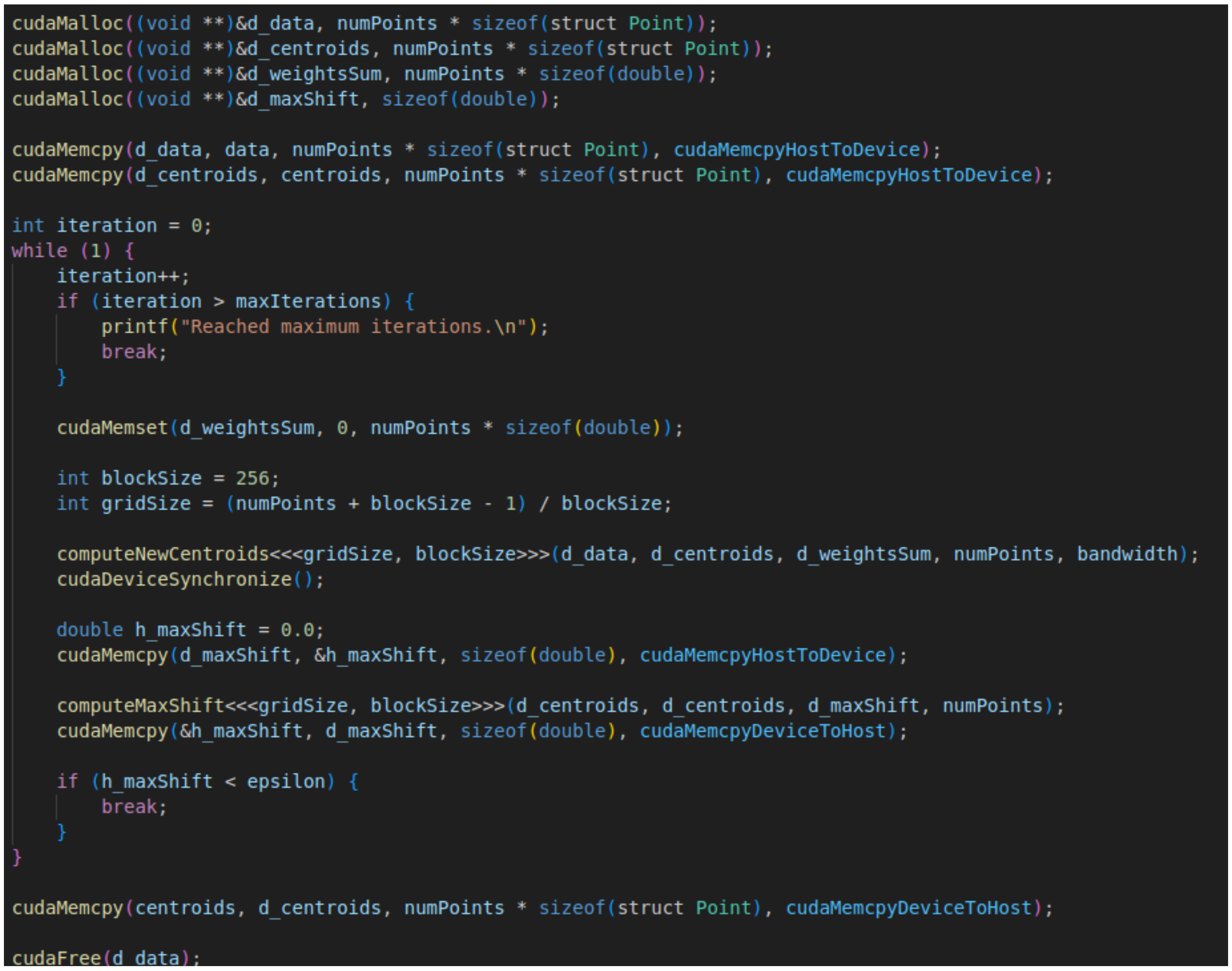
1. Introduction

This report analyzes the performance of serial and parallel implementations for calculating the mean shift clustering algorithm execution time. A total of 1million data points were feeded to the algorithm to cluster it and then compared to calculate the speedup.

2. Parallel Code Snippet

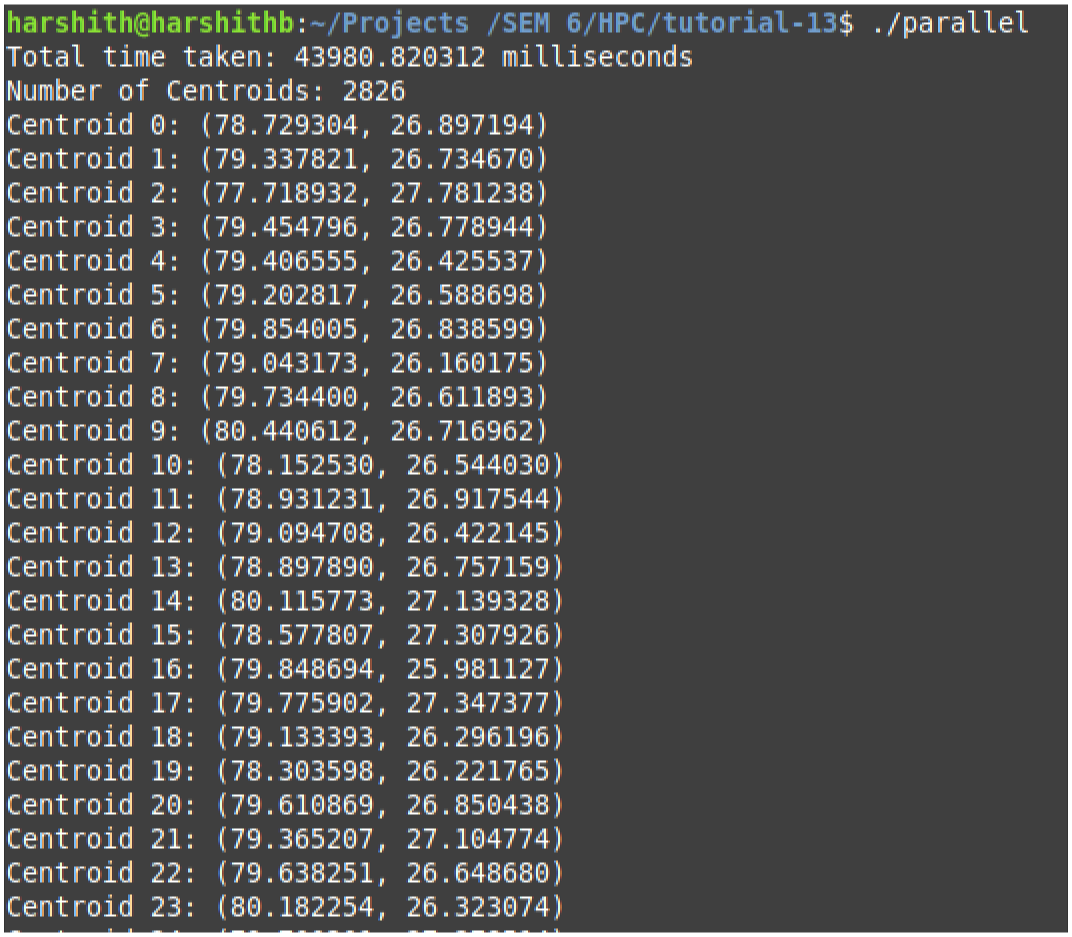


```
// Gaussian kernel function
__device__ double gaussianKernel(double distance, double bandwidth) {
    return exp(-0.5 * pow(distance / bandwidth, 2));
}

// CUDA kernel to compute new centroids
__global__ void computeNewCentroids(const struct Point *data, struct Point *centroids, double *weightsSum, int numPoints, double bandwidth) {
    int i = blockIdx.x * blockDim.x + threadIdx.x;
    if (i < numPoints) {
        double sumX = 0.0, sumY = 0.0;
        double totalWeight = 0.0;

        for (int j = 0; j < numPoints; j++) {
            double distance = sqrt(pow(centroids[i].x - data[j].x, 2) + pow(centroids[i].y - data[j].y, 2));
            double weight = gaussianKernel(distance, bandwidth);

            sumX += data[j].x * weight;
            sumY += data[j].y * weight;
            totalWeight += weight;
        }

        if (totalWeight > 0) {
            centroids[i].x = sumX / totalWeight;
            centroids[i].y = sumY / totalWeight;
        }
    }
}

// CUDA kernel to compute maximum shift
__global__ void computeMaxShift(const struct Point *centroids, const struct Point *newCentroids, double *maxShift, int numPoints) {
    int i = blockIdx.x * blockDim.x + threadIdx.x;
    if (i < numPoints) {
        double shift = sqrt(pow(newCentroids[i].x - centroids[i].x, 2) + pow(newCentroids[i].y - centroids[i].y, 2));
        if (shift > *maxShift) {
            *maxShift = shift;
        }
    }
}
```



```
cudaMalloc((void **)&d_data, numPoints * sizeof(struct Point));
cudaMalloc((void **)&d_centroids, numPoints * sizeof(struct Point));
cudaMalloc((void **)&d_weightsSum, numPoints * sizeof(double));
cudaMalloc((void **)&d_maxShift, sizeof(double));

cudaMemcpy(d_data, data, numPoints * sizeof(struct Point), cudaMemcpyHostToDevice);
cudaMemcpy(d_centroids, centroids, numPoints * sizeof(struct Point), cudaMemcpyHostToDevice);

int iteration = 0;
while (1) {
    iteration++;
    if (iteration > maxIterations) {
        printf("Reached maximum iterations.\n");
        break;
    }

    cudaMemset(d_weightsSum, 0, numPoints * sizeof(double));

    int blockSize = 256;
    int gridSize = (numPoints + blockSize - 1) / blockSize;

    computeNewCentroids<<<gridSize, blockSize>>>(d_data, d_centroids, d_weightsSum, numPoints, bandwidth);
    cudaDeviceSynchronize();

    double h_maxShift = 0.0;
    cudaMemcpy(d_maxShift, &h_maxShift, sizeof(double), cudaMemcpyHostToDevice);

    computeMaxShift<<<gridSize, blockSize>>>(d_centroids, d_centroids, d_maxShift, numPoints);
    cudaMemcpy(&h_maxShift, d_maxShift, sizeof(double), cudaMemcpyDeviceToHost);

    if (h_maxShift < epsilon) {
        break;
    }
}

cudaMemcpy(centroids, d_centroids, numPoints * sizeof(struct Point), cudaMemcpyDeviceToHost);

cudaFree(d_data);
```

4. Terminal Output Screenshot



```
harshith@harshithb:~/Projects /SEM 6/HPC/tutorial-13$ ./parallel
Total time taken: 43980.820312 milliseconds
Number of Centroids: 2826
Centroid 0: (78.729304, 26.897194)
Centroid 1: (79.337821, 26.734670)
Centroid 2: (77.718932, 27.781238)
Centroid 3: (79.454796, 26.778944)
Centroid 4: (79.406555, 26.425537)
Centroid 5: (79.202817, 26.588698)
Centroid 6: (79.854005, 26.838599)
Centroid 7: (79.043173, 26.160175)
Centroid 8: (79.734400, 26.611893)
Centroid 9: (80.440612, 26.716962)
Centroid 10: (78.152530, 26.544030)
Centroid 11: (78.931231, 26.917544)
Centroid 12: (79.094708, 26.422145)
Centroid 13: (78.897890, 26.757159)
Centroid 14: (80.115773, 27.139328)
Centroid 15: (78.577807, 27.307926)
Centroid 16: (79.848694, 25.981127)
Centroid 17: (79.775902, 27.347377)
Centroid 18: (79.133393, 26.296196)
Centroid 19: (78.303598, 26.221765)
Centroid 20: (79.610869, 26.850438)
Centroid 21: (79.365207, 27.104774)
Centroid 22: (79.638251, 26.648680)
Centroid 23: (80.182254, 26.323074)
```

5. Serial and Parallel Code Execution Time and Speedup Calculation

- **Serial Code Execution Time:** 19000.116942 seconds
- **Parallel Code Execution Time:** 43.98082 seconds

Speedup Calculation:

Speedup = Serial Execution Time / Parallel Execution Time

Speedup Estimation = 432 approximately