HPC Project

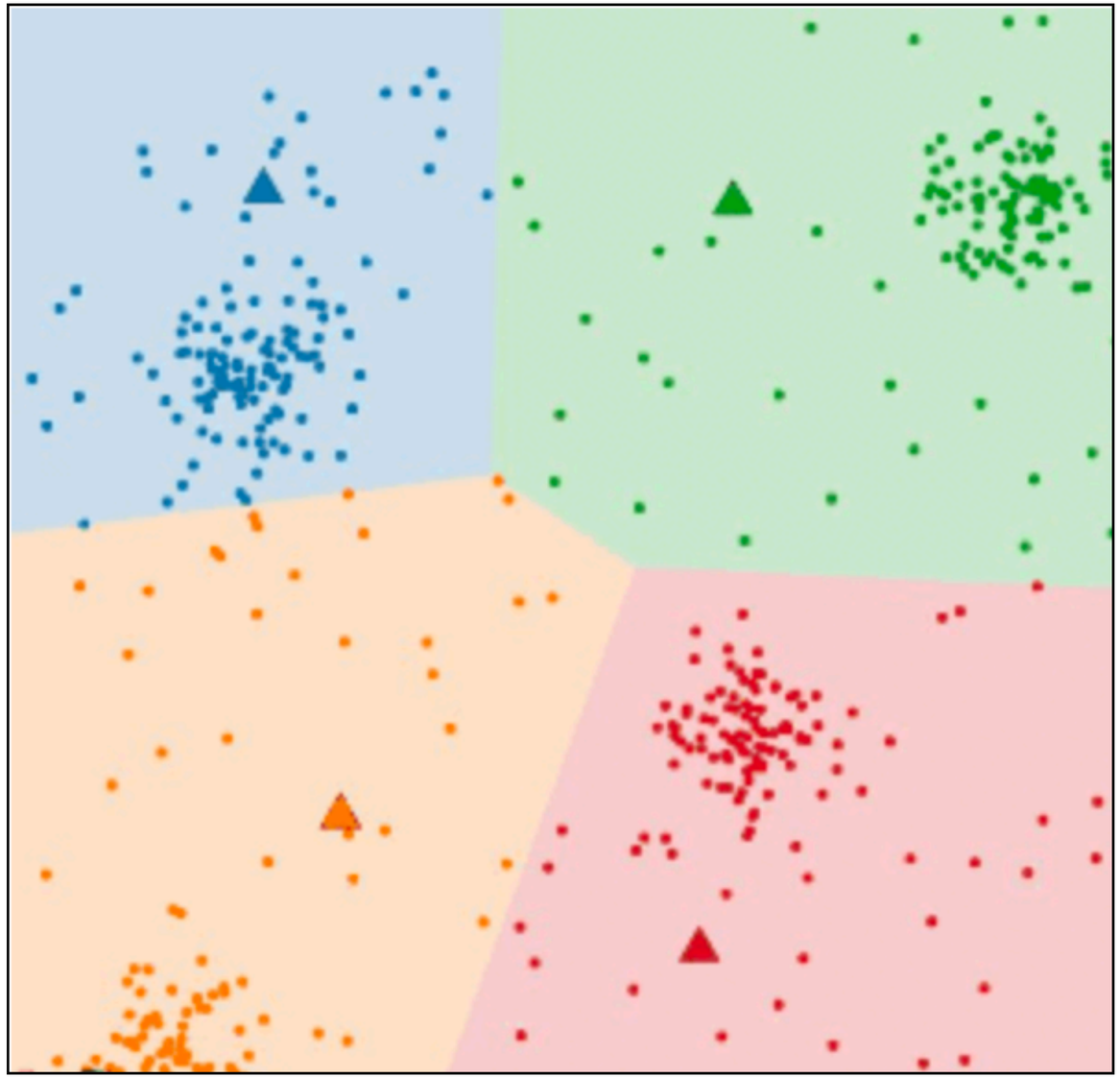
K-means using openmp c++

Filitov Mikhail

K-means clustering

Algorithm

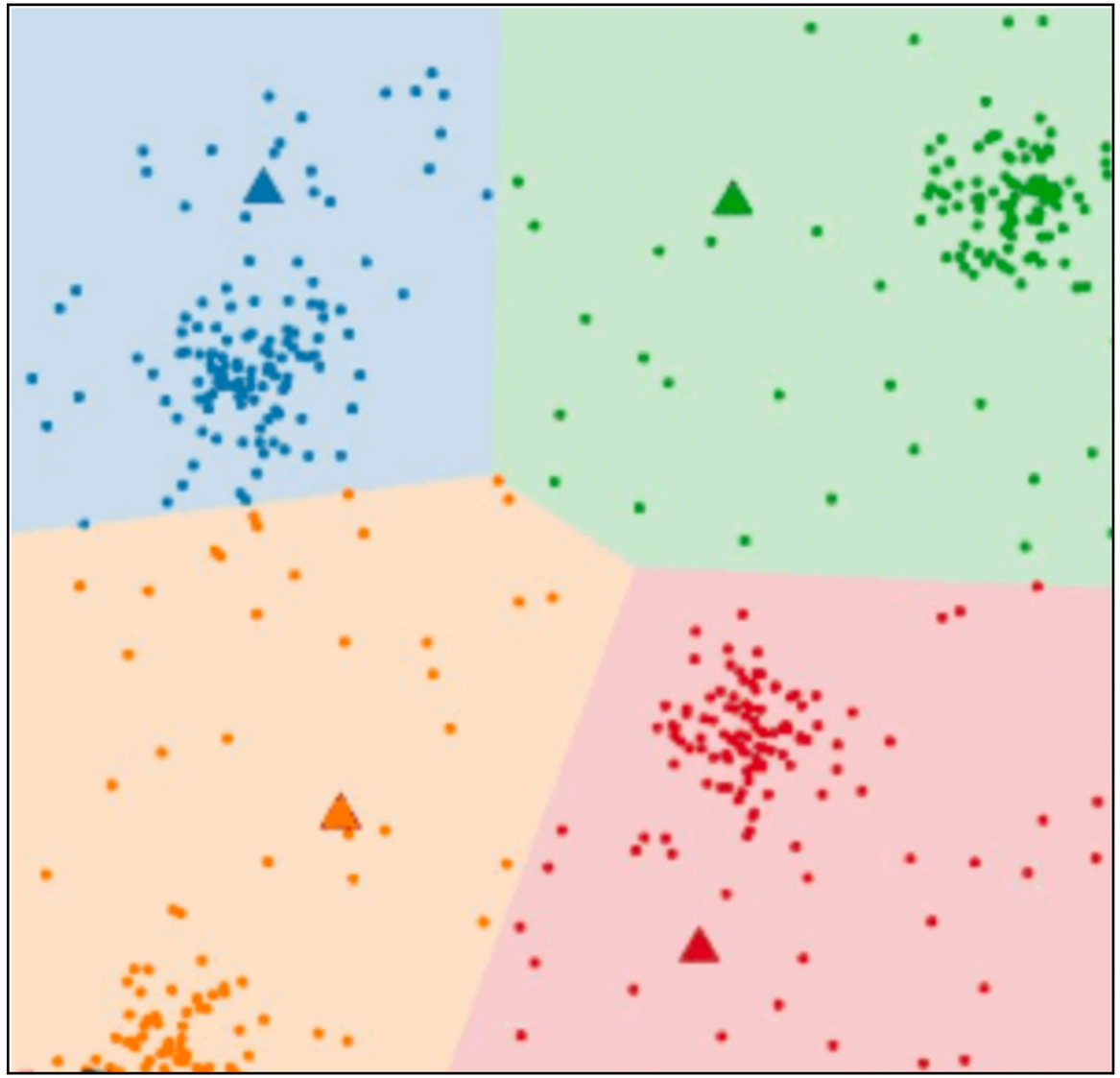
- Choose the number of clusters
- Place the centroids randomly
- Repeat steps 4 and 5 until convergence
- for each data point find the nearest centroid and assign the point to that cluster 5.
- for each cluster new centroid = mean of all points assigned to that cluster



K-means clustering

Weak steps

- Choose the number of clusters
- Place the centroids randomly
- Repeat steps 4 and 5 until convergence
- For each data point find the nearest centroid and assign the point to that cluster 5.
- For each cluster new centroid = mean of all points assigned to that cluster



Pragmas

```
#pragma omp declare reduction(vec_plus_size_t: vector<size_t> : \
                        transform(omp_out.begin(), omp_out.end(), omp_in.begin(), omp_out.begin(), plus<size_t>())) \
                    initializer(omp_priv = omp_orig)


#pragma omp declare reduction(vec_plus_double: vector<double> : \
                        transform(omp_out.begin(), omp_out.end(), omp_in.begin(), omp_out.begin(), plus<double>())) \
                    initializer(omp_priv = omp_orig)
```

Used for assigning dots to clusters

Use of pragmas

```
#pragma omp parallel
{
    #pragma omp for reduction(vec_plus_size_t: clusters_sizes) reduction(vec_plus_double: centroids)
    for (size_t i = 0; i < data_size_normal; ++i) {
        double dot1 = data_opt[2 * i];
        double dot2 = data_opt[2 * i + 1];
        centroids[clusters[i] * 2] += dot1;
        centroids[clusters[i] * 2 + 1] += dot2;
        ++clusters_sizes[clusters[i]];
    }
}
```

Used for assigning dots to centroids

```
#pragma omp parallel for reduction(&:converged)
for (size_t i = 0; i < data_size_normal; ++i) {
    size_t nearest_cluster = FindNearestCentroid2D(centroids, data_opt[2 * i], data_opt[2 * i + 1]);
    if (clusters[i] != nearest_cluster) {
        clusters[i] = nearest_cluster;
        converged = false;
    }
}
```

Used for checking if the algorithm converges

Test data generation

Box-muller transform

$$z_0 = \sqrt{-2\ln U_1}\cos(2\pi U_2) = \sqrt{-2\ln s}\left(\frac{u}{\sqrt{s}}\right) = u\cdot\sqrt{\frac{-2\ln s}{s}}$$

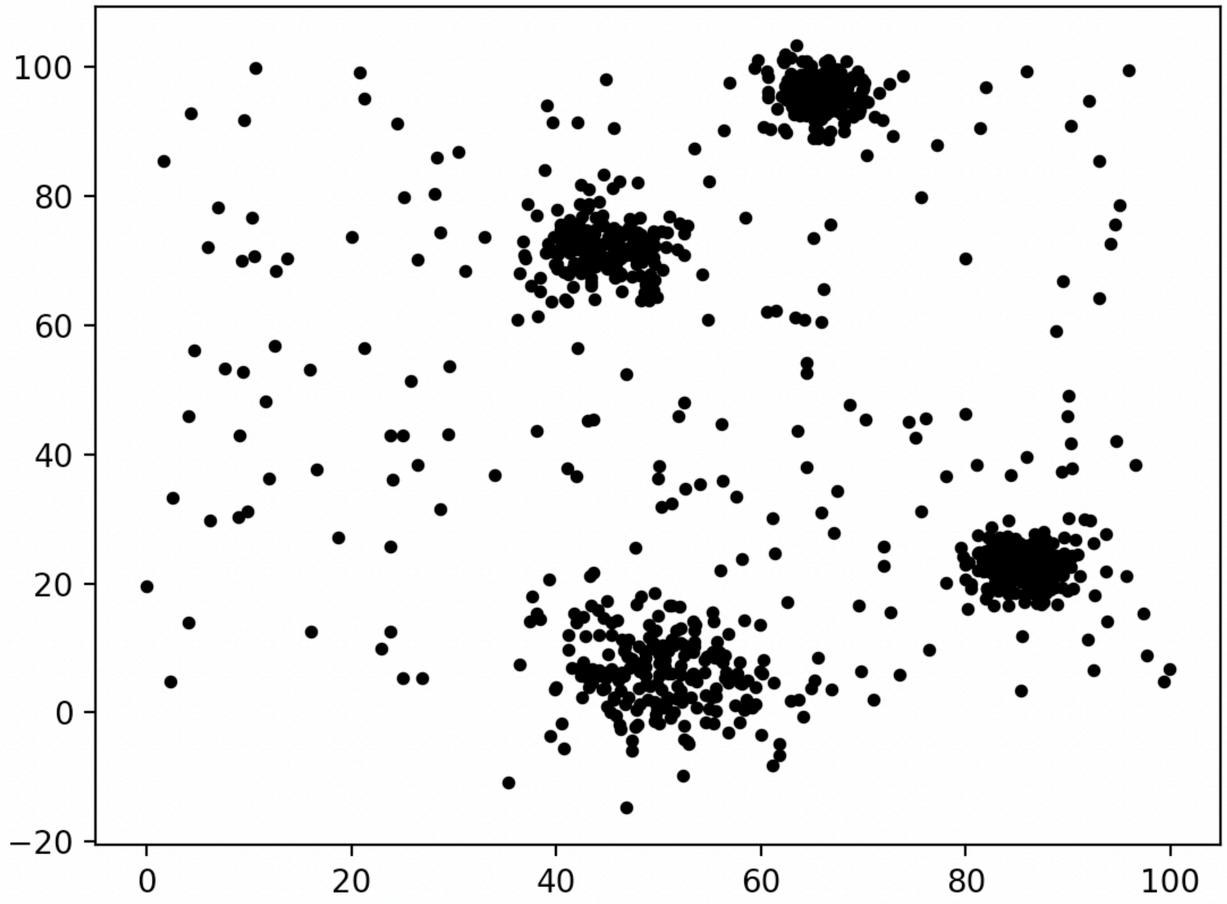
and

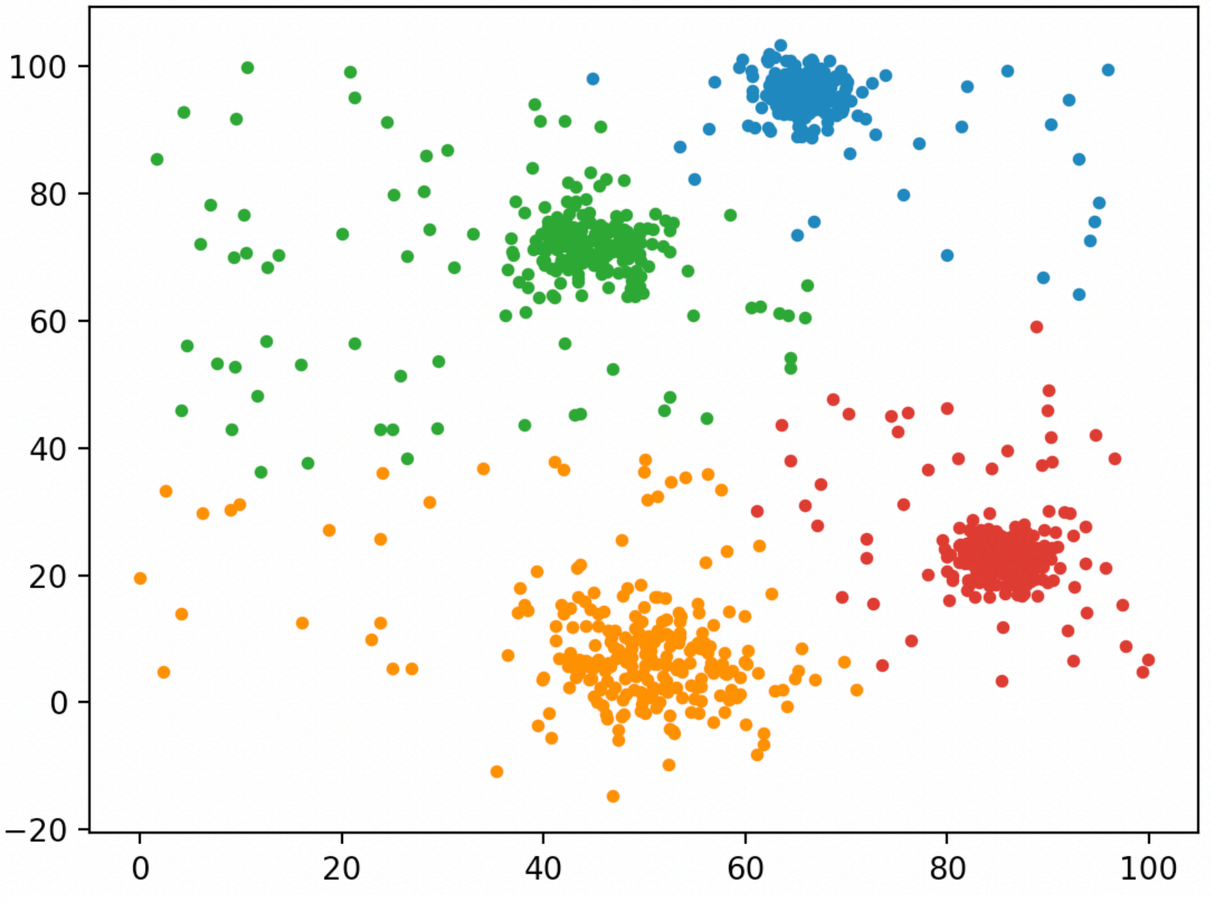
$$z_1 = \sqrt{-2\ln U_1}\sin(2\pi U_2) = \sqrt{-2\ln s}\left(\frac{v}{\sqrt{s}}\right) = v\cdot\sqrt{\frac{-2\ln s}{s}}.$$

```cpp
// Box-Muller transform
double RandNormal(double mean, double sigma) {
    double x, y, r;
    do {
        x = 2 * RandUniform01() - 1;
        y = 2 * RandUniform01() - 1;
        r = x * x + y * y;
    } while (r == 0.0 || r > 1.0);
    return sigma * x * sqrt(-2 * log(r) / r) + mean;
}

struct ClusterParams {
    Point mean;
    double var;
};

Point RandomPointGauss(ClusterParams params) {
    size_t dimensions = params.mean.size();
    Point coord(dimensions);
    for (size_t i = 0; i < dimensions; ++i) {
        coord[i] = RandNormal(params.mean[i], params.var);
    }
    return coord;
}
```

Solution check

Unclustered dots vs clustered

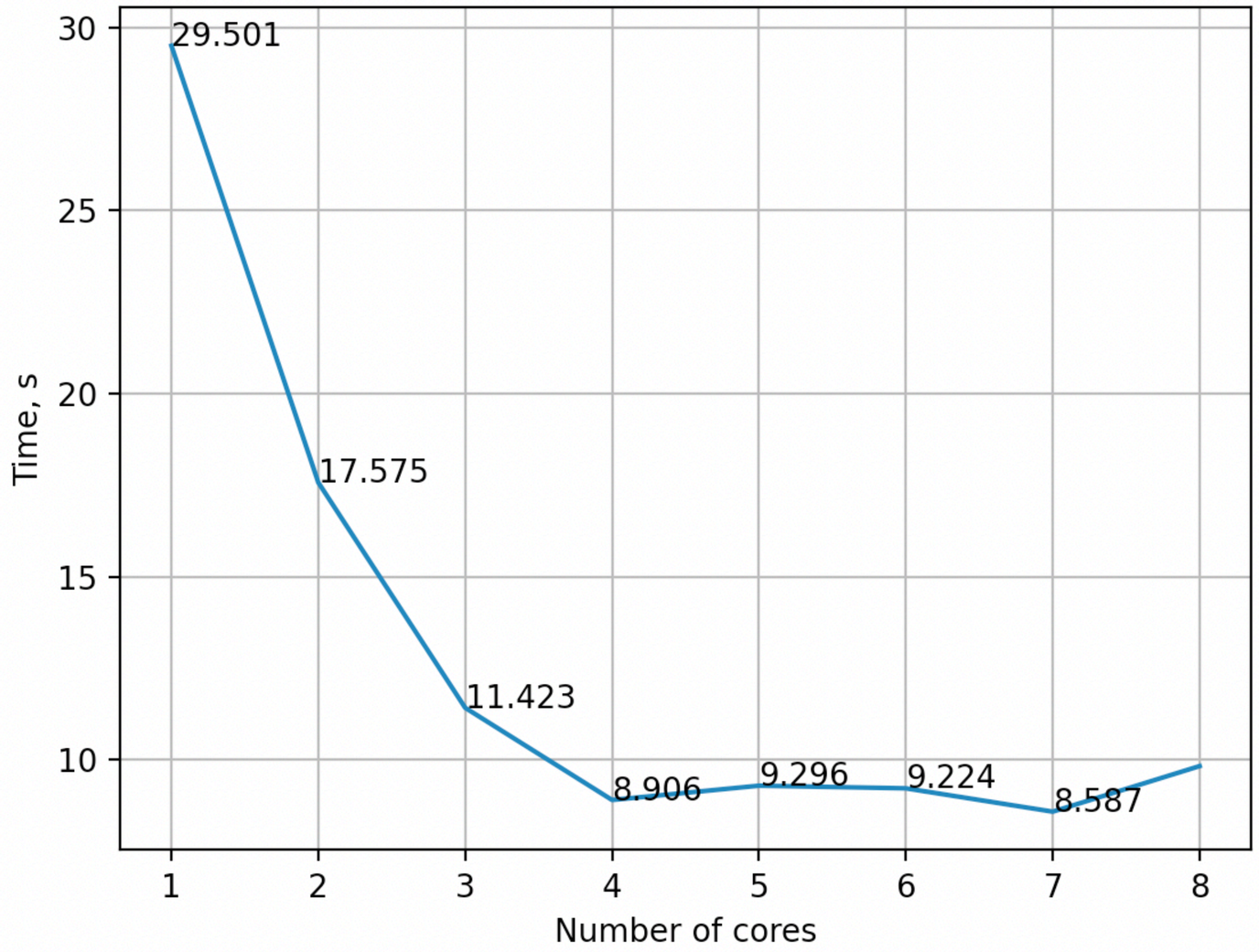




Results

Processor info

- Intel(R) Core(TM) i5-8259U CPU @ 2.30GHz
- cpu.core_count: 4
- cpu.thread_count: 8



Thanks!