Internals

Program 5 — Write a parallel program for points classification

```
#include <stdio.h>
#include <stdlib.h>
#include <omp.h>
#include <math.h>
#define CLUSTER_SIZE 4
int cluster[CLUSTER_SIZE][2] = {{75,25}, {25,25}, {25,75},
int long cluster_count[CLUSTER_SIZE];          {75,75}};
unsigned long
int points[POINTS_SIZE][2];
void populate_points() {
    long long i;
    for(i=0; i< CLUSTER_SIZE ; i++) {
        cluster_count[i] = 0;
    }
  for(i=0; i<POINTS_SIZE ; i++){
      srand(i)
      points[i][0] = rand() %.100;
      points[i][i] = rand() %.100;
    }
}

double get_distance(int x1, int y1, int x2, int y2) {
    int x = x2 - x1, y = y2-y1;
    return (double) sqrt((x*x)+(y*y));
}

int main() {
    double t;
    populate_points();
    long long i;
```

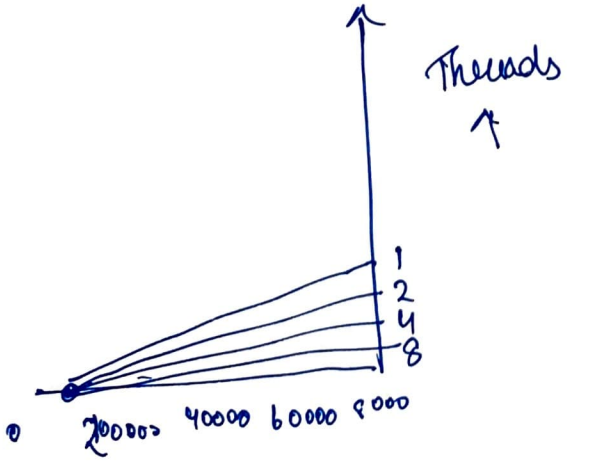
```
if (PRINTS_POINTS != 0) {
    for (i=0; i< CLUSTER_SIZE; i++) {
        printf ("\n Cluster %d: (%d, %d)", i+1, cluster[i][0],
                cluster[i][1]);
    }
    printf ("\n\n");
}

int nt=0;
printf ("Enter number of threads: ");
scanf ("%d", &nt);
t = omp_get_wtime();
#pragma omp parallel for private(i) shared (points, cluster)
    reduction (+: cluster_count) num_threads (nt)
for (i=0; i<POINTS_SIZE; i++) {
    double min_dist = 100, cur_dist = -1;
    int j, cluster_index = -1;
    for (j=0; j< CLUSTER_SIZE; j++) {
        cur_dist = get_distance (points[i][0], points[i][1], cluster
                [j][0], cluster[j][1]);
        if (cur_dist < min_dist) {
            min_dist = cur_dist;
            cluster_index = j;
    }}
    if (PRINT_POINTS != 0) {
        printf ("\n (%d %d) belongs to (%d %d)", points[i][0],
            points[i][1], cluster[cluster_index][0], cluster[cluster_
                index][1]);
    }
    cluster_count[cluster_index]++;
}
```

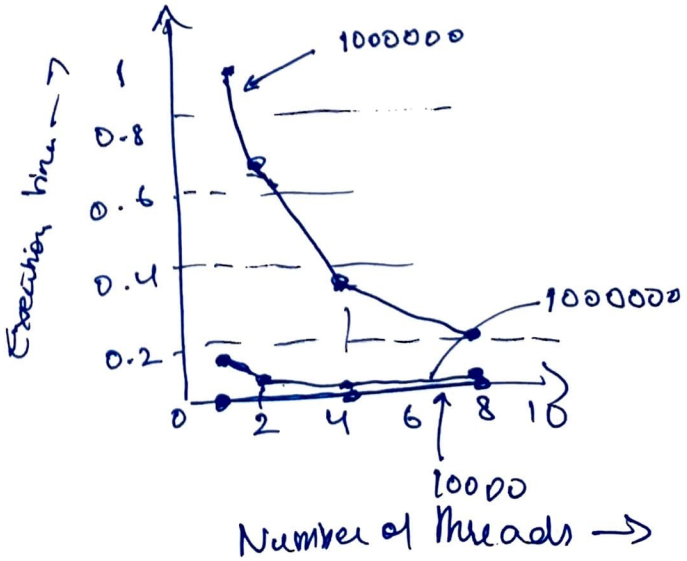
```
t = omp_get_wtime() - t;
for (i=0; i< CLUSTER_SIZE; i++) {
    printf("\n Cluster (%d, %d) : %lld", cluster[i][0],
        cluster_count[i]);
}
printf("\n\n Time taken : %.1f\n", t);
return 0;
}
```

Execution time

Input Size	1	2	4	8
10000	0.001208	0.001096	0.000922	0.00122
100000	0.011513	0.00773	0.00442	0.002891
1000000	0.114586	0.074563	0.039273	0.020269
10000000	1.141697	0.074243	0.383055	0.193872



Input size vs execution time



Number of threads vs execution time.