**Parallel implementation of K-Means**

**Linor Dolev**

Final project

Course 10324, Parallel and Distributed Computation

2017 FALL Semester

**Problem Definition**

Given a set of points in **2**-dimensional space. Initial position **(xi, yi)** and velocity **(vxi, vyi)** are known for each point **Pi**. Its position at the given time **t** can be calculated as follows:

xi(t) = xi +t\*vxi

yi(t) = yi + t\*vyi

Implement simplified K-Means algorithm to find K clusters. Find a first occurrence during given time interval [0, T] when a system of K clusters has a Quality Measure q that is less than given value QM.

**Simplified K-Means algorithm**

1. Choose first **K** points as a cluster centers.
2. Group points around the given cluster centers - for each point define a center that is most close to the point.
3. Recalculate the cluster centers (average of all points in the cluster)
4. Check the termination condition – no points move to other clusters or maximum iteration LIMIT was made.
5. Repeat from 2 till the termination condition fulfills.
6. Evaluate the Quality of the clusters found. The Quality is equal to an average of diameters of the cluster divided by distance to other clusters. For example, in case of k = 3 the quality is equal

**q = (d1/D12 + d1/D13 + d2/D21 + d2/D23 + d3/D31 + d3/D32) / 6**,

where di is a diameter of cluster **i** and Dij is a distance between centers of cluster **i** and cluster **j**.

**Algorithm Parallelization**

***Main Idea***

This solution is based on giving different processes different dt's, so each process applies K-Means with a different value of t (time), each process parallelize the task of moving all the points in time using ***NVIDIA*** GPU and the other tasks using CPU threads.

***Algorithm Steps***

1) The root process reads all points and meta-data from file

2) The root process broadcasts all the data (all points and meta-data) to the other processes

3) All the processes initiating their clusters and allocates their memory, using first k points as their centers, each cluster has his own dynamic array of points

4) Each process moves his points to his initial time using GPU, each process has his own time offset, process 0 with 0\*dT, process 1 with 1\*dT, process 2 with 2\*dT and so on.

5) Each process activates K-Means on his moved points

*Inside K-Means:*

1. Initiating or reset the clusters with the first k points as centers
2. Matching points to clusters is made by finding for each point the cluster with the closest coordinate as center, each point is independent from the others so different points can find their cluster in the same time using ***OpenMP***
3. After points moved between clusters, we recalculate their centers as the average of all their points coordinates, the different clusters are independent because their groups of points are foreign, which means we can calculate different cluster's centers in the same time using ***OpenMP***
4. If one or more of the points has changed her cluster and we haven't reached the limit of iterations, we go back to a, otherwise we stop the loop.

6) Each process calculates his quality

*Quality Measurement*:

1. First, for each cluster we calculate his diameter which is the longest distance between pair of points in that cluster, here again, we can calculate the diameter of different clusters in the same time using ***OpenMP***.
2. The calculation of the quality is the average of the clusters diameters divided by their distance to the other clusters; this task can be parallelized by using ***OpenMP******reduction*** for the sum.

7) Each process checks if his quality is less than the desired quality from the file, or he has reached the time limit from the file (T), the **stop** condition is defined as:

Stop = (Calculated Quality < Desired Quality) **OR** (Current Time >= Time Limit)

8)All the processes sharing their stop condition to check if any of them has reached quality or time limit, in case that’s true, the process that reached the stop condition sending his clusters centers coordinate, quality and time to the root process.

9) In case no one has stopped, each process moves his points in time, with time interval of: dT \* number of processes and then goes back to step 5 with the moved points.

10) In case some process actually reached quality or time limit that process sends his

clusters centers, quality and the time he reached coordinates to the root process.

11) The root process writes the quality, time and the centers of the clusters to file.

***The Rational of Choosing this Specific Architecture***

This solution offers minimum as possible communication between processes and an effective load balancing.

***Complexity evaluation***

K – number of clusters

N – number of points

L – LIMIT

T – Max time

t – number of threads

p – number of processes

*Kmeans:*

Matching points:

Calculating Centers:

Kmeans total: =

*Quality Measure:*

Calculating Diameters:

Calculating quality using diameters:

Quality Measure total:

**Total**: