Parallel K-means Algorithm Using MPI

Fan Xie fx349@nyu.edu

1. Clustering Problem Description

1.1 Description

Given a set of data points X = {x1, x2, … xn}, where each data point is a d-dimensional real vector, clustering algorithm aims to partition those data points into k sets C = {C1, C2, … Cn} so as to minimize the within-cluster sum of squares. Each point is closer to points inside than outside of its subset[1].

1.2 Mathematical Definition:

$$
\begin{aligned}
J_{k\text{-means}}(C_1,\ldots,C_k) &= \sum_{i=1}^{k}\sum_{x\in C_i} d(x,\mu(C_i))^2 \\
&= \min_{\mu_1,\ldots,\mu_k\in\mathcal{X}} \sum_{i=1}^{k}\sum_{x\in C_i} d(x,\mu_i)^2
\end{aligned}
$$

2. K-means Algorithm Description

2.1 Description

Finding the best Partitions in a clustering problem is actually an NP-hard problem[1], so we need some kind of approximation algorithm to solve it in practice. K-means is one of the heuristic algorithms to solve the clustering problem. It converges quickly to a local optimum and its performance heavily depends on the initialization step. So usually we run K-means algorithm several times to get a better result.

2.2 Algorithm

```
Input: a set of data points X = {x1, x2, .. xn} and a parameter k

Initialize: Randomly choose initial k centroids

Repeat until convergence:

    Assignment step: Compute the distance from each point xi to
    each cluster centroid Cj and assign each point to the centroid it
    is closest to

    Update step: Recompute each centroid as the mean of all
    points assigned to it
```

3. Implement K-means Algorithm Using MPI

3.1 Description

As the number of data points grows, the k-means algorithm can be very slow. So we need to find some way to parallel it. As we can see, the assignment step is the most time-cost step in this algorithm. In the assignment step, each data points does their own things individually. and no information except the k centers needs to be shared[2]. So we can easily adopt a data parallelism method to speed up the process.

3.2 Algorithm

Initialize*:*

1. *Master process read all the data points and randomly initialize k centroids.*
2. *Master process divide those data points evenly and send them to the corresponding processes (MPI_Bcast, MPI_Scatter)*

Repeated until converge:

Assignment step

1. *Master process broadcast k centroids to all the processes(MPI_Bcast)*
2. *Each process calculate new centroids for all the data points that belong to it*

Update Step

3. *Each process tells the master process what's the new centroid of each data point (MPI_Gather)*
4. *Master process recompute k new centroids base on the new assignment*

4. Code Repository

https://github.com/xiefan46/homework2/tree/master/project/k-means

5. Reference

[1] https://en.wikipedia.org/wiki/K-means_clustering

[2] http://www.goldsborough.me/c++/python/cuda/2017/09/10/20-32-46-exploring_k-means_in_python,_c++_and_cuda/