COMP 633: Parallel Computing
Programming Assignment PA2

Assigned: Mon Nov 1, 2021
Due: Tue Nov 30, 2021

Assignment

For programming assignment PA2, you can choose to implement the Lloyd k-means algorithm described in the attachment, or you can implement a computationally intensive problem of your choice, relevant to your interests. For the latter, we need to meet briefly to agree on the suitability of the problem. You can work on your own, or with a partner in the class. Please be sure you have identified the project and personnel by the end of this week (Nov 5).

You can use Phaedra for shared memory and/or the V100 GPU. If you choose to build an implementation using MPI, you will need to request permission to use the Dogwood computing cluster at UNC Research Computing, which will take some time to request and to complete runs.

Submission

You need to describe

(a) The problem you chose
(b) The implementation platform used
(c) The performance scaling of the problem at different problem size (e.g. different values of K and n, in the k-means algorithm), and with different numbers of processors. Discuss the results.

Also, please include your implementation file(s) with your submission.

Appendix A. K-means Clustering Algorithm

This section is a brief introduction to the K-means clustering algorithm. The goal of K-means clustering is to take a set of feature vectors $(x_1, x_2, ..., x_n)$ and group them into K clusters $(C_1, C_2, ..., C_3)$ so that the sum of variances within clusters is minimized:

$$\sum_{i=1}^{k} \sum_{x \in C_i} ||x - \mu_i||^2 \to \min. \tag{1}$$

Here μ_i is the centroid of cluster C_i, computed as the mean of the vectors assigned to it.

The standard algorithm for solving this workload was introduced in the 1960s, and is often referred to as Lloyd's algorithm. Various efforts have been made to improve on this standard algorithm. In 2003 Charles Elkan introduced a new algorithm that is optimized for large values of K [15]. Then in 2010, George Hamerly introduced an algorithm that is optimized for smaller values of K [6]. CFXKMeans library uses the Hamerly's algorithm for the K-means clustering, whereas scikit-learn uses Elkan's implementation. Though there have been further development in K-means clusterng algorithms, only the algorithms relevant to the paper will be described in this section: the standard algorithm first, followed by the Hamerly's algorithm.

A.1. Standard Algorithm

Standard algorithm is an iterative algorithm to find the minimum variance clusters. It starts with initial "guesses" for the cluster centroids, and these centroids are refined at each step until the algorithm converges. K-means clustering algorithm convergence time strongly depends on these initial conditions. There are numerous studies on how best to select the initial guesses. However, this is beyond the scope of this discussion. In the benchmarks presented in the paper, "guesses" are selected from the feature vectors (input).

Each step can be divided into two phases: assignment and update.

- Assignment Phase: Each feature vector (input) is assigned to the cluster that minimizes the variance. Because the Euclidean distance is the square root of the sum of squares, this is equivalent to assigning a feature vector to the closest cluster centroid.
- Update Phase: Each centroid is updated based on its current members. In other words, the centroid position is set to the vector average of the feature vectors assigned to it.

When there are no changes to the assignment in the assignment phase, then the algorithm has converged to the local minimum.

Standard algorithm is shown in Algorithm 1.

Algorithm 1 Standard algorithm

$N \leftarrow$ number of feature vectors
$K \leftarrow$ number of clusters
$F \leftarrow$ number of features
$\vec{x}_n \in \mathbb{R}^{\mathrm{N}\times\mathrm{F}}$ is feature vector n
$\vec{\mu}_c \in \mathbb{R}^{\mathrm{N}\times\mathrm{K}}$ is cluster centroid c
$A_n \in \mathbb{R}^{\mathrm{N}}$ is assignment of vector n
$\vec{C}_c \in \mathbb{R}^{\mathrm{K}\times\mathrm{F}}$ is sum of member vectors of centroid c
$M_c \in \mathbb{R}^{\mathrm{K}}$ is number of member vectors of centroid c
Initialize$(\vec{C}, A, M, \vec{x})$
while not converged **do**
    converged $\leftarrow$ true
    for $n \leftarrow 0, N$ **do**
        $c_{\min} \leftarrow$GetNearestCentroid$(\vec{x}_n, \vec{\mu})$
        if $c_{\min} \neq A_n$ **then**
            converged $\leftarrow$ false
            $\vec{C}_{A_n} \leftarrow \vec{C}_{A_n} - \vec{x}_n$
            $\vec{C}_{c_{\min}} \leftarrow \vec{C}_{c_{\min}} + \vec{x}_n$
            $M_{A_n} \leftarrow M_{A_n} - 1$
            $M_{c_{\min}} \leftarrow M_{c_{\min}} + 1$
            $A_n \leftarrow c_{\min}$
    for $c \leftarrow 0, K$ **do**
        $\vec{\mu}_c \leftarrow \vec{C}_c / M_c$

procedure GETNEARESTCENTROID$(\vec{x}_n, \vec{\mu})$
    $d_{\min} \leftarrow \infty$
    $c_{\min} \leftarrow 0$
    for $c \leftarrow 0, K$ **do**
        $d \leftarrow ||\vec{x}_n - \vec{\mu}_c||$
        if $d < d_{\min}$ **then**
            $d_{\min} \leftarrow d$
            $c_{\min} \leftarrow c$
    return $c_{\min}$
end procedure
procedure INITIALIZE$(\vec{C}, A, M, \vec{x})$
    for $n \leftarrow 0, N$ **do**
        $A_n \leftarrow 0$
        $\vec{C}_0 \leftarrow \vec{C}_0 + \vec{x}_n$
    $M_c \leftarrow N$
    for $c \leftarrow 1, K$ **do**
        $\vec{C}_c \leftarrow 0$
        $M_c \leftarrow 0$
end procedure