Parallel k-means Clustering

SF2568 – Parallel Computations for Large-Scale Problems

Lukas Bjarre **Gabriel Carrizo**
lbjarre@kth.se **carrizo@kth.se**

Abstract

This report details the implementation and analysis of a parallelisation of the k-means clustering method. A serial version of Lloyd's algorithm for computing the k-means clustering was analysed and a parallelisation stategy of dividing data points over multiple processes was proposed. The parallel implementation was a single program multiple data scheme on distributed memory using the Message Passing Interface (MPI) standard. The implementation was tested on the super computer cluster Tegner located at the PDC Center for High Performance Computing at the Royal Institute of Technology in Stockholm. <something about the results>.

1 Introduction

We have done parallelisation woop wooop

2 Theory

2.1 k-means clustering

k-means clustering is a data clustering method which clusters input data from the data set $\mathcal{X}$ into k different classes. The classes are represented by the class means μ_i and points are considered to be in a class S_i if the squared distance to the class mean is the minimum compared to the squared distance to the other class means. Formally:

$$S_i = \{\boldsymbol{x} \in \mathcal{X} : ||\boldsymbol{x} - \boldsymbol{\mu}_i||^2 \leq ||\boldsymbol{x} - \boldsymbol{\mu}_j||^2, \forall 1 \leq j \leq k\}$$

A clustering method aims to find a selection of these classes $\mathcal{S} = \{S_1, S_2, \ldots, S_k\}$ which divides the data points in some favorable way. k-means finds the placement of the class means by minimization of the summed squared distance of all class points to the class mean for all k classes:

$$\mathcal{S}_{k\text{-means}} = \arg\min_{\mathcal{S}} \sum_{i=1}^{k} \sum_{\boldsymbol{x} \in S_i} ||\boldsymbol{x} - \boldsymbol{\mu}_i||^2$$

A common algorithm to find this is Lloyd's algorithm, which iteratively classifies points according to current class means and updates them with the average of all classified points until convergence. Algorithm 1 describes this procedure in pseudocode.

Lloyd's algorithm requires multiple loops for each iteration. The most costly part is the classification of all data points, which requires computation on all d dimensions of the data set for all k classes over all n points, giving the algorithm a $\Theta(nkd)$ complexity.

Algorithm 1: Lloyd's algorithm for finding the k-means clustering class means.

```
Input: Data points $\mathcal{X} = \{\boldsymbol{x}_1, \boldsymbol{x}_2, \dots, \boldsymbol{x}_n\}$ with $\boldsymbol{x}_i \in \mathbb{R}^d \; \forall \boldsymbol{x}_i \in \mathcal{X}$, number of clusters $k$
Output: Class means $\boldsymbol{\mu}_1, \boldsymbol{\mu}_2, \dots, \boldsymbol{\mu}_k$
initialize $\bar{\boldsymbol{\mu}}_i$ on random points in $\mathcal{X} \; \forall i = 1, \dots, k$
while $\boldsymbol{\mu}_i \neq \bar{\boldsymbol{\mu}}_i \; \forall i = 1, \dots, k$ do
    for $i = 1, \dots, k$ do
        $\boldsymbol{\mu}_i \leftarrow \bar{\boldsymbol{\mu}}_i$
        $\bar{\boldsymbol{\mu}}_i \leftarrow 0$
    forall $\boldsymbol{x} \in \mathcal{X}$ do
        class $\leftarrow \arg\min_k ||\boldsymbol{x} - \boldsymbol{\mu}_k||^2$
        count$_{\text{class}} \leftarrow$ count$_{\text{class}} + 1$
        $\bar{\boldsymbol{\mu}}_{\text{class}} \leftarrow \bar{\boldsymbol{\mu}}_{\text{class}} + \boldsymbol{x}$
    for $i = 1, \dots, k$ do
        $\bar{\boldsymbol{\mu}}_i \leftarrow \frac{\bar{\boldsymbol{\mu}}_i}{\text{count}_i}$
        count$_i \leftarrow 0$
return $\{\boldsymbol{\mu}_1, \boldsymbol{\mu}_2, \dots, \boldsymbol{\mu}_k\}$
```