15-440: Project 3

Clustering Data Points and DNA Strands Using MPI

Fall 2015

Project posted on: November 2, 2015

Due date: November 16, 2015

Intended Learning Outcomes

This project applies the theory of the popular programming model, Message Passing. The main learning outcome of the project is to apply Message Passing Interface (MPI), a library standard for writing message passing programs, to a popular real problem, namely cluster analysis using the K-Means algorithm.

Project Objectives

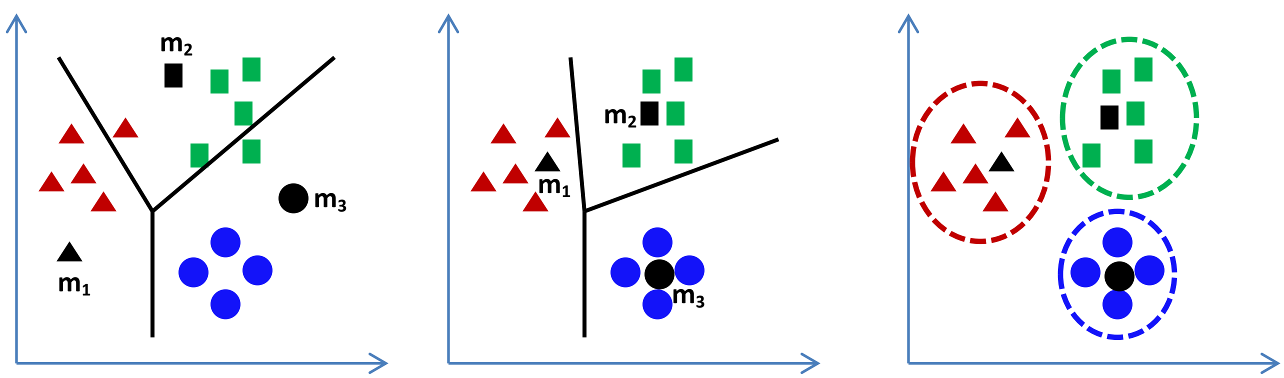
The overall goal of this project is to get a clear understating on how to apply MPI to real problems. We have chosen a clustering analysis algorithm (i.e., K-Means) due to its significance and importance in various domains including, but not limited to, data mining and statistical data analysis. For whatever domain our students will be in, the chances are that sooner or later they will run into a clustering problem. The project potentially provides our students with a practical experience augmented with a methodology for solving clustering and other similar problems on a distributed system using MPI. The students will also conduct and analyze some scalability studies on various degrees of parallelism and data set sizes.

Cluster Analysis

Cluster analysis or **clustering** is the task of assigning a set of objects into groups (called **clusters**) so that the degree of similarity can be strong between members of the same cluster and weak between members of different clusters. In short, clustering has to define some notion of "similarity" among objects. The objective is to maximize intra-cluster similarity and minimize inter-cluster similarity.

Clustering problems arise in many different applications such as visualization (e.g., visualizing the stock market data to give individuals/institutions useful information about the market behavior for investment decisions), data mining and statistical data analysis including machine learning, pattern recognition, image analysis, information retrieval, and bioinformatics.

Among clustering formulations that are based on minimizing a formal objective function, perhaps the most widely used and studied one is K-Means algorithm. Simply put, K-Means is an iterative algorithm that attempts to find K similar groups in a given data set via minimizing a mean squared distance function. Initial guesses of K means ($\mathbf{m_1}$, $\mathbf{m_2}$, …, $\mathbf{m_K}$) is firstly made (see Fig. 1(a)). These estimated means are then used to classify the data set objects into K clusters. Afterwards, each mean is recomputed so as to reflect the true mean of its constituent objects (see Fig. 1(b)). The algorithm keeps iterating until the recomputed means (almost) stop varying (see Fig. 1(c)).



(a) Initial Means **(b) Recalculated Means** **(c) Final Clusters**

Fig. 1

In this project we will apply K-Means clustering to two different applications, data points in a 2D plane and DNA strands in biology.

Clustering Data Points

Considering a case of a data set composed of data points in ***d-dimensional*** space $\boldsymbol{R^d}$. In K-Means clustering, we specify a set of ***n*** data points and an integer ***k***. The problem then is to determine a set of ***k*** points in $\boldsymbol{R^d}$, called ***centroids***, so as to minimize the

mean squared distance from each data point to its nearest center. In pseudo code, it is shown by Alpaydin (Introduction to Machine Learning, page 139) that K-Means essentially follows the following procedure:

Initialize m_i to k random x^d, for i = 1, …., k and x^d Є X that contains each of our d-dimensional data point.

Repeat:

For all x^d in X

$b_i^d \leftarrow 1$ if $||x^d - m_i|| = \min_j ||x^d - m_j||$

$b_i^d \leftarrow 0$ otherwise

For all m_i, i = 1, …., k

$m_i \leftarrow$ sum over $b_i^d x^d$/sum over b_i^d

Until m_i converge

Explained in plain English, K-Means roughly follows this approach:

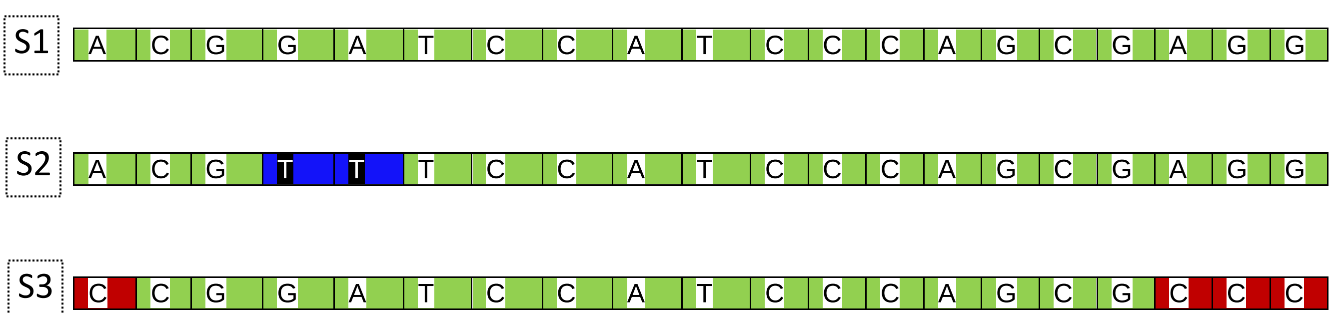
1. We start by deciding how many clusters we would like to form from our data. We call this value ***k***. The value of ***k*** is generally a small integer, such as 2, 3, 4, or 5, but may be larger.
2. Next we select ***k*** points to be the centroids of ***k*** clusters which at present have no members. The list of centroids can be selected by any method (e.g., randomly from the set of data points). It is usually better to pick centroids that are far apart.
3. We then compute the ***Euclidean distance*** (*the similarity function with a data set of data points*) from each data point to each centroid. A data point is assigned to a cluster such that its distance to that cluster is the smallest among all other distances.
4. After associating every data point with one of ***k*** clusters, each centroid is recalculated so as to reflect the true *mean* of its constituent data points.
5. Steps 3 and 4 are repeated for a number of times (say **μ**), essentially until the centroids start varying very little.

The positive integer **μ** is known as number of K-Means iterations. The precise value of **μ** can vary depending on the initial starting cluster centroids, even on the same data set.

In this project, you will provide sequential and parallel implementations of the above K-Means algorithm with a data set of data points as input and K centroids as output.

Clustering DNA Strands

Bioinformatics involves the manipulation, searching, and data mining of biological data, and this includes DNA sequence data. A strand of DNA consists of a string of molecules called bases, where the possible bases are adenine (A), guanine (G), cytosine (C), and thymine (T). We can express a strand of DNA as a string over the finite set **{**A, C, G, T**}**. String searching or matching algorithms, which find an occurrence of a sequence of letters inside a larger sequence of letters, or simply match two sequences of letters, is widely used in genetics (e.g., for studying various phylogenetic relationships and protein functions). In many studies, we often want to compare the DNA of two (or more) different organisms. One goal of comparing two strands of DNA is to determine how "similar" the two strands are, as some measure of how closely related the two organisms are. ***Similarity in such a scenario can be defined as a function F(. , .) of the number bases in a strand subtracted from the number of changes required to turn one strand into the other***. For example consider the following three DNA strands:



The similarity between S1 and S2 is denoted as ***F*(S1, S2)** and is equal to 18. On the other hand, ***F*(S1, S3)** = 16. The K-Means algorithm, described in the previous section, can be applied to DNA strands with this given similarity function ***F(. , .)*** to compare DNA of two or more different organisms.

In this project, you will provide sequential and parallel implementations of the K-Means algorithm with a data set of DNA strands as input and K centroids as output.

Implementation Guidelines

As stated earlier, in this project, you will provide sequential and parallel implementations for K-Means with two types of data sets, a data set of data points and a data set of DNA strands. For simplicity we assume 2D data points. Furthermore, we assume that strands in the DNA data set are equal in size, and that strands in the list of centroids are also equal in size to each other and equal in size to every strand in the data set.

For the sequential implementation, use C, C++, or Java. For the parallel implementation you will provide an ***MPI-based*** version using **MPICH2**, a high performance and widely portable implementation of the Message Passing Interface (MPI) standard (both MPI-1 and MPI-2).

In addition, you have to write your own data set generator that generates a random number ***P*** of DNA strands per cluster for ***k*** clusters (use any programming language you like). We will provide you with a data set generator that generates a random set of 2D data points.

Your sequential and MPI-based K-Means implementations should be tested and run on a data set of 2D data points, generated by the given data set generator, and on a data set of DNA strands, generated by your DNA strands data generator.

Experimentation and Analysis

Please conduct and provide the following:

- A comparison between your 2 different K-Means implementations in terms of performance and development effort.

- Three scalability studies for your MPI version on:
    - The number of processes with a fixed data set size (use only the data set of the 2D data points for this study). Specifically, use 2, 4, 8, and 12 processes on 4 virtual machines (VMs).
    - The number of VMs with a fixed data set size (again, use only the data set of the 2D data points for this study). Specifically, use 1, 2, 3 and 4 VMs with a fixed number of processes (e.g., 8).
    - The number of data points in your data set of the 2D data points with a fixed number of processes (e.g., 8) and a fixed number of VMs (e.g., 4). Specifically, use 20 million, 30 million, and 40 million data points.

- A discussion on:

    - Your experience in applying MPI to the K-Means clustering algorithm.
    - Your insights concerning the performance trade-offs of MPI and sequential programming with K-Means.
    - Your scalability studies (a detailed analysis on the collected results should be provided).
    - Your thoughts on the applicability of K-Means to MPI.
    - Your recommendations regarding the usage of MPI for algorithms similar to K-means.

Deliverables

As deliverables, you should submit on November 16, 2015 (by midnight):

1- An archive containing a fully tested and debugged code for your data set generator as well as your sequential and MPI K-Means implementations.

2- An article with a maximum of 5 pages (similar to research articles) that presents your solution, findings, observations and analysis.

Handing In the Project

Submit your code using the AFS file system:
/afs/qatar.cmu.edu/usr10/mhhammou/www/15440-f15/handin/p3/userid/,
where userid is your andrew user id.

Late Policy

- If you hand in on time, there is no penalty (duh!).
- 0-24 hours late = 25% penalty.
- 24-48 hours late = 50% penalty.
- More than 48 hours late = you lose all the points for this project.

NOTE: You can use your grace-days quota. For details about the grace-days quota, please refer to the course syllabus.