High-Performance Computing

MPI: K-Means Clustering

Anand Kamble
Department of Scientific Computing
Florida State University

1 Introduction

This project implements the K-Means clustering algorithm using MPI (Message Passing Interface) for parallel processing of images. The algorithm aims to segment an input image into k clusters based on the color values of the pixels.

2 Implementation

The source code for this program is written in the file `main.cpp`. It utilizes the `JpegLib` library to read and write JPEG image files.

2.1 Data distribution

The input image data is divided equally among the processes, with the master process handling any remaining rows. Each process receives its portion of the data through an `MPI_Scatter` operation. And the remaining part is processed by master, this part is manually added to the workerData variable which belongs to the master process.

2.2 K-Means clustering

The K-Means clustering algorithm is performed in an iterative manner for a fixed number of iterations (defined by the `ITERATIONS` constant). The following steps are executed in each iteration:
1. The initial cluster centers (generators) are selected from the first k pixels of the image by the master process.
2. The generators are broadcasted to all processes using `MPI_Bcast`.
3. Each process computes the distance between its assigned pixels and the generators, assigning each pixel to the closest generator (cluster).
4. The new generator values are computed as the average of the pixels assigned to each cluster.
5. The new generator values are reduced to the master process using `MPI_Reduce`.

After the iterations, the master process assigns the color of the corresponding generator to each pixel, creating the segmented image.

3 Errors and Debugging

I am not able to get a good image output using this code. Although, I am getting correct values for the generators and the distances, the problem is with the reduce and gather methods, where I have some errors in the types of the variables.
This is the output I got

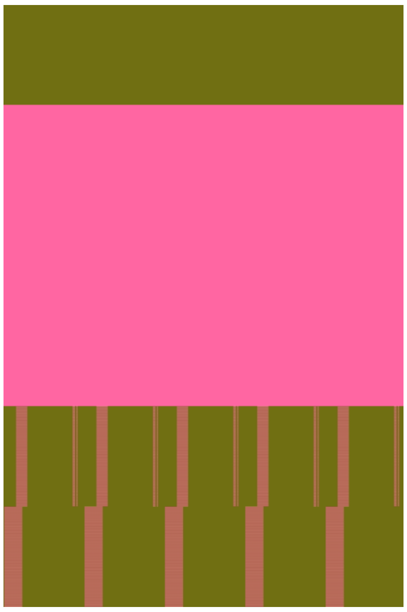

Figure 1: Caption

4 Results

The program is scaling well on multiple number of processes even though it might not produce the desired output image.

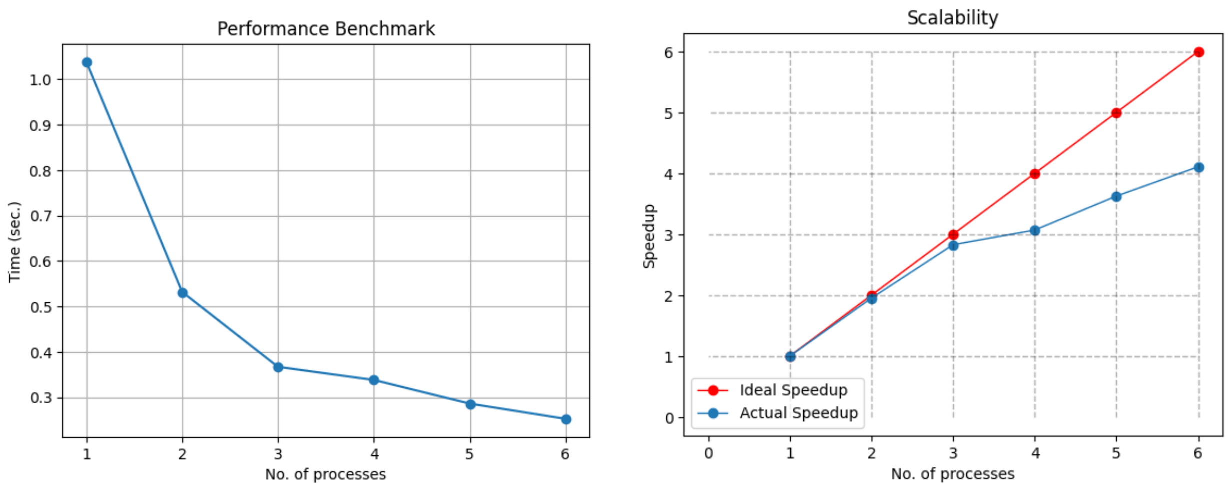

Figure 2: Timing and Scaling of the program