Task 2 - distributed

How to run this task

First, this project uses ‘cmake’ and ‘make’. Make sure you have both of them installed. Second, we are going to use ‘c++ 11’ so make sure you have the correct version and updated ‘gcc’ compiler installed.

For the image processing, we are using the OpenCV library (for loading/ storing/ writing images). Install the library using the following command (on macOS) ‘bash brew install opencv’. Note: This will also install all the dependencies required for the opencv library.

We are going to use ‘OpenMPI 3.0’ library for distributing the work among multiples nodes. In order to install this library please check the link: https://wiki.helsinki.fi/display/HUGG/Open+MPI+install+on+Mac+OS+X

For building the project use the following commands:
‘bash’, ‘cmake .’ and ‘make’

A ‘Filters.o’ executable file should be created by now. In order to apply the grayscale filter and the other tree filters to your image, run:
‘mpirun --host list_of_hosts ./Filters.o path_to_your_image’

For example run ‘mpirun --host localhost,localhost ./Filters.o ~/image.jpg’ to apply the filters to image.jpg on home directory with two nodes, one on the localhost machine.

Algorithm

We will have a master node that dicatates work to the other worker nodes.
After the master has distributed the work to the nodes, it keeps one part to himself, solves it and then assembles all of the results together.

We split the image on row chunks. And each worker will get a chunk of image rows.
For example if we have the following matrix:
‘”
```
------row-0-------
------row-1-------
------row-2-------
------row-3-------
------row-4-------
------row-5-------
```
‘”
For `n = 3` nodes we can split the matrix in the following way:
‘”
```
------chunk-1-----
------chunk-1-----
------chunk-2-----
------chunk-2-----
------chunk-3-----
------chunk-3-----
```
‘”

First, the master will keep the first chunk (`chunk-1`), will send the two other chunks to the slaves, will compute it's part and in the end collect the result from the slaves.

Synchronization

If the master finishes before the other nodes, he must wait for all of the others to finish. This is automatically done by the `MPI_Recv()` method.

Performance measurements

In order to test the perfomance, we can also make the master generate an arbitrary sized image.
In order to run the performance measurements type into terminal the following command:
‘chmod +x performance.sh’ and ‘./performance.sh’
Here is an example of the output of the above commands:

Number of Nodes	Time
1	.31610800
2	.330647000
3	.369201000
4	.373882000