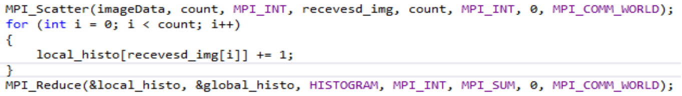
Histogram Equalization

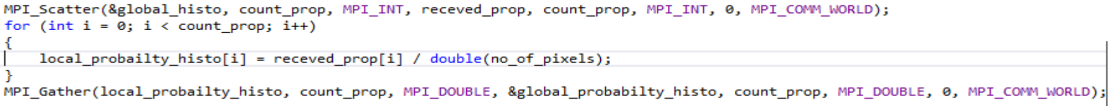
Flow of Code:

- Master node → 0, send for the rest of others
- Count pixels in image by use scatter&reduce according to (count = imageSize/#processors)



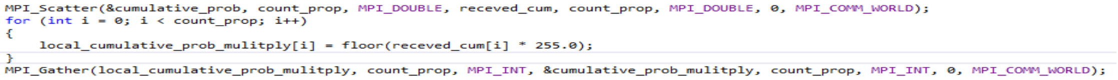
```
MPI_Scatter(imageData, count, MPI_INT, recevesd_img, count, MPI_INT, 0, MPI_COMM_WORLD);
for (int i = 0; i < count; i++)
{
    local_histo[recevesd_img[i]] += 1;
}
MPI_Reduce(&local_histo, &global_histo, HISTOGRAM, MPI_INT, MPI_SUM, 0, MPI_COMM_WORLD);
```

- Calculate Probability by use scatter&gather according to (count = #Histogram/#processors)



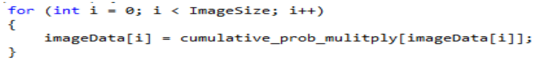
```
MPI_Scatter(&global_histo, count_prop, MPI_INT, receved_prop, count_prop, MPI_INT, 0, MPI_COMM_WORLD);
for (int i = 0; i < count_prop; i++)
{
    local_probailty_histo[i] = receved_prop[i] / double(no_of_pixels);
}
MPI_Gather(local_probailty_histo, count_prop, MPI_DOUBLE, &global_probabilty_histo, count_prop, MPI_DOUBLE, 0, MPI_COMM_WORLD);
```

- Calculate Cumulative probability by use ‘`global_probabilty_histo`’
- Then scale Cumulative probability by floor(Cumulative probability * 255).



```
MPI_Scatter(&cumulative_prob, count_prop, MPI_DOUBLE, receved_cum, count_prop, MPI_DOUBLE, 0, MPI_COMM_WORLD);
for (int i = 0; i < count_prop; i++)
{
    local_cumulative_prob_mulitply[i] = floor(receved_cum[i] * 255.0);
}
MPI_Gather(local_cumulative_prob_mulitply, count_prop, MPI_INT, &cumulative_prob_mulitply, count_prop, MPI_INT, 0, MPI_COMM_WORLD);
```

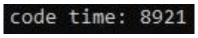
- Map scaled values to new image



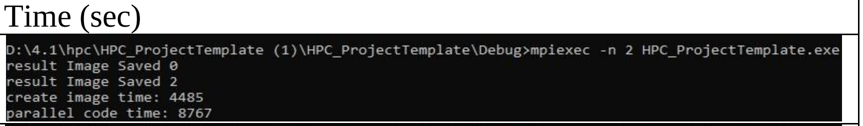
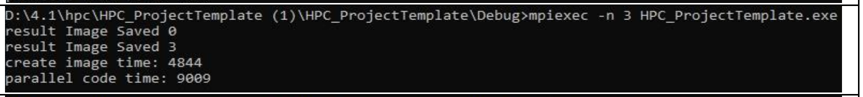
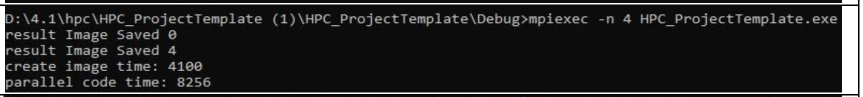
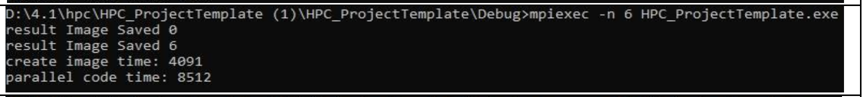
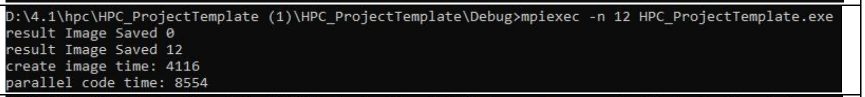
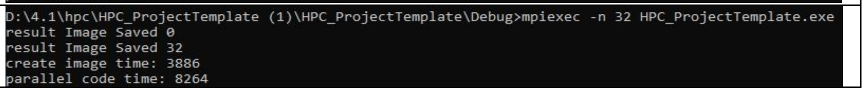
```
for (int i = 0; i < ImageSize; i++)
{
    imageData[i] = cumulative_prob_mulitply[imageData[i]];
}
```

Calculate Time:

- Sequential time: `code time: 8921` sec.

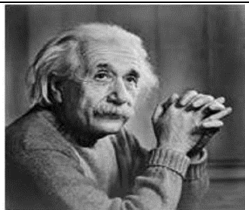


- At parallel time:

Number of processors (n)	Time (sec)
n = 2	 D:\4.1\hpc\HPC_ProjectTemplate (1)\HPC_ProjectTemplate\Debug>mpiexec -n 2 HPC_ProjectTemplate.exe result Image Saved 0 result Image Saved 2 create image time: 4485 parallel code time: 8767
n = 3	 D:\4.1\hpc\HPC_ProjectTemplate (1)\HPC_ProjectTemplate\Debug>mpiexec -n 3 HPC_ProjectTemplate.exe result Image Saved 0 result Image Saved 3 create image time: 4844 parallel code time: 9009
n = 4	 D:\4.1\hpc\HPC_ProjectTemplate (1)\HPC_ProjectTemplate\Debug>mpiexec -n 4 HPC_ProjectTemplate.exe result Image Saved 0 result Image Saved 4 create image time: 4100 parallel code time: 8256
n = 6	 D:\4.1\hpc\HPC_ProjectTemplate (1)\HPC_ProjectTemplate\Debug>mpiexec -n 6 HPC_ProjectTemplate.exe result Image Saved 0 result Image Saved 6 create image time: 4091 parallel code time: 8512
n = 12	 D:\4.1\hpc\HPC_ProjectTemplate (1)\HPC_ProjectTemplate\Debug>mpiexec -n 12 HPC_ProjectTemplate.exe result Image Saved 0 result Image Saved 12 create image time: 4116 parallel code time: 8554
n = 32	 D:\4.1\hpc\HPC_ProjectTemplate (1)\HPC_ProjectTemplate\Debug>mpiexec -n 32 HPC_ProjectTemplate.exe result Image Saved 0 result Image Saved 32 create image time: 3886 parallel code time: 8264

Conclusion:

Time increasing from start until #processors = 4 time decrease, when #p > 4 then time increasing. **(#p = 4 is the best number of processors to solve this problem in the least time)**

	→