Lab 5 – Hybrid sobel operator implementation – MPI + OpenMP

Submitted by – Shivani Sabhlok [2:20PM batch]
500237896

Question 1 - Compare your run-time with your serial and cuda parallel versions from lab4. Did your achieve the best performance with the serial or a parallel version? Why?

Answer 1- All the versions of sobel operator implementation on converged in 50 iterations on coins.bmp.
Timing results from previous lab –

Device/Host	Time(sec)
CPU	.139
GPU	.014 (best result)

Timing result on lab 5

Parallelization technique	Time(sec)
MPI	.007
MPI + OpenMP	.32 (best result) [32 threads]

It is evident that the best timing results were achieved on the MPI program.
It was expected that the parallel version would perform better than the serial version. With MPI, it was equivalent to running 16 serial problems and thus faster results.
With OpenMP in the MPI program, the performance degraded. I believe that the overload of creating and destroying threads on each MPI process on each row iteration was way more than the benefit of parallelizing column processing for each row. This was observed on not just the coins image(246X300) but also on other sample images too of size 445X640 and 768X1366.
Probably the advantage of hybrid MPI+OpenMP model would be evident on images that are wide and not too long in height because I used OpenMP to parallelize columns on each row and used disposable version on OpenMP.

Question 2 - Explain your workload distribution strategy and why you selected it.

Answer 2 - For the MPI version, I allocated blocks of rows to each MPI process and for the OpenMP , I used the default OpenMP block distribution for parallelizing columns of each row.
Each MPI process except the last one was allocated same number of rows equal to the ceil(height/16). This was a straightforward distribution strategy. I tried using OpenMP threads for parallelizing the rows of each process , that is , each OpenMP threads process a row of a MPI process. I observed better performance with parallelizing columns using OpenMP.

Question 3 - Report and comment any reduction and/or synchronization mechanisms you used.

Answer 3 - To reduce the percent black cells from all the MPI processes, I used MPI_Allreduce MPI_IN_INPLACE.
I tried using MPI_Allreduce to rank 0 MPI process but that was complicated to manage with OpenMP parallelization.

Upon convergence, I used MPI_Gather to gather the chunks of rows from all MPI processes at the rank 0 MPI process. As mentioned earlier, all but last MPI process have same number of rows chunk. Thus an if.. else block was added to handle the case when the last MPI process might send lesser data.
Instead of MPI_Allreduce, I might have used reduce on MPI process 0 and then broadcast to all MPI processes.
Instead of using an if..else on MPI gather, I might have given a variable named size to each MPI process and could have gathered data same as the size of that MPI process but I chose not to do it just to make it explicit how the distribution is.

Question 4 - Were there any surprises you encountered in this exercise?

Answer 4 - It was expected that MPI would perform better than serial, but it was surprising that it outperformed CUDA version too. Probably this is because of the overhead of kernel calls. I would believe, CUDA would perform better if we have large images. On a small image, the overhead overshadowed the power of CUDA.
Even the observation that including OpenMP on MPI degraded the performance was slightly surprising. In the lab on OpenMP persistent and disposable versions we did learn that the overhead incurred on thread creation and disposal is high but none of the OpenMP thread combinations helped in this case. I tried 2,4,8,16,32,64 and 128 threads.

Results summary for lab 5 –

Only MPI on coins.bmp

```
********************************************************************************
Time taken for MPI operation: 0.007371 secs
Threshold during convergence:: 50


********************************************************************************
```

MPI With OpenMP with 4 threads on coins.bmp

```
********************************************************************************
Time taken for MPI operation: 5.129242 secs
Threshold during convergence:: 50


********************************************************************************
```

MPI With OpenMP with 64 threads on coins.bmp

```

********************************************************************************
Time taken for MPI operation: 0.582120 secs
Threshold during convergence:: 50


********************************************************************************
```

MPI With OpenMP with 32 threads on coins.bmp

```
********************************************************************************
Time taken for MPI operation: 0.320503 secs
Threshold during convergence:: 50


********************************************************************************
```

Only MPI on other test image

```
Image Info ::
         Height=768 Width=1366


********************************************************************************
Time taken for MPI operation: 0.231134 secs
Threshold during convergence:: 390


********************************************************************************
```

MPI With OpenMP with 32 threads on other test image

```
Image Info ::
         Height=768 Width=1366


********************************************************************************
Time taken for MPI operation: 10.373679 secs
Threshold during convergence:: 390


********************************************************************************
```