Shiny Patel CSE 5441 – Section 12:45 pm

Lab 5 Report

Output:

```
---------------------------------------------------------------------
Image Info ::
        Height=246 Width=300

Time taken by master process during convergence loop: 1.49 s
Sum of time taken by all MPI processes during convergence loop: 23.89
s
Threshold: 49
---------------------------------------------------------------------
```

coins.bmp	**Serial**	**Cuda**	**MPI (and OMP)**
Time	100 ms	3.7 ms	1.49 s

Other MPI results:

imgs	**Time (taken by master process) (s)**	**Time taken by all processes (s)**
fox.bmp	1.43	22.90
map.bmp	2.79	44.74
cave.bmp	2.93	46.96
cliff.bmp	3.77	60.44

- **Compare your run-time with your serial and cuda parallel versions from lab4. Did you achieve the best performance with the serial or a parallel version? Why?**
  For coins.bmp
  The Cuda parallel version gave the best run-time (3.7 ms) followed by serial version (100 ms) followed by MPI (1.49 s). MPI was slower than Cuda as all the work was done on only 1 GPU. Overall, MPI gave the worst run-time amongst the three and the advantage of using it was not apparent for such a small file as coins.bmp.

- **Explain your workload distribution strategy and why you selected it.**
  The program was designed using block distribution. Each MPI process handled (rank * quotient + 1) 'i' values from serial version (where quotient = (height – 2)/# of processes).
  The inner j-for loop was parallelized using 2 (which gave the best results; see table below) OMP threads for each process.
  This implementation provides the advantage of not using any barriers in program which significantly improves the running time.

  on cliff.bmp (83.18 MB)

# OMP threads	**Time (taken by master process) (s)**	**Time taken by all processes (s)**
2	3.77	60.44
4	5.86	93.87
8	29.36	469.88
16	71.68	1147.55

32	5.03	80.47
64	8.39	134.19
128	16.19	259.03
256	32.82	525.09

- **Report and comment onany reduction and/or synchronization mechanisms you used.**
  The program reported aggregate time taken by all MPI processes calculated using MPI_Reduce. MPI_Reduce and OMP reduction were used during convergence loop to sum black_cells into percent_black_cells which was then broadcasted to all processes using MPI_Broadcast to check the condition of convergence loop.
  At the end of convergence loop, all the slave processes sent their corresponding new pixel values to master process using MPI_Send who collected them using MPI_Recv.

- **Were there any surprises you encountered in this exercise?**
  No.

  Note: The extra test images are available here:- https://osu.app.box.com/v/CSE5441TestImages