MyOcean

Lunatic Ian

September 2023

1 Process

Shown bellow is the main portions of both the MyOcean and MyOcean_OMP algorithm as described in class. Each non-border index is averaged using itself and the values in the index to the north, south, east, and west. This implementation uses the Red-Black method of traversal in where each index of the matrix is either red or black. On even time steps, the red indexes are updated and averaged, and the same is done to the black indexes on odd time steps.

myocean.cpp > main()

```
57      for(i = 0; i < time; ++i)
58      {
59          if(i % 2 == 0)
60          {
61              for(j = 1; j < (int)mesh.size()-1; ++j)
62              {
63                  if(j % 2 == 0) r = 1;
64                  else           r = 2;
65                  for(; r < (int)mesh[j].size()-1; r += 2)
66                  {
67                      float north, south, east, west;
68                      north = mesh[j-1][r];
69                      south = mesh[j+1][r];
70                      east = mesh[j][r+1];
71                      west = mesh[j][r-1];
72                      mesh[j][r] = (north + south + east + west + mesh[j][r])/5.0;
73                  }
74              }
75          }
76          else
77          {
78              for(j = 1; j < (int)mesh.size()-1; ++j)
79              {
80                  if(j % 2 == 0) b = 2;
81                  else           b = 1;
82                  for(; b < (int)mesh[j].size()-1; b += 2)
83                  {
84                      float north, south, east, west;
85                      north = mesh[j-1][b];
86                      south = mesh[j+1][b];
87                      east = mesh[j][b+1];
88                      west = mesh[j][b-1];
89                      mesh[j][b] = (north + south + east + west + mesh[j][b])/5.0;
90                  }
91              }
92          }
```

myocean_omp.cpp > main(int, char **)

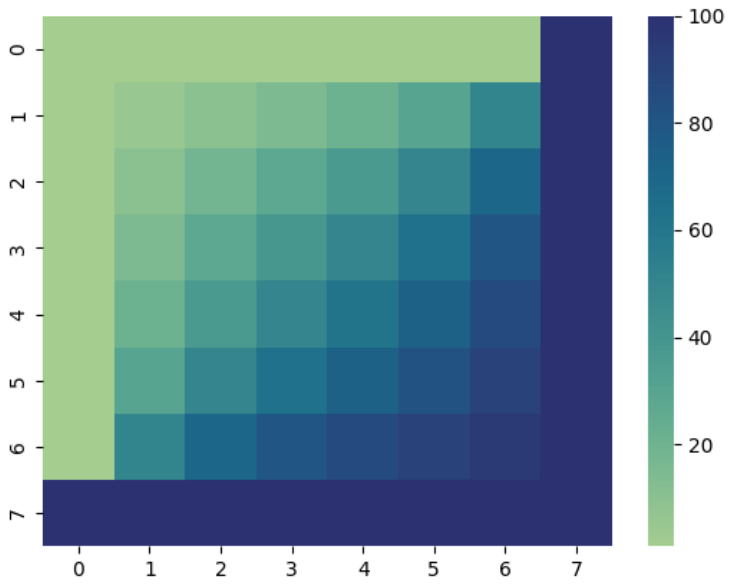
```
71      start = omp_get_wtime();
72      for(i = 0; i < time; ++i)
73      {
74          if(i % 2 == 0)
75          {
76              #pragma omp parallel for
77              for(j = 1; j < (int)mesh.size()-1; ++j)
78              {
79                  #pragma omp parallel for
80                  for(r = j % 2 + 1; r < (int)mesh[j].size()-1; r += 2)
81                  {
82                      float north, south, east, west;
83                      north = mesh[j-1][r];
84                      south = mesh[j+1][r];
85                      east = mesh[j][r+1];
86                      west = mesh[j][r-1];
87                      mesh[j][r] = (north + south + east + west + mesh[j][r])/5.0;
88                  }
89              }
90          }
91          else
92          {
93              #pragma omp parallel for
94              for(j = 1; j < (int)mesh.size()-1; ++j)
95              {
96                  #pragma omp parallel for
97                  for(b = 2 - j % 2; b < (int)mesh[j].size()-1; b += 2)
98                  {
99                      float north, south, east, west;
100                     north = mesh[j-1][b];
101                     south = mesh[j+1][b];
102                     east = mesh[j][b+1];
103                     west = mesh[j][b-1];
104                     mesh[j][b] = (north + south + east + west + mesh[j][b])/5.0;
105                 }
106             }
107         }
```

The only difference between the two programs being the use of the OMP library to parallelize the traversal and the calculations of which indices to average. Bellow are the identical resulting mesh matrices using the provided myocean.in.short file, the serialized one on the left and the parallelized one on the right using 8 threads. As seen they are symmetrical through the (x, x) diagonal and converges to the boundary condition of 100.

```
Finalized Matrix:
    1     1     1     1     1     1     1   100
    1  5.51    10  14.9    21  30.7  50.5   100
    1    10  18.7  27.4  37.3  50.4  70.2   100
    1  14.9  27.4  38.9  50.4  63.6    80   100
    1    21  37.3  50.4  61.9  73.4  86.1   100
    1  30.7  50.4  63.6  73.4  82.2  90.9   100
    1  50.5  70.2    80  86.1  90.9  95.5   100
  100   100   100   100   100   100   100   100
```

```
Initialization TIME 0.00103s
Red+Black TIME 0.00116s

Finalized Matrix:
    1     1     1     1     1     1     1   100
    1  5.51    10  14.9    21  30.7  50.5   100
    1    10  18.7  27.4  37.3  50.4  70.2   100
    1  14.9  27.4  38.9  50.4  63.6    80   100
    1    21  37.3  50.4  61.9  73.4  86.1   100
    1  30.7  50.4  63.6  73.4  82.2  90.9   100
    1  50.5  70.2    80  86.1  90.9  95.5   100
  100   100   100   100   100   100   100   100
```



The above heat map is another visual representation of the resulting mesh matrix using colors to show that the algorithm does result in a gradient on the permeter of the grid as described in the lab write up.

2 Report

For the OMP of this lab, we were asked to report on the times it took for the program to finish on the provided myocean.in file (a 64x64 matrix over 1000 timesteps) using 1, 2, 4, 6, and 8 total threads. The initializing process of the whole matrix on my system generally takes about .001 seconds as that portion was not parallelized. The Red-Black traversal sequential version (1 thread) took about .05561 seconds, 2 threads took .02919 seconds almost halfing the time, 4 threads took 0.162 seconds, and 8 threads took .01386 seconds. As seen in the total time for 4 and 8 threads there seems to be a diminishing return in the use of threads.

```
lobelia@DESKTOP-HJ0FHTJ:/mnt/d/Documents/College Classes/Comp Sci 462 Parallel Programing/Parallel-OpenMP$ ./myocean_omp 1 < myocean.in
Please enter all in one line: X_max, Y_max, time_steps> Please enter each row of the matrix (ctrl+D to stop):
Initialization TIME 0.00101s
Red+Black TIME 0.05561s
lobelia@DESKTOP-HJ0FHTJ:/mnt/d/Documents/College Classes/Comp Sci 462 Parallel Programing/Parallel-OpenMP$ ./myocean_omp 2 < myocean.in
Please enter all in one line: X_max, Y_max, time_steps> Please enter each row of the matrix (ctrl+D to stop):
Initialization TIME 0.00099s
Red+Black TIME 0.02919s
lobelia@DESKTOP-HJ0FHTJ:/mnt/d/Documents/College Classes/Comp Sci 462 Parallel Programing/Parallel-OpenMP$ ./myocean_omp 4 < myocean.in
Please enter all in one line: X_max, Y_max, time_steps> Please enter each row of the matrix (ctrl+D to stop):
Initialization TIME 0.00100s
Red+Black TIME 0.01620s
lobelia@DESKTOP-HJ0FHTJ:/mnt/d/Documents/College Classes/Comp Sci 462 Parallel Programing/Parallel-OpenMP$ ./myocean_omp 8 < myocean.in
Please enter all in one line: X_max, Y_max, time_steps> Please enter each row of the matrix (ctrl+D to stop):
Initialization TIME 0.00103s
Red+Black TIME 0.01386s
lobelia@DESKTOP-HJ0FHTJ:/mnt/d/Documents/College Classes/Comp Sci 462 Parallel Programing/Parallel-OpenMP$
```