Michael Gibson
9/27/2023
Cosc 462

For myocean.c, I used scanf to get all my input.

```
void main(){
    double start, end;
    start = omp_get_wtime();
    int xsize, ysize, stepcount;
    scanf("%d", &xsize);
    scanf("%d", &ysize);
    scanf("%d", &stepcount);
    float grid[xsize][ysize];
    for(int i = 0; i < ysize; i++){
        for(int p = 0; p < xsize; p++){
            scanf("%f", &grid[p][i]);
        }
    }
```

This part takes minimal time compared to the computation of average temperature.

```
for(int curstep = 0; curstep < stepcount; curstep++){
    int redblack = (curstep%2);
    for(int i = 1; i < ysize - 1; i++){
        for(int p = redblack+(i%2); p < xsize - 1; p = p+2){
            if(p != 0){
            float averagetmp;
            averagetmp = (grid[p][i+1] + grid[p][i-1] + grid[p-1][i] + grid[p+1][i] + grid[p][i])/5;
            grid[p][i] = averagetmp;
            }
        }
    }
}
```

This step calculates the average temperature of either an even or odd part of the grid every count. Then it places the average of the north, south, west, east, and center temp of the grid to average the temp at the center. Redblack is what keeps track of whether we are on even or odd.

Lastly, we print out the grid and the time.

```
    for(int i = 0; i < ysize; i++){
        printf("\n");
        for(int p = 0; p < xsize; p++){
            printf(" %f", grid[p][i]);
        }
    }
    end = omp_get_wtime();
    printf("\n");
    printf("TIME %.5f s\n", end-start);
}
```

This program would take about .00250 seconds to run myocean2.in

For myocean-omp.c, the input has changed slightly. Now the first thing on the input is the amount of threads. So 2 64 64 100 grid would be 2 threads, 64x64 size, 100 steps, and then the grid itself.

```
    scanf("%d", &threadnum);
```

I implemented #pragma omp parallel right before the curstep for loop, and implement the #pragma omp for right before the outer nested loop for i. I had to pad the grid by 8 to help improve performance, as my program suffers from false share, and this is the best solution I could find.

```
int i, p;
#pragma omp parallel private(i, p) shared(grid)
for(int curstep = 0; curstep < stepcount; curstep++){
    int redblack = (curstep%2);
    #pragma omp for
    for(i = 1; i < ysize-1; i++){

        for(p = redblack+(i%2); p < xsize - 1; p = p+2){
            if(p != 0){
            float averagetmp;
            averagetmp = (grid[8*(p+1)][8*i] + grid[8*(p-1)][8*i] + grid[8*p][8*(i+1)] + grid[8*p][8*(i-1)] + grid[8*p][8*i])/5;

            grid[p*8][i*8] = averagetmp;
        }
        }

    }
    #pragma omp barrier
}
```

The performance is worse at lower amounts of threads, and only marginally better at higher amounts

The amount of time threads took are as followed:

1 thread = .00270s

2 threads = .00244s

4 threads = .00195s

8 threads = .00220s
The output code has not changed.

To use Makefile, type *make myocean* for myocean, and *make myocean-opm* for myocean-opm.