Maseeh College of Engineering and Computer Science
PORTLAND STATE UNIVERSITY

ECE 688: WINTER 2024 - HW3
ADVANCED COMPUTER ARCHITECTURE II

HEAT DIFFUION GRID HOMEWORK 3 PTHREADS-IMPLEMENTATION

MOHAMED GHONIM
AHLIAH NORDSTROM

02/18/2024

Table of Contents

Introduction:

In this homework, we will use of POSIX threads (pthreads) to model the temperature distribution across a 1000x1000 grid, where we explore the temperature evolution under specified initial conditions over 4000 time steps. This program, through its iterations, explores the performance advantages and complexities of concurrency and synchronization, with an emphasis on optimizing computational efficiency. We experiment with different computational strategies, from basic row-major approaches to symmetry-based optimizations to compare performance and analyze results.

Problem statement:

This is a program using pthreads to estimate the temperature of all points on a grid.

Parameters and specifications:

a) Grid size = 1000 x 1000, spanning all points in the square between coordinates (1,1) and (1000,1000).

b) Initial condition: All center points in the region (200, 200) to (800, 800) have a temperature of 500 degrees, and all other points have a temperature of zero. Points outside the grid (i.e., neighbors of points on the boundary) always have a temperature of zero that does not change.

c) At each time step t, the temperature of a point at coordinates (x,y) is computed from the temperatures of the neighboring points in the previous time step (t-1) according to the following equation:

$$\begin{aligned} T(x,y)(t) = {} & T(x,y)(t-1) \\ & + Cx * (T(x+1,y)(t-1) + T(x-1,y)(t-1) - 2\,T(x,y)(t-1)) \\ & + Cy * (T(x,y+1)\,(t-1) + T(x,y-1)\,(t-1) - 2\,T(x,y)\,(t-1)) \end{aligned}$$

Where Cx=0.125 and Cy=0.11

d) Run the program for 4000 time steps. Note that depending on how you split your data, you may need to communicate information to neighboring processors after each time step.

e) After each 200 time steps, you should print the temperatures of the following points: (1, 1), (150,150), (400, 400), (500, 500), (750, 750), and (900,900).

4000/200 = 20 lines. => Temperature of 6 points

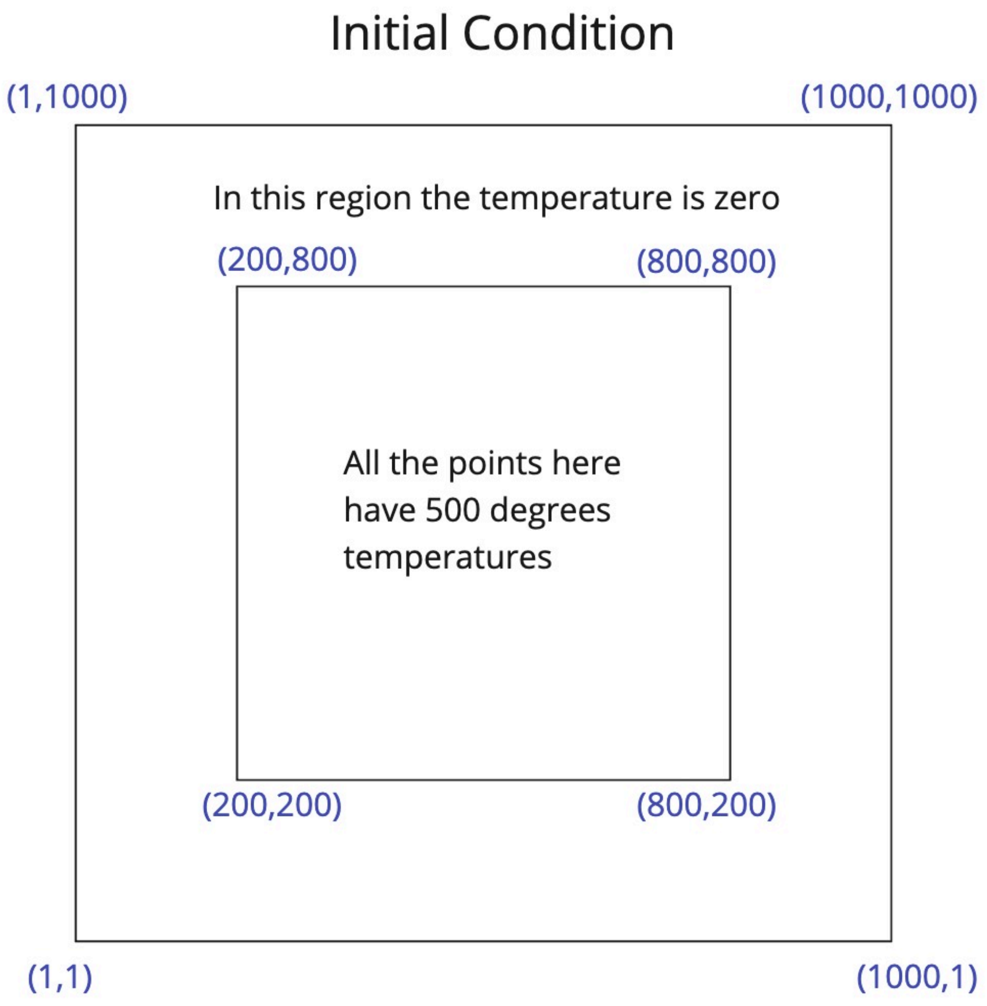
Solution

There are so many ways to divide up and process this problem, one might be more efficient than other given the hardware structure on which the program is running. To really compare the different ways this pthreads code could be structured, we need to experiment and run the program in different configurations.

Disclaimer:

Those programs were run on a Mac M3 pro machine with 11 cores, as well as the mo/auto PSU server. As expected, we noticed that the performance saturates beyond 11 threads on the mac, while it improves a little bit on Linux, overall, the mac was much more efficient than the Linux server, even with a fewer core.

Initial conditions and how to run the programs:


Initial Condition
(1,1000)
(1000,1000)
In this region the temperature is zero
(200,800)
(800,800)
All the points here have 500 degrees temperatures
(200,200)
(800,200)
(1,1)
(1000,1)

To compile the code, we use:

```
cc -lpthread -lrt HW3_updated.c -o HW3_updated1
```

On macos, the -lrt flag is not needed.

To run the code executable, we can use:

```
./HW3_updated1 16
```

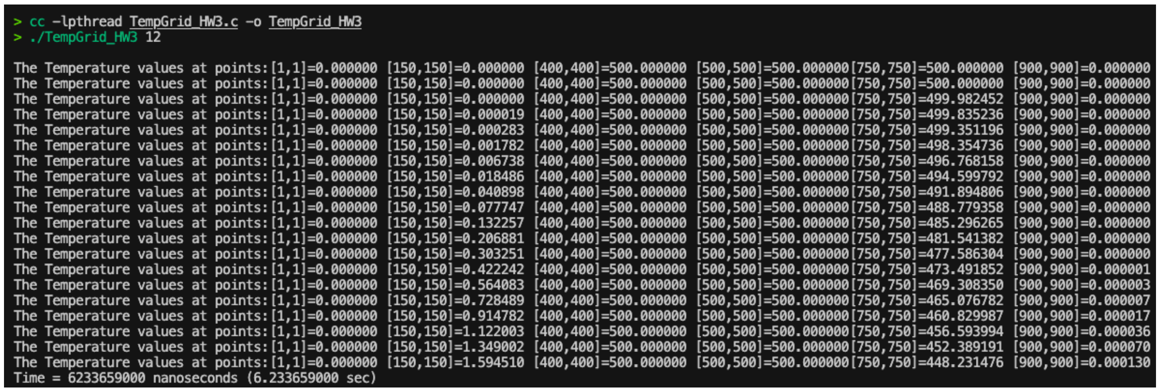
To run 16 threads for example

Different Implementations and analysis of the program:

1- A basic row-major approach. (TempGrid_HW3.c)

- We updated the code to ensure that only one thread (thread 0) is responsible for updating the **old** grid with **new** values after each timestep, which prevents race conditions.

- The initialization of the **old** grid to set initial conditions is streamlined with a single nested loop and a conditional statement. Compared to the given example which assigned 0 to the whole grid, then did an override and assigned 500 to the center grid, wasting computational resources.
- Added sanity checks to make sure we're not performing any calculations outside the grid.

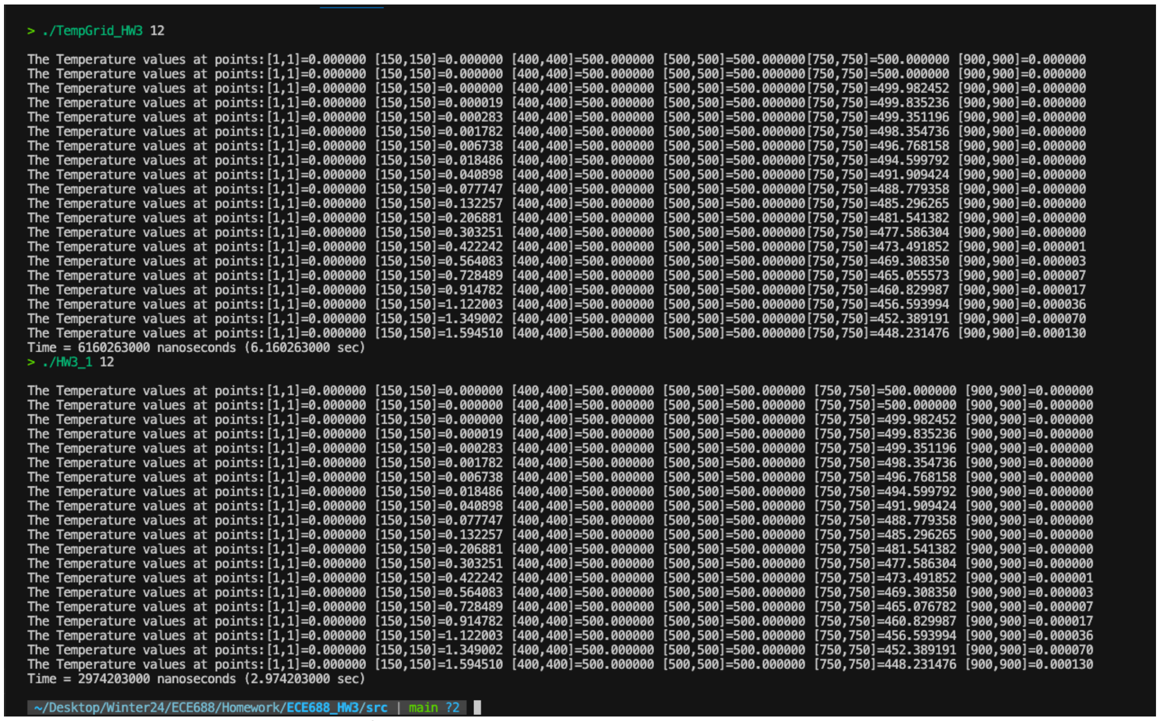


```
> cc -lpthread TempGrid_HW3.c -o TempGrid_HW3
> ./TempGrid_HW3 12

The Temperature values at points:[1,1]=0.000000 [150,150]=0.000000 [400,400]=500.000000 [500,500]=500.000000[750,750]=500.000000 [900,900]=0.000000
The Temperature values at points:[1,1]=0.000000 [150,150]=0.000000 [400,400]=500.000000 [500,500]=500.000000[750,750]=500.000000 [900,900]=0.000000
The Temperature values at points:[1,1]=0.000000 [150,150]=0.000000 [400,400]=500.000000 [500,500]=500.000000[750,750]=499.982452 [900,900]=0.000000
The Temperature values at points:[1,1]=0.000000 [150,150]=0.000019 [400,400]=500.000000 [500,500]=500.000000[750,750]=499.835236 [900,900]=0.000000
The Temperature values at points:[1,1]=0.000000 [150,150]=0.000283 [400,400]=500.000000 [500,500]=500.000000[750,750]=499.351196 [900,900]=0.000000
The Temperature values at points:[1,1]=0.000000 [150,150]=0.001782 [400,400]=500.000000 [500,500]=500.000000[750,750]=498.354736 [900,900]=0.000000
The Temperature values at points:[1,1]=0.000000 [150,150]=0.006738 [400,400]=500.000000 [500,500]=500.000000[750,750]=496.768158 [900,900]=0.000000
The Temperature values at points:[1,1]=0.000000 [150,150]=0.018486 [400,400]=500.000000 [500,500]=500.000000[750,750]=494.599792 [900,900]=0.000000
The Temperature values at points:[1,1]=0.000000 [150,150]=0.040898 [400,400]=500.000000 [500,500]=500.000000[750,750]=491.894806 [900,900]=0.000000
The Temperature values at points:[1,1]=0.000000 [150,150]=0.077747 [400,400]=500.000000 [500,500]=500.000000[750,750]=488.779358 [900,900]=0.000000
The Temperature values at points:[1,1]=0.000000 [150,150]=0.132257 [400,400]=500.000000 [500,500]=500.000000[750,750]=485.296265 [900,900]=0.000000
The Temperature values at points:[1,1]=0.000000 [150,150]=0.206881 [400,400]=500.000000 [500,500]=500.000000[750,750]=481.541382 [900,900]=0.000000
The Temperature values at points:[1,1]=0.000000 [150,150]=0.303251 [400,400]=500.000000 [500,500]=500.000000[750,750]=477.586304 [900,900]=0.000000
The Temperature values at points:[1,1]=0.000000 [150,150]=0.422242 [400,400]=500.000000 [500,500]=500.000000[750,750]=473.491852 [900,900]=0.000001
The Temperature values at points:[1,1]=0.000000 [150,150]=0.564083 [400,400]=500.000000 [500,500]=500.000000[750,750]=469.308350 [900,900]=0.000003
The Temperature values at points:[1,1]=0.000000 [150,150]=0.728489 [400,400]=500.000000 [500,500]=500.000000[750,750]=465.076782 [900,900]=0.000007
The Temperature values at points:[1,1]=0.000000 [150,150]=0.914782 [400,400]=500.000000 [500,500]=500.000000[750,750]=460.829987 [900,900]=0.000017
The Temperature values at points:[1,1]=0.000000 [150,150]=1.122003 [400,400]=500.000000 [500,500]=500.000000[750,750]=456.593994 [900,900]=0.000036
The Temperature values at points:[1,1]=0.000000 [150,150]=1.349002 [400,400]=500.000000 [500,500]=500.000000[750,750]=452.389191 [900,900]=0.000070
The Temperature values at points:[1,1]=0.000000 [150,150]=1.594510 [400,400]=500.000000 [500,500]=500.000000[750,750]=448.231476 [900,900]=0.000130
Time = 6233659000 nanoseconds (6.233659000 sec)
```

2- Enhancing the code by using swappable arrays for old and new, and fewer barriers [only when necessary] (HW3_1.c)

- **Dynamic Allocation**: Both **old** and **new** grids are dynamically allocated, allowing for pointer swapping.
- **Single Barrier**: Only one synchronization point (barrier) per iteration is used, reducing synchronization overhead.

- **Pointer Swapping**: After each iteration and synchronization point, pointers to **old** and **new** grids are swapped, eliminating the need for element-wise copying.



```
> ./TempGrid_HW3 12

The Temperature values at points:[1,1]=0.000000 [150,150]=0.000000 [400,400]=500.000000 [500,500]=500.000000[750,750]=500.000000 [900,900]=0.000000
The Temperature values at points:[1,1]=0.000000 [150,150]=0.000000 [400,400]=500.000000 [500,500]=500.000000[750,750]=500.000000 [900,900]=0.000000
The Temperature values at points:[1,1]=0.000000 [150,150]=0.000000 [400,400]=500.000000 [500,500]=500.000000[750,750]=499.982452 [900,900]=0.000000
The Temperature values at points:[1,1]=0.000000 [150,150]=0.000019 [400,400]=500.000000 [500,500]=500.000000[750,750]=499.835236 [900,900]=0.000000
The Temperature values at points:[1,1]=0.000000 [150,150]=0.000283 [400,400]=500.000000 [500,500]=500.000000[750,750]=499.351196 [900,900]=0.000000
The Temperature values at points:[1,1]=0.000000 [150,150]=0.001782 [400,400]=500.000000 [500,500]=500.000000[750,750]=498.354736 [900,900]=0.000000
The Temperature values at points:[1,1]=0.000000 [150,150]=0.006738 [400,400]=500.000000 [500,500]=500.000000[750,750]=496.768158 [900,900]=0.000000
The Temperature values at points:[1,1]=0.000000 [150,150]=0.018486 [400,400]=500.000000 [500,500]=500.000000[750,750]=494.599792 [900,900]=0.000000
The Temperature values at points:[1,1]=0.000000 [150,150]=0.040898 [400,400]=500.000000 [500,500]=500.000000[750,750]=491.909424 [900,900]=0.000000
The Temperature values at points:[1,1]=0.000000 [150,150]=0.077747 [400,400]=500.000000 [500,500]=500.000000[750,750]=488.779358 [900,900]=0.000000
The Temperature values at points:[1,1]=0.000000 [150,150]=0.132257 [400,400]=500.000000 [500,500]=500.000000[750,750]=485.296265 [900,900]=0.000000
The Temperature values at points:[1,1]=0.000000 [150,150]=0.206881 [400,400]=500.000000 [500,500]=500.000000[750,750]=481.541382 [900,900]=0.000000
The Temperature values at points:[1,1]=0.000000 [150,150]=0.303251 [400,400]=500.000000 [500,500]=500.000000[750,750]=477.586304 [900,900]=0.000000
The Temperature values at points:[1,1]=0.000000 [150,150]=0.422242 [400,400]=500.000000 [500,500]=500.000000[750,750]=473.491852 [900,900]=0.000001
The Temperature values at points:[1,1]=0.000000 [150,150]=0.564083 [400,400]=500.000000 [500,500]=500.000000[750,750]=469.308350 [900,900]=0.000003
The Temperature values at points:[1,1]=0.000000 [150,150]=0.728489 [400,400]=500.000000 [500,500]=500.000000[750,750]=465.055573 [900,900]=0.000007
The Temperature values at points:[1,1]=0.000000 [150,150]=0.914782 [400,400]=500.000000 [500,500]=500.000000[750,750]=460.829987 [900,900]=0.000017
The Temperature values at points:[1,1]=0.000000 [150,150]=1.122003 [400,400]=500.000000 [500,500]=500.000000[750,750]=456.593994 [900,900]=0.000036
The Temperature values at points:[1,1]=0.000000 [150,150]=1.349002 [400,400]=500.000000 [500,500]=500.000000[750,750]=452.389191 [900,900]=0.000070
The Temperature values at points:[1,1]=0.000000 [150,150]=1.594510 [400,400]=500.000000 [500,500]=500.000000[750,750]=448.231476 [900,900]=0.000130
Time = 6160263000 nanoseconds (6.160263000 sec)
> ./HW3_1 12

The Temperature values at points:[1,1]=0.000000 [150,150]=0.000000 [400,400]=500.000000 [500,500]=500.000000 [750,750]=500.000000 [900,900]=0.000000
The Temperature values at points:[1,1]=0.000000 [150,150]=0.000000 [400,400]=500.000000 [500,500]=500.000000 [750,750]=500.000000 [900,900]=0.000000
The Temperature values at points:[1,1]=0.000000 [150,150]=0.000000 [400,400]=500.000000 [500,500]=500.000000 [750,750]=499.982452 [900,900]=0.000000
The Temperature values at points:[1,1]=0.000000 [150,150]=0.000019 [400,400]=500.000000 [500,500]=500.000000 [750,750]=499.835236 [900,900]=0.000000
The Temperature values at points:[1,1]=0.000000 [150,150]=0.000283 [400,400]=500.000000 [500,500]=500.000000 [750,750]=499.351196 [900,900]=0.000000
The Temperature values at points:[1,1]=0.000000 [150,150]=0.001782 [400,400]=500.000000 [500,500]=500.000000 [750,750]=498.354736 [900,900]=0.000000
The Temperature values at points:[1,1]=0.000000 [150,150]=0.006738 [400,400]=500.000000 [500,500]=500.000000 [750,750]=496.768158 [900,900]=0.000000
The Temperature values at points:[1,1]=0.000000 [150,150]=0.018486 [400,400]=500.000000 [500,500]=500.000000 [750,750]=494.599792 [900,900]=0.000000
The Temperature values at points:[1,1]=0.000000 [150,150]=0.040898 [400,400]=500.000000 [500,500]=500.000000 [750,750]=491.909424 [900,900]=0.000000
The Temperature values at points:[1,1]=0.000000 [150,150]=0.077747 [400,400]=500.000000 [500,500]=500.000000 [750,750]=488.779358 [900,900]=0.000000
The Temperature values at points:[1,1]=0.000000 [150,150]=0.132257 [400,400]=500.000000 [500,500]=500.000000 [750,750]=485.296265 [900,900]=0.000000
The Temperature values at points:[1,1]=0.000000 [150,150]=0.206881 [400,400]=500.000000 [500,500]=500.000000 [750,750]=481.541382 [900,900]=0.000000
The Temperature values at points:[1,1]=0.000000 [150,150]=0.303251 [400,400]=500.000000 [500,500]=500.000000 [750,750]=477.586304 [900,900]=0.000000
The Temperature values at points:[1,1]=0.000000 [150,150]=0.422242 [400,400]=500.000000 [500,500]=500.000000 [750,750]=473.491852 [900,900]=0.000001
The Temperature values at points:[1,1]=0.000000 [150,150]=0.564083 [400,400]=500.000000 [500,500]=500.000000 [750,750]=469.308350 [900,900]=0.000003
The Temperature values at points:[1,1]=0.000000 [150,150]=0.728489 [400,400]=500.000000 [500,500]=500.000000 [750,750]=465.076782 [900,900]=0.000007
The Temperature values at points:[1,1]=0.000000 [150,150]=0.914782 [400,400]=500.000000 [500,500]=500.000000 [750,750]=460.829987 [900,900]=0.000017
The Temperature values at points:[1,1]=0.000000 [150,150]=1.122003 [400,400]=500.000000 [500,500]=500.000000 [750,750]=456.593994 [900,900]=0.000036
The Temperature values at points:[1,1]=0.000000 [150,150]=1.349002 [400,400]=500.000000 [500,500]=500.000000 [750,750]=452.389191 [900,900]=0.000070
The Temperature values at points:[1,1]=0.000000 [150,150]=1.594510 [400,400]=500.000000 [500,500]=500.000000 [750,750]=448.231476 [900,900]=0.000130
Time = 2974203000 nanoseconds (2.974203000 sec)

 ~/Desktop/Winter24/ECE688/Homework/ECE688_HW3/src | main ?2
```

Comparing this version of the program to the initial one, we can see that we're generating the same temperatures, but now the time is much, much less.

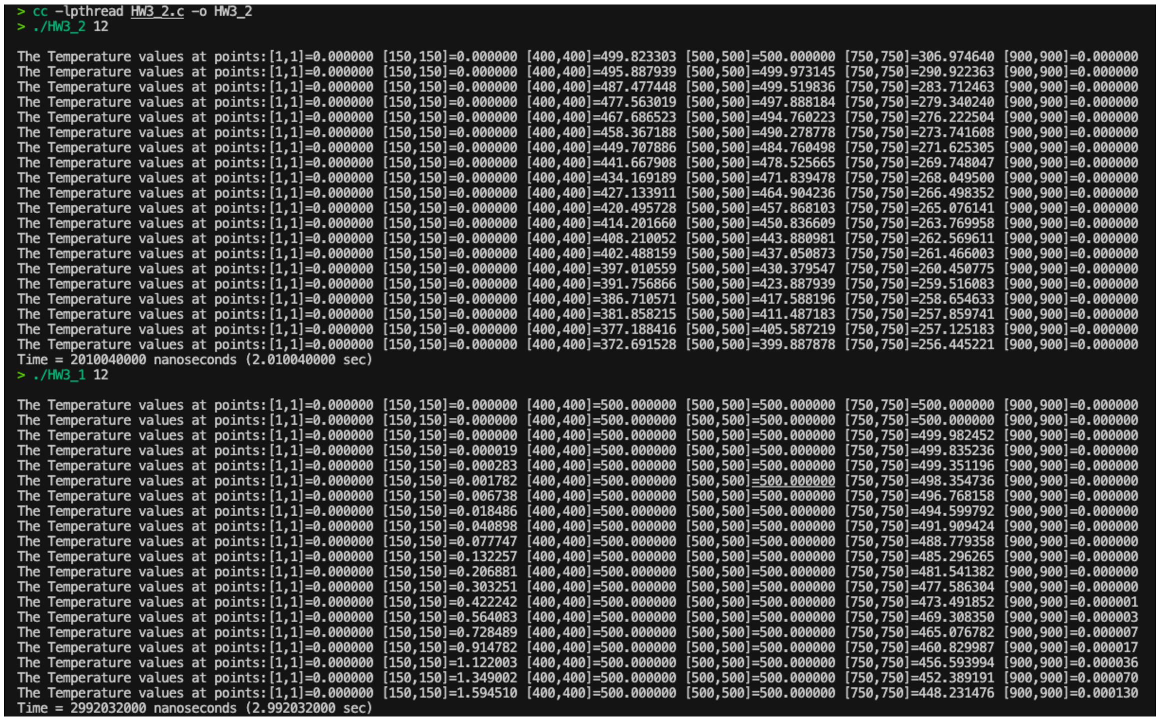
3- Exploring Further enhancement by utilizing the code symmetry to minimize calculations.

To further enhance the code by utilizing the symmetry of the grid, we can calculate the temperature for just a quarter or half of the grid and mirror these values to the other parts. This approach can potentially halve or quarter the computation time.
Given the grid's symmetry, we will start by calculating temperatures for half of the grid (either vertically or horizontally) and then mirroring these values. For simplicity, let's assume we calculate for the left half and mirror to the right half. We need to adjust the logic slightly if the symmetry or the pattern of temperature distribution is different.

Here's what the temp function is when calculating only the left half of the grid and mirroring the values to the right half.

```
void* Temp(void* tmp) {
    long int threadId = (long int)tmp;
    int halfX = X_SIZE / 2; // Calculate only for the left half
    int start = threadId * (halfX - 1) / NumThreads + (threadId < Remainder ?
threadId : Remainder);
    int end = start + (halfX - 1) / NumThreads - 1 + (threadId < Remainder ? 1
: 0);

    for (int block = 1; block <= TIMESTEPS; block++) {
        for (int j = start; j <= end; j++) {
            for (int k = 1; k < Y_SIZE - 1; k++) {
                if (j > 0 && j < halfX) {
                    new[j][k] = old[j][k] + Cx * (old[j + 1][k] + old[j - 1][k]
- 2 * old[j][k]) + Cy * (old[j][k + 1] + old[j][k - 1] - 2 * old[j][k]);
                    // Mirror the calculated value to the right half
                    new[X_SIZE - 1 - j][k] = new[j][k];
                }
            }
        }

        Barrier();
        if (threadId == 0) {
            float (*temp)[Y_SIZE] = old;
            old = new;
            new = temp;

            if (block % PRINT_STEPS == 0) {
                printf("\nThe Temperature values at points:[1,1]=%f
[150,150]=%f [400,400]=%f [500,500]=%f [750,750]=%f [900,900]=%f", old[1][1],
old[150][150], old[400][400], old[500][500], old[750][750], old[900][900]);
            }
        }
        Barrier();
    }
    return NULL;
}
```

The new program is called HW3_2, as compared to the program from step (2), which is called HW3_1 here.



```
> cc -lpthread HW3_2.c -o HW3_2
> ./HW3_2 12

 The Temperature values at points:[1,1]=0.000000 [150,150]=0.000000 [400,400]=499.823303 [500,500]=500.000000 [750,750]=306.974640 [900,900]=0.000000
 The Temperature values at points:[1,1]=0.000000 [150,150]=0.000000 [400,400]=495.887939 [500,500]=499.973145 [750,750]=290.922363 [900,900]=0.000000
 The Temperature values at points:[1,1]=0.000000 [150,150]=0.000000 [400,400]=487.477448 [500,500]=499.519836 [750,750]=283.712463 [900,900]=0.000000
 The Temperature values at points:[1,1]=0.000000 [150,150]=0.000000 [400,400]=477.563019 [500,500]=497.888184 [750,750]=279.340240 [900,900]=0.000000
 The Temperature values at points:[1,1]=0.000000 [150,150]=0.000000 [400,400]=467.686523 [500,500]=494.760223 [750,750]=276.222504 [900,900]=0.000000
 The Temperature values at points:[1,1]=0.000000 [150,150]=0.000000 [400,400]=458.367188 [500,500]=490.278778 [750,750]=273.741608 [900,900]=0.000000
 The Temperature values at points:[1,1]=0.000000 [150,150]=0.000000 [400,400]=449.707886 [500,500]=484.760498 [750,750]=271.625305 [900,900]=0.000000
 The Temperature values at points:[1,1]=0.000000 [150,150]=0.000000 [400,400]=441.667908 [500,500]=478.525665 [750,750]=269.748047 [900,900]=0.000000
 The Temperature values at points:[1,1]=0.000000 [150,150]=0.000000 [400,400]=434.169189 [500,500]=471.839478 [750,750]=268.049500 [900,900]=0.000000
 The Temperature values at points:[1,1]=0.000000 [150,150]=0.000000 [400,400]=427.133911 [500,500]=464.904236 [750,750]=266.498352 [900,900]=0.000000
 The Temperature values at points:[1,1]=0.000000 [150,150]=0.000000 [400,400]=420.495728 [500,500]=457.868103 [750,750]=265.076141 [900,900]=0.000000
 The Temperature values at points:[1,1]=0.000000 [150,150]=0.000000 [400,400]=414.201660 [500,500]=450.836609 [750,750]=263.769958 [900,900]=0.000000
 The Temperature values at points:[1,1]=0.000000 [150,150]=0.000000 [400,400]=408.210052 [500,500]=443.880981 [750,750]=262.569611 [900,900]=0.000000
 The Temperature values at points:[1,1]=0.000000 [150,150]=0.000000 [400,400]=402.488159 [500,500]=437.050873 [750,750]=261.466003 [900,900]=0.000000
 The Temperature values at points:[1,1]=0.000000 [150,150]=0.000000 [400,400]=397.010559 [500,500]=430.379547 [750,750]=260.450775 [900,900]=0.000000
 The Temperature values at points:[1,1]=0.000000 [150,150]=0.000000 [400,400]=391.756866 [500,500]=423.887939 [750,750]=259.516083 [900,900]=0.000000
 The Temperature values at points:[1,1]=0.000000 [150,150]=0.000000 [400,400]=386.710571 [500,500]=417.588196 [750,750]=258.654633 [900,900]=0.000000
 The Temperature values at points:[1,1]=0.000000 [150,150]=0.000000 [400,400]=381.858215 [500,500]=411.487183 [750,750]=257.859741 [900,900]=0.000000
 The Temperature values at points:[1,1]=0.000000 [150,150]=0.000000 [400,400]=377.188416 [500,500]=405.587219 [750,750]=257.125183 [900,900]=0.000000
 The Temperature values at points:[1,1]=0.000000 [150,150]=0.000000 [400,400]=372.691528 [500,500]=399.887878 [750,750]=256.445221 [900,900]=0.000000
Time = 2010040000 nanoseconds (2.010040000 sec)
> ./HW3_1 12

 The Temperature values at points:[1,1]=0.000000 [150,150]=0.000000 [400,400]=500.000000 [500,500]=500.000000 [750,750]=500.000000 [900,900]=0.000000
 The Temperature values at points:[1,1]=0.000000 [150,150]=0.000000 [400,400]=500.000000 [500,500]=500.000000 [750,750]=500.000000 [900,900]=0.000000
 The Temperature values at points:[1,1]=0.000000 [150,150]=0.000000 [400,400]=500.000000 [500,500]=500.000000 [750,750]=499.982452 [900,900]=0.000000
 The Temperature values at points:[1,1]=0.000000 [150,150]=0.000019 [400,400]=500.000000 [500,500]=500.000000 [750,750]=499.835236 [900,900]=0.000000
 The Temperature values at points:[1,1]=0.000000 [150,150]=0.000283 [400,400]=500.000000 [500,500]=500.000000 [750,750]=499.351196 [900,900]=0.000000
 The Temperature values at points:[1,1]=0.000000 [150,150]=0.001782 [400,400]=500.000000 [500,500]=500.000000 [750,750]=498.354736 [900,900]=0.000000
 The Temperature values at points:[1,1]=0.000000 [150,150]=0.006738 [400,400]=500.000000 [500,500]=500.000000 [750,750]=496.768158 [900,900]=0.000000
 The Temperature values at points:[1,1]=0.000000 [150,150]=0.018486 [400,400]=500.000000 [500,500]=500.000000 [750,750]=494.599792 [900,900]=0.000000
 The Temperature values at points:[1,1]=0.000000 [150,150]=0.040898 [400,400]=500.000000 [500,500]=500.000000 [750,750]=491.909424 [900,900]=0.000000
 The Temperature values at points:[1,1]=0.000000 [150,150]=0.077747 [400,400]=500.000000 [500,500]=500.000000 [750,750]=488.779358 [900,900]=0.000000
 The Temperature values at points:[1,1]=0.000000 [150,150]=0.132257 [400,400]=500.000000 [500,500]=500.000000 [750,750]=485.296265 [900,900]=0.000000
 The Temperature values at points:[1,1]=0.000000 [150,150]=0.206881 [400,400]=500.000000 [500,500]=500.000000 [750,750]=481.541382 [900,900]=0.000000
 The Temperature values at points:[1,1]=0.000000 [150,150]=0.303251 [400,400]=500.000000 [500,500]=500.000000 [750,750]=477.586304 [900,900]=0.000000
 The Temperature values at points:[1,1]=0.000000 [150,150]=0.422242 [400,400]=500.000000 [500,500]=500.000000 [750,750]=473.491852 [900,900]=0.000001
 The Temperature values at points:[1,1]=0.000000 [150,150]=0.564083 [400,400]=500.000000 [500,500]=500.000000 [750,750]=469.308350 [900,900]=0.000003
 The Temperature values at points:[1,1]=0.000000 [150,150]=0.728489 [400,400]=500.000000 [500,500]=500.000000 [750,750]=465.076782 [900,900]=0.000007
 The Temperature values at points:[1,1]=0.000000 [150,150]=0.914782 [400,400]=500.000000 [500,500]=500.000000 [750,750]=460.829987 [900,900]=0.000017
 The Temperature values at points:[1,1]=0.000000 [150,150]=1.122003 [400,400]=500.000000 [500,500]=500.000000 [750,750]=456.593994 [900,900]=0.000036
 The Temperature values at points:[1,1]=0.000000 [150,150]=1.349002 [400,400]=500.000000 [500,500]=500.000000 [750,750]=452.389191 [900,900]=0.000070
 The Temperature values at points:[1,1]=0.000000 [150,150]=1.594510 [400,400]=500.000000 [500,500]=500.000000 [750,750]=448.231476 [900,900]=0.000130
Time = 2992032000 nanoseconds (2.992032000 sec)
```

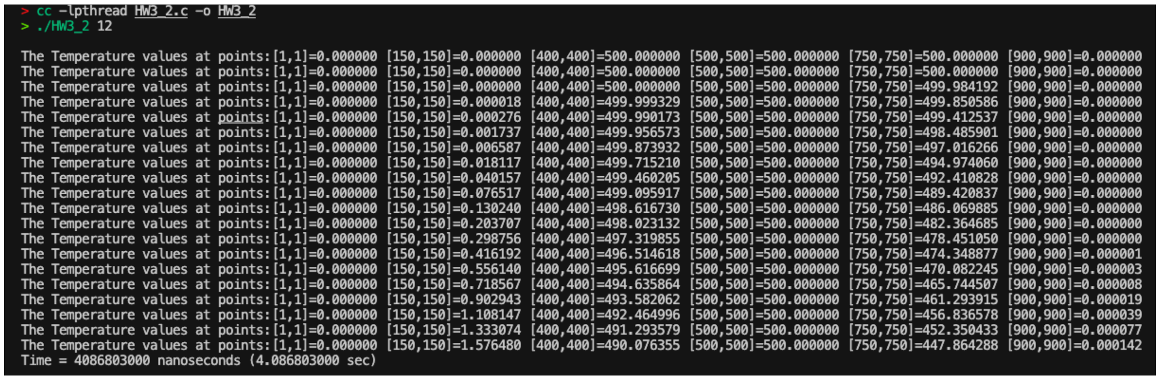
We see that we are able to cut some time, not quite 50% of the time because we have introduced some overhead logic to "mirror" the left half into the right half, but we also notice that the calculated values are not exactly the same, though in general they seem to be on the right direction. We explored what might be causing this discrepancy:

1. **Barrier Synchronization Overhead**: While the intention was to reduce computation by leveraging symmetry, the additional overhead of synchronizing threads (especially with multiple barriers) and the added complexity of mirroring the data might not offset the gains from reduced computation, particularly if the computation itself is not the primary bottleneck.
2. **Data Consistency Issues**: If the mirroring of data occurs while other threads are still computing their portions, or if the mirroring itself is not correctly synchronized, this could lead to inconsistencies in the grid data, affecting accuracy.

We further explored mirroring vertically instead of horizontally, to see if maybe this works better with our cache.

```
void* Temp(void* tmp) {
    long int threadId = (long int)tmp;
    int start = threadId * (X_SIZE - 1) / NumThreads + (threadId < Remainder ? 
threadId : Remainder);
    int end = start + (X_SIZE - 1) / NumThreads - 1 + (threadId < Remainder ? 1 
: 0);
```

```
    int halfY = Y_SIZE / 2; // Only calculate for the top half

    for (int block = 1; block <= TIMESTEPS; block++) {
        for (int j = start; j <= end; j++) {
            for (int k = 1; k < halfY; k++) { // Iterate only over the top half
                if (k > 0 && k < halfY) {
                    new[j][k] = old[j][k] + Cx * (old[j + 1][k] + old[j - 1][k]
- 2 * old[j][k]) + Cy * (old[j][k + 1] + old[j][k - 1] - 2 * old[j][k]);
                }
            }
        }

        Barrier();

        // Mirror the calculated values to the bottom half
        if (threadId == 0) {
            for (int x = 0; x < X_SIZE; x++) {
                for (int y = 1; y < halfY; y++) {
                    new[x][Y_SIZE - y - 1] = new[x][y];
                }
            }
        }

        Barrier();

        if (threadId == 0) {
            // Swap the pointers
            float (*temp)[Y_SIZE] = old;
            old = new;
            new = temp;

            if (block % PRINT_STEPS == 0) {
                printf("\nThe Temperature values at points:[1,1]=%f
[150,150]=%f [400,400]=%f [500,500]=%f [750,750]=%f [900,900]=%f", old[1][1],
old[150][150], old[400][400], old[500][500], old[750][750], old[900][900]);
            }
        }
    }
    return NULL;
}
```
This however degraded the performance.



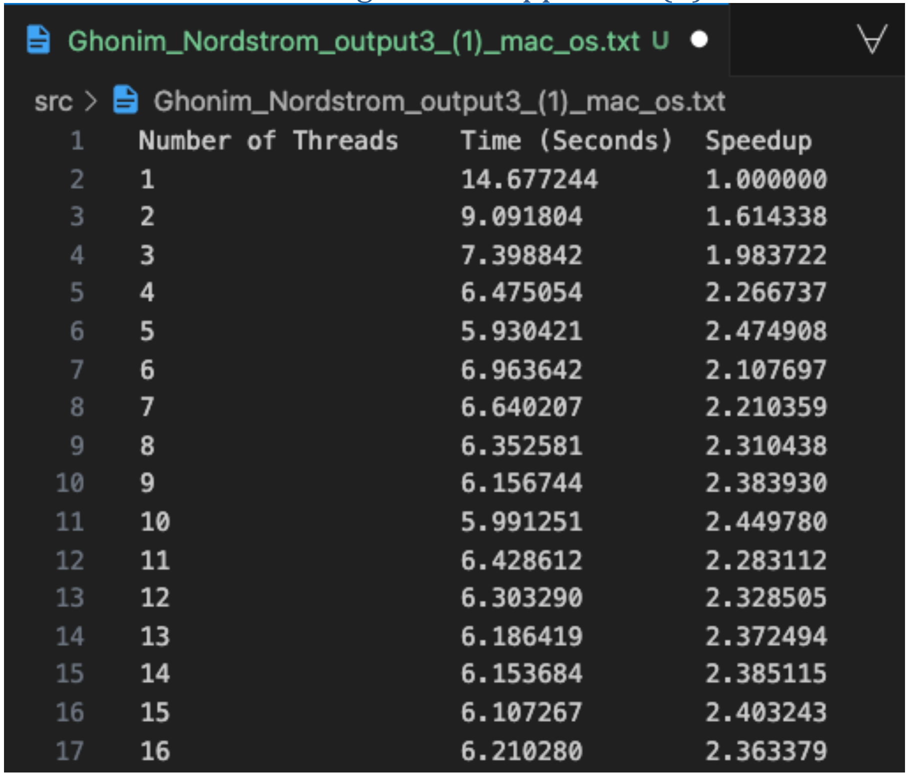
```
> cc -lpthread HW3_2.c -o HW3_2
> ./HW3_2 12

The Temperature values at points:[1,1]=0.000000 [150,150]=0.000000 [400,400]=500.000000 [500,500]=500.000000 [750,750]=500.000000 [900,900]=0.000000
The Temperature values at points:[1,1]=0.000000 [150,150]=0.000000 [400,400]=500.000000 [500,500]=500.000000 [750,750]=500.000000 [900,900]=0.000000
The Temperature values at points:[1,1]=0.000000 [150,150]=0.000000 [400,400]=500.000000 [500,500]=500.000000 [750,750]=499.984192 [900,900]=0.000000
The Temperature values at points:[1,1]=0.000000 [150,150]=0.000018 [400,400]=499.999329 [500,500]=500.000000 [750,750]=499.850586 [900,900]=0.000000
The Temperature values at points:[1,1]=0.000000 [150,150]=0.000276 [400,400]=499.990173 [500,500]=500.000000 [750,750]=499.412537 [900,900]=0.000000
The Temperature values at points:[1,1]=0.000000 [150,150]=0.001737 [400,400]=499.956573 [500,500]=500.000000 [750,750]=498.485901 [900,900]=0.000000
The Temperature values at points:[1,1]=0.000000 [150,150]=0.006587 [400,400]=499.873932 [500,500]=500.000000 [750,750]=497.016266 [900,900]=0.000000
The Temperature values at points:[1,1]=0.000000 [150,150]=0.018117 [400,400]=499.715210 [500,500]=500.000000 [750,750]=494.974060 [900,900]=0.000000
The Temperature values at points:[1,1]=0.000000 [150,150]=0.040157 [400,400]=499.460205 [500,500]=500.000000 [750,750]=492.410828 [900,900]=0.000000
The Temperature values at points:[1,1]=0.000000 [150,150]=0.076517 [400,400]=499.095917 [500,500]=500.000000 [750,750]=489.420837 [900,900]=0.000000
The Temperature values at points:[1,1]=0.000000 [150,150]=0.130240 [400,400]=498.616730 [500,500]=500.000000 [750,750]=486.069885 [900,900]=0.000000
The Temperature values at points:[1,1]=0.000000 [150,150]=0.203707 [400,400]=498.023132 [500,500]=500.000000 [750,750]=482.364685 [900,900]=0.000000
The Temperature values at points:[1,1]=0.000000 [150,150]=0.298756 [400,400]=497.319855 [500,500]=500.000000 [750,750]=478.451050 [900,900]=0.000000
The Temperature values at points:[1,1]=0.000000 [150,150]=0.416192 [400,400]=496.514618 [500,500]=500.000000 [750,750]=474.348877 [900,900]=0.000001
The Temperature values at points:[1,1]=0.000000 [150,150]=0.556140 [400,400]=495.616699 [500,500]=500.000000 [750,750]=470.082245 [900,900]=0.000003
The Temperature values at points:[1,1]=0.000000 [150,150]=0.718567 [400,400]=494.635864 [500,500]=500.000000 [750,750]=465.744507 [900,900]=0.000008
The Temperature values at points:[1,1]=0.000000 [150,150]=0.902943 [400,400]=493.582062 [500,500]=500.000000 [750,750]=461.293915 [900,900]=0.000019
The Temperature values at points:[1,1]=0.000000 [150,150]=1.108147 [400,400]=492.464996 [500,500]=500.000000 [750,750]=456.836578 [900,900]=0.000039
The Temperature values at points:[1,1]=0.000000 [150,150]=1.333074 [400,400]=491.293579 [500,500]=500.000000 [750,750]=452.350433 [900,900]=0.000077
The Temperature values at points:[1,1]=0.000000 [150,150]=1.576480 [400,400]=490.076355 [500,500]=500.000000 [750,750]=447.864288 [900,900]=0.000142
Time = 4086803000 nanoseconds (4.086803000 sec)
```

Automation and output3.txt creation:

We created a simple Perl script to automate the creation of the output3.txt file and run the HW3 program from 1 to 16 threads and calculate their speedup. In large, this was a modification from the Perl automating script in HW2.

```perl
#!/usr/bin/perl
use strict;
use warnings;
# Output file
my $output_file = 'Ghonim_Nordstrom_output3.txt';

# Open the file for writing
open(my $fh, '>', $output_file) or die "Could not open file '$output_file' $!";

# Print the header to the file with fixed widths for each column
printf $fh "%-20s %-15s %-15s\n", "Number of Threads", "Time (Seconds)",
"Speedup";

# Variable to store time for single-thread execution
my $single_thread_time = 0;

# Run the program with different numbers of threads
for (my $num_threads = 1; $num_threads <= 16; $num_threads++) {
    # Execute the program and capture its output
    my $output = `./TempGrid_HW3 $num_threads`;

    # Extract the time from the output
    my ($time_in_seconds) = $output =~ /Time = \d+ nanoseconds\s+\(([\d\.]+)
sec\)/;

    # Calculate speedup
    my $speedup = $num_threads == 1 ? 1 : $single_thread_time /
$time_in_seconds;
    $single_thread_time = $time_in_seconds if $num_threads == 1;
    # Write the results to the output file with fixed widths for each column
    printf $fh "%-20d %-15f %-15f\n", $num_threads, $time_in_seconds, $speedup;
}

# Close the file
close $fh;

print "Results saved in $output_file\n";
```

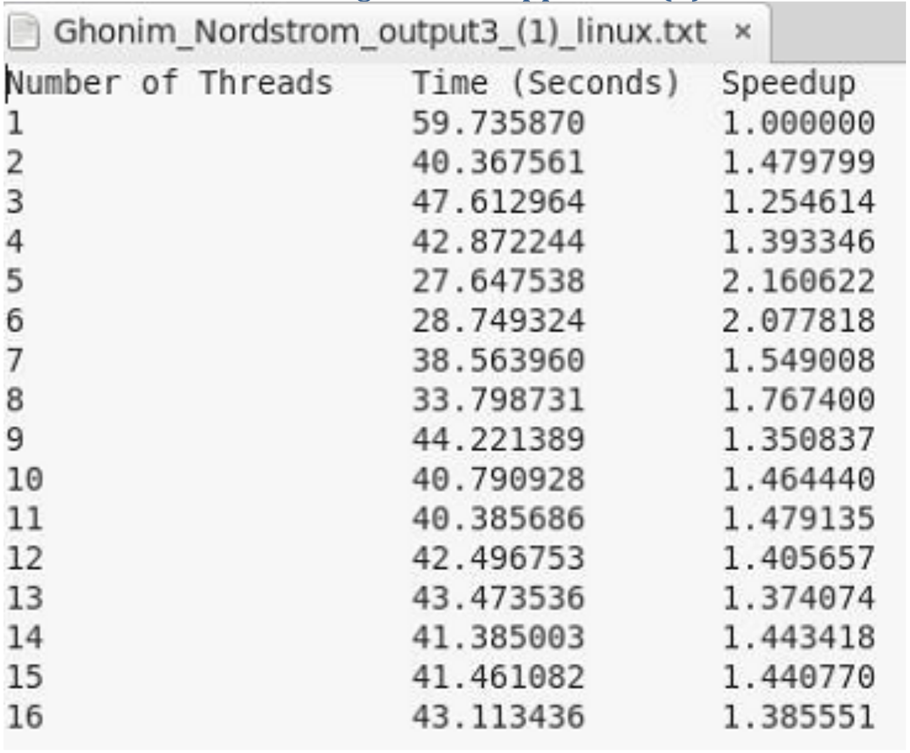
The results after running the first approach. (1) on a Mac with 11 cores.



Ghonim_Nordstrom_output3_(1)_mac_os.txt U
src > Ghonim_Nordstrom_output3_(1)_mac_os.txt

```
Number of Threads    Time (Seconds)  Speedup
1                    14.677244       1.000000
2                    9.091804        1.614338
3                    7.398842        1.983722
4                    6.475054        2.266737
5                    5.930421        2.474908
6                    6.963642        2.107697
7                    6.640207        2.210359
8                    6.352581        2.310438
9                    6.156744        2.383930
10                   5.991251        2.449780
11                   6.428612        2.283112
12                   6.303290        2.328505
13                   6.186419        2.372494
14                   6.153684        2.385115
15                   6.107267        2.403243
16                   6.210280        2.363379
```

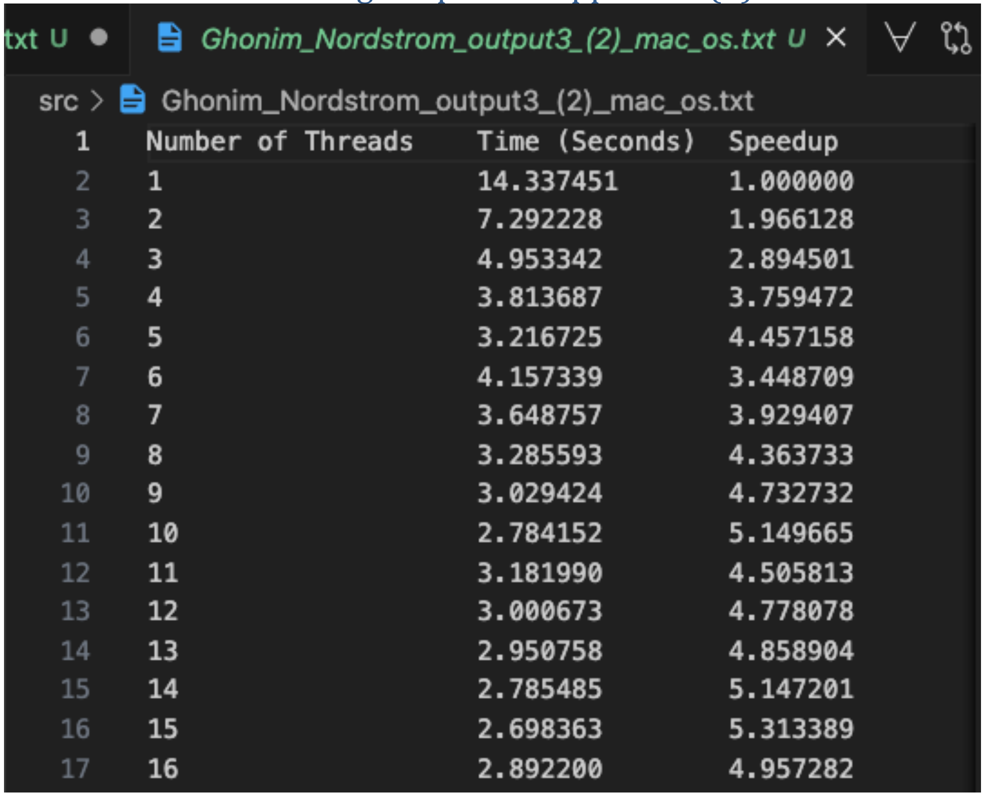
The results after running the first approach. (1) Linux with 16 cores.



Ghonim_Nordstrom_output3_(1)_linux.txt ×

```
Number of Threads	Time (Seconds)	Speedup
1			59.735870	1.000000
2			40.367561	1.479799
3			47.612964	1.254614
4			42.872244	1.393346
5			27.647538	2.160622
6			28.749324	2.077818
7			38.563960	1.549008
8			33.798731	1.767400
9			44.221389	1.350837
10			40.790928	1.464440
11			40.385686	1.479135
12			42.496753	1.405657
13			43.473536	1.374074
14			41.385003	1.443418
15			41.461082	1.440770
16			43.113436	1.385551
```

The results after running the pointer approach. (2) on a Mac with 11 cores.

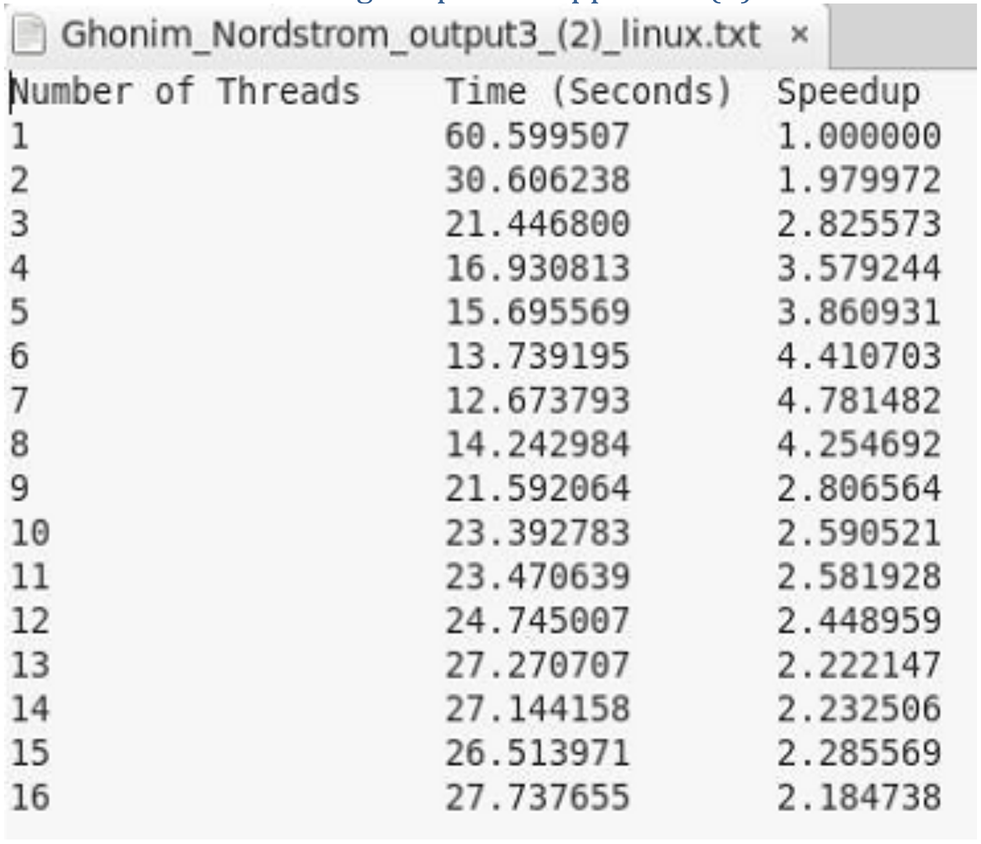


txt U ● Ghonim_Nordstrom_output3_(2)_mac_os.txt U ×

src > Ghonim_Nordstrom_output3_(2)_mac_os.txt

```
Number of Threads    Time (Seconds)  Speedup
1                    14.337451       1.000000
2                    7.292228        1.966128
3                    4.953342        2.894501
4                    3.813687        3.759472
5                    3.216725        4.457158
6                    4.157339        3.448709
7                    3.648757        3.929407
8                    3.285593        4.363733
9                    3.029424        4.732732
10                   2.784152        5.149665
11                   3.181990        4.505813
12                   3.000673        4.778078
13                   2.950758        4.858904
14                   2.785485        5.147201
15                   2.698363        5.313389
16                   2.892200        4.957282
```

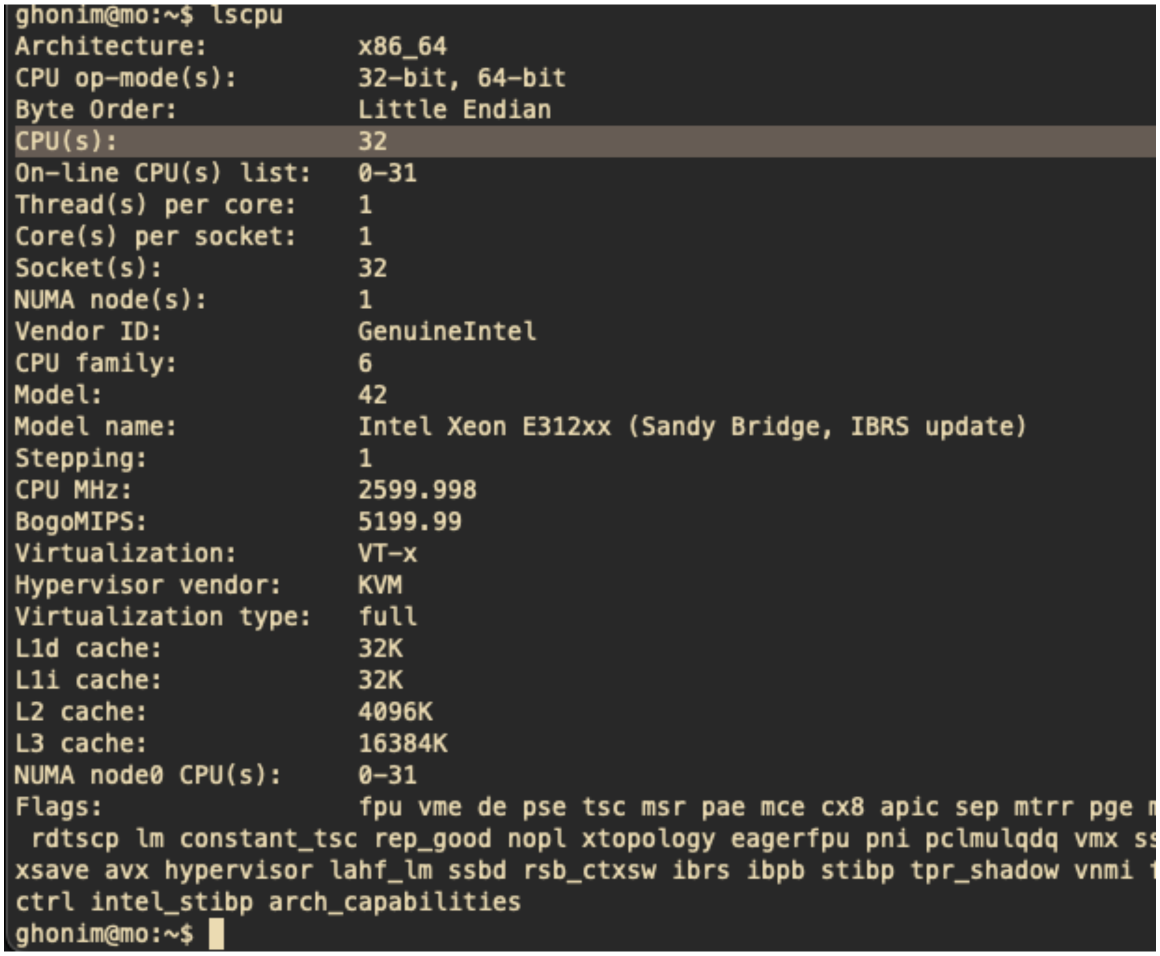
The results after running the pointer approach. (2) Linux with 16 cores.



Ghonim_Nordstrom_output3_(2)_linux.txt ×

```
Number of Threads       Time (Seconds)  Speedup
1                       60.599507       1.000000
2                       30.606238       1.979972
3                       21.446800       2.825573
4                       16.930813       3.579244
5                       15.695569       3.860931
6                       13.739195       4.410703
7                       12.673793       4.781482
8                       14.242984       4.254692
9                       21.592064       2.806564
10                      23.392783       2.590521
11                      23.470639       2.581928
12                      24.745007       2.448959
13                      27.270707       2.222147
14                      27.144158       2.232506
15                      26.513971       2.285569
16                      27.737655       2.184738
```

It looks like either the linux system has only 8 cores, or the performance drops after 8 threads due to the communication overhead on linux, as compared to mac. We checked the linux system and found that it has 32 cores.



```
ghonim@mo:~$ lscpu
Architecture:          x86_64
CPU op-mode(s):        32-bit, 64-bit
Byte Order:            Little Endian
CPU(s):                32
On-line CPU(s) list:   0-31
Thread(s) per core:    1
Core(s) per socket:    1
Socket(s):             32
NUMA node(s):          1
Vendor ID:             GenuineIntel
CPU family:            6
Model:                 42
Model name:            Intel Xeon E312xx (Sandy Bridge, IBRS update)
Stepping:              1
CPU MHz:               2599.998
BogoMIPS:              5199.99
Virtualization:        VT-x
Hypervisor vendor:     KVM
Virtualization type:   full
L1d cache:             32K
L1i cache:             32K
L2 cache:              4096K
L3 cache:              16384K
NUMA node0 CPU(s):     0-31
Flags:                 fpu vme de pse tsc msr pae mce cx8 apic sep mtrr pge m
 rdtscp lm constant_tsc rep_good nopl xtopology eagerfpu pni pclmulqdq vmx ss
xsave avx hypervisor lahf_lm ssbd rsb_ctxsw ibrs ibpb stibp tpr_shadow vnmi f
ctrl intel_stibp arch_capabilities
ghonim@mo:~$
```

Therefore it's probably due to more communication overhead that causes performance to degrade on the linux systems as compared to macOS.

Summary and closing notes:

In conclusion, we have experimented with different implementations of threads starting an obvious implementation that uses multiple loops to copy the old grid into the new grid every timestep to an enhanced program utilizing pointers, then the exploitation of symmetry. This has revealed the intricate balance between computation, synchronization, and memory management in parallel processing. The optimization techniques employed, from swappable arrays to symmetry-based calculations have provided us valuable insights into the behavior of parallel systems. The experimental results highlighted the importance of thread management and data consistency, showcasing the potential for significant speedups while also cautioning against the pitfalls of overhead resulting from complexity.