CS310 CDP

Group 6 - Assignment 5

Aryan Garg B19153

Anukool Dwivedi B19071

Divyansh Vinayak B19080

Kailash Kumar B19087

1. **Sequential vs multi-threading - 1 thread per row:**

```
aryan@LAPTOP-114CPH5S:/mnt/c/Users/HP/Desktop/sem_5/CS310_CDP/A5$ ./q1 2
FOR N = 2
A:
7 7
6 1
B:
1 3
7 2
C:
56 35
13 20
ThreadC:
56 35
13 20

[+] Verified(element-wise)! Sequential and multi-threaded solution are the same.

FOR N = 2
*** Sequential implementation took 0.000002 seconds ***
*** Multi-thread implementation(1 thread per row) took 0.000221 seconds ***


Speed up: 0.009050
aryan@LAPTOP-114CPH5S:/mnt/c/Users/HP/Desktop/sem_5/CS310_CDP/A5$ ./q1 20
FOR N = 20

[+] Verified(element-wise)! Sequential and multi-threaded solution are the same.

FOR N = 20
*** Sequential implementation took 0.000057 seconds ***
*** Multi-thread implementation(1 thread per row) took 0.004120 seconds ***


Speed up: 0.013835
```

N = 100 (100 threads created)

```
aryan@LAPTOP-114CPH5S:/mnt/c/Users/HP/Desktop/sem_5/CS310_CDP/A5$ ./q1 100
FOR N = 100

[+] Verified(element-wise)! Sequential and multi-threaded solution are the same.

FOR N = 100
*** Sequential implementation took 0.006848 seconds ***
*** Multi-thread implementation(1 thread per row) took 0.029490 seconds ***


Speed up: 0.232214
```

N = 200 (200 threads created)

```
aryan@LAPTOP-114CPH5S:/mnt/c/Users/HP/Desktop/sem_5/CS310_CDP/A5$ ./q1 200
FOR N = 200

[+] Verified(element-wise)! Sequential and multi-threaded solution are the same.

FOR N = 200
*** Sequential implementation took 0.078415 seconds ***
*** Multi-thread implementation(1 thread per row) took 0.156601 seconds ***


Speed up: 0.500731
```

<u>Inference:</u> More threads mean more distribution of task at hand but it definitely adds on time in context switching and creating more than 500 threads is a huge load on memory (at least on my architecture.)

2. **4 threads doing N/4 rows each.**

aryan@LAPTOP-114CPH5S: /mr

```
aryan@LAPTOP-114CPH5S:/mnt/c/Users/HP/Desktop/sem_5/CS310_CDP/A5$ ./q2 4
FOR N = 4
Starting row: 0
Starting row: 1
Starting row: 2
Starting row: 3

[+] Verified(element-wise)! Sequential and multi-threaded solution are the same.

FOR N = 4
*** Sequential implementation took 0.000002 seconds ***
*** Multi-thread implementation(4 threads) took 0.002399 seconds ***


Speed up: 0.000834
aryan@LAPTOP-114CPH5S:/mnt/c/Users/HP/Desktop/sem_5/CS310_CDP/A5$ ./q2 20
FOR N = 20
Starting row: 0
Starting row: 5
Starting row: 10
Starting row: 15

[+] Verified(element-wise)! Sequential and multi-threaded solution are the same.

FOR N = 20
*** Sequential implementation took 0.000065 seconds ***
*** Multi-thread implementation(4 threads) took 0.001406 seconds ***


Speed up: 0.046230
```

N = 100 and 500

```
aryan@LAPTOP-114CPH5S:/mnt/c/Users/HP/Desktop/sem_5/CS310_CDP/A5$ ./q2 100
FOR N = 100
Starting row: 0
Starting row: 25
Starting row: 50
Starting row: 75

[+] Verified(element-wise)! Sequential and multi-threaded solution are the same.

FOR N = 100
*** Sequential implementation took 0.007805 seconds ***
*** Multi-thread implementation(4 threads) took 0.004436 seconds ***


Speed up: 1.759468
aryan@LAPTOP-114CPH5S:/mnt/c/Users/HP/Desktop/sem_5/CS310_CDP/A5$ ./q2 500
FOR N = 500
Starting row: 0
Starting row: 125
Starting row: 250
Starting row: 375

[+] Verified(element-wise)! Sequential and multi-threaded solution are the same.

FOR N = 500
*** Sequential implementation took 1.269900 seconds ***
*** Multi-thread implementation(4 threads) took 1.612874 seconds ***


Speed up: 0.787352
```

N = 1000

```
aryan@LAPTOP-114CPH5S: /mr
aryan@LAPTOP-114CPH5S:/mnt/c/Users/HP/Desktop/sem_5/CS310_CDP/A5$ ./q2 1000
FOR N = 1000
Starting row: 0
Starting row: 750
Starting row: 250
Starting row: 500

[+] Verified(element-wise)! Sequential and multi-threaded solution are the same.

FOR N = 1000
*** Sequential implementation took 15.780090 seconds ***
*** Multi-thread implementation(4 threads) took 14.785737 seconds ***


Speed up: 1.067251
aryan@LAPTOP-114CPH5S:/mnt/c/Users/HP/Desktop/sem_5/CS310_CDP/A5$
```

N >= 1000 was taking a lot of time. Hence skipped.

<u>Inference:</u> We see for n = 100, the speed up is maximum, then it falls again as the context switching catches up!

3. **Best methodology so far → Consistent speed up > 1 for large N observed.**

```
aryan@LAPTOP-114CPH5S:/mnt/c/Users/HP/Desktop/sem_5/CS310_CDP/A5$ ./q3 20
FOR N = 20
Thread id: 0
It solves the block: (0, 0) to (10, 10)
Thread id: 1
It solves the block: (0, 10) to (10, 20)
Thread id: 2
It solves the block: (10, 0) to (20, 10)
Thread id: 3
It solves the block: (10, 10) to (20, 20)

[+] Verified(element-wise)! Sequential and multi-threaded solution are the same.

FOR N = 20
*** Sequential implementation took 0.000079 seconds ***
*** Multi-thread implementation(1 thread per square-block) took 0.001562 seconds ***


Speed up: 0.050576
aryan@LAPTOP-114CPH5S:/mnt/c/Users/HP/Desktop/sem_5/CS310_CDP/A5$ ./q3 100
FOR N = 100
Thread id: 0
It solves the block: (0, 0) to (50, 50)
Thread id: 1
It solves the block: (0, 50) to (50, 100)
Thread id: 2
It solves the block: (50, 0) to (100, 50)
Thread id: 3
It solves the block: (50, 50) to (100, 100)

[+] Verified(element-wise)! Sequential and multi-threaded solution are the same.

FOR N = 100
*** Sequential implementation took 0.009295 seconds ***
*** Multi-thread implementation(1 thread per square-block) took 0.005675 seconds ***


Speed up: 1.637885
```

```
aryan@LAPTOP-114CPH5S:/mnt/c/Users/HP/Desktop/sem_5/CS310_CDP/A5$ ./q3 500
FOR N = 500
Thread id: 0
It solves the block: (0, 0) to (250, 250)
Thread id: 1
It solves the block: (0, 250) to (250, 500)
Thread id: 2
It solves the block: (250, 0) to (500, 250)
Thread id: 3
It solves the block: (250, 250) to (500, 500)

[+] Verified(element-wise)! Sequential and multi-threaded solution are the same.

FOR N = 500
*** Sequential implementation took 1.339734 seconds ***
*** Multi-thread implementation(1 thread per square-block) took 1.286237 seconds ***


Speed up: 1.041592
aryan@LAPTOP-114CPH5S:/mnt/c/Users/HP/Desktop/sem_5/CS310_CDP/A5$ ./q3 1000
FOR N = 1000
Thread id: 0
It solves the block: (0, 0) to (500, 500)
Thread id: 2
It solves the block: (500, 0) to (1000, 500)
Thread id: 1
It solves the block: (0, 500) to (500, 1000)
Thread id: 3
It solves the block: (500, 500) to (1000, 1000)

[+] Verified(element-wise)! Sequential and multi-threaded solution are the same.

FOR N = 1000
*** Sequential implementation took 13.604080 seconds ***
*** Multi-thread implementation(1 thread per square-block) took 11.475117 seconds ***


Speed up: 1.185529
aryan@LAPTOP-114CPH5S:/mnt/c/Users/HP/Desktop/sem_5/CS310_CDP/A5$
```