Trinity College Dublin
Coláiste na Tríonóide, Baile Átha Cliath
The University of Dublin

CS3014 – Lab Report

Name: Davy Nolan
Student No: 17330208
Date: 14/4/2020

Sparse Parallel Multi-channel Multi-kernel Convolution

Function Modifications

The first improvements made to the program were modifications to the provided code.
The function **float ***new_empty_3d_matrix()** was modified to work similarly to the given **float ****new_empty_4d_matrix()** function. The given function to create an empty 3-dimensional worked by using a 4-dimensional matrix so it was not really functioning correctly.
The given code contained a function **copy_4d_matrix()**, however, there was no function for copying a 3-D matrix so the code was modified to include a **copy_3d_matrix()** method.
The given code had a function for generating a random 4-D matrix **gen_random_4d_matrix()** that functions correctly, however the **gen_random_3d_matrix()** did not function correctly so this was modified to work similarly to the 4-D matrix generator.

OpenMp:

The next improvement made was the addition of the **#pragma omp parallel** command. This allowed parts of the program to run using multiple threads concurrently. This split the work of the program and almost doubled code runtime. **Parallel For** divided up the iterations of a for loop between the threads while making use of the extension **collapse()** merged several for loops into an iteration space and divided according to the schedule clause. The sequential execution of the iteration in all associated loops determines the order of the iterations in the collapsed iteration space. This increased the total number of iterations that will be partitioned across the threads by reducing the granularity of work to be done by each thread.

Runtime

```
dnolan5@stoker:~/assignment1$ ./a.out 128 128 1 512 512 1
Team conv time: 9474007 microseconds
```
- ⇨ 9.474007 secs

```
dnolan5@stoker:~/assignment1$ ./a.out 128 128 3 512 512 2
Team conv time: 2365653 microseconds
```
- ⇨ 2.365653secs

```
dnolan5@stoker:~/assignment1$ ./a.out 32 32 5 512 512 1
Team conv time: 9556712 microseconds
```
- ⇨ 9.556712secs

```
dnolan5@stoker:~/assignment1$ ./a.out 16 16 1 32 32 1
Team conv time: 893 microseconds
```
- ⇨ 0.000893 secs

```
dnolan5@stoker:~/assignment1$ ./a.out 16 16 5 32 32 1
Team conv time: 19318 microseconds
```
- ⇨ 0.019318 secs

```
dnolan5@stoker:~/assignment1$ ./a.out 16 16 1 32 32 4
Team conv time: 33141 microseconds
```
- ⇨ 0.033141secs

```
dnolan5@stoker:~/assignment1$ ./a.out 16 16 5 32 32 4
Team conv time: 40119 microseconds
```
- ⇨ 0.040119secs

Conclusion

As a result of doing this lab, I have learned that optimisation of code and algorithms is absolutely key in different areas of computer science. However if the optimisation of code requires so much effort and little reward, it is oftentimes redundant, especially if execution time is not critical to the functionality of the application.