CM30225 Parallel Computing
Assessed Courseork Assignment 1

November 13, 2016

1 Parralisation Technique

In order to parallelise the problem I decided to spilt the matrix up into rows then give each thread a number of these rows. More specifically each thread is given a starting row, which is applies relaxation too, then adds the number of threads to the starting row to get the next row to compute on. So if 4 threads are used on a 14, the two end rows are fixed and don't require relaxation, row matrix the rows are split up like so:

rowNumber	1	2	3	4	5	6	7	8	9	10	11	12	13	14
thread		1	2	3	4	1	2	3	4	1	2	3	4	

The threads are then all syncronised using a barrier so after all rows have been computed the read and write matrixs are swapped and the computation continues.

A quick note is that the program will actual use the number of thread plus the main thread, it was decided not to change this as the main thread just will be desheduled while it waits for all the other threads to return so won't effect the investigation significantly.

2 Avoiding Race Conditions

There are a few places where race conditions are possible:

1. Firstly two matrixs are used, one to read from and then one to write to. This avoids threads trying to write and read to the same cell simultaniously, similtanious reads are possible but not a problem. Also if one matrix was used without a lock the program would yeild diffrent results each time depending on if a cell was computed before of after its neighbours are.

2. Once a thread has finished it sets a global variable "cont" to 1 if the precision was not met. this variable does not need a lock as if one thread resets it after it doesn't matter as its still 1.

3. After all threads have finished they check if the continue value is 0 and if so return. this is preceded by a pthread_barrier_wait (line 128) to make sure all threads have finished relaxation.

4. The swapping of the matrix's and resetting the continue back to 0 needs to be done by a single thread after all threads have finished relaxing. therefore before a pthread_barrier_wait appears before and after this serial code (lines 135 and 142).

3 Correctness Testing

In order to test the program is correctly computing the answer the problem was split into three seperate sections which can be tested seperatly. Firstly the relaxation, in order to test this the first three iterations of a 5 x 5 matrix bounded by all 1's where hand calculated then compared to the output from my program running with one thread. For example in the first iteration you would expect 0.5 at the corners (1 + 1 + 0 + 0 / 4) , 0.25 on the edges (1 + 0 + 0 + 0 / 4) and zero in the middle. As you can see below the program outputs the correct value.

1	1	1	1	1
1	0.5	0.25	0.5	1
1	0.25	0.0	0.25	1
1	0.5	0.25	0.5	1
1	1	1	1	1

This was continued for the next two iterations, after this it was concluded that the program can correctly calculate each relaxation iteration.
The second step to the correctness testing is that the relaxation stops when a given precision has been reached. To test this a precision is picked, 0.01 for example, then two 3 x 3 matrix was then chosen with bounding values 0.011 and 0.01 when relaxation is performed the first matrix should iterate twice and the second only once as the precision is imediatly met.
Below are two tables showing the two runs on the program, firstly with 0.011 bounding and secondly with 0.01.

iteration	diffrence	precision	continue
1	0.011	0.01	1
2	0.0	0.01	0

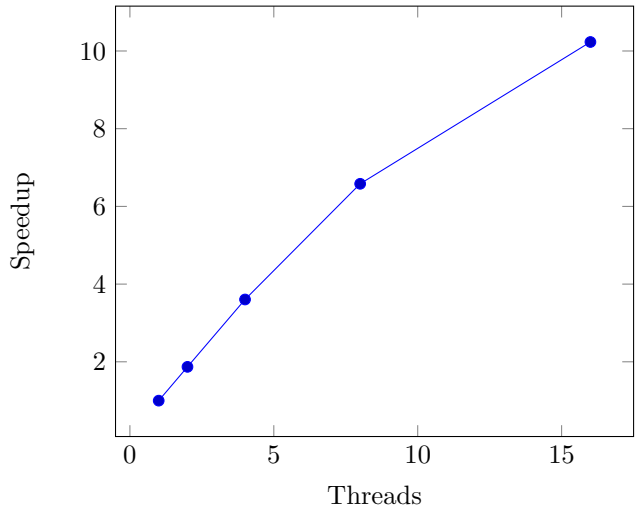
iteration	diffrence	precision	continue
1	0.01	0.01	0

This was tried with precisions 0.1, 0.01, 0.001 and according bounding values and then it was concluded that the relaxation correctly stops when the precision is met.

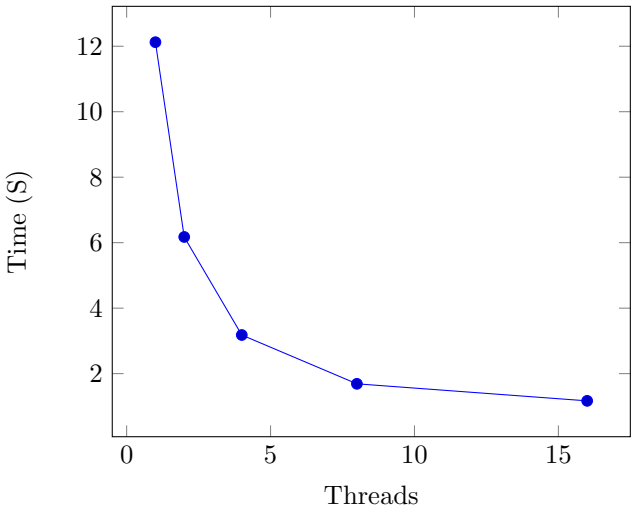
The Final thing to test is that all of the above still holds for multiple threads, to check this various number of threads were run using the same matrix size and the results were checked for equality. Once this was done for 4x4, 8x8, 10x10 and 20x20 with 2, 4, 8 and 16 threads it was conculded that the program was correctly calculating the relaxation of a matrix to a given precision with any number of threads.

4 Scalability Investigation

Threads	Time (S)	Speedup
1	0.48	1
2	0.25	1.87
4	0.13	3.6
8	$7.23 \cdot 10^{-2}$	6.58
16	$4.65 \cdot 10^{-2}$	10.23



Threads	Time (S)
1	12.12
2	6.17
4	3.18
8	1.69
16	1.17



Threads	Time (S)
1	48.38
2	24.67
4	12.69
8	6.72
16	4.43