CM30225 Parallel Computing
Assessed Courseork Assignment 1

December 12, 2016

1 Approach

In order to parallalise the relaxation problem we need to be able to split the matrix up into chunks then let each node relax its own chunk with as little communication between nodes as possible.

In order to chunk the matrix up we want to give each node as similar numbers of rows as possible to distribute work evenly. In my implementation first we calculate the rounded up value of the number of rows minus 2, because the border rows arn't relaxed, divided by the number of nodes. Each node is the allocated this many rows until there are no rows left in the matrix. For example if we run 5 nodes on a 100 x 100 matrix the rounded value is 20 so the first 4 nodes are allocated 20 and the last is allocated 18.

The least each node needs to know after each iteration is the values in the row to the left and right of its chunk. For example given a 6 x 6 matrix and 2 nodes, the first is allocated row 2 and 3 and the second rows 4 and 5, after one iteration all the first nodes needs is row 4 and the second node only needs row 3. Therefore after each iteration each node sends its outside rows to the appropriate nodes.

We can improve this by calculating the two outside rows before any others and sending them, so the node can be calculating the rest of the chunk while the communication happens. To do this MPI_Isend and MPI_Irecv are used as these are non blocking so the node won't wait for the other node to receive the value before continuing.

2 Testing

3 Scalability Investigation

3.1 Speedup

The speedup was calculated using a 10,000 by 10,000 matrix using a selection of nodes between 1 and 64. The results are given in the table below.

processors	Time (S)	Speedup on P processors
1	477	1
2	239	2
4	120	3.98
8	61	7.82
16	31	15.39
32	18	26.5
48	13	36.69
64	10	47.7

As the results show the speedup on P processors is very close to the number of processors used. This is true until over 16 processors, at this point the extra communication overhead added by using another processor starts to outweigh the extra processing power gained. Figure 1 shows this in graph form.

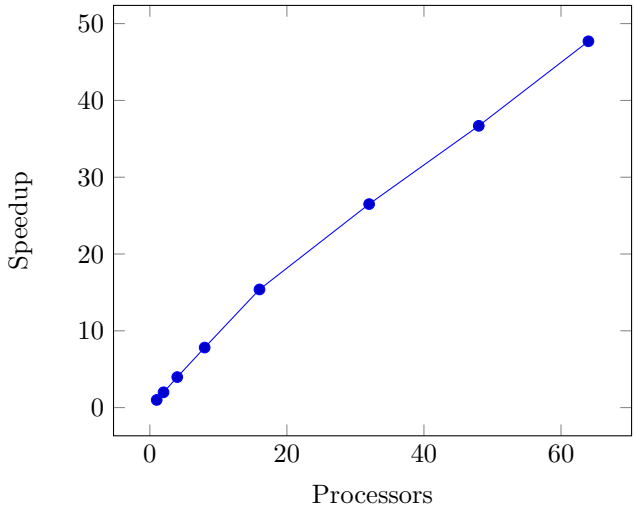

Figure 1: Speedup

3.2 Amdahls Limit

Amdahls Limit is the maximum limit speedup can reach, this is because a certain ammount of the computation must be done in serial and therefor adding more

processors will not reduce the time this takes.
The only parts of the system that have to run in parallel is the checking if all the nodes have finished and the rebuilding of the matrix at the end of the computation. However the creation of the matrix and the swapping of the read and write matrix happens on all processors so adding more will not decrease this time. Also something to note is as the number of processors increases the amount of time taken to check if each processor has done and rebuild the matrix will increase as there are more processors to communicate with.
Due to the fact that the serial parts of the problem increase as the number of processors are added the theoretical amdahl limit was not calculated.

3.3 Slowdown

When smalled matrix's were used it was noticed that as more threads were added the time time taken to complete would increase. This is due to the overhead of communicating between processors is more than the time saved by splitting the problem up further. Slowdown can be observed when a 1000 by 1000 matrix is used, results are show in Figure 2 below.

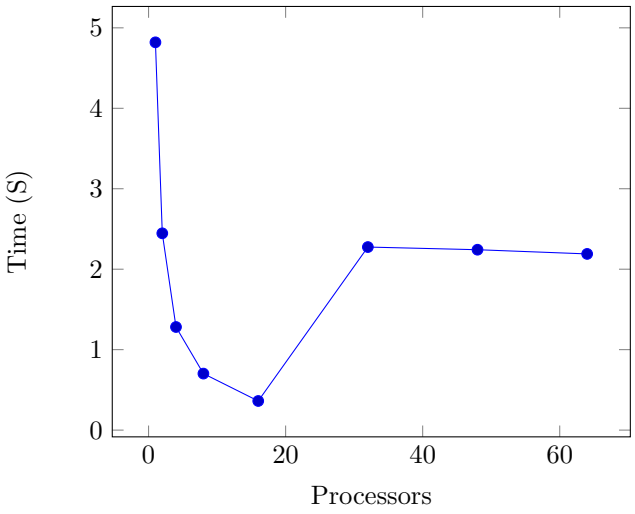

Figure 2: Slowdown

As Figure 2 shows when more that around 16 processors were used the amount of time required to complete the computation increased.

3.4 Gustafson's Law

Gustafson's Law states that if you increase the size of the problem then the amount of time spent on any sequential parts will be a less significant chunk of overall runtime. This means better speedup values can be obtained. To show

this the speedup values were calculated for differing sizes of matrices to show as the size increase so does speedup.

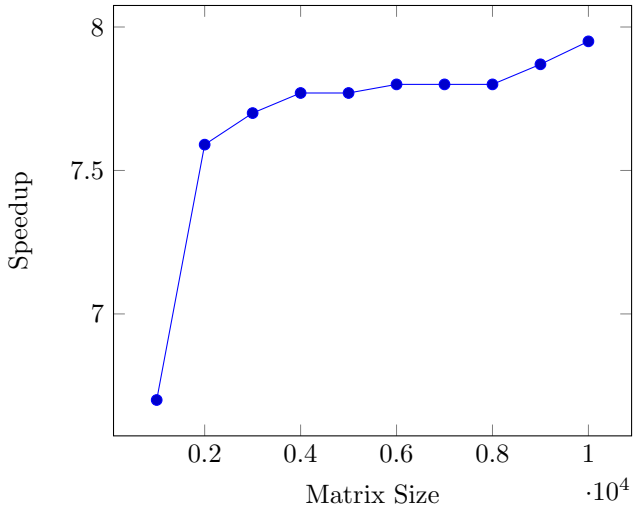

Figure 3: Gustafson's Law 8 Processors

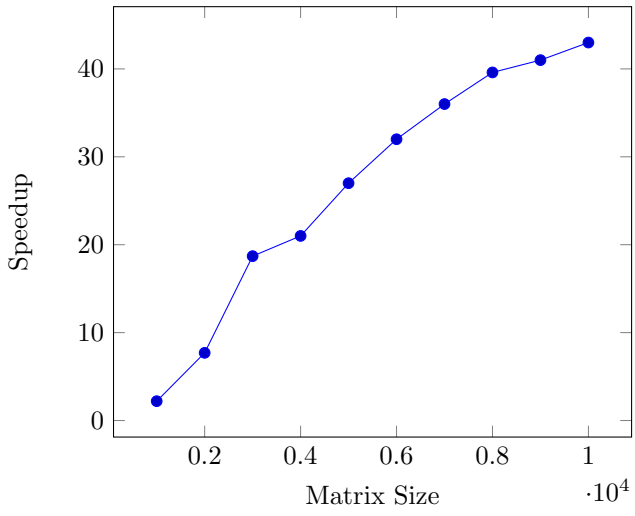

Figure 4: Gustafson's Law 64 Processors

As you can see from Figure 3 and 4 as Gustafson's Law suggests as the size of the problem is increased the speed up on P processors also increases, until the speedup reaches the number of processors used.

3.5 Efficiency

The efficiency was calculated for a number of processors and problem sizes, the results are shown in Figure 5.

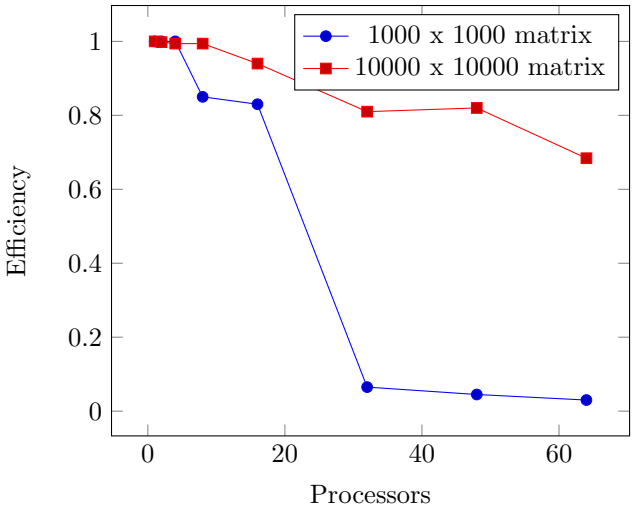

Figure 5: Efficiency

As expected from the results of the speedup investigation earlier the efficiency drops as more processors are added and the problem size is kept constant. This is because as some of the program has to be done in serial for that part every other processor is idle, therefore if more processors are added the total idle time increases.