Multiprocessor Systems (DV2544) – Pthreads programming- Project Implementation Report

Name	Robin Gustafsson	Trishala Chalasani
Personal Number	931220-1271	930602-0281
Email	roga15@student.bth.se	trch15@student.bth.se

AIM: To implement a parallel version of an algorithm-Gaussian Elimination using Pthreads and measure the speedup of parallel version on 8 cpus.

ASSUMPTIONS: A square matrix A is taken along with the vectors y and b. The values of the matrix and the two vectors must be selected reasonably.

IMPLEMENTATION: The parallel version of the implementation runs on one CPU and eight CPUs. A predetermined number of threads (determined in a compiler macro) are created. The threads first perform the division step in parallel. All threads then synchronize using a barrier. A single thread performs a small bit of work that must be performed sequentially, after which the threads again synchronize on a barrier. Elimination is then performed in parallel. They then synchronize again before starting the next iteration. Once all iterations have been completed, the threads will be joined as the work is finished.

The division of work is as follows:
For the division step, the current row is divided into continuous sections as evenly as possible. Each thread then handles one such section. The section sizes are rounded up so that all elements will always be covered. If it is not evenly divisible, the last thread will get less work.

For the elimination step, the rows are divided between the threads in a cyclical fashion. For example, if eight threads are used then the first thread will take rows 0, 8, 16, etc. As the rows are decreasing in size, this will give a more even workload than simply giving the first thread the rows 0, 1, 2, etc.

TIME MEASUREMENTS:

The command used for the execution is `/usr/bin/time -f %E ./p2 -n matrixsize`, e.g. `/usr/bin/time -f %E ./p2 -n 4096`.

All times are measured in seconds.

	Sequential			Parallel (8 threads)			Parallel (1 thread)		
Size	1024	2048	4096	1024	2048	4096	1024	2048	4096
Test 1	1.57	11.39	89.57	1.27	6.47	47.72	1.50	11.44	89.80
Test 2	1.57	11.39	89.70	1.32	6.47	47.79	1.50	11.45	89.70
Test 3	1.58	11.41	96.72	1.30	6.58	47.94	1.49	11.44	89.77
Test 4	1.84	11.39	94.97	1.30	6.49	47.78	1.50	11.44	89.77
Test 5	1.84	11.39	90.93	1.26	6.48	47.86	1.49	11.46	89.81
Average	**1.68**	**11.39**	**92.38**	**1.29**	**6.50**	**47.82**	**1.50**	**11.45**	**89.77**

SPEEDUP:

The average times were used to calculate speedup.

All times are measured in seconds.

Size	Sequential	Parallel (8 threads)	Speedup
1024	1.68	1.29	**1.30**
2048	11.39	6.50	**1.75**
4096	92.38	47.82	**1.93**

CONCLUSION:

The run time values of the sequential version and parallel implementation with 8 threads and 1 thread were slightly unpredictable at times, so we ran each configuration five times and calculated the average. We figured that it would give more reliable results while calculating speed up.

As the matrix size increases, the relative values for the parallel implementation and the sequential execution times increased and the speedup values also increased. For a particular matrix size, in most of the cases the parallel implementation (both on 8 threads and 1 thread) seems to work faster than the sequential version. We further notice that most of the cases the execution time for the parallel implementation is less if there is increase in number of threads. This implies that the synchronization mechanism that we have implemented for the threads works well.

Finally, the speedup goal of 1.5 is achieved in two of the three measured cases. With a size of 1024, the speedup falls short of the goal with a value of only 1.3. However, in the cases where the matrix size is 2048 and 4096, the speedup goal is achieved with some margin with values of 1.75 and 1.93, respectively. As the speedup values differ with size and no particular matrix size has been specified in the formulation of the goal criteria, we have interpreted it to consider the default size used in the reference implementation, i.e. 2048. Under this assumption, the criteria is satisfied.