Cameron Calv and Nicholas Sica
ECEC 413 Intro to Parallel Computer Architecture
Assignment 2: Iterative Jacobi Solver (pthreads)

Parallelization

The code was parallelized by splitting the operations for rows and columns between threads. Row operations were done in a chunking fashion split between all threads while column operations were done in a striding manner.

Evaluating Parallelization

Below are tables outlining the runtimes of the program run for parallelization. Following the tables are a plot showing the difference of the execution times between chunking and striding plotted for each data size.

Table 1: Evaluating the execution time of the program on Drexel CS college's Tux server.

	Number of Threads - Jacobi						
Data Size	1	2	4	8	16	32	64
512x512	3.61631	4.18068	4.25505	6.84831	10.92428	21.07064	47.12507
1024x1024	22.78492	16.51619	15.55491	16.12169	22.67003	41.40640	92.05264
2048x2048	184.42822	100.10813	80.27589	67.00982	59.85125	92.83042	168.76685

Table 2: Evaluating the speed-up of the program with parallel threads on Drexel CS college's Tux server.

	Number of Threads - Jacobi						
Data Size	1	2	4	8	16	32	64
512x512	-15.61%	-17.66%	-89.37%	-202.08%	-482.66%	-1203.13%	-15.61%
1024x1024	27.51%	31.73%	29.24%	0.50%	-81.73%	-304.01%	27.51%
2048x2048	45.72%	56.47%	63.67%	67.55%	49.67%	8.49%	45.72%

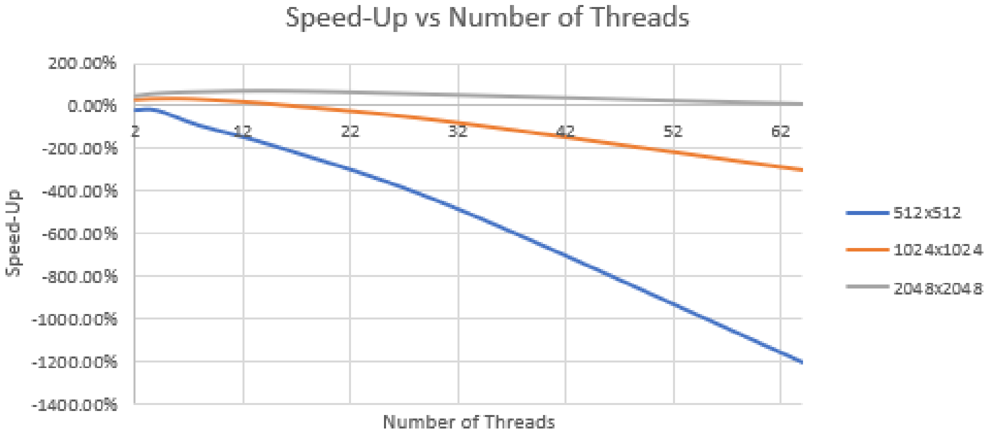

Figure 1: Plot showing the speedup per number of threads for all sizes.

It appears that speedup seem to decrease with the number of threads with the greatest speed up occurring somewhere around 16 threads for the greatest data size. The increasing rate of slowdown seems to come about because of cache sharing and a great number of context switching.