Ananiya Deneke - Jacobi Solver report 5/10/23

The parallel formulation of the solver using the chunking design pattern involves dividing the solution vector into equal-sized chunks and assigning each chunk to a separate thread. Each thread performs the Jacobi iteration for its assigned chunk in parallel with the other threads. To ensure that each thread has access to the previous iteration values of the entire solution vector, we use ping-pong buffers that are swapped between iterations.

The parallel formulation of the solver using the striding design pattern involves dividing the computation of each Jacobi iteration by threads instead of dividing the solution vector.

The following table illustrates the difference in performance achieved with each method for given matrix sizes and number of threads (Average diff b/n LHS and RHS)

Matrix sizes	#threads	Serial	Chunking	Striding
512 x 512	4	0.001352	0.000805	0.000561
512 x 512	8	0.001354	0.000486	0.000515
512 x 512	16	0.001354	0.000290	0.000494
512 x 512	32	0.001352	0.000584	0.000698
1024 x 1024	4	0.001917	0.002810	0.001330
1024 x 1024	8	0.001917	0.000905	0.001323
1024 x 1024	16	0.001916	0.001238	0.001075
1024 x 1024	32	0.001916	0.001061	0.000991
2048 x 2048	4	0.002713	0.004329	0.002815
2048 x 2048	8	0.002714	0.008137	0.004500
2048 x 2048	16	0.002714	0.009212	0.003960
2048 x 2048	32	0.002714	0.008807	0.003581

We can see that serial version stays the same across different # of threads for the same input size because it doesn't take advantage of parallel processing. Chunking performs better when the number of threads is small, and the problem size is large enough to keep all threads busy, it can reduce the overhead of thread creation and joining and can help minimize cache contention by allowing each thread to work on a contiguous block of data. On the other hand, striding can be more efficient when the number of threads is large