Cameron Calv and Nicholas Sica
ECEC 413 Intro to Parallel Computer Architecture
Assignment 1: Ways to SAXPY (Single-Precision AX Plus Y)

Version 1 and Version 2 Explanation

Version one of the code uses a chunking method. In this method the elements of the data array are split into chunks and distributed amongst the threads to do their computations on. Version two of the code uses a striding method which assigns a starting element for each thread. Each thread works on the *i*th element of the data. For instance, for four threads, thread 0 takes 0, 4, 8, thread 1 takes 1, 5, 9, thread 2 takes 2, 6, 10, thread 3 takes 3, 7, 11.

Evaluating 'Chunking' vs 'Striding'

Below are tables outlining the runtimes of the program run for both versions of 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 performance of Version 1, the chunking technique on Drexel CS college's Tux server.

	Number of Threads						
Data Size	1	2	4	8	16	32	64
10	0.000689	0.000389	0.00056	0.001045	0.001933	0.003405	0.008544
100	0.00054	0.000545	0.000701	0.001317	0.001673	0.003872	0.011075
1,000	0.000222	0.000434	0.000487	0.001278	0.003151	0.003622	0.004924
100,000	0.000303	0.000545	0.000459	0.00086	0.002034	0.003287	0.006067
1,000,000	0.001472	0.00086	0.000902	0.001113	0.001793	0.003268	0.006162
10,000,000	0.007658	0.004894	0.004009	0.003586	0.003369	0.007273	0.007966
100,000,000	0.086148	0.047156	0.028256	0.035901	0.034409	0.022496	0.01619

Table 2: Evaluating the performance of Version 2, the striding technique on Drexel CS college's Tux server.

	Number of Threads						
Data Size	1	2	4	8	16	32	64
10	0.000082	0.000336	0.000478	0.000973	0.001547	0.003355	0.005366
100	0.000413	0.000452	0.000295	0.001032	0.001555	0.003054	0.004335
1,000	0.000101	0.000148	0.000298	0.000841	0.001987	0.003283	0.006241
100,000	0.000269	0.000412	0.000549	0.000866	0.001908	0.002937	0.007462
1,000,000	0.001136	0.001978	0.001765	0.002086	0.004921	0.008336	0.006216
10,000,000	0.010462	0.010364	0.009861	0.020132	0.032277	0.034367	0.026166
100,000,000	0.084935	0.096193	0.110201	0.156752	0.283934	0.363786	0.176287

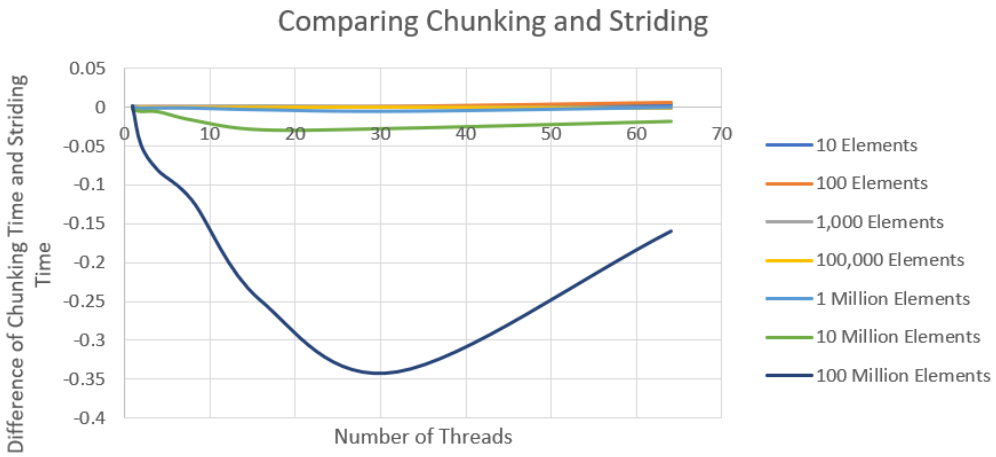

Figure 1: Plot showing the difference between chunking time and striding time.

As the number of data points get large, there is an increasingly great difference between chunking and striding. One of the main reasons for this drop in performance is that for striding false sharing can occur in caches which leads to a lot of invalidation of the caches leading to greater execution times. It can also be seen that for lower numbers of elements, the difference is not so great.