

Cole Bardin
ECEC-413
1/31/25

SAXPY PThread Report

Introduction

This project explores two parallelization methods for the SAXPY operation using the pthread library. SAXPY, which stands for "Single-Precision AX Plus Y," is a core operation in the Basic Linear Algebra Subprograms (BLAS) library. The operation computes:

$$y[i] = a * x[i] + y[i]$$

for two input vectors, x and y, and a scalar a. The objective is to implement and compare the performance of two parallel versions of SAXPY—the **chunking** method and the **striding** method—against the serial implementation.

Approach

1. **Chunking Method:**

In this approach, the input vectors are divided into equal-sized chunks, and each thread is responsible for computing SAXPY on one chunk. The division ensures contiguous processing, reducing thread coordination overhead.

2. **Striding Method:**

Here, threads process elements of the vectors in a strided fashion, with each thread operating on every kth element, where k is the total number of threads. This involves more scattered memory access.

Results

Both methods were evaluated on vector sizes of 10^4 , 10^6 , and 10^8 elements, using 4, 8, and 16 threads, and the results were compared to the serial implementation.

4 Thread SAXPY: Series, Chunking and Striding

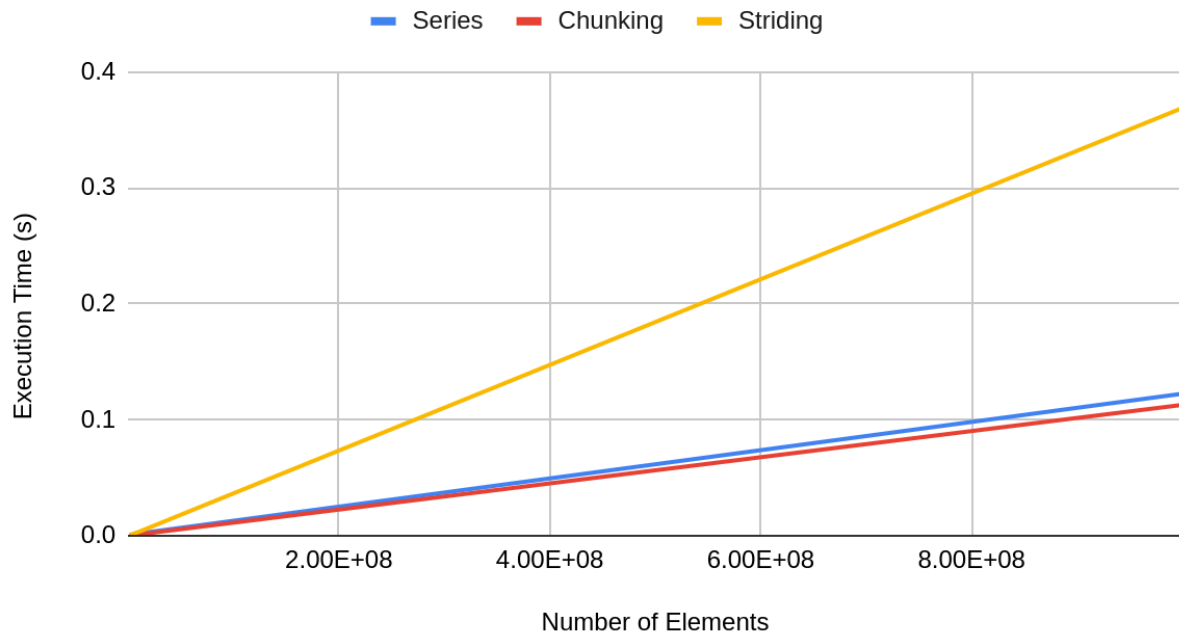


Figure 1: 4 Thread SAXPY execution times versus number of elements comparing single thread, chunking and striding

8 Thread SAXPY: Series, Chunking and Striding

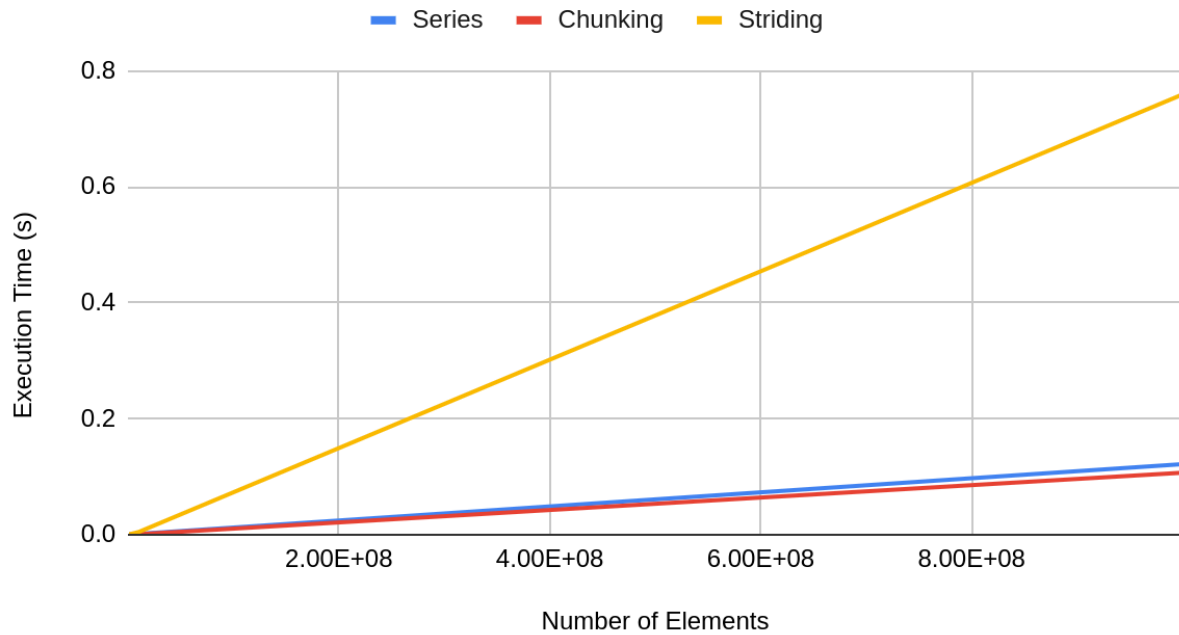


Figure 2: 8 Thread SAXPY execution times versus number of elements comparing single thread, chunking and striding

16 Thread SAXPY: Series, Chunking and Striding

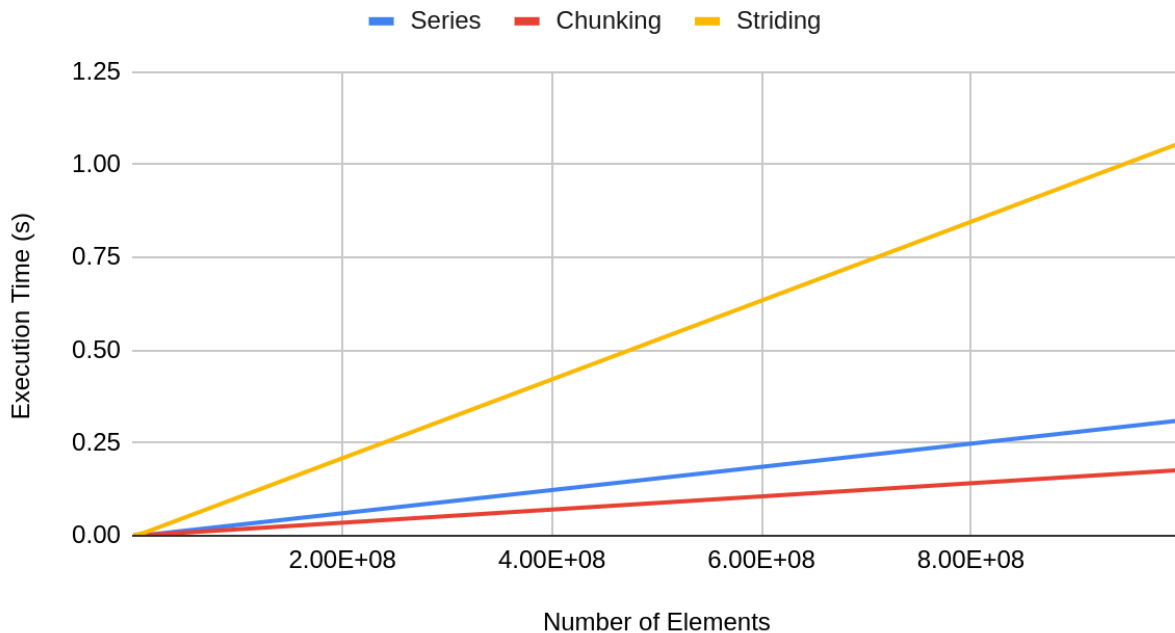


Figure 3: 16 Thread SAXPY execution times versus number of elements comparing single thread, chunking and striding

Conclusion

It can be seen that as the number of elements increases, the execution time also increases. Due to the cache conflicts created by the striding method, its performance is consistently worse than a single threaded model. Using a cache-aware policy such as chunking provides performance benefits over serial execution, unlike striding. The performance benefits of parallelism scales with the number of threads and the size of the array.

The array sizes resulted in short durations of elapsed time. This led to comparisons that were pushing the limits of the granularity of the deprecated and non-monotonic `gettimeofday()` function. Therefore, the values recorded by this program should not be considered statistically significant.