**Student Names**

Trung Hoang

Cortney Reed

# **Assignment Discussions**

1. The design of your multi-threaded implementations, using code or pseudocode to clarify the discussion.

There are 2 multi-threaded implementations created to parallelize a SAXPY loop. The following is the SAXPY loop

Chunking was implemented such that there are k chunks where k is the number of threads and each chunk has the size of at most n/k where n is the number of elements in the array. Each thread is responsible for performing SAXPY loop at its chunk. The following codes show how each thread does SAXPY loop with chunking method



Striding was implemented such that k threads “strides” over each element of the vectors calculating the SAXPY along the way. The length of the stride is n/k. The number of strides is equal to the number of threads and using a while loop the function below calculates the SAXPY corresponding to the thread id. The following codes show how each thread does SAXPY loop with striding method



1. The performance achieved by 4, 8, and 16 threads, for 104, 106, and 108 elements, relative to the serial version. As part of your report, provide a clear reasoning for any differences that you might observe in the performance achieved by your parallel versions.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **No of elements** | **No of threads** | **Execution Time (s)** | | |
| **Chunking** | **Striding** | **Serial Version** |
| **10^4** | **4** | 0.000797 | 0.000277 | 0.000010 |
| **8** | 0.001310 | 0.000691 |
| **16** | 0.002131 | 0.001272 |
| **10^6** | **4** | 0.001712 | 0.003438 | 0.002168 |
| **8** | 0.001820 | 0.005276 |
| **16** | 0.002228 | 0.009296 |
| **10^8** | **4** | 0.152252 | 0.406253 | 0.296312 |
| **8** | 0.118589 | 0.757925 |
| **16** | 0.124142 | 1.082177 |

From the table, it can be seen that both versions of parallelization are slower than the serial version. This is caused by false sharing when we are reading and writing on the same cache line. In SAXPY loop, we read and write y in the same calculation step. This creates false sharing meaning multiple threads are not sharing the same version/value of a memory in the cache. This adds computational time and make both versions of parallelization slower.

There are a few solutions we can do to improve this behavior. One solution is to allocate the result to a different cache line before writing it back to the original cache line that we are reading y from. Another solution is to add padding to our arrays such that the array lies on multiple cache lines where we have each thread responsible for each cache line.