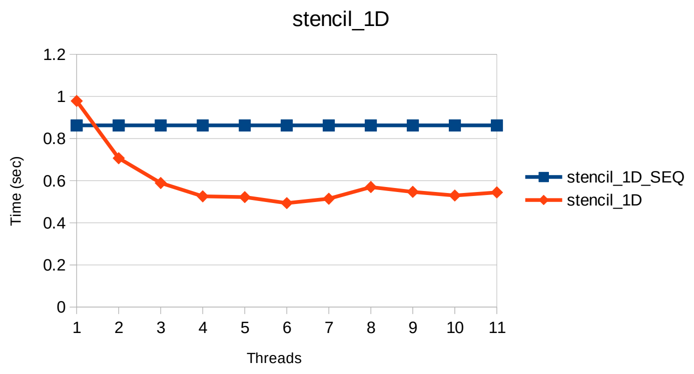
Algorithm description: The stencil_1D, stencil_2D, and mat_vec programs seem to all share the same purpose of creating an array or 2D array that calculates and fills in numbers. They don't seem to serve any more purpose than to take time looping and making calculations.

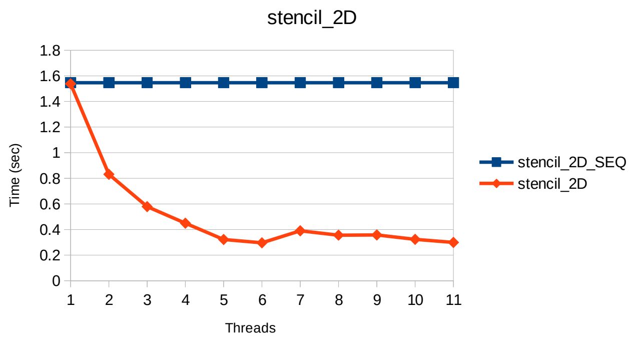
Description of parallel approach: The approach uses parallel for loops in order to simultaneously calculate each index of an array, thus lowering the total time it takes to calculate every index. This will be used when we go through each data/array point and for the iterations we are going to do.

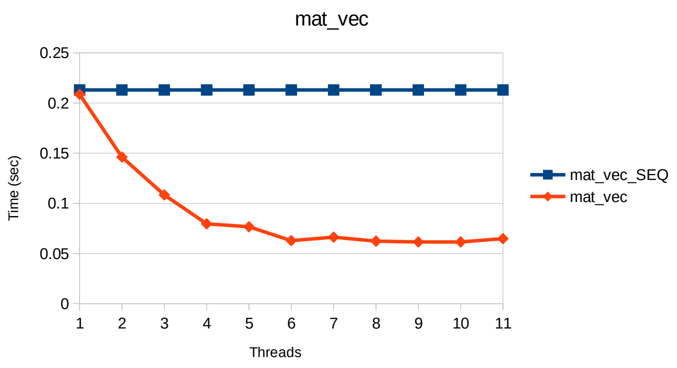
Experimental setup:

- **Machine description**:
  - **Name**: buttermilk
  - **Number of cores**: 6
  - **Cache sizes**:
    - L1d cache: 32K
    - L1i cache: 32K
    - L2 cache: 256K
    - L3 cache: 15360K
- **Experiments planned**:
  - I will run each sequential vs parallel program with the same number of data points and iterations. So I will run stencil_1D_SEQ and stencil_1D with the same numbers. I will run each program with their numbers 11 times (removing the max and min data points), taking the average, and incrementally increase the number of threads used so we can see the increase more threads has on a parallel program.
  - The stencil_1D program with have 4000 data points and 200,000 iterations.
  - The stencil_2D program with have 800 data points and 2000 iterations.
  - The mat_vec program with have 25,000 data points and 10,000 iterations.

Experimental results:







Conclusion and observations: The speed of each program decreased significantly when parallel for loops were added. In each case the time it took to complete the program was more than cut in half by the time 6 threads were being used. Although, when only one thread was used the time it took the complete the program seemed to stay the same or increase. It also seems that using more than 6 threads doesn’t lead to a decrease in time to complete the program because the overhead is greater than the time saved from the parallel execution. In conclusion, you should only use a parallel for loop for computations that are time and memory intensive, otherwise you are creating a lot of extra work for little to no benefit.