Scaling Statement

In strong scaling, the problem size stays fixed (in this case 100 mil), but the number of processers change. The goal is for the same problem size to run more efficiently by adding processors.

I figured that I would make a chart of some sort to chronicle the program times. Basically, from my timing tests, the performance got better for each function as more processors were added. As can be seen in the following data, for each function, the time to complete the function got better as the processors increased. The timings differed with respect to creating the array, however.

Figure 1

Thread count	Create array	Sum array	Standard deviation	Smooth
1	1.997912 seconds	0.269570 seconds	1.806752 seconds	0.985024 seconds
2	6.026697 seconds	0.140628 seconds	1.065419 seconds	0.511617 seconds
4	21.519553 seconds	0.083628 seconds	0.683274 seconds	0.266861 seconds
8	15.112328 seconds	0.039096 seconds	0.497991 seconds	0.140621 seconds
16	13.778166 seconds	0.033306 seconds	0.454945 seconds	0.109811 seconds
32	14.355054 seconds	0.030182 seconds	0.452548 seconds	0.093699 seconds

Figure 2

Threads: 1

```
>>Total time to create array: 1.997912 seconds
>>Total time to calculate the sum of the array: 0.269570 seconds
>>Total time to calculate parallelized standard deviation: 1.806752 seconds
>>Total time to calculate original/serial standard deviation: 1.833107
seconds
Total time to calculate parallelized array after smooth: 0.985024 seconds
```

Threads: 2

```
>>Total time to create array: 6.026697 seconds
>>Total time to calculate the sum of the array: 0.140628 seconds
>>Total time to calculate parallelized standard deviation: 1.065419 seconds
>>Total time to calculate original/serial standard deviation: 1.814152
seconds
```

```
Total time to calculate parallelized array after smooth: 0.511617 seconds
```

Threads: 4

```
>>Total time to create array: 21.519553 seconds
>>Total time to calculate the sum of the array: 0.083628 seconds
>>Total time to create array: 21.519553 seconds

>>Total time to calculate parallelized standard deviation: 0.683274 seconds
>>Total time to calculate original/serial standard deviation: 1.811824
seconds
Total time to calculate parallelized array after smooth: 0.266861 seconds
```

Threads: 8

```
>>Total time to create array: 15.112328 seconds
>>Total time to calculate the sum of the array: 0.039096 seconds
>>Total time to calculate parallelized standard deviation: 0.497991 seconds
>>Total time to calculate original/serial standard deviation: 1.815412
seconds
Total time to calculate parallelized array after smooth: 0.140621 seconds
```

Threads: 16

```
>>Total time to create array: 13.778166 seconds
>>Total time to calculate the sum of the array: 0.033306 seconds
>>Total time to calculate parallelized standard deviation: 0.454945 seconds
>>Total time to calculate original/serial standard deviation: 1.884992
seconds
Total time to calculate parallelized array after smooth: 0.109811 seconds
```

Threads: 32

```
>>Total time to create array: 14.355054 seconds
>>Total time to calculate the sum of the array: 0.030182 seconds
>>Total time to calculate parallelized standard deviation: 0.452548 seconds
>>Total time to calculate original/serial standard deviation: 2.029504
seconds
Total time to calculate parallelized array after smooth: 0.093699 seconds
```