Coulby Nguyen

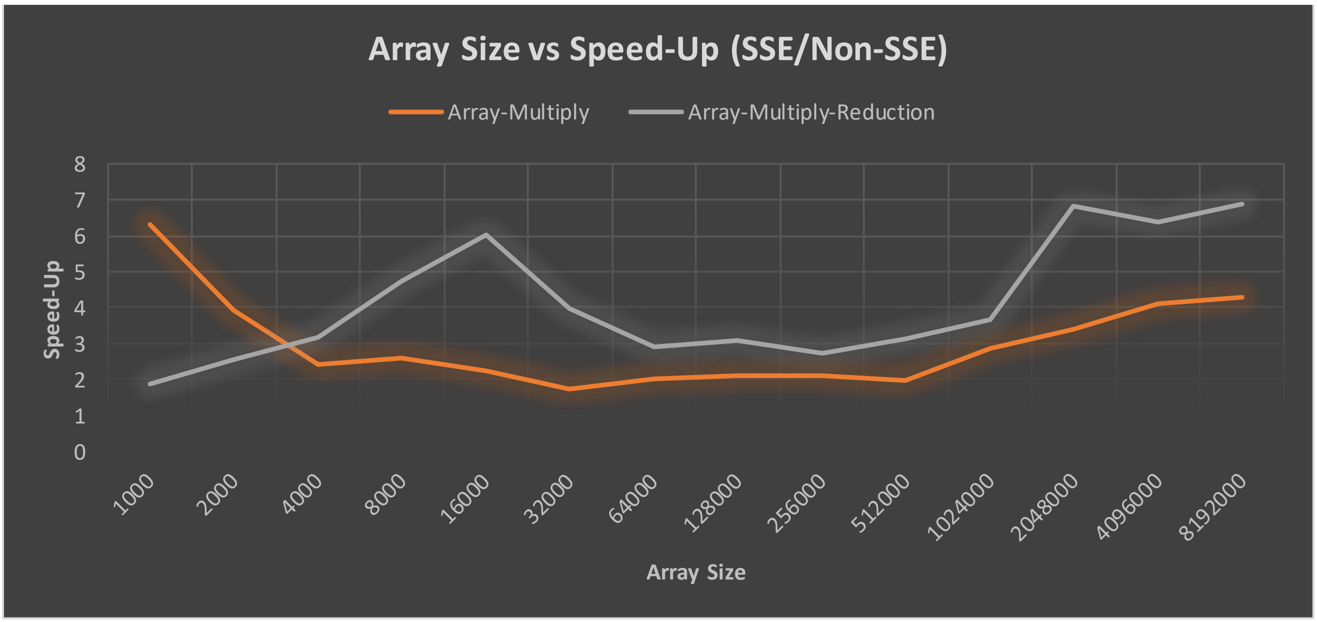
CS 475

Due 5/13/2019

# Project 4 Write Up

1. What Machine You ran this on
    a. I ran this project on the Oregon State University flip server using an SSH through puTTy.
2. Show the Table and Graph

| Array Size | Array-Multiply | Array-Multiply-Reduction |
|---:|---:|---:|
| 1000 | 6.31078 | 1.87718 |
| 2000 | 3.91265 | 2.53476 |
| 4000 | 2.41943 | 3.19354 |
| 8000 | 2.60978 | 4.74438 |
| 16000 | 2.23751 | 6.03189 |
| 32000 | 1.73656 | 3.99546 |
| 64000 | 1.99919 | 2.92713 |
| 128000 | 2.11927 | 3.0786 |
| 256000 | 2.12456 | 2.72778 |
| 512000 | 1.96941 | 3.15339 |
| 1024000 | 2.84349 | 3.65843 |
| 2048000 | 3.41931 | 6.82715 |
| 4096000 | 4.0943 | 6.38199 |
| 8192000 | 4.30964 | 6.88093 |



3. What patterns are you seeing in speedups?
    a. With small array sizes there is greater speedups with the Array-Multiply, however when the array size grows large there is greater speed-ups in Array-Multiply-reduction.
4. Are they consistent across a variety of array sizes?
    a. I do not believe it is necessarily consistent in respect to array sizes, as there is a dip in speed up with the array-multiply-reduction after rising to good performance at array size of 16000.
5. Why or Why not do you think?
    a. I think this inconsistency could be due to the SIMD reaching its limit and then still needing to do multiplies in an Non sse multiplication reduction way. It could also be inconsistent based on the array size because of the limit is based off the array size divided by the SSE_WIDTH then multiplied by the SSE_WIDTH which means that in some array sizes there is no remainder past the limit, and in other cases there is a lot of remainder that needs to be done a non sse multiplication reduction way.
6. Knowing that SSE SIMD is 4-float-at-atime, why could you get a speed up of < 4.0 or > 4.0 in the array multiplication?
    a. I think that one of the main reasons that I got scores above and below the 4.0 mark was based on the size of the array and that if there was not many overhang based on the SSE_WIDTH it would provide a great speed up, and on the contrary if there was a lot of overhang, it would not provide over the 4.0 speed up mark.
7. Knowing that SSE SIMD is 4-float-at-atime, why could you get a speed up of < 4.0 or > 4.0 in the array-multiplication-reduction?
    a. Again, I think the majority of this is based off the overhang, and how much of that data would have to be computed through a non-sse method after the limit. However, I think that the speed up that was achieved with the larger data sets was due to the Fused Multiply Add concept in which both operations are done in a single step. This would result in great speed up in large sizes as there would be less operations taking place.