Lab 2

Dr. Shirley Moore

CS 5334

Oscar Galindo

1. You will be assigned one configuration to run to obtain the serial execution time for the vectorized code or two configurations to run to obtain the non-vectorized parallel execution times. Please put your results into the tables on the Lab 2 collaboration page.

Done, I did place my run times and took the liberty to include more run times than mine.

1. Complete the table below. To complete the Pmeasured and Vmeasured columns, you should run a fully vectorized and fully parallel executable for the configuration in each row and compute the ratios for Pmeasured and Vmeasured using the times from task 1 for the denominators.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Data Layout | Precision | Vectorization | Pmeasured | Vmeasured | B = Pmeasured \* Vmeasured | GFLOPS/s | Runtime (sec) |
| SOA | Single | AVX | 189.96/12.38 = 15.344 | 104.82/12.38 = 8.4668 | 129.91 | 807.996 | 12.38 |
| SOA | Single | SSE | 513.3/31.653 = 16.216 | 104.82/31.653 = 3.311 | 53.69 | 315.956 | 31.653 |
| AOS | Single | AVX | 2462.7/153.92 = 15.990 | 108.52/153.92 = 0.705 | 11.272 | 64.962 | 153.92 |
| AOS | Single | SSE | 871.84/56.48 = 15.431 | 108.52/56.48 = 1.921 | 29.642 | 176.836 | 56.48 |
| SOA | Double | AVX | 476.23/31.413 = 15.160 | 102.325/31.413 = 3.257 | 49.376 | 20.999 | 31.413 |
| SOA | Double | SSE | 1123.2/70.966 = 15.827 | 102.325/70.966 = 1.441 | 22.806 | 140.926 | 70.966 |
| AOS | Double | AVX | 974.82/60.929 = 15.999 | 92.136/60.929 = 1.512 | 24.190 | 164.129 | 60.929 |
| AOS | Double | SSE | 1484.9/92.418 = 16.06 | 92.136/92.418 = 0.996 | 16.010 | 108.210 | 92.418 |

1. Compute E, E\_par, and E\_vec for each of the 8 configurations in the table. Discuss the results. Do you think that using additional cores or wider vector units would be beneficial for this code?

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Data Layout | Precision | Vectorization | Pmeasured | Vmeasured | Epar = Pmeasured/#cores | Evec = Vmeasured /Videal | E |
| SOA | Single | AVX | 189.96/12.38 = 15.344 | 104.82/12.38 = 8.4668 | 0.959 | 1.05835 | 1.0149097 |
| SOA | Single | SSE | 513.3/31.653 = 16.216 | 104.82/31.653 = 3.311 | 1.0135 | 0.827 | 0.8381645 |
| AOS | Single | AVX | 2462.7/153.92 = 15.990 | 108.52/153.92 = 0.705 | 0.999 | 0.088 | 0.07992 |
| AOS | Single | SSE | 871.84/56.48 = 15.431 | 108.52/56.48 = 1.921 | 0.964 | 0.24 | 0.23136 |
| SOA | Double | AVX | 476.23/31.413 = 15.160 | 102.325/31.413 = 3.257 | 0.947 | 0.814 | 0.770858 |
| SOA | Double | SSE | 1123.2/70.966 = 15.827 | 102.325/70.966 = 1.441 | 0.989 | 0.495 | 0.489555 |
| AOS | Double | AVX | 974.82/60.929 = 15.999 | 92.136/60.929 = 1.512 | 0.999 | 0.378 | 0.377622 |
| AOS | Double | SSE | 1484.9/92.418 = 16.06 | 92.136/92.418 = 0.996 | 1.003 | 0.5015 | 0.5030045 |

The results show that the inclusion of more cores and the use of vectorization in general help to decrease the time cost of running all configurations. Notably, for SOA we do see in all cases a more efficient use of the cores. And, also in general we can see how the configurations that use AOS do speedup with the use of cores but not when the code is vectorized and also parallelized there isn’t as much efficient use of the vectorization (as expected). There was a weird result in the configuration of SOA, Single, AVX, which indicates that the overall efficiency is over 1. This result of course doesn’t make sense, but these are the results I obtained. For the AOS I don’t believe a wider vectorization would help because as we know vectorization doesn’t improve performance with AOS. On the contrary, the use of Vectorization does help and it may help if we increase the size of the vectorization so that we may manipulate more data at once.

1. Explain the difference between SOA and AOS. Describe your results for SOA vs. AOS and explain why one or the other works better for this code. You may want to look at the generated assembly code to answer this question.

The difference between Structure of Arrays and Array of Structures is the ordering of memory and the contents in the memory constructions. In the AOS we construct an array/a space in memory where we place data of different size, for example floats, then characters, etc. On the contrary, but still similarly we have SOA which means we open a memory construction that contains references to other spaces of memory. The use of SOA aids vectorization more than AOS, so, even though the use of parallel runs aids but kinds of experiments SOA further is optimized through vectorization which does show in the results.

1. Generate the compiler vectorization reports for each of the 8 configurations and view the reports. Discuss the results in terms of how many and which loops were vectorized and why some loops were not vectorized.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Data Layout | Precision | Vectorization | Loops Vectorized | Loops partially vectorized | Loops Peeled | Ignored Loops | Total Loops |
| SOA | Single | AVX | 4 | 0 | 2 | 14 | 19 |
| SOA | Single | SSE | 2 | 0 | 2 | 15 | 19 |
| AOS | Single | AVX | 3 | 2 | 3 | 11 | 19 |
| AOS | Single | SSE | 5 | 2 | 3 | 9 | 19 |
| SOA | Double | AVX | 5 | 0 | 1 | 13 | 19 |
| SOA | Double | SSE | 6 | 0 | 2 | 10 | 19 |
| AOS | Double | AVX | 7 | 2 | 2 | 8 | 19 |
| AOS | Double | SSE | 5 | 2 | 2 | 10 | 19 |

In general, on these codes, we can see that even though it is known that if we implement code using AOS we will get less vectorization here it is not necessarily the case. As we can see that more vectorizations are done in the AOS implementation than in the SOA implementations. Nevertheless, quality can also not necessarily mean quality, and as seen in these reports there is a major area which vectorization can occur. The lines for such area are/start at line 451:

Table

Description automatically generated

This area, as seen in the previous picture, is a high-loop-density-region, this means that most of the work is done in this piece of code. What is different about the vectorization between AOS, and SOA is that in AOS implementations some loops of this area can only be partially vectorized whereas in the SOA implementations this area attains a complete vectorization. This, as the runtimes suggest, account for most of the speedup and difference in time that it takes to complete the task under an SOA implementation to an AOS implementation with similar vectorization schemes and precisions. Possibly, the reason why the vectorizations are not complete in this are for AOS is because of the inability to transform some of the operations from single/fixed units of operation to a large concatenation of many operations at once. This last normally happens with AOS since the “arrays” contain individual pieces of the array instead of holding the reference to where the data resides.

1. Experiment with additional vectorization using directives to see if you can improve E\_vec and E. You need only use the better performing choice of SOA vs. AOS for this task.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Data Layout | Precision | Vectorization | Pmeasured | Vmeasured | Epar = Pmeasured/#cores | Evec = Vmeasured /Videal | E |
| SOA | Double | AVX | 476.23/31.413 = 15.160 | 102.325/31.413 = 3.257 | 0.947 | 0.814 | 0.770858 |
| AOS | Double | SSE | 1484.9/92.418 = 16.06 | 92.136/92.418 = 0.996 | 1.003 | 0.5015 | 0.5030045 |

I tried modifying AVX2 to AVX3 and AVX 4.2 which resulted in the execution for SOA taking double the time that it takes for AVX2. For the AVX

I modified the line of compilation for the AOS DOUBLE SSE implementation by compiling with xSSE4.2, xSSSE3, SSE2 and optimization 4, as well as xSSE3 optimization 4, and nothing helped.