Intrinics based data parallelism in Array of Structure for Gridding and Degridding

M. Nimalan

Introduction

Structure of Array vs Array of Structure

Array of structure and Structure of Arrays are contrasing way of arranging sequence of data to exploit data parallelism [Wikipedia, 2013].

Array of Structure of Array(AOSOA) or Tiled Array of Structs

There is also a third form of structuring Array of Structure of Array [Sharp, 2013] which promises the vectorization benefits of SoA and preserves much of the simpler interface of AoS.

```
// Fits within SSE 128 bit registers
struct complex2 {
  alignas(8) float imag[2];
  alignas(8) float real[2];
};

// Fits within AVX 256 bit registers
struct complex4 {
  alignas(16) float imag[4];
  alignas(16) float real[4];
};
```

Shortcomings:

1. Although this is faster than AoS is slower than a having the arrays seperate in SoA
2. The reason for the difference in performance is due to the generated ASM. The compiler tries it's best to optimize this, but the ASM generated in the SoA is much simpler

3. We also need to transform the data into the tiled format, and convert it back

SIMD Intrinics based data parallelism to enable Tiled Array of Structs

One observation in all of the AoS, SoA and AoSoA techniques is that the generated ASM isn't the best that can, and the previous approach needs the data to be in a tiled format.

The idea is if intrinics could be used to transform the data as well as process it then we save on data movement(transformation) and still get vectorization

The structure would be as follows

```
struct complex2 {
  std::complex<float> d[2];
}
```

Shortcomings:

1. Intrinsics have to be written for each algorithm

References

Amanda K Sharp. Memory layout transformations, 2013. URL https://www.intel.com/content/www/us/en/developer/articles/technical/memory-layout-transformations.html.

Wikipedia. Aos and soa, 2013. URL https://en.wikipedia.org/wiki/AoS_and_SoA.