Kokkos SIMD

Dong Hun Lee, Sandia National Laboratories

Kokkos User Group Meeting 2023

December 14, 2023

Sandia National Laboratories is a multi-mission laboratory managed and operated by National Technology and Engineering Solutions of Sandia, LLC., a wholly owned subsidiary of Honeywell International, Inc., for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-NA0003525.
SAND2023-14460C

SIMD

- Utilization of vector registers in processors to operate on vector lanes in parallel
- ISA and Intrinsics
- Compiler auto-vectorization

Standard C++

- *C++ Extensions for Parallelism Version 2*[1]
- *std::simd — merge data -parallel types from the Parallelism TS 2*[2]

[1] N4808 Working Draft, C++ Extensions for Parallelism Version 2 by Jared Hoberock
[2] P1928 std::simd - merge data-parallel types from the Parallelism TS 2 by Matthias Kretz

Kokkos SIMD

- Abstraction layer of platform-specific vector types for SIMD intrinsics
- Data-parallel types based on *Extensions for Parallelism, Version 2*[1]
- Alignment with `std::simd`[2] targeted for ISO standard C++26

[1] N4808 Working Draft, C++ Extensions for Parallelism Version 2 by Jared Hoberock

[2] P1928 std::simd - merge data-parallel types from the Parallelism TS 2 by Matthias Kretz

Supported SIMD Types

	NEON	AVX2	AVX512
float	✓	✓	✓
double	✓	✓	✓
int32_t	✓	✓	✓
uint32_t			✓
int64_t	✓	✓	✓
uint64_t	✓	✓	✓

Data Parallel Types

Kokkos::Experimental::simd

- template <class T, class Abi> class simd

Kokkos::Experimental::simd_mask

- template <class T, class Abi> class simd_mask

Kokkos::Experimental::native_simd<T>
Kokkos::Experimental::native_simd_mask<T>

native is determined by KOKKOS_ARCH and Kokkos::DefaultExecutionSpace

Copy functions

- `template <class U, class Flags> void copy_from(const U* mem, Flags flags)`
- `template <class U, class Flags> void copy_to(U* mem, Flags flags)`

Subscript operators

- `reference operator[](std::size_t)`
- `value_type operator[](std::size_t)`

Supported operations

- Basic arithmetic operators
- Shift operators (logical or arithmetic shifts)
- Comparison operators
- Rounding and remainder functions
- `simd_mask` reductions
- Subset of `<cmath>` functions

Supported operations

copysign	max	tan	asinh	pow
abs	min	asin	acosh	hypot
sqrt	exp2	acos	atanh	atan2
cbrt*	log10	atan	erf	
exp*	log2	sinh	erfc	
log*	sin	cosh	tgamma	
fma*	cos	tanh	lgamma	

- Not all functions are implemented using intrinsics

** requires Intel SVML; otherwise uses a serial fallback implementation*

Conditionals: `where_expression`

- Provides references to the simd values as described by a mask
- Used to perform conditional logic using a mask

```
template <class M, class T> class const_where_expression;
template <class M, class T> class where_expression;

template <class T, class Abi>
const_where_expression<simd_mask<T, Abi>, simd<T, Abi>>
where(typename simd<T, Abi>::mask_type const&, simd<T, Abi> const&);

template <class T, class Abi>
where_expression<simd_mask<T, Abi>, simd<T, Abi>>
where(typename simd<T, Abi>::mask_type const&, simd<T, Abi>&);
```

Conditionals: `where_expression`

Copy functions

- `copy_from`
- `copy_to`
- `gather_from`
- `scatter_to`

Assignment operators

- Assignment
- Compound assignments

Example

```
double x[N];
double y[N];
double z[N];
double r[N];

for (int i = 0; i < N; ++i) {
  r[i] = sqrt(x[i] * x[i] + y[i] * y[i] + z[i] * z[i]);
}
```

Example

```cpp
#include <Kokkos_SIMD.hpp>

using simd_type = Kokkos::Experimental::native_simd<double>;
using tag_type = Kokkos::Experimental::element_aligned_tag;

constexpr int width = int(simd_type::size());
simd_type simd_x, simd_y, simd_z, simd_r;

for (int i = 0; i < N; i += width) {
  simd_x.copy_from(x + i, tag_type());
  simd_y.copy_from(y + i, tag_type());
  simd_z.copy_from(z + i, tag_type());
  simd_r = Kokkos::sqrt(simd_x * simd_x + simd_y * simd_y + simd_z * simd_z);
  simd_r.copy_to(r + i, tag_type());
}
```

Example

```
#include <Kokkos_SIMD.hpp>

using simd_type      = Kokkos::Experimental::native_simd<double>;
using simd_view_type = Kokkos::View<simd_type*>;
using tag_type       = Kokkos::Experimental::element_aligned_tag;

constexpr int simd_view_length = N / simd_type::size();

simd_view_type simd_vx("simd_view_x", simd_view_length);
/*...*/

Kokkos::RangePolicy<> policy(0, simd_view_length);
Sqrt_Functor sqrt_functor(simd_vx, simd_vy, simd_vz, simd_vr);
Kokkos::parallel_for(policy, [&](int i) {
  int offset = i * simd_type::size();
  simdv_x(i).copy_from(x + offset, tag_type());
  simdv_y(i).copy_from(y + offset, tag_type());
  simdv_z(i).copy_from(z + offset, tag_type());
  sqrt_functor(i);
  simdv_r(i).copy_to(r + offset, tag_type());
});
```

Missing capabilities

Currently unavailable operations

- `simd` compound assignments
- `simd` reductions
- `simd` constructor taking a mem pointer as an argument