OAK RIDGE National Laboratory

Kokkos and standard C++

Damien Lebrun-Grandie

ORNL is managed by UT-Battelle LLC
for the US Department of Energy

U.S. DEPARTMENT OF ENERGY

Kokkos

```
template<
  class DataType
  [, class Layout = @see below@]
  [, class MemorySpace = @see below@]
  [, class MemoryTraits = @see below@]
> Kokkos::View;
```

Made up syntax ™Kokkos to signify variable-length template arguments à la **kwargs in Python except that the specified arguments must appear in order.

```
Layout = typename MemorySpace::layout,
MemorySpace =
    Kokkos::DefaultExecutionSpace::memory,
MemoryTraits = Kokkos::MemoryManaged
```

C++ standard library

```
template<
    class T,
    class Extents,
    class LayoutPolicy =
        std::layout_right,
    class AccessorPolicy =
        std::default_accessor<T>
> class mdspan;  // (since C++23)
```

Translate std to Kokkos and vice versa

- `std::mdspan` can be seen as an unmanaged `Kokkos::View`
  - `MemoryTraits = Kokkos::MemoryTraits<Kokkos::Unmanaged[|OtherTrait]...>;`

```
std::vector v{1, 2, 3, 4, 5, 6};
std::mdspan ms2(v.data(), 2, 3);
Kokkos::View<int**, Kokkos::HostSpace> v2(v.data(), 2, 3);
```

- `Kokkos::View` can be seen as a `std::mdspan` paired with a `std::shared_ptr`
  - `LayoutPolicy = std::layout_{right,left}_padded`
  - Turning reference counting off before launching kernels

```
template <class T, class E, class L, class A>
class MyView {
  std::shared_ptr<T[]> sh_;
  std::mdspan<T, L, E, A> ms_;
public:
  template <class... IndexTypes>
  MyView(std::string lbl, IndexTypes... exts);
};
```

Refactor `Kokkos::View` to use `Kokkos::mdspan` (C++14 backport of `std::mdspan`)

COMING SOON!

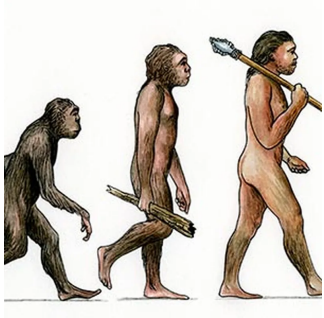
KokkosKernels

```
template<
 [class ExecutionHandle,]
  class InMat,
  class InVec,
  class OutVec
> void KokkosBLAS::gemv(
 [const ExecutionHandle& exec,]
  const char trans[],
  typename InMat::const_value_type& alpha,
  InMat A,
  Invec x,
  typename OutVec::const_value_type& beta,
  OutVec y);
```

C++ standard library

```
template<
 [class ExecutionPolicy,]
  InMatrix InMat,
  InVector InVec,
  OutVector OutVec
> void matrix_vector_product (
   [ExecutionPolicy&& exec,]
    InMat A,
    InVec x,
    OutVec y );  // (since C++26)
```

```
// BLAS
dgemv('N', M, N, 1., A, 1, x, 1, 0., y, 1);  // 11 parameters
// KokkosKernels
KokkosBLAS::gemv('N', 1., A, x, 0., y);
// Standard C++
std::matrix_vector_product(A, x, y);
```



- Change the underlying accessor
  - `std::scaled`
  - `std::transpose`
  - `std::conjugated`

```
// z = alpha * x + y
add(scaled(alpha, x), y, z);
```