Kokkos Kernels Math Library

Luc Berger-Vergiat, S. Rajamanickam, V. Dang, N. Ellingwood, J. Foucar, E. Harvey, B. Kelley, K. Liegeois, J. Loe, C. Pearson

ECP Annual Meeting

May 5th 2022

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.
SAND2020-9315 TR

Introducing Kokkos Kernels

The aims of Kokkos Kernels are to:

- deliver **portable** sparse/dense linear algebra and graph kernels,
- deliver **robust software ecosystem** for other software technology projects and applications,
- serve as **reference implementation** of key kernel needs of applications,
- partner with libraries, applications and vendors to identify new opportunities for performance.

Major partners and customers: Trilinos, PETSc, ExaWind, ORNL, ANL, QMCPACK, Nvidia, Intel, AMD

Online Resources

- https://github.com/kokkos/kokkos-kernels:
  - Kokkos Kernels GitHub repository,
  - https://github.com/kokkos/kokkos-kernels/wiki,
  - The wiki provides API calls, examples and build instructions.
- https://kokkosteam.slack.com:
  - Slack workspace for Kokkos, includes a kokkos-kernels channel,
  - Please join: fastest way to get your questions answered.

A focus on device BLAS and batched BLAS kernels

Learning objectives:

- Motivation for batched functions
- Two namespaces with BLAS and LAPACK functions
- Calling batched functions

Two namespaces with BLAS and LAPACK functions

KokkosBlas namespace

- **KokkosBlas:** device and functor level functions
  - *Intended Use Case:*
    - Caller uses optimal amount of parallelism to work on single input data
  - *Multiple Interfaces: Serial, Team, TeamVector, Device*
    - Device: all levels of nested parallelism are used on whole device
    - TeamVector: two-level nested parallelism is used with *TeamThreadRange and TeamVectorRange*
    - Team: one-level nested parallelism is used with *TeamThreadRange*
    - Serial: no nested parallelism is used internally

Two namespaces with BLAS and LAPACK functions

KokkosBatched namespace

- **KokkosBatched:** functor level functions
  - *Intended Use Case:*
    - Caller is within parallel kernel body with a batch of input data
  - *Multiple Interfaces: Serial, Team, TeamVector*
    - Serial: no nested parallelism is used internally
    - Team: one-level nested parallelism is used with *TeamThreadRange*
    - TeamVector: two-level nested parallelism is used with *TeamThreadRange and TeamVectorRange*

Parallel Batched BLAS/LAPACK Interface

Batched BLAS/LAPACK is **simple** i.e., BLAS/LAPACK in a parallel loop

```
auto A = Kokkos::View<double***>(''A'', N, Blk, Blk);
Kokkos::parallel_for( RangePolicy(N), /// users' parallel execution policy
 KOKKOS_LAMBDA(int &i) {
  auto AA = Kokkos::subview(A, i, ALL, ALL);
  KokkosBatched::SerialLU(AA);  /// functor-level interface
});
```

Kokkos batched BLAS/LAPACK is made up of following two components

- Kokkos parallel execution policy with `parallel_for`
- A functor-level interface to be used in `operator()`

Hierarchical functor interface is required to match Kokkos' hierarchical parallelism

Layered Hierarchical Functor-level Interface

Device Interface

- internally uses `TeamPolicy`
- is used for large input data that occupies an entire device
- can use an execution space instance to launch in a stream

Device with ExecutionSpace

```
    Kokkos::Cuda execution_space(myCudaStream);
    KokkosBlas(execution_space);
```

TeamVector Interface

- internally uses two nested `parallel_for` with `TeamThreadRange` and `ThreadVectorRange`
- requires the member (thread communicator) as an input argument

TeamVector with TeamPolicy

```
parallel_for(TeamPolicy,
 KOKKOS_LAMBDA(member_type &member){
    KokkosBatched::TeamVectorDoSomething(member);
});
```

Layered Hierarchical Functor-level Interface

Team Interface

- internally uses `TeamThreadRange` only
- in general is used with SIMD or Ensemble types where vector parallelism is expressed within the type
- can include `ThreadVectorRange`

Team without ThreadVectorRange

```
parallel_for(TeamPolicy,
  KOKKOS_LAMBDA(member_type &member){
   KokkosBatched::TeamDoThing(member);
});
```

Team with ThreadVectorRange outside

```
parallel_for(TeamPolicy,
 KOKKOS_LAMBDA(member_type &member){
    parallel_for(ThreadVectorRange) {
      KokkosBatched::TeamDoSomething(
            member);
}); });
```

Layered Hierarchical Functor-level Interface

Serial Interface

- can be used in a flat `parallel_for` i.e., `Kokkos::RangePolicy`
- can be used in the most inner loop of nested `parallel_for`'s

Serial with RangePolicy

```
parallel_for(RangePolicy,
 KOKKOS_LAMBDA(int &idx){
   KokkosBatched::SerialDoThing();
});
```

Serial in Hierarchical parallel loops

```
parallel_for(TeamPolicy,
 KOKKOS_LAMBDA(member_type &member){
   parallel_for(TeamThreadRange) {
      parallel_for(ThreadVectorRange) {
         KokkosBatched::
                 SerialDoSomething();
}); }); });
```

Summary: Batched BLAS/LAPACK

- User composable (batched) BLAS interface: parallel execution policy + functor-level interface
- Performance on GPUs is tunable:
  - Launching light-weight kernels multiple times can cause overhead
  - Fusing too many functor-level BLAS/LAPACK operations is difficult to do while maintaining optimal performance with a single team size