Kokkos User Group Meeting 2023, Dec 12–15, 2023, Albuquerque, NM

INTEGRATING PETSC WITH THE KOKKOS ECOSYSTEM

Junchao Zhang
(jczhang@anl.gov)

Mathematics and Computer Science Division
Argonne National Laboratory
Dec. 12, 2023

Outline

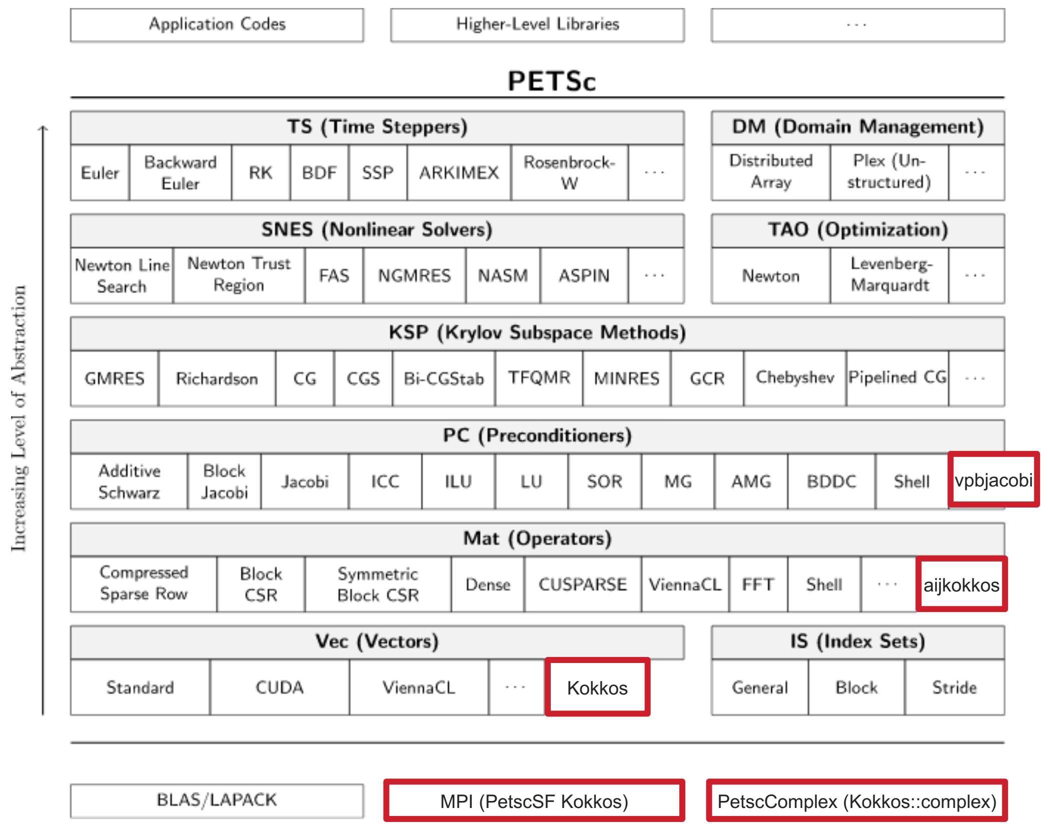
- What is PETSc?
- Integration status of PETSc and Kokkos & Kokkos-Kernels
  - Build PETSc with Kokkos
  - Useful Kokkos features w.r.t PETSc
  - PETSc APIs specially for Kokkos users
- Problems I met

What is PETSc?

- The *Portable, Extensible Toolkit for Scientific Computation* (https://petsc.org) is a popular math library for scalable solution of scientific applications modeled by partial differential equations (PDEs)
  - Linear/non-linear solvers, optimizers, etc
- Written in C but with object-oriented design
- Has C, Fortran, Python, Rust (WIP) bindings
- Runs on Linux, Mac and Windows
- Supports Kokkos and Kokkos-Kernels (4.2.0) and runs on Summit / Frontier / Aurora (WIP) with or without GPU-aware MPI (-use_gpu_aware_mpi <bool>)

Integrating PETSc with Kokkos

- Use Kokkos::complex for PetscComplex
- Use Kokkos in the PETSc communication module (PetscSF)
  - (simple) pack/unpack kernels
  - Allocate/free device send/recv buffers
  - Kokkos::atomic in unpack kernels
- Vector type (VecKokkos) and matrix type (MatAIJKokkos)
  - *Extensive* calls to KokkosBlas/Sparse
- Preconditioner VPBJACOBI
  - Call Kokkos *batched* BLAS to operate on a batch of small dense matrices



Building PETSc along with Kokkos

- Build PETSc with its bespoke BuildSystem, which spits cmake options to build Kokkos

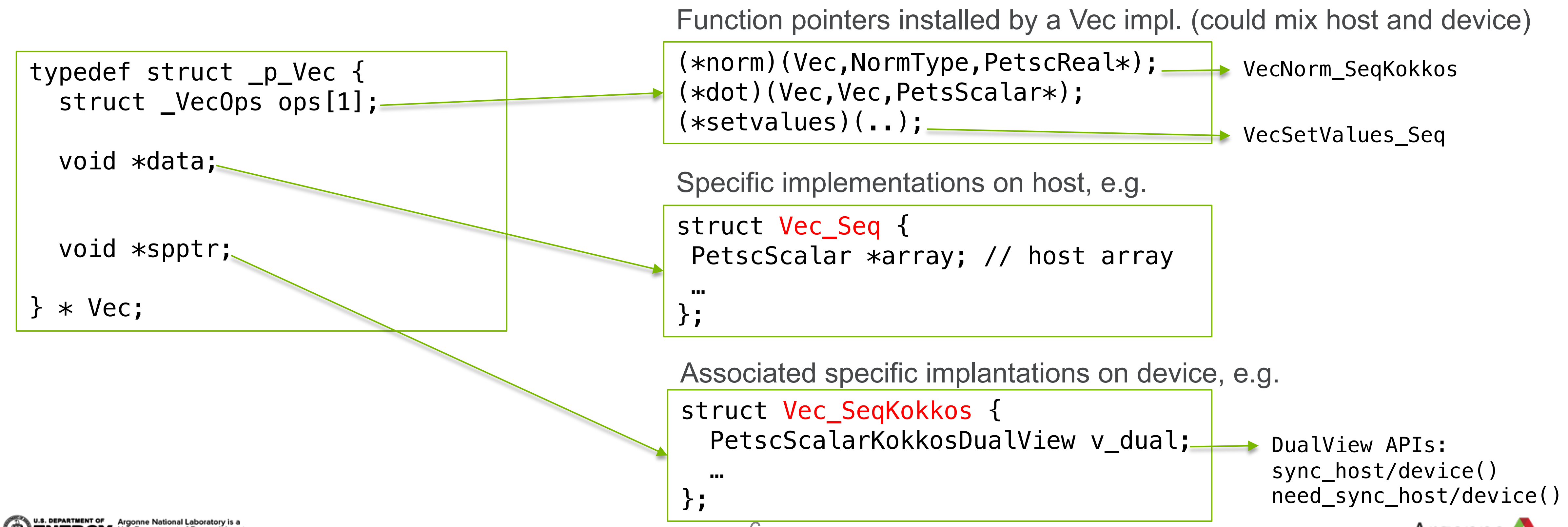
```
$ ./configure
--with-cc=gcc
--with-cxx=g++
--with-fc=gfortran
--download-mpich
--with-{cuda, hip, sycl}
--with-{cudac, hipc, syclc}={nvcc, hipcc, icpx}
--with-{cuda, hip, sycl}-arch={80, gfx908, pvc}
--download-kokkos // default 4.2.0
--download-kokkos-kernels
```

- Treat Kokkos as a language and set up a “Kokkos compiler” for **.kokkos.cxx* files in PETSc
  - No need to compile C files with C++ compilers
  - No need to compile regular PETSc C++ files with device compilers
- *Fancy C++ language features used in Kokkos could impair PETSc BuildSystem*

```
// gmakefile.test

ifneq ($(KOKKOS_USE_CUDA_COMPILER),)

KOKC = PATH=`dirname $(CUDAC)`:$(PATH) NVCC_WRAPPER_DEFAULT_COMPILER="$(CUDA_CXX)"
$(KOKKOS_BIN)/nvcc_wrapper --expt-extended-lambda

  KOKKOS_COMPILE = $(call quiet,KOKC) -c $(CUDAC_FLAGS) ${PETSC_CXXCPPFLAGS} $(CUDACPPFLAGS)
$(CUDA_CXXFLAGS) $(MPICXX_INCLUDES)

else ifneq ($(KOKKOS_USE_CUDACLANG_COMPILER),)

  KOKKOS_COMPILE = $(PETSC_COMPILE.cu)

else ifneq ($(KOKKOS_USE_HIP_COMPILER),)

  KOKKOS_COMPILE = $(PETSC_COMPILE.hip.cpp)

else ifneq ($(KOKKOS_USE_SYCL_COMPILER),)

  KOKKOS_COMPILE = $(PETSC_COMPILE.sycl.cxx)

else

  KOKKOS_COMPILE = $(PETSC_COMPILE.cxx)

endif

# Workaround for Kokkos PR 5473 introducing inline static variables

PETSC_COMPILE.kokkos.cxx = $(filter-out -fvisibility=hidden,$(subst -Xcompiler -fvisibility=hidden ,,$(strip
$(KOKKOS_COMPILE))))

KOKKOS_LINKER = $(filter-out -fvisibility=hidden,$(subst -Xcompiler -fvisibility=hidden ,,$(strip $(CLINKER))))
```

Using Kokkos DualView to dance with host and device

- Some PETSc Mat/Vec operations are not feasible on device
- We always have two copies of data on host and device;
- Operations just pull data and work on memory where they are designed with



```
typedef struct _p_Vec {
  struct _VecOps ops[1];

  void *data;


  void *spptr;

} * Vec;
```

Function pointers installed by a Vec impl. (could mix host and device)

```
(*norm)(Vec,NormType,PetscReal*);      VecNorm_SeqKokkos
(*dot)(Vec,Vec,PetsScalar*);
(*setvalues)(..);                      VecSetValues_Seq
```

Specific implementations on host, e.g.

```
struct Vec_Seq {
 PetscScalar *array; // host array
 …
};
```

Associated specific implantations on device, e.g.

```
struct Vec_SeqKokkos {
  PetscScalarKokkosDualView v_dual;    DualView APIs:
  …                                    sync_host/device()
};                                     need_sync_host/device()
```

PETSc APIs specially for Kokkos Users

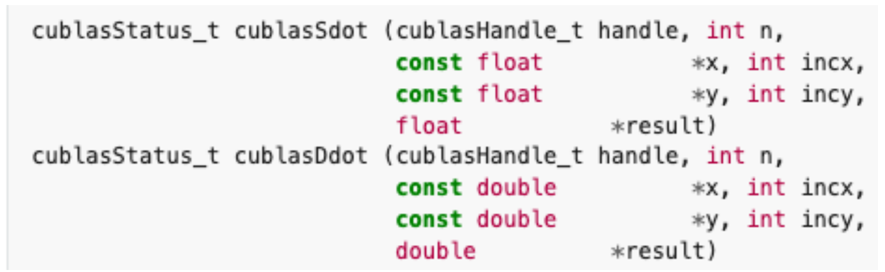
- Get Kokkos Veiws from PETSc vectors
  - `VecGetKokkosView(Vec x, Kokkos::View<PetscScalar*, MemorySpace> *kv)`
- Get Kokkos OffsetViews from PETSc MPI-parallel structured grids
  - `DMDAVecGetKokkosOffsetView(DM da, Vec x, Kokkos::Experimental::OffsetView<PetscScalar*,MemorySpace>* kv)`
    - Return a 1~4D view with the latest data in the specified memory space
    - The View uses global indices (to be consistent with PETSc host APIs)

Problem 1: not totally satisfied with KK performance

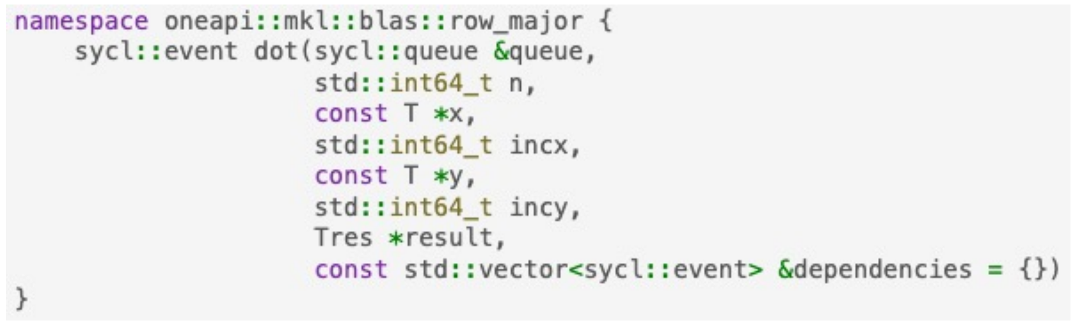
- PETSc GPU backends with code duplication
  - (before ECP) native CUDA backend
  - (ECP) Kokkos backend
  - (ECP) native HIP backend
    - Contributed by AMD after they found PETSc/Kokkos performance was bad for AMD to compare with Nvidia using PETSc/CUDA
    - Why? Some key KK kernels either
      - With incorrect native implementation (e.g., spgemm)
      - or, native implementation was not performant (e.g., dot)
      - or, did not have TPL interface to vendor libraries
      - or, the TPL interface was not designed with its use in solvers in mind (e.g., malloc/free temp buffers in spmv)

Problem 2: KK TPL boilerplate code is too much

- E.g., vector dot product has very simple APIs in cuBLAS and MKL

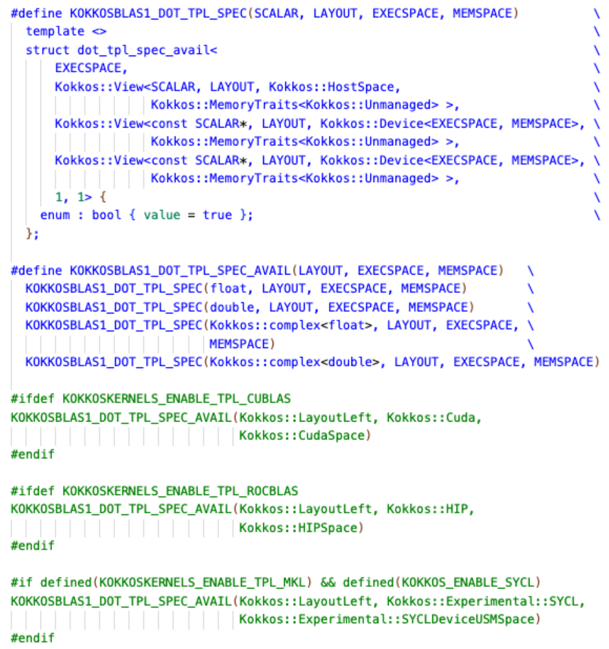


```
cublasStatus_t cublasSdot (cublasHandle_t handle, int n,
                           const float           *x, int incx,
                           const float           *y, int incy,
                           float           *result)
cublasStatus_t cublasDdot (cublasHandle_t handle, int n,
                           const double          *x, int incx,
                           const double          *y, int incy,
                           double          *result)
```

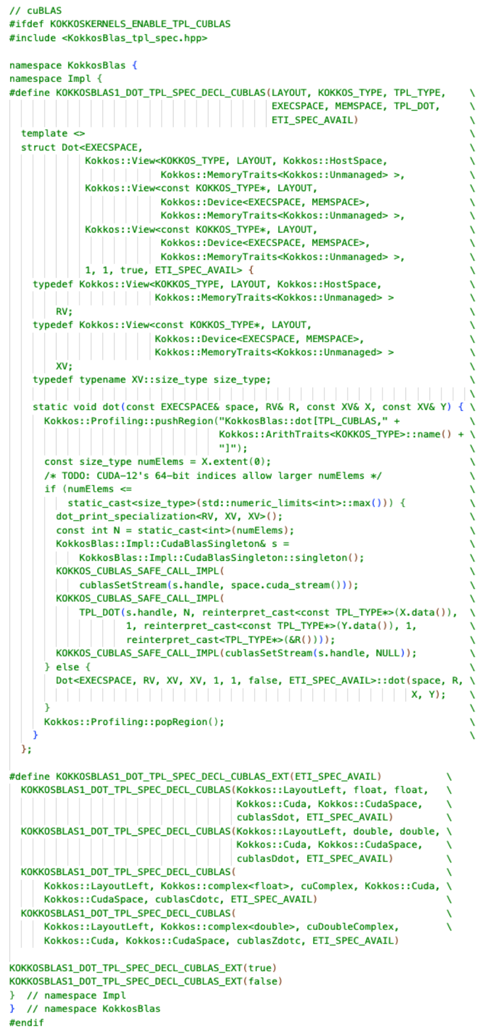


```
namespace oneapi::mkl::blas::row_major {
    sycl::event dot(sycl::queue &queue,
                    std::int64_t n,
                    const T *x,
                    std::int64_t incx,
                    const T *y,
                    std::int64_t incy,
                    Tres *result,
                    const std::vector<sycl::event> &dependencies = {})
}
```

KokkosBlas1_dot_tpl_spec_{avail, decl}.hpp



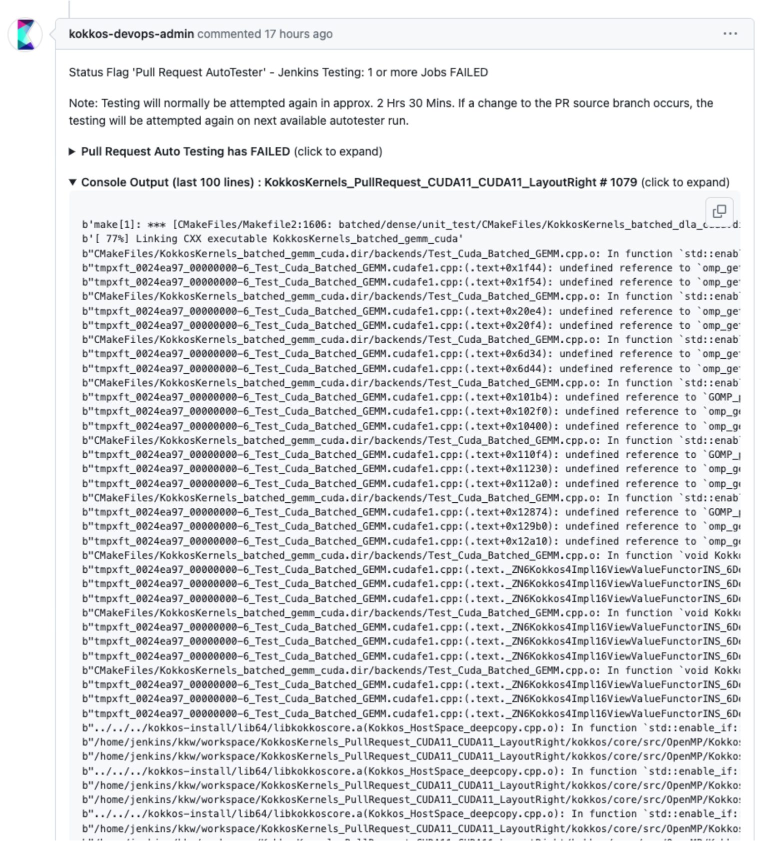
```
#define KOKKOSBLAS1_DOT_TPL_SPEC(SCALAR, LAYOUT, EXECSPACE, MEMSPACE)         \
  template <>                                                                 \
  struct dot_tpl_spec_avail<                                                  \
      EXECSPACE,                                                              \
      Kokkos::View<SCALAR, LAYOUT, Kokkos::HostSpace,                         \
                   Kokkos::MemoryTraits<Kokkos::Unmanaged> >,                 \
      Kokkos::View<const SCALAR*, LAYOUT, Kokkos::Device<EXECSPACE, MEMSPACE>, \
                   Kokkos::MemoryTraits<Kokkos::Unmanaged> >,                 \
      Kokkos::View<const SCALAR*, LAYOUT, Kokkos::Device<EXECSPACE, MEMSPACE>, \
                   Kokkos::MemoryTraits<Kokkos::Unmanaged> >,                 \
      1, 1> {                                                                 \
    enum : bool { value = true };                                             \
  };

#define KOKKOSBLAS1_DOT_TPL_SPEC_AVAIL(LAYOUT, EXECSPACE, MEMSPACE)   \
  KOKKOSBLAS1_DOT_TPL_SPEC(float, LAYOUT, EXECSPACE, MEMSPACE)        \
  KOKKOSBLAS1_DOT_TPL_SPEC(double, LAYOUT, EXECSPACE, MEMSPACE)       \
  KOKKOSBLAS1_DOT_TPL_SPEC(Kokkos::complex<float>, LAYOUT, EXECSPACE, \
                           MEMSPACE)                                  \
  KOKKOSBLAS1_DOT_TPL_SPEC(Kokkos::complex<double>, LAYOUT, EXECSPACE, MEMSPACE)

#ifdef KOKKOSKERNELS_ENABLE_TPL_CUBLAS
KOKKOSBLAS1_DOT_TPL_SPEC_AVAIL(Kokkos::LayoutLeft, Kokkos::Cuda,
                               Kokkos::CudaSpace)
#endif

#ifdef KOKKOSKERNELS_ENABLE_TPL_ROCBLAS
KOKKOSBLAS1_DOT_TPL_SPEC_AVAIL(Kokkos::LayoutLeft, Kokkos::HIP,
                               Kokkos::HIPSpace)
#endif

#if defined(KOKKOSKERNELS_ENABLE_TPL_MKL) && defined(KOKKOS_ENABLE_SYCL)
KOKKOSBLAS1_DOT_TPL_SPEC_AVAIL(Kokkos::LayoutLeft, Kokkos::Experimental::SYCL,
                               Kokkos::Experimental::SYCLDeviceUSMSpace)
#endif
```



```
// cuBLAS
#ifdef KOKKOSKERNELS_ENABLE_TPL_CUBLAS
#include <KokkosBlas_tpl_spec.hpp>

namespace KokkosBlas {
namespace Impl {
#define KOKKOSBLAS1_DOT_TPL_SPEC_DECL_CUBLAS(LAYOUT, KOKKOS_TYPE, TPL_TYPE,    \
                                             EXECSPACE, MEMSPACE, TPL_DOT,     \
                                             ETI_SPEC_AVAIL)                   \
  template <>                                                                  \
  struct Dot<EXECSPACE,                                                        \
             Kokkos::View<KOKKOS_TYPE, LAYOUT, Kokkos::HostSpace,              \
                          Kokkos::MemoryTraits<Kokkos::Unmanaged> >,           \
             Kokkos::View<const KOKKOS_TYPE*, LAYOUT,                          \
                          Kokkos::Device<EXECSPACE, MEMSPACE>,                 \
                          Kokkos::MemoryTraits<Kokkos::Unmanaged> >,           \
             Kokkos::View<const KOKKOS_TYPE*, LAYOUT,                          \
                          Kokkos::Device<EXECSPACE, MEMSPACE>,                 \
                          Kokkos::MemoryTraits<Kokkos::Unmanaged> >,           \
             1, 1, true, ETI_SPEC_AVAIL> {                                     \
    typedef Kokkos::View<KOKKOS_TYPE, LAYOUT, Kokkos::HostSpace,               \
                         Kokkos::MemoryTraits<Kokkos::Unmanaged> >             \
        RV;                                                                    \
    typedef Kokkos::View<const KOKKOS_TYPE*, LAYOUT,                           \
                         Kokkos::Device<EXECSPACE, MEMSPACE>,                  \
                         Kokkos::MemoryTraits<Kokkos::Unmanaged> >             \
        XV;                                                                    \
    typedef typename XV::size_type size_type;                                  \
                                                                               \
    static void dot(const EXECSPACE& space, RV& R, const XV& X, const XV& Y) { \
      Kokkos::Profiling::pushRegion("KokkosBlas::dot[TPL_CUBLAS," +            \
                                    Kokkos::ArithTraits<KOKKOS_TYPE>::name() + \
                                    "]");                                      \
      const size_type numElems = X.extent(0);                                  \
      /* TODO: CUDA-12's 64-bit indices allow larger numElems */               \
      if (numElems <=                                                          \
          static_cast<size_type>(std::numeric_limits<int>::max())) {           \
        dot_print_specialization<RV, XV, XV>();                                \
        const int N = static_cast<int>(numElems);                              \
        KokkosBlas::Impl::CudaBlasSingleton& s =                               \
            KokkosBlas::Impl::CudaBlasSingleton::singleton();                  \
        KOKKOS_CUBLAS_SAFE_CALL_IMPL(                                          \
            cublasSetStream(s.handle, space.cuda_stream()));                   \
        KOKKOS_CUBLAS_SAFE_CALL_IMPL(                                          \
            TPL_DOT(s.handle, N, reinterpret_cast<const TPL_TYPE*>(X.data()),  \
                    1, reinterpret_cast<const TPL_TYPE*>(Y.data()), 1,         \
                    reinterpret_cast<TPL_TYPE*>(&R())));                       \
        KOKKOS_CUBLAS_SAFE_CALL_IMPL(cublasSetStream(s.handle, NULL));         \
      } else {                                                                 \
        Dot<EXECSPACE, RV, XV, XV, 1, 1, false, ETI_SPEC_AVAIL>::dot(space, R, \
                                                                     X, Y);    \
      }                                                                        \
      Kokkos::Profiling::popRegion();                                          \
    }                                                                          \
  };

#define KOKKOSBLAS1_DOT_TPL_SPEC_DECL_CUBLAS_EXT(ETI_SPEC_AVAIL)          \
  KOKKOSBLAS1_DOT_TPL_SPEC_DECL_CUBLAS(Kokkos::LayoutLeft, float, float,   \
                                       Kokkos::Cuda, Kokkos::CudaSpace,    \
                                       cublasSdot, ETI_SPEC_AVAIL)         \
  KOKKOSBLAS1_DOT_TPL_SPEC_DECL_CUBLAS(Kokkos::LayoutLeft, double, double, \
                                       Kokkos::Cuda, Kokkos::CudaSpace,    \
                                       cublasDdot, ETI_SPEC_AVAIL)         \
  KOKKOSBLAS1_DOT_TPL_SPEC_DECL_CUBLAS(                                    \
      Kokkos::LayoutLeft, Kokkos::complex<float>, cuComplex, Kokkos::Cuda, \
      Kokkos::CudaSpace, cublasCdotc, ETI_SPEC_AVAIL)                      \
  KOKKOSBLAS1_DOT_TPL_SPEC_DECL_CUBLAS(                                    \
      Kokkos::LayoutLeft, Kokkos::complex<double>, cuDoubleComplex,        \
      Kokkos::Cuda, Kokkos::CudaSpace, cublasZdotc, ETI_SPEC_AVAIL)

KOKKOSBLAS1_DOT_TPL_SPEC_DECL_CUBLAS_EXT(true)
KOKKOSBLAS1_DOT_TPL_SPEC_DECL_CUBLAS_EXT(false)
}  // namespace Impl
}  // namespace KokkosBlas
#endif
```

Problem 3: KK format and CI is not contributor friendly

- 80-character column width is not for 4K monitors
- Clang-format 8.0 is ancient
  - Need to ask admins to specially install it
- Need to ask KK developers to test PRs
  - Back and forth communication hurts productivity
- Github CI output is difficult to navigate
  - Need to horizontally scroll in a tall & skinny window



kokkos-devops-admin commented 17 hours ago

Status Flag 'Pull Request AutoTester' - Jenkins Testing: 1 or more Jobs FAILED

Note: Testing will normally be attempted again in approx. 2 Hrs 30 Mins. If a change to the PR source branch occurs, the testing will be attempted again on next available autotester run.

▶ **Pull Request Auto Testing has FAILED** (click to expand)

▼ **Console Output (last 100 lines) : KokkosKernels_PullRequest_CUDA11_CUDA11_LayoutRight # 1079** (click to expand)

```
b'make[1]: *** [CMakeFiles/Makefile2:1606: batched/dense/unit_test/CMakeFiles/KokkosKernels_batched_dla_
b'[ 77%] Linking CXX executable KokkosKernels_batched_gemm_cuda'
b"CMakeFiles/KokkosKernels_batched_gemm_cuda.dir/backends/Test_Cuda_Batched_GEMM.cpp.o: In function `std::enab
b"tmpxft_0024ea97_00000000-6_Test_Cuda_Batched_GEMM.cudafe1.cpp:(.text+0x1f44): undefined reference to `omp_ge
b"tmpxft_0024ea97_00000000-6_Test_Cuda_Batched_GEMM.cudafe1.cpp:(.text+0x1f54): undefined reference to `omp_ge
b"CMakeFiles/KokkosKernels_batched_gemm_cuda.dir/backends/Test_Cuda_Batched_GEMM.cpp.o: In function `std::enab
b"tmpxft_0024ea97_00000000-6_Test_Cuda_Batched_GEMM.cudafe1.cpp:(.text+0x20e4): undefined reference to `omp_ge
b"tmpxft_0024ea97_00000000-6_Test_Cuda_Batched_GEMM.cudafe1.cpp:(.text+0x20f4): undefined reference to `omp_ge
b"CMakeFiles/KokkosKernels_batched_gemm_cuda.dir/backends/Test_Cuda_Batched_GEMM.cpp.o: In function `std::enab
b"tmpxft_0024ea97_00000000-6_Test_Cuda_Batched_GEMM.cudafe1.cpp:(.text+0x6d34): undefined reference to `omp_ge
b"tmpxft_0024ea97_00000000-6_Test_Cuda_Batched_GEMM.cudafe1.cpp:(.text+0x6d44): undefined reference to `omp_ge
b"CMakeFiles/KokkosKernels_batched_gemm_cuda.dir/backends/Test_Cuda_Batched_GEMM.cpp.o: In function `std::enab
b"tmpxft_0024ea97_00000000-6_Test_Cuda_Batched_GEMM.cudafe1.cpp:(.text+0x101b4): undefined reference to `GOMP_
b"tmpxft_0024ea97_00000000-6_Test_Cuda_Batched_GEMM.cudafe1.cpp:(.text+0x102f0): undefined reference to `omp_g
b"tmpxft_0024ea97_00000000-6_Test_Cuda_Batched_GEMM.cudafe1.cpp:(.text+0x10400): undefined reference to `omp_g
b"CMakeFiles/KokkosKernels_batched_gemm_cuda.dir/backends/Test_Cuda_Batched_GEMM.cpp.o: In function `std::enab
b"tmpxft_0024ea97_00000000-6_Test_Cuda_Batched_GEMM.cudafe1.cpp:(.text+0x110f4): undefined reference to `GOMP_
b"tmpxft_0024ea97_00000000-6_Test_Cuda_Batched_GEMM.cudafe1.cpp:(.text+0x11230): undefined reference to `omp_g
b"tmpxft_0024ea97_00000000-6_Test_Cuda_Batched_GEMM.cudafe1.cpp:(.text+0x112a0): undefined reference to `omp_g
b"CMakeFiles/KokkosKernels_batched_gemm_cuda.dir/backends/Test_Cuda_Batched_GEMM.cpp.o: In function `std::enab
b"tmpxft_0024ea97_00000000-6_Test_Cuda_Batched_GEMM.cudafe1.cpp:(.text+0x12874): undefined reference to `GOMP_
b"tmpxft_0024ea97_00000000-6_Test_Cuda_Batched_GEMM.cudafe1.cpp:(.text+0x129b0): undefined reference to `omp_g
b"tmpxft_0024ea97_00000000-6_Test_Cuda_Batched_GEMM.cudafe1.cpp:(.text+0x12a10): undefined reference to `omp_g
b"CMakeFiles/KokkosKernels_batched_gemm_cuda.dir/backends/Test_Cuda_Batched_GEMM.cpp.o: In function `void Kokk
b"tmpxft_0024ea97_00000000-6_Test_Cuda_Batched_GEMM.cudafe1.cpp:(.text._ZN6Kokkos4Impl16ViewValueFunctorINS_6D
b"tmpxft_0024ea97_00000000-6_Test_Cuda_Batched_GEMM.cudafe1.cpp:(.text._ZN6Kokkos4Impl16ViewValueFunctorINS_6D
b"tmpxft_0024ea97_00000000-6_Test_Cuda_Batched_GEMM.cudafe1.cpp:(.text._ZN6Kokkos4Impl16ViewValueFunctorINS_6D
b"CMakeFiles/KokkosKernels_batched_gemm_cuda.dir/backends/Test_Cuda_Batched_GEMM.cpp.o: In function `void Kokk
b"tmpxft_0024ea97_00000000-6_Test_Cuda_Batched_GEMM.cudafe1.cpp:(.text._ZN6Kokkos4Impl16ViewValueFunctorINS_6D
b"tmpxft_0024ea97_00000000-6_Test_Cuda_Batched_GEMM.cudafe1.cpp:(.text._ZN6Kokkos4Impl16ViewValueFunctorINS_6D
b"tmpxft_0024ea97_00000000-6_Test_Cuda_Batched_GEMM.cudafe1.cpp:(.text._ZN6Kokkos4Impl16ViewValueFunctorINS_6D
b"CMakeFiles/KokkosKernels_batched_gemm_cuda.dir/backends/Test_Cuda_Batched_GEMM.cpp.o: In function `void Kokk
b"tmpxft_0024ea97_00000000-6_Test_Cuda_Batched_GEMM.cudafe1.cpp:(.text._ZN6Kokkos4Impl16ViewValueFunctorINS_6D
b"tmpxft_0024ea97_00000000-6_Test_Cuda_Batched_GEMM.cudafe1.cpp:(.text._ZN6Kokkos4Impl16ViewValueFunctorINS_6D
b"tmpxft_0024ea97_00000000-6_Test_Cuda_Batched_GEMM.cudafe1.cpp:(.text._ZN6Kokkos4Impl16ViewValueFunctorINS_6D
b"../../../kokkos-install/lib64/libkokkoscore.a(Kokkos_HostSpace_deepcopy.cpp.o): In function `std::enable_if:
b"/home/jenkins/kkw/workspace/KokkosKernels_PullRequest_CUDA11_CUDA11_LayoutRight/kokkos/core/src/OpenMP/Kokko
b"/home/jenkins/kkw/workspace/KokkosKernels_PullRequest_CUDA11_CUDA11_LayoutRight/kokkos/core/src/OpenMP/Kokko
b"../../../kokkos-install/lib64/libkokkoscore.a(Kokkos_HostSpace_deepcopy.cpp.o): In function `std::enable_if:
b"/home/jenkins/kkw/workspace/KokkosKernels_PullRequest_CUDA11_CUDA11_LayoutRight/kokkos/core/src/OpenMP/Kokko
b"/home/jenkins/kkw/workspace/KokkosKernels_PullRequest_CUDA11_CUDA11_LayoutRight/kokkos/core/src/OpenMP/Kokko
b"../../../kokkos-install/lib64/libkokkoscore.a(Kokkos_HostSpace_deepcopy.cpp.o): In function `std::enable_if:
b"/home/jenkins/kkw/workspace/KokkosKernels_PullRequest_CUDA11_CUDA11_LayoutRight/kokkos/core/src/OpenMP/Kokko
```

Conclusion and Thanks to Kokkos developers

- PETSc already has deep integration with Kokkos
- Kokkos & Kokkos-Kernels are key to make PETSc device portability
- Let’s work together and make the Kokkos ecosystem greater!