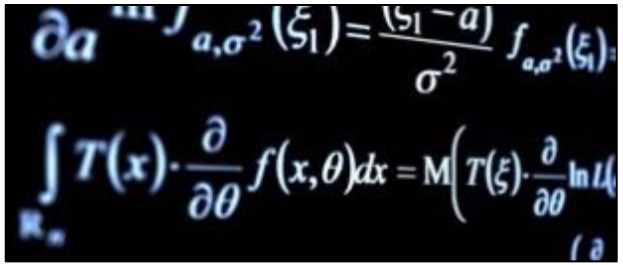
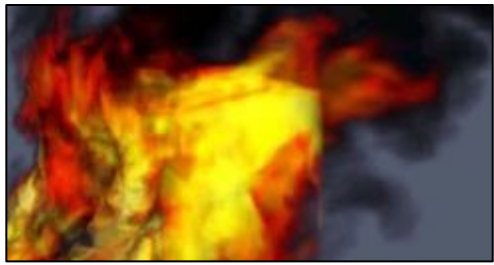
Exceptional service in the national interest

Sandia National Laboratories



$$\int T(x)\cdot\frac{\partial}{\partial\theta}f(x,\theta)dx = \mathrm{M}\left(T(\xi)\cdot\frac{\partial}{\partial\theta}\ln L\right.$$



Trilinos Support on AMD and Intel GPUs: SAKE Project

Unclassified Unlimited Release

Brian Kelley, Luc Berger-Vergiat and Ichitaro Yamazaki
Center for Computing Research
Sandia National Laboratories/NM

U.S. DEPARTMENT OF ENERGY

NNSA National Nuclear Security Administration

Sandia National Laboratories is a multimission 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-NA-0003525.

Introduction

Sandia National Laboratories

- SAKE Solvers Project
  - Kokkos Kernels + Tpetra solver stack in Trilinos
  - Compatibility and performance on accelerator architectures other than NVIDIA's Cuda
- Upcoming DOE Exascale machines
- Kokkos backends for each architecture
- HIP Status, and how to use it in Trilinos
- OpenMPTarget Status
- SYCL Status
- Lessons learned for being portable

Upcoming DOE Exascale Machines

Sandia National Laboratories

- Frontier (ORNL)
  - AMD MI250x GPUs
  - 2022
- Aurora (Argonne)
  - Intel Xe Ponte Vecchio GPUs
  - 2022
- El Capitan (LLNL)
  - AMD GPUs
  - 2023
- No NVIDIA/Cuda: Trilinos (especially Kokkos/Tpetra stack) must be ready to support completely new architectures.
- Relying heavily on natural portability of Kokkos, and work of the Kokkos Core team to get fundamental operations (parallel_for, parallel_reduce, etc.) working on these architectures

Kokkos Backends for GPUs

Sandia National Laboratories

	NVIDIA	AMD	Intel
Cuda	✓		
HIP		✓	
SYCL			✓
OpenMPTarget	✓	✓	✓

- Preferred backends (best vendor support and performance):
  - NVIDIA – Cuda
  - AMD – HIP
  - Intel – SYCL
- Kokkos+OpenMPTarget still useful for applications that already rely on OpenMPTarget for GPU offloading

Available Vendor Libraries

Sandia National Laboratories

	Dense BLAS	Sparse Linear Algebra
Cuda	cuBLAS	cuSPARSE
HIP	rocBLAS	rocSPARSE
SYCL (on Intel)	oneAPI MKL	

- Trilinos has mature support for cuBLAS/cuSPARSE (enabled by default)
- CMake infrastructure for rocBLAS/rocSPARSE was recently added to Kokkos Kernels
  - SpMV (sparse matrix-vector) and GEMV (dense matrix-vector) wrappers are almost ready, other wrappers will come soon
- oneAPI MKL not supported by KokkosKernels yet

Enabling HIP Backend in Trilinos

Sandia National Laboratories

- Can follow these steps today on Caraway (Sandia testbed), Spock (ORNL testbed), or any machine with AMD GPUs in the MI25…MI100 range
- Set up environment
  - Load ROCm or Cray compiler module
  - If using ROCm directly, set underlying MPI compiler
    - `export OMPI_CXX=hipcc` (OpenMPI)
    - `export MPICH_CXX=hipcc` (MPICH, MVAPICH)
  - No analogue of nvcc_wrapper needed.
- Configuration
  - Architecture flag for Kokkos. For MI100: `Kokkos_ARCH_VEGA908=ON`
  - `Kokkos_ENABLE_HIP=ON`
  - `Tpetra_INST_HIP=ON`
- That's it!

Developing for AMD/HIP with Kokkos

Sandia National Laboratories

- Mostly identical to Cuda:
  - `Kokkos::Cuda -> Kokkos::Experimental::HIP`
  - `Kokkos::CudaSpace -> Kokkos::Experimental::HIPSpace`
  - `KokkosCudaWrapperNode -> KokkosHIPWrapperNode`
  - Annotate device functions with `KOKKOS_LAMBDA`, `KOKKOS_INLINE_FUNCTION`, `etc.`
- Defaults will be set to HIP automatically when you enable it:
  - `Kokkos::DefaultExecutionSpace == Kokkos::Experimental::HIP`
  - `Tpetra::Map<>::node_type == KokkosHIPWrapperNode`
- Important difference: no UVM equivalent currently exists
  - Use Kokkos::DualView and the new Tpetra interfaces that Karen just presented.
  - HIPHostPinnedSpace is accessible from both host and device, but every access from GPU goes over PCIe. Cuda UVM keeps pages where they were most recently used.

What works with HIP today

Sandia National Laboratories

- All the CMake/build related infrastructure
- Kokkos, Kokkos Kernels (except for two Kernels unit tests)
- Tpetra, Amesos2, Belos, Ifpack2 (49/51 tests), Zoltan2, MueLu (100/102 tests)

Future work with HIP

Sandia National Laboratories

- Performance tuning (Kokkos Kernels is still sprinkled with hardcoded heuristics that were chosen to perform well on NVIDIA V100)
- Wrappers for rocBLAS and rocSPARSE kernels
  - SpMV and GEMV are almost ready
- Performance preview, 10,000 x 10,000 GEMV (matrix*vector) in FP64:
  - NVIDIA V100 + cuBLAS: 218 GFLOP/s
  - AMD MI100 + Kokkos Kernels: 174 GFLOP/s
    - Kokkos Kernels portable GEMV implementation
  - AMD MI100 + rocBLAS: 248 GFLOP/s
    - rocBLAS implementation, called through Kokkos Kernels wrapper

OpenMP Target: Current Support

Sandia National Laboratories

- Requirements
  - Testing with clang 13 + cuda 10.2 mostly
  - Requires C++17 and libopenmp
- Restrictions
  - No vendor TPL support at planned so far (cuBLAS, cuSPARSE, rocBLAS, rocSPARSE, MKL)
  - Might not deliver as much performance as underlying backend (prefer using CUDA, ROCm/HIP, or oneAPI/SYCL directly)

OpenMP Target: backend status

Sandia National Laboratories

- Kokkos status
  - building library with tests
  - Some tests disabled: grep -rin "FIXME_OPENMPTARGET" core/unit_test/
- Kokkos Kernels status
  - library builds without tests
  - Unit-tests: nvlink linking errors related to Kokkos:ALL()
  - Also experimenting with icpx/dpcpp builds on Intel hardware at Argonne JLSE

OpenMP Target: upcoming work

Sandia National Laboratories

- Kokkos/Kokkos Kernels
  - Fix nvlink issues with Kokkos::ALL()
    → should allow Kokkos Kernels tests to build
  - Tackle FIXME_OPENMPTARGET to enable more regression tests
  - Setup testing for Kokkos Kernels (Weaver clang 13)
- Trilinos solvers
  - Add infrastructure support in Tpetra (Tpetra_INST_OPENMPTARGET, etc…)
  - Enable new backend in solver stack (Belos, Ifpack2, Amesos2, …)
- Support in spack packages for Kokkos, Kokkos Kernels and Trilinos
  - Allow ECP to deploy on early access systems
  - Makes life easier for users

Enabling SYCL Backend in Trilinos

Sandia National Laboratories

- Load Intel oneAPI toolchain module
- Use dpcpp (data parallel C++ compiler) as MPI's underlying compiler
- Set `Kokkos_ENABLE_SYCL=ON` and `Tpetra_INST_SYCL=ON`
- Set architecture flag, e.g. `Kokkos_ARCH_INTEL_GEN9=ON` for Gen9, or `Kokkos_ARCH_INTEL_XEHP=ON` for Xe GPUs.

What works with SYCL today

Sandia National Laboratories

- CMake/build related infrastructure in Kokkos Kernels and Tpetra
- Roughly half of Kokkos Kernels tests pass: all BLAS, SpMV for single vectors, all graph kernels except distance-2 coloring, SpADD (sparse matrix addition)

Lessons Learned

Sandia National Laboratories

- Use portable Kokkos facilities instead of hardcoding values:
  - Launch kernel with `Kokkos::AUTO` for team size
  - Or if you need a specific value before launching: `teamSize = TeamPolicy<…>::team_size_recommended(functor, ParallelForTag())`
  - Use `TeamPolicy<…>::vector_length_max()` as an upper bound for vector length, if you are using 3 levels of parallelism
- Avoid duplicating nontrivial source code for different backends. The bulk of Tpetra-stack code should be in generic templated classes.

Getting Support

Sandia National Laboratories

- SAKE Team
  - Brian Kelley (bmkelle@sandia.gov)
  - Luc Berger-Vergiat (lberge@sandia.gov)
  - Ichi Yamazaki (iyamaza@sandia.gov)
- Join the Kokkos slack channel: kokkosteam.slack.com
  - Not just for Kokkos core – also for Kokkos Kernels and Trilinos/application questions involving Kokkos