Module 5: Kokkos Kernels Math Library

August 27, 2020

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.
SAND2020-7475 PE

Online Resources:

- https://github.com/kokkos:
  - Primary Kokkos GitHub Organization
- https://github.com/kokkos/kokkos-tutorials/wiki/Kokkos-Lecture-Series:
  - Slides, recording and Q&A for the Lectures
- https://github.com/kokkos/kokkos/wiki:
  - Wiki including API reference
- https://kokkosteam.slack.com:
  - Slack channel for Kokkos.
  - Please join: fastest way to get your questions answered.
  - Can whitelist domains, or invite individual people.

- 07/17 Module 1: Introduction, Building and Parallel Dispatch
- 07/24 Module 2: Views and Spaces
- 07/31 Module 3: Data Structures + MultiDimensional Loops
- 08/07 Module 4: Hierarchical Parallelism
- 08/14 Module 5: Tasking, Streams and SIMD
- 08/21 Module 6: Internode: MPI and PGAS
- 08/28 Module 7: Tools: Profiling, Tuning and Debugging
- **09/04 Module 8: Kernels: Sparse and Dense Linear Algebra**
- 09/11 Reserve Day

Tools Stuff

Dense Linear Algebra (BLAS)

- BLAH.
- More BLAH.

Sparse Linear Algebra

- Sparse BLAH.
- Sparse BLAH2.

Graph Functions

- Graph BLAH.

Dense Linear Algebra (BLAS)

Kokkos Kernels dense linear algebra capabilities.

Learning objectives:

- Calling BLAS functions with Views
- Calling BLAS functions with Views
- Calling BLAS functions with Views
- Calling BLAS functions with Views

HPC world owns many Fortran LOC!

- We generally cannot port it all at once.
- We need an incremental porting strategy
  - Keep our e.g. Fortran mains, drivers, physics packages
  - But port relevant infrastructure, or hotspot kernels to C++ and Kokkos

HPC world owns many Fortran LOC!

- We generally cannot port it all at once.
- We need an incremental porting strategy
  - Keep our e.g. Fortran mains, drivers, physics packages
  - But port relevant infrastructure, or hotspot kernels to C++ and Kokkos

How do we make Kokkos and Fortran talk with each other?

Fortran Language Compatibility Layer (FLCL)

- Pass multidimensional arrays accross the C++/Fortran boundary
  - See Fortran arrays as Kokkos Views and vice versa
- Create Kokkos View and DualView from Fortran
  - Allows Fortran to be the memory owner but call C++ functions with Kokkos kernels for CUDA/HIP
- Initialize and Finalize Kokkos from Fortran
- FortranIndex<T> scalar type to deal with 1 vs 0 based indexing in sparse data structures

FLCL

The Fortran Language Compatibility Layer allows an incremental porting of a Fortran code to Kokkos.

nd_array_t

The compatibility glue between Fortran arrays and Kokkos Views.

Keeps Track of:

- An array's rank
- Extents of the array
- Strides of the array
- Pointer to the allocation

nd_array_t

The compatibility glue between Fortran arrays and Kokkos Views.

Keeps Track of:

- An array's rank
- Extents of the array
- Strides of the array
- Pointer to the allocation

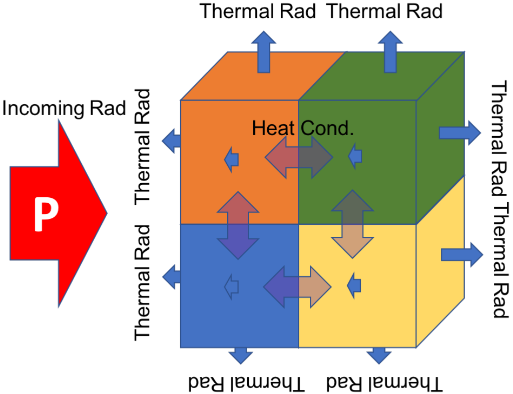
How do we create an `nd_array_t`?

- Explicit routines like `to_nd_array_i64_6`
- Simple interface taking a fortran array as argument

```
array = to_nd_array(foo);
```

3D Heat Conduction

- Heat conduction inside the body
- Thermal radiation (Black Body) on surface
- Incoming power flow from one direction



Sparse Linear Algebra

Sparse linear algebra data structures and functions.

Learning objectives:

- Calling BLAS functions with Views
- Calling BLAS functions with Views
- Calling BLAS functions with Views
- Calling BLAS functions with Views

HPC world owns many Fortran LOC!

- We generally cannot port it all at once.
- We need an incremental porting strategy
    - Keep our e.g. Fortran mains, drivers, physics packages
    - But port relevant infrastructure, or hotspot kernels to C++ and Kokkos

HPC world owns many Fortran LOC!

- We generally cannot port it all at once.
- We need an incremental porting strategy
  - Keep our e.g. Fortran mains, drivers, physics packages
  - But port relevant infrastructure, or hotspot kernels to C++ and Kokkos

How do we make Kokkos and Fortran talk with each other?

Fortran Language Compatibility Layer (FLCL)

- Pass multidimensional arrays accross the C++/Fortran boundary
    - See Fortran arrays as Kokkos Views and vice versa
- Create Kokkos View and DualView from Fortran
    - Allows Fortran to be the memory owner but call C++ functions with Kokkos kernels for CUDA/HIP
- Initialize and Finalize Kokkos from Fortran
- FortranIndex<T> scalar type to deal with 1 vs 0 based indexing in sparse data structures

FLCL

The Fortran Language Compatibility Layer allows an incremental porting of a Fortran code to Kokkos.

nd_array_t

The compatibility glue between Fortran arrays and Kokkos Views.

Keeps Track of:

- An array's rank
- Extents of the array
- Strides of the array
- Pointer to the allocation

nd_array_t

The compatibility glue between Fortran arrays and Kokkos Views.

Keeps Track of:

- An array's rank
- Extents of the array
- Strides of the array
- Pointer to the allocation

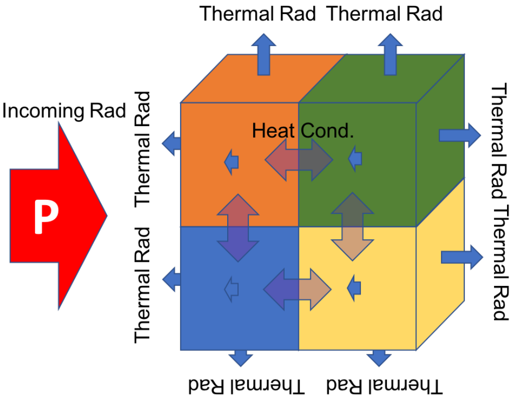
How do we create an `nd_array_t`?

- Explicit routines like `to_nd_array_i64_6`
- Simple interface taking a fortran array as argument

```
array = to_nd_array(foo);
```

3D Heat Conduction

- Heat conduction inside the body
- Thermal radiation (Black Body) on surface
- Incoming power flow from one direction



Sparse Solvers

Sparse linear algebra data structures and functions.

Learning objectives:

- Calling BLAS functions with Views
- Calling BLAS functions with Views
- Calling BLAS functions with Views
- Calling BLAS functions with Views

HPC world owns many Fortran LOC!

- We generally cannot port it all at once.
- We need an incremental porting strategy
  - Keep our e.g. Fortran mains, drivers, physics packages
  - But port relevant infrastructure, or hotspot kernels to C++ and Kokkos

HPC world owns many Fortran LOC!

- We generally cannot port it all at once.
- We need an incremental porting strategy
  - Keep our e.g. Fortran mains, drivers, physics packages
  - But port relevant infrastructure, or hotspot kernels to C++ and Kokkos

How do we make Kokkos and Fortran talk with each other?

Fortran Language Compatibility Layer (FLCL)

- Pass multidimensional arrays accross the C++/Fortran boundary
  - See Fortran arrays as Kokkos Views and vice versa
- Create Kokkos View and DualView from Fortran
  - Allows Fortran to be the memory owner but call C++ functions with Kokkos kernels for CUDA/HIP
- Initialize and Finalize Kokkos from Fortran
- FortranIndex<T> scalar type to deal with 1 vs 0 based indexing in sparse data structures

FLCL

The Fortran Language Compatibility Layer allows an incremental porting of a Fortran code to Kokkos.

nd_array_t

The compatibility glue between Fortran arrays and Kokkos Views.

Keeps Track of:

- An array's rank
- Extents of the array
- Strides of the array
- Pointer to the allocation

nd_array_t

The compatibility glue between Fortran arrays and Kokkos Views.

Keeps Track of:

- An array's rank
- Extents of the array
- Strides of the array
- Pointer to the allocation

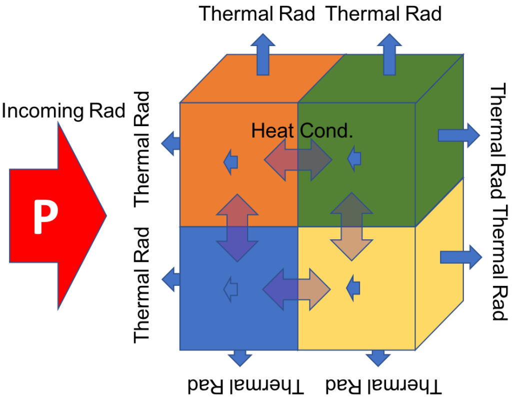
How do we create an `nd_array_t`?

- Explicit routines like `to_nd_array_i64_6`
- Simple interface taking a fortran array as argument

```
array = to_nd_array(foo);
```

3D Heat Conduction

- Heat conduction inside the body
- Thermal radiation (Black Body) on surface
- Incoming power flow from one direction



Graph Kernels

Kokkos Kernels functionality for graph computations.

Learning objectives:

- Calling BLAS functions with Views
- Calling BLAS functions with Views
- Calling BLAS functions with Views
- Calling BLAS functions with Views

HPC world owns many Fortran LOC!

- We generally cannot port it all at once.
- We need an incremental porting strategy
  - Keep our e.g. Fortran mains, drivers, physics packages
  - But port relevant infrastructure, or hotspot kernels to C++ and Kokkos

HPC world owns many Fortran LOC!

- We generally cannot port it all at once.
- We need an incremental porting strategy
  - Keep our e.g. Fortran mains, drivers, physics packages
  - But port relevant infrastructure, or hotspot kernels to C++ and Kokkos

How do we make Kokkos and Fortran talk with each other?

Fortran Language Compatibility Layer (FLCL)

- Pass multidimensional arrays accross the C++/Fortran boundary
  - See Fortran arrays as Kokkos Views and vice versa
- Create Kokkos View and DualView from Fortran
  - Allows Fortran to be the memory owner but call C++ functions with Kokkos kernels for CUDA/HIP
- Initialize and Finalize Kokkos from Fortran
- FortranIndex<T> scalar type to deal with 1 vs 0 based indexing in sparse data structures

FLCL

The Fortran Language Compatibility Layer allows an incremental porting of a Fortran code to Kokkos.

nd_array_t

The compatibility glue between Fortran arrays and Kokkos Views.

Keeps Track of:

- An array's rank
- Extents of the array
- Strides of the array
- Pointer to the allocation

nd_array_t

The compatibility glue between Fortran arrays and Kokkos Views.

Keeps Track of:

- An array's rank
- Extents of the array
- Strides of the array
- Pointer to the allocation

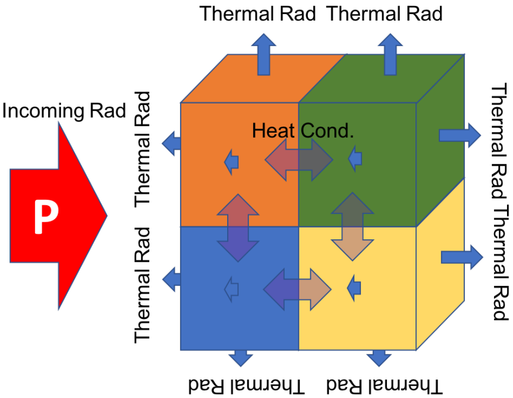
How do we create an `nd_array_t`?

- Explicit routines like `to_nd_array_i64_6`
- Simple interface taking a fortran array as argument

```
    array = to_nd_array(foo);
```

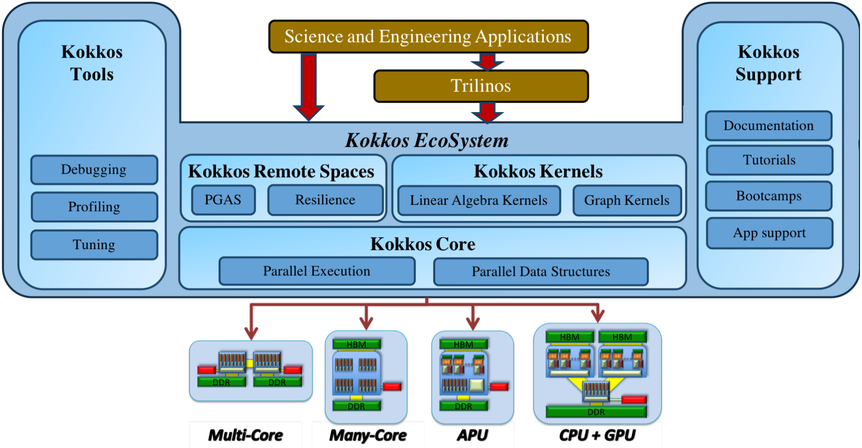
3D Heat Conduction

- Heat conduction inside the body
- Thermal radiation (Black Body) on surface
- Incoming power flow from one direction



Summary

More Summary



Science and Engineering Applications

Trilinos

Kokkos Tools

Debugging

Profiling

Tuning

Kokkos EcoSystem

Kokkos Remote Spaces

PGAS

Resilience

Kokkos Kernels

Linear Algebra Kernels

Graph Kernels

Kokkos Core

Parallel Execution

Parallel Data Structures

Kokkos Support

Documentation

Tutorials

Bootcamps

App support

kokkos

Los Alamos NATIONAL LABORATORY EST.1943

Sandia National Laboratories

Argonne NATIONAL LABORATORY

OAK RIDGE National Laboratory

BERKELEY LAB

CSCS

Kokkos Core:	**C.R.Trott**, J. Ciesko, V. Dang, N. Ellingwood, D.S. Hollman, D. Ibanez, J. Miles, J. Wilke, , H. Finkel, N. Liber, D. Lebrun-Grandie, D. Arndt, B. Turcksin, J. Madsen, R. Gayatri `former:` H.C. Edwards, D. Labreche, G. Mackey, S. Bova, D. Sunderland
Kokkos Kernels:	**S. Rajamanickam**, L. Berger, V. Dang, N. Ellingwood, E. Harvey, B. Kelley, K. Kim, C.R. Trott, J. Wilke, S. Acer
Kokkos Tools	**D. Poliakoff**, C. Lewis, S. Hammond, D. Ibanez, J. Madsen, S. Moore, C.R. Trott
Kokkos Support	**C.R. Trott**, G. Shipmann, G. Womeldorff, and all of the above `former:` H.C. Edwards, G. Lopez, F. Foertter

Online Resources:

- https://github.com/kokkos:
  - Primary Kokkos GitHub Organization
- https://github.com/kokkos/kokkos-tutorials/wiki/Kokkos-Lecture-Series:
  - Slides, recording and Q&A for the Lectures
- https://github.com/kokkos/kokkos/wiki:
  - Wiki including API reference
- https://kokkosteam.slack.com:
  - Slack channel for Kokkos.
  - Please join: fastest way to get your questions answered.
  - Can whitelist domains, or invite individual people.