

Kokkos

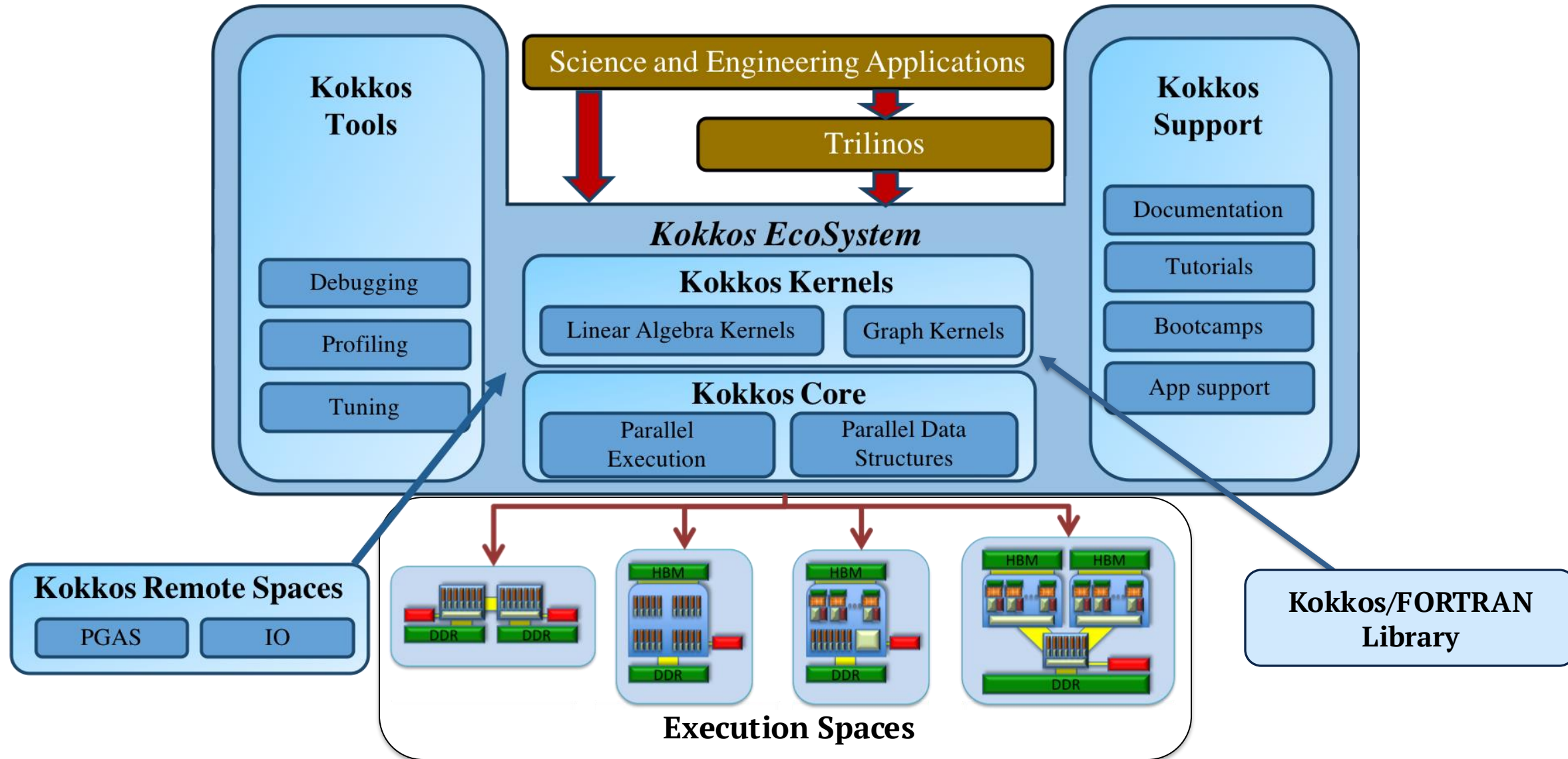
Brian Homerding
Argonne Leadership Computing Facility



What is Kokkos?

- A C++ Programming Model for Performance Portability
 - Implemented as a template library on top of CUDA, OpenMP, SYCL, ...
 - Aims to be descriptive not prescriptive
 - Aligns with developments in the C++ standard
- Expanding solution for common needs of modern science/engineering codes
 - Math libraries based on Kokkos
 - Tools which allow inside into Kokkos
- It is Open Source
 - Maintained and developed at <https://github.com/kokkos>
- It has many users at wide range of institutions.

Kokkos EcoSystem





Kokkos Core Abstractions

Kokkos

Data Structures

Memory Spaces (“Where”)

- HBM, DDR, Non-Volatile, Scratch

Memory Layouts

- Row/Column-Major, Tiled, Strided

Memory Traits (“How”)

- Streaming, Atomic, Restrict

Parallel Execution

Execution Spaces (“Where”)

- CPU, GPU, Executor Mechanism

Execution Patterns (“How”)

- parallel_for/reduce/scan, task-spawn

Execution Policies

- Range, Team, Task-Graph



Example: Conjugent Gradient Solver

- Simple Iterative Linear Solver
- For example used in MiniFE
- Uses only three math operations:
 - Vector addition (AXPBY)
 - Dot product (DOT)
 - Sparse Matrix Vector multiply (SPMV)
- Data management with Kokkos Views:

```
View<double*,HostSpace,MemoryTraits<Unmanaged>> h_x(x_in, n_rows);  
View<double*> x("x",n_rows);  
deep_copy(x,h_x);
```

CG Solve: The AXPBY

- Simple data parallel loop: Kokkos::parallel_for
- Easy to express in most programming models
- Bandwidth bound
- Serial Implementation:

```
void axpby(int n, double* z, double alpha, const double* x,  
           double beta, const double* y) {  
  for(int i=0; i<n; i++)  
    z[i] = alpha*x[i] + beta*y[i];  
}
```

Parallel Pattern: for loop

- Kokkos Implementation:

```
void axpby(int n, View<double*> z, double alpha, View<const double*> x,  
           double beta, View<const double*> y) {  
  parallel_for("AXpBY", n, KOKKOS_LAMBDA (const int i) {  
    z(i) = alpha*x(i) + beta*y(i);  
  });  
}
```

String Label: Profiling/Debugging

Execution Policy: do n iterations

Loop Body

Iteration handle: integer index

Kokkos Parallelism

- **Simple** usage is similar to OpenMP, advanced features are also straightforward
- Three common **data-parallel patterns** are parallel for, parallel reduce, and parallel scan.
- A parallel computation is characterized by its **pattern, policy, and body**.
- User **provides computational bodies** as functors or lambdas which handle a single work item.

```
double * x = new double[N]; // also y and z
parallel_for("AXpBY", N, [=] (const int64_t i) {
    z[i] = alpha * x[i] + beta * y[i];
});
```

Lambda

```
struct Functor {
    double *_x, *_y, *_z, _a, _b;
    void operator()(const int64_t i) const {
        _z[i] = _a * _x[i] + _b * _y[i]; }
};
```

Functor

Kokkos Memory Management

- Data is stored in Views that are “pointers” to **multi-dimensional arrays** residing in **memory spaces**.
- Views **abstract away** platform-dependent allocation, (automatic) deallocation, and access.
- **Heterogeneous nodes** have one or more memory spaces.
- **Mirroring** is used for performant access to views in host and device memory.

```
// Create view array in device memory space.
```

```
using view_type = Kokkos::View<double*, Space>;
```

```
view_type view (...);
```

```
// Mirror view in host memory space.
```

```
view_type::HostMirror hostView = Kokkos::create_mirror_view(view);
```

```
// Fill host view from input.
```

```
populateView(hostView);
```

```
// Copy host view to view on device.
```

```
Kokkos::deep_copy(view, hostView);
```

```
// Use the view on the device.
```

```
Kokkos::parallel_for(...
```


Quickstart

```
kokkos_user@aurora:> git clone https://github.com/kokkos/kokkos-tutorials.git
```

```
kokkos_user@aurora:> module load kokkos cmake
```

```
kokkos_user@aurora:> env | grep KOKKOS
```

```
KOKKOS_ROOT=/opt/aurora/24.347.0/spack/unified/0.9.2/install/linux-sles15-x86_64/oneapi-2025.0.5/kokkos-4.5.01-kcmoh6f
```

```
kokkos_user@aurora:> cd kokkos-tutorials/Exercises/04
```

```
kokkos_user@aurora:> ls
```

Begin Solution

```
kokkos_user@aurora:> cd Solution
```

```
kokkos_user@aurora:> cmake -B build_sycl -DKokkos_ENABLE_SYCL=ON
```

```
kokkos_user@aurora:> cmake --build build_sycl
```

Resources

- **Getting Started**

- Kokkos Tutorials: <https://github.com/kokkos/kokkos-tutorials>
- Kokkos Lectures: <https://kokkos.org/kokkos-core-wiki/tutorials-and-examples/video-lectures.html>

- **Documentation**

- Kokkos Documentation: <https://kokkos.org/kokkos-core-wiki/>
- Kokkos git: <https://github.com/kokkos/kokkos>

- **Running on Aurora**

- Kokkos on Aurora: <https://docs.alcf.anl.gov/aurora/programming-models/kokkos-aurora/>
- Running jobs on Aurora: <https://docs.alcf.anl.gov/aurora/running-jobs-aurora/>

- **Getting Help**

- ALCF Help: <https://docs.alcf.anl.gov/support/>
- Kokkos Slack: <https://kokkosteam.slack.com>

