kokkos Programming Model Update

Sandia National Laboratories
C.R. Trott, D. Sunderland, N. Ellingwood, D. Ibanez,
S. Bova, J. Miles, D. Hollman, D. Lanbreche, V. Dang

SAND2019-0170D

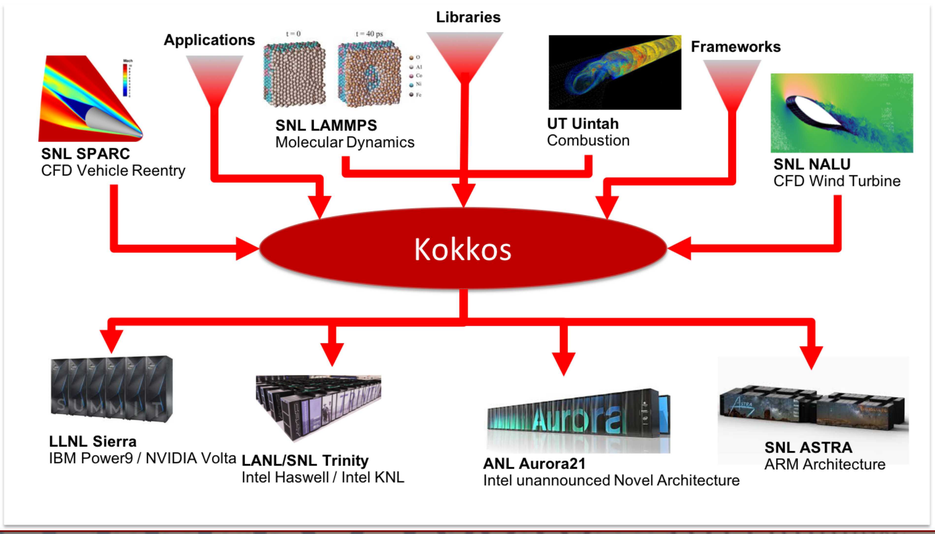
C++ Performance Portability or Bust

- Typical Industry estimate for Developer Productivity:

10 LOC / hour ~ 20k LOC / year

- Optimistic estimate: 10% of an application needs to get rewritten for adoption of a new Shared Memory Parallel Programming Model
- Typical Apps: 300k – 600k Lines
  - Typical App Port => **2-3 Man-Years**
- Large Scientific Libraries
  - E3SM: 1,000k Lines x 10% => **5 Man-Years**
  - Trilinos: 4,000k Lines x 10% => **20 Man-Years**
- **Example:** Change from NVIDIA Pascal to Volta architecture broke synchronization
  - 3 code places in Kokkos needed fixing => 2 man-months work
  - Trilinos implemented as native CUDA would have had 400 places

Kokkos Provides: *Vendor-Independent Isolation Layer*



Fundamental Capabilities

Concept	Example
Parallel Loops	parallel_for(N, KOKKOS_LAMBDA (int i) { ...BODY... });
Parallel Reduction	parallel_reduce(RangePolicy<ExecSpace>(0,N), KOKKOS_LAMBDA (int i, double& upd) { ...BODY... upd += ... }, result);
Tightly Nested Loops (exp)	parallel_for(MDRangePolicy<Rank<3> > ({0,0,0},{N1,N2,N3},{T1,T2,T3}, KOKKOS_LAMBDA (int i, int j, int k) {...BODY...});

Non-Tightly Nested Loops	parallel_for(TeamPolicy<Schedule<Dynamic>>(N, TS), KOKKOS_LAMBDA (Team team) {
	... COMMON CODE 1 ...
	parallel_for(TeamThreadRange(team, M(N)), [&] (int j) { ... INNER BODY... });
	... COMMON CODE 2 ...
	});

Task Dag	task_spawn(TaskTeam(scheduler , priority), KOKKOS_LAMBDA (Team team) { ... BODY });
Data Allocation	View<double**, Layout, MemSpace> a("A",N,M);
Data Transfer	deep_copy(a,b);
Exec Spaces	Serial, Threads, OpenMP, Cuda, ROCm *(experimental)*

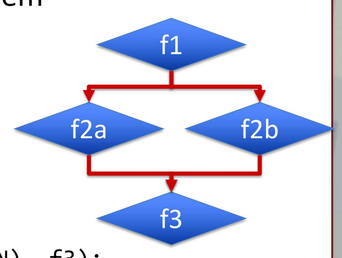
Upcoming: Asynchronous Dispatch with Dependencies

- C++ standard is moving towards more asynchronicity
  - Executors proposal for supporting heterogeneous computing
  - Dispatch of parallel work returns new kind of `future`
  - Dispatch consumes future for dependency ordering
- Aligning Kokkos with this development means:
  - Introduction of Execution space instances

```
DefaultExecutionSpace spaces[2];
partition( DefaultExecutionSpace(), 2, spaces);
// f1 and f2 are executed simultaneously
parallel_for( RangePolicy<>(spaces[0], 0, N), f1);
parallel_for( RangePolicy<>(spaces[1], 0, N), f2);
// wait for all work to finish
fence();
```

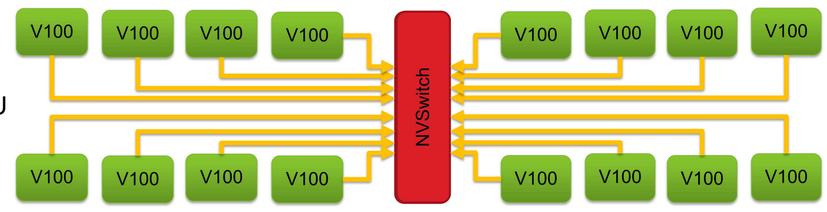
  - Patterns return futures and Execution Policies consume them

```
auto fut_1 = parallel_for(
          RangePolicy<>("Funct1", 0, N), f1);
auto fut_2a = parallel_for(
          RangePolicy<>("Funct2a", fut_1,0, N), f2a);
auto fut_2b = parallel_for(
          RangePolicy<>("Funct2b", fut_1,0, N), f2b);
auto fut_3 = parallel_for(
          RangePolicy<>("Funct3", all(fut_2a,fut2_b),0, N), f3);
fence(fut_3);
```



Upcoming: Kokkos Remote Spaces

- PGAS Models may become more viable for HPC with both changes in network architectures and the emergence of "super-node" architectures
  - Example DGX2
  - First SuperNode
  - 300GB/s per GPU



- Idea: Add new memory spaces which return data handles with shmem semantics to Kokkos View
  - View<double**[3], LayoutLeft, NVShmemSpace> a("A",N,M);
  - Operator a(i,j,k) returns:

```
template<>
struct NVShmemElement<double> {
  NVShmemElement(int pe_, double* ptr_):pe(pe_),ptr(ptr_) {}
  int pe; double* ptr;
  void operator = (double val) { shmem_double_p(ptr,val,pe); }
};
```

- Test Problem: CG-Solve
  - Using the miniFE problem N^3
  - Compare to optimized CUDA
  - MPI version is using overlapping
  - DGX2 4 GPU workstation
- 3 Variants
  - Full use of SHMEM
  - Inline functions by ptr mapping
  - Explicit by-rank indexing

Development Roadmap

Q2 FY19: CUDA stream support as precursor for instances
- User is responsible for creating streams

Q3 FY19: Initial ExecSpace Instances
- Cuda fully supported, OpenMP will serialize,

Q4 FY19: Initial RemoteSpaces for PGAS
Q4 FY19: Working AMD GPU support for Apps (compilers are main issue)
H1FY20: Initial RemoteSpaces for Resilience
H1FY20: Initial Dependency support for parallel dispatch
H1FY20: Initial Support for ANL Aurora Architecture

National Nuclear Security Administration
U.S. DEPARTMENT OF ENERGY
SAND2019-XXXX C

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.

ST 2.3.1.04 SNL ATDM PMR