



Kokkos EcoSystem Update



Kokkos Core:

C.R. Trott, N. Ellingwood, D. Ibanez, V. Dang, Jan Ciesko, L. Cannada, N. Liber, D. Lebrun-Grandie, B. Turcksin, J. Madsen, D. Arndt, R. Gayatri

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, S. Hammond, C.R. Trott, D. Ibanez, S. Moore, L. Cannada

Kokkos Support:

C.R. Trott, G. Shipman, G. Womeldorff, All of the above!

pyKokkos:

M. Gligoric, N. Al Awar, S. Zhu, J. Madsen



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.

Sandia National Laboratories is a multimission laboratory managed and operated by National Technology & 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.



U.S. DEPARTMENT OF
ENERGY

Office of
Science



Kokkos EcoSystem

- EcoSystem
- Core Backend Status
- Core Graphs
- Remote Spaces
- Kernels
- Tools
- Resilience
- pyKokkos

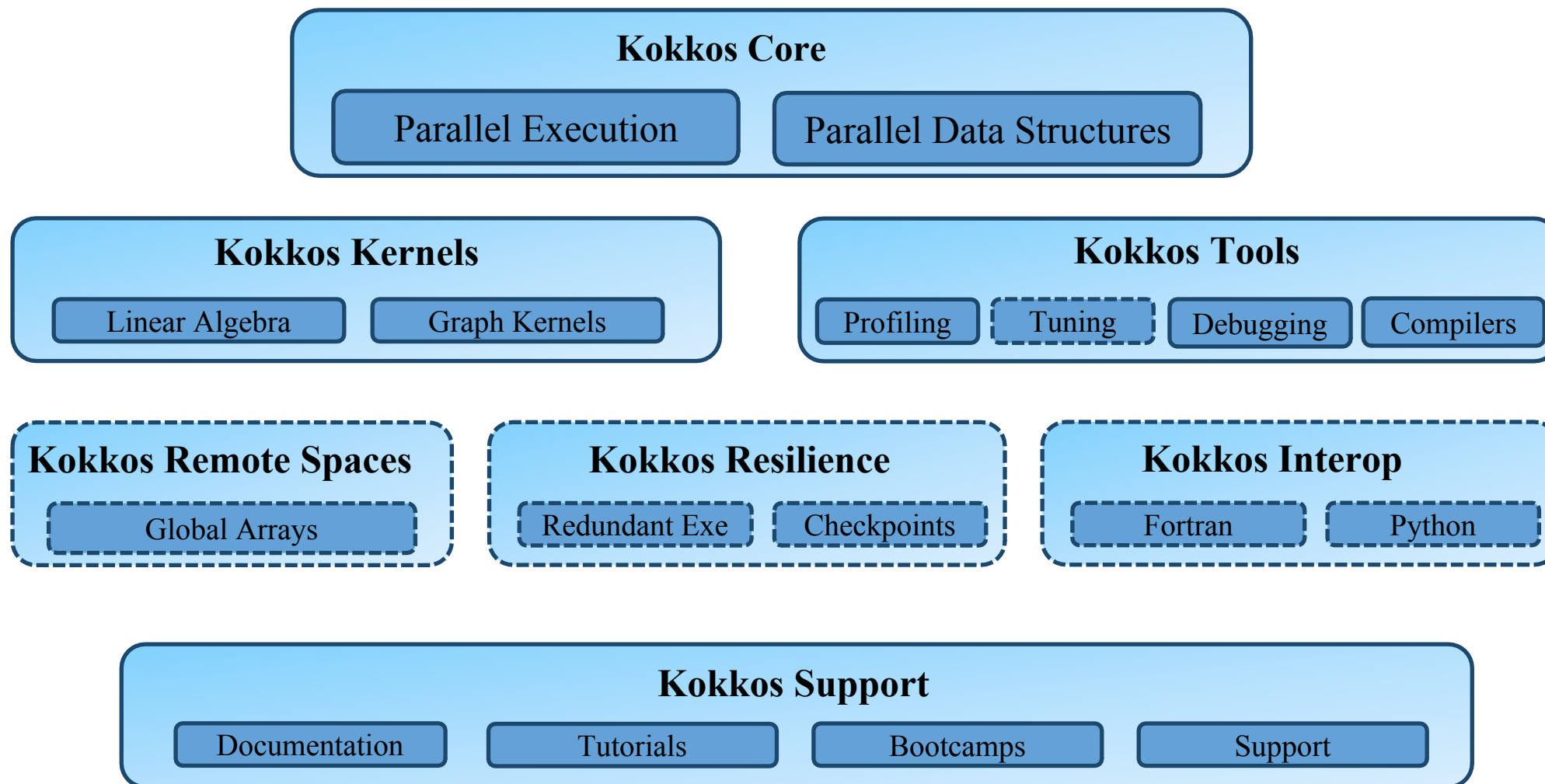




Kokkos EcoSystem

Mature

Early Users





Developer Team

- **Kokkos Core:** 12 Developers (6 SNL)
 - Trying to fill some recent gaps
- More code contributions from non-SNL
 - >50% of commits from non-Sandians
- Sandia leads API design
- Other labs lead backend implementations
- Other subprojects largely by Sandia so far



Kokkos Core:

C.R. Trott, N. Ellingwood, D. Ibanez, V. Dang, Jan Ciesko, L. Cannada,
N. Liber, D. Lebrun-Grandie, B. Turcksin, J. Madsen, D. Arndt, R. Gayatri
former: **H.C. Edwards**, D. Labreche, G. Mackey, S. Bova, D. Sunderland, J. Miles, D. Hollman, J. Wilke

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, S. Hammond, C.R. Trott, D. Ibanez, S. Moore, L. Cannada

Kokkos Support:

C.R. Trott, G. Shipman, G. Lopez, G. Womeldorff,
former: **H.C. Edwards**, D. Labreche, Fernanda Foertter



Users and Uptake

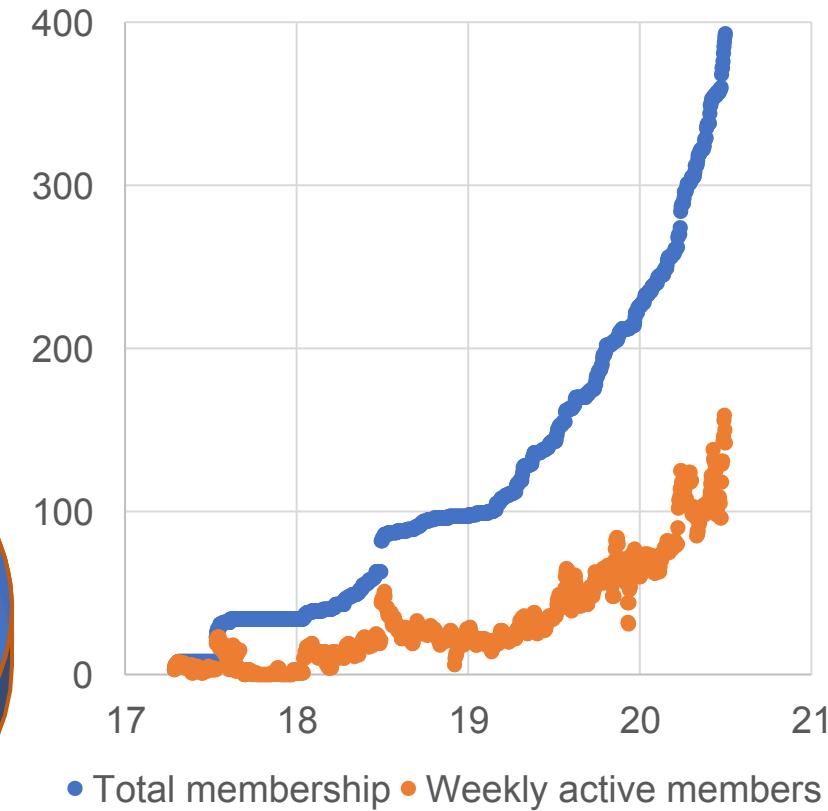
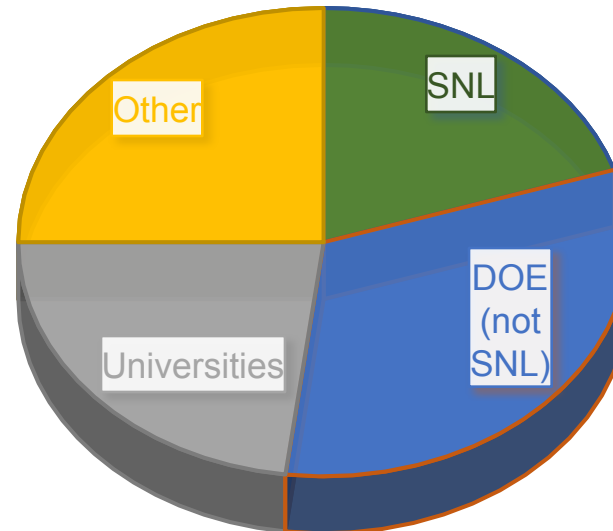
ECP Critical Dependencies

MPI	60
LLVM	57
C++	41
OpenMP	34
LAPACK	24
CUDA	22
Fortran	21
HDF5	21
BLAS	21
Kokkos	18
C	14
ALPINE	12

hypre	11
DAV-SDK	11
VTK-m	11
Trilinos	10
ADIOS	8
SPACK	8
SCALAPACK	8
FFT	7
OpenACC	7
MPI-IO	6
PnetCDF	6
Tau	6

Kokkos Slack Users

- 560 registered users
 - 90 Institutions
 - Every continent
 - (-Antarctica)





Kokkos: The Lectures

- 8 lectures covering most aspects of Kokkos
- 15 hours of recordings
- > 500 slides
- >20 exercises
- Extensive Wiki
 - API Reference
 - Programming Guide
- Slack as primary direct support

<https://kokkos.link/the-lectures>

- **Module 1: *Introduction***
 - Introduction, Basic Parallelism, Build System
- **Module 2: *Views and Spaces***
 - Execution and Memory Spaces, Data Layout
- **Module 3: *Data Structures and MDRangePolicy***
 - Tightly Nested Loops, Subviews, ScatterView,...
- **Module 4: *Hierarchical Parallelism***
 - Nested Parallelism, Scratch Pads, Unique Token
- **Module 5: *Advanced Optimizations***
 - Streams, Tasking and SIMD
- **Module 6: *Language Interoperability***
 - Fortran, Python, MPI and PGAS
- **Module 7: *Tools***
 - Profiling, Tuning , Debugging, Static Analysis
- **Module 8: *Kokkos Kernels***
 - Dense LA, Sparse LA, Solvers, Graph Kernels



Kokkos Core Backend Status





Platform Strategy

Why OpenMP too?

- Fallback in case of compiler issues
- Interoperability with OpenMP libraries



Name	Perlmutter	Frontier	Aurora	ElCapitan	Crossroads	Fugaku
Facility	NERSC	ORNL	ANL	LLNL	LANL	RIKEN
Year	2021	2021	2022	2023	2022	2020
Hardware	NVIDIA GPU + AMD CPU	AMD GPU + AMD CPU	Intel GPU + Intel CPU	AMD GPU + AMD CPU	Intel CPU	ARM CPU
Toolchain	NVCC, NVC++	ROCM, Cray	Intel OneAPI	ROCM, Cray	Intel OneAPI, Cray, GCC	Fujitsu, GCC, ARM Clang
CUDA						
OpenMP						
DPC++/SYCL						
HIP						



Kokkos HIP backend

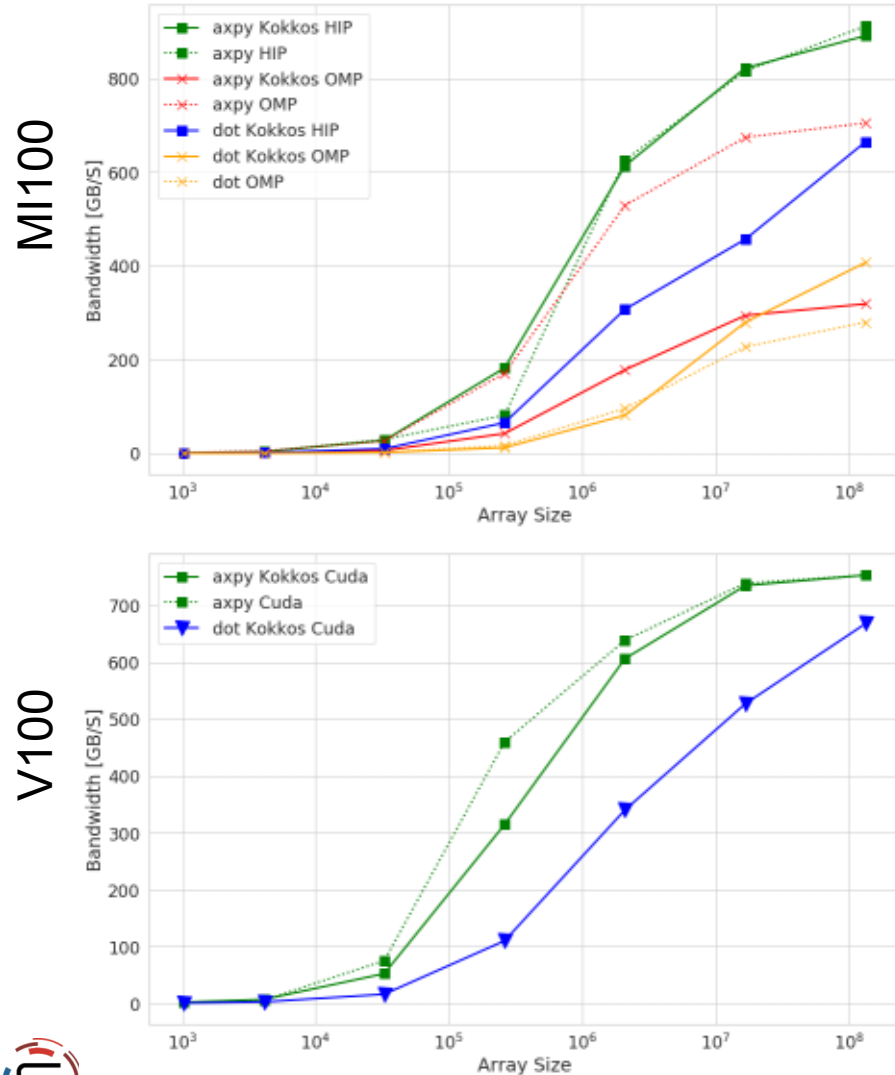
- HIP:
 - native language of AMD's GPU
 - very similar to CUDA
 - supports both Nvidia and AMD's GPU
 - Still compiler issues -> tracking latest release very closely
- Primary backend for AMD GPUs
 - Do not support NVIDIA GPUs (use CUDA backend instead)
- Introduced in Kokkos-3.1(released in Apr'20)
- Feature complete except for dynamic tasking
 - Focus so far has been on adding feature, now moving to performance
- We require ROCm 3.8, we will require 4.1 soon

AMD
ROCm



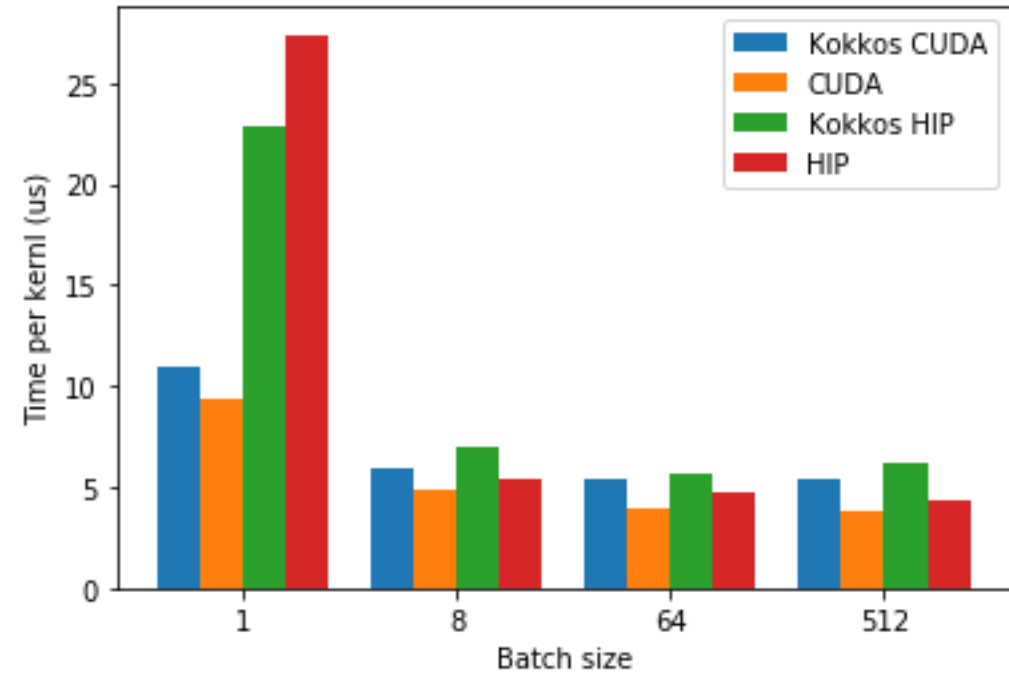
Performance Comparison: V100 vs MI100

AXPBY and DOT



Launch Latency Benchmark

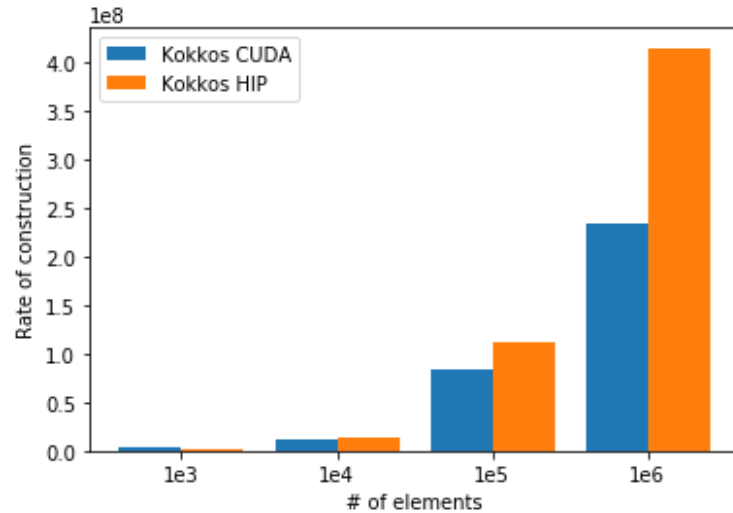
(Batch Size = # of kernels before fence)



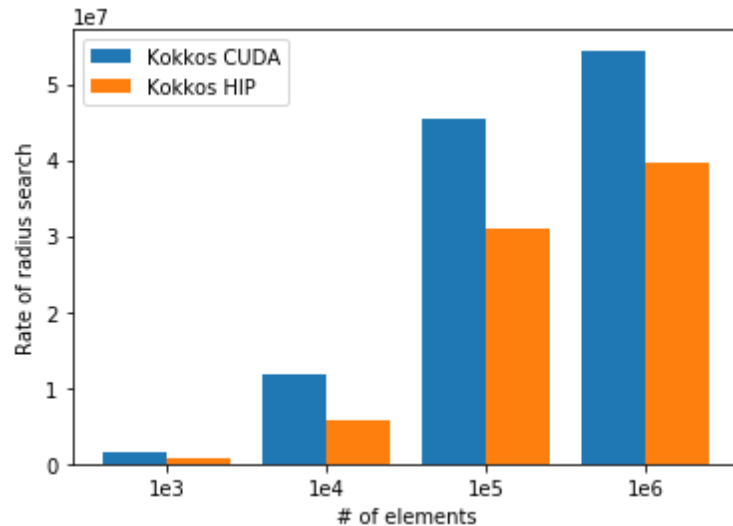
MI100 results are from our own system (lascaux02), not Tulip



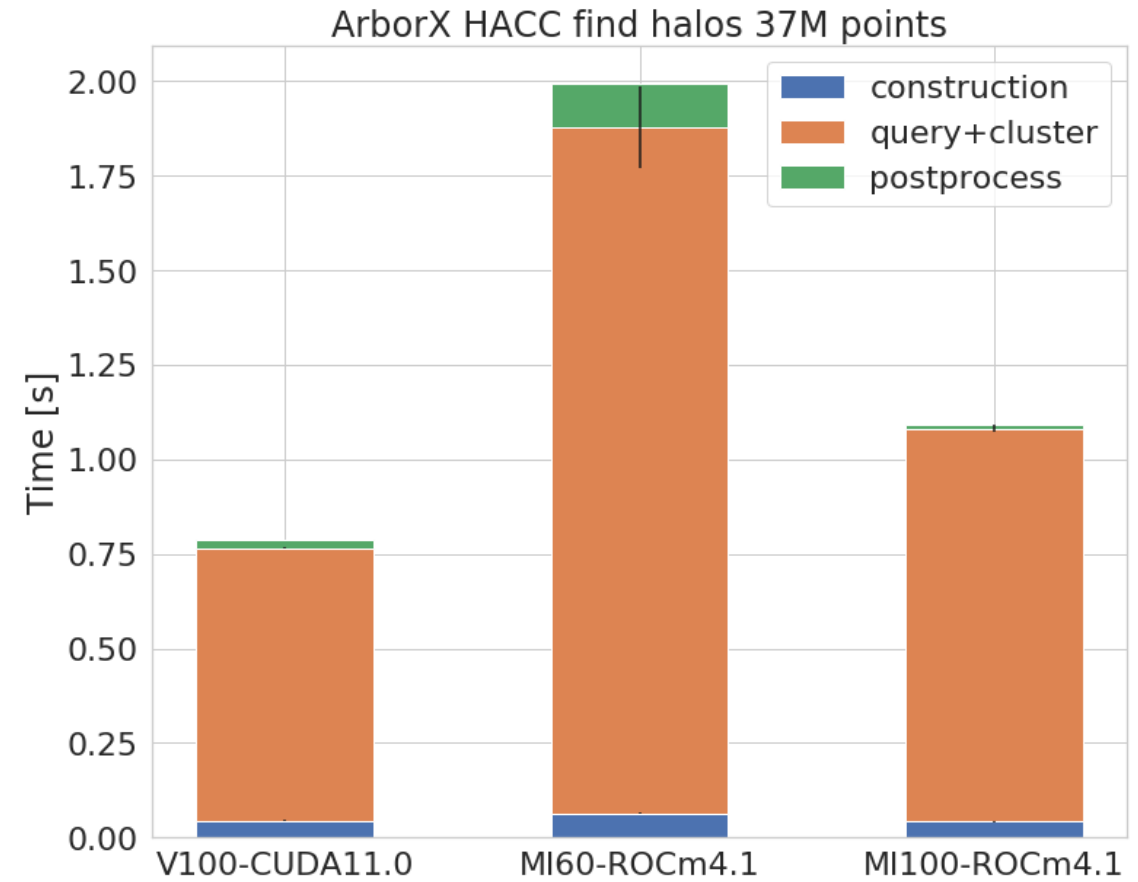
Performance Comparison: V100 vs MI100 (ArborX)



BVH construction



BVH query (radius search)





Kokkos SYCL backend

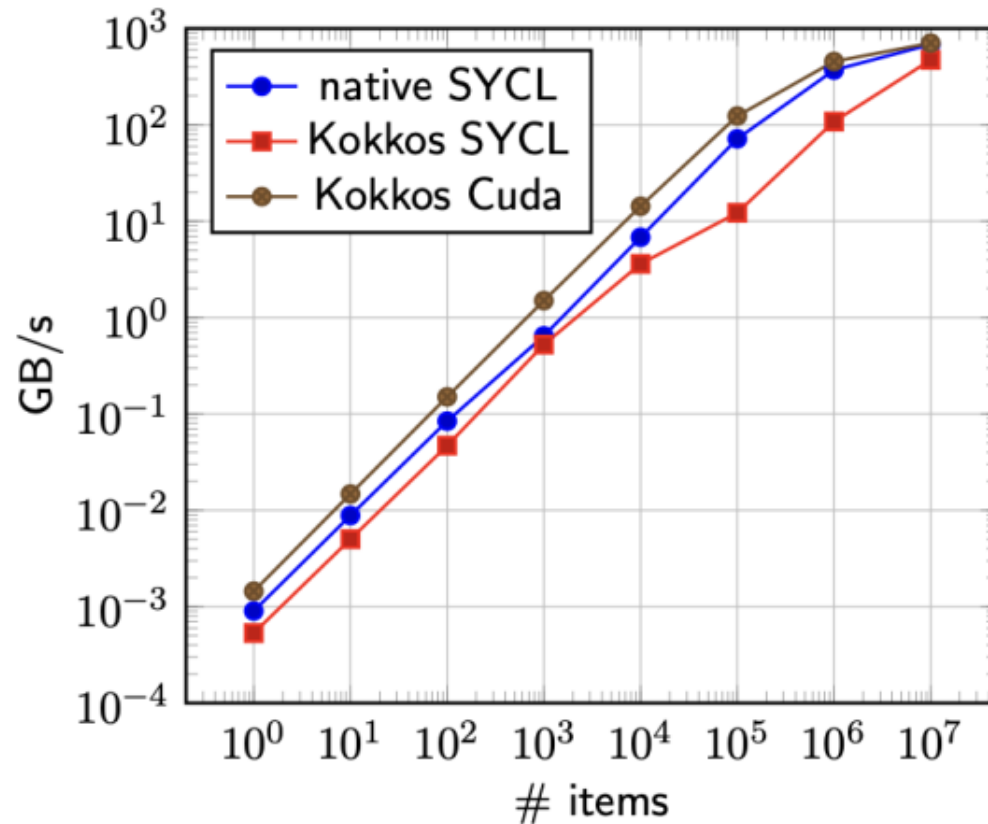
- SYCL
 - Open standard (Khronos group)
 - Parallel heterogeneous computing
 - Intel implementation called DPC++
- Hardware support
 - Targeting Intel GPUs
 - CI running on NVIDIA V100
- Introduced in Kokkos-3.3 (released in Dec'20)
- Near feature complete in Kokkos-3.4 (Apr'21)
- Using DPC++ extensions of SYCL – may not work with other SYCL implementations
- Not yet implemented
 - atomics for types larger than 64bits
 - WorkGraphPolicy
 - Dynamic Task Graphs
- Future work will focus on performance
- Known to work with SYCL backend: ArborX, Cabana, LAMMPS





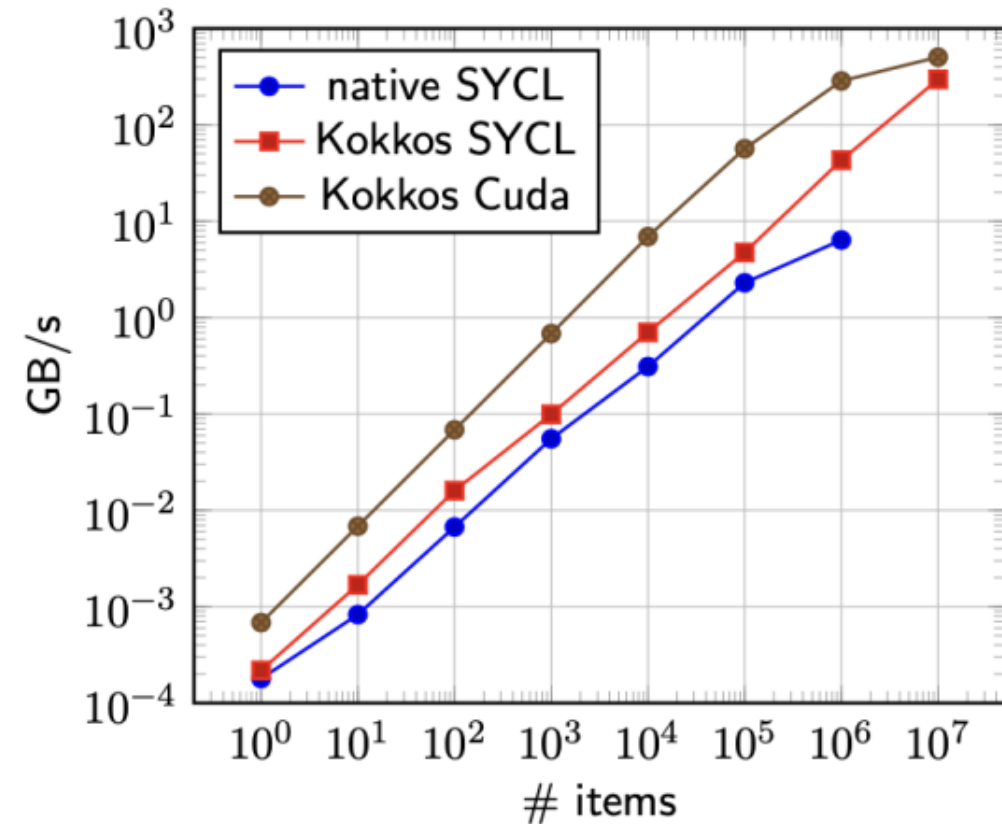
Kokkos SYCL backend – Performance on V100

AXPBY
(double scalar type)



parallel_for, AXPY

DOT
(double scalar type)



parallel_reduce, DOT



OpenMPTarget Backend

- Secondary backend for all GPU platforms
- Majority of features are available
- Started working with applications to test the backend

Compiler/Architecture Support Matrix

Architecture	clang	icpx	rocm	nvhpc	cce
NVIDIA (V100)	support			partial support	
AMD (MI60/MI100)			support		partial support
Intel (Gen9)		support			

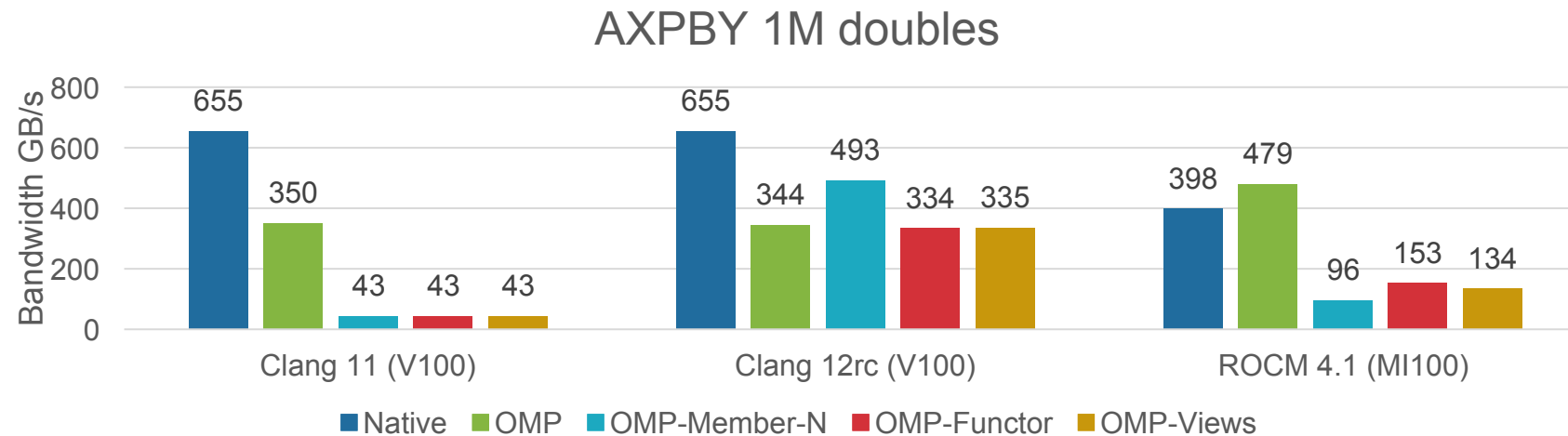
 support  partial support



OpenMPTarget: Performance Fragility

- OpenMP Offload Compilers still fragile in performance
 - Small changes can have dramatic impacts
 - Have not finetuned Kokkos backend yet, since this rapidly changes
- Example: AXPBY
 - No Functor, Raw Ptrs, Loop Bound local
 - No Functor, Raw Ptrs, Loop Bound member
 - Functor, Raw Ptrs, Loop Bound local
 - Functor, Views, Loop Bound Local

```
struct AXPY {  
    template <class ExecutionSpace, class View>  
    AXPY(ExecutionSpace const &s, View x, View y) {  
        using T = typename View::value_type;  
        T const a = 2;  
        auto xp = x.data();  
        auto yp = y.data();  
        int N = x.extent(0);  
        #pragma omp target teams distribute parallel \  
            for is_device_ptr(xp, yp)  
        for (int i = 0; i < N; ++i) {  
            yp[i] = a * xp[i] + yp[i];  
        }  
        s.fence();  
    }  
};
```





OpenMPTarget: What's Next?

Challenges

- Reductions
 - Mandatory `shared` clauses on reduction items makes hierarchical reduction a challenge
 - Declare reductions support non-uniform
- Non-Uniform Support of SIMD
 - Causes performance issues for 3-level hierarchical parallelism in Kokkos
 - Intel supports SIMD, but many other compilers don't on GPUs.

Next Steps

- Work with ECP apps to build with OpenMPTarget backend
 - Already started in particular for Aurora
- Optimize implementation
- Work with compiler vendors to support Unified Memory and SIMD
- Work to support more compilers fully (GCC, NVHPC, CCE)



Backend Feature Support Matrix

Feature	HIP	SYCL	OMPT
Range par_for	X	X	X
Range par_red	X	X	X
Range par_scan	X	X	X
MDRange par_for	X	X	X
MDRange par_red	X	X	X
Team par_for	X	X	X
Team par_red	X	X	X
TeamThread par_for	X	X	X
TeamThread par_red	X	X	X
ThreadVector par_for	X	X	X
ThreadVector par_red	X	X	X
WorkGraphPolicy	X	--	--
Graphs	X	X	X

Feature	HIP	SYCL	OMPT
Team Scratch	X	X	X
Reducers Build-In	X	X	X
Reducers Custom	X	X	--
Reducers Multiple	X	X	--
Reducers Team Nested	X	X	X
Atomics 32/64bit build-in	X	X	X
Atomics 32/64bit custom	X	X	X
Atomics arbitrary size	X	--	--
Unique Token Global	X	X	X
Unique Token Instance	X	X	--
Dynamic Tasking	--	--	--
Tools: Profiling Hooks	X	X	X
Tools: Tuning Hooks	X	X	X



Kokkos Graphs





CUDA Graphs & Kokkos Graphs

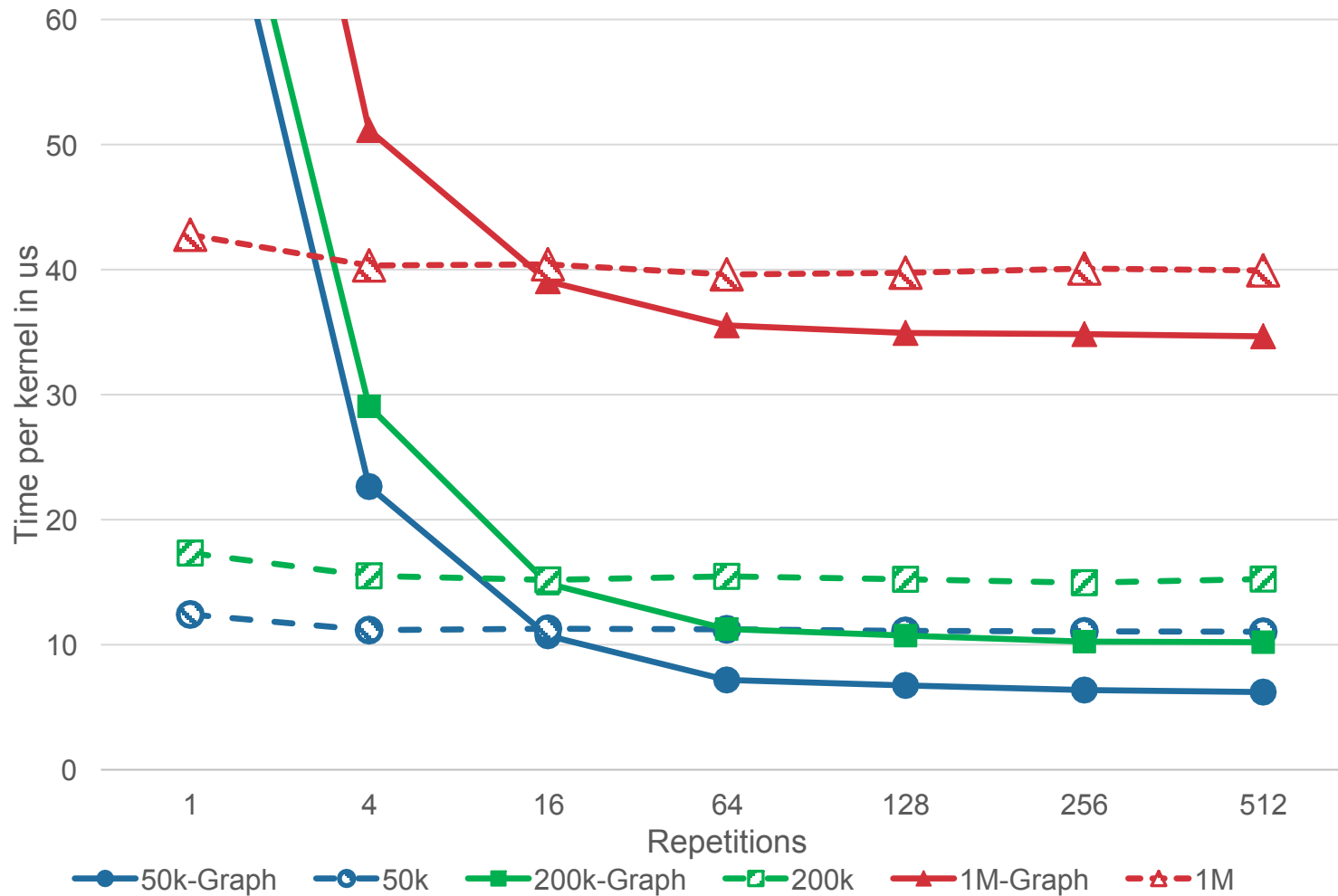
- Kokkos Graphs in Release 3.3 (January 2021) expose CUDA Graphs
 - One Design Principle: less foot guns than CUDA Graphs
- API Design: Scoped creation/Explicit dependencies/no implicit capture

```
View<double*> x, y, z;  
View<double, CudaHostPinnedSpace> x_dot_z;  
double alpha, beta, gamma, threshold;  
DefaultExecutionSpace ex();  
  
auto graph = create_graph(ex, [&](auto root) {  
    auto f_xpy = root.then_parallel_for(N, axpby_func{alpha,x,beta,y});  
    auto f_zpy = root.then_parallel_for(N, axpby_func{gamma,z,beta,y});  
  
    //-----  
    auto ready = when_all(f_xpy, f_zpy);  
  
    //-----  
    ready.then_parallel_reduce(N, dot_func{x,z}, x_dot_z);  
};  
  
while(x_dot_z()<threshold) { graph.submit(); Kokkos::fence(); }
```



Kokkos Graphs Benchmark

Solid: Graphs
Dashed: Simple Dispatch



Can reuse graph:

- In solver iterations
 - Between solves if matrix structure unchanged
- >100 reuses could be realistic

Throughput Improvement:

- 50K 78%
- 200k 49%
- 1M 15%

Next: look at reducing graph creation time



Remote Spaces





Kokkos Remote Spaces

Add support for distributed memory spaces to facilitate software development for multi-node and multi-GPU executions.

Allow code reuse to scale across multiple memory address spaces at minimal development effort

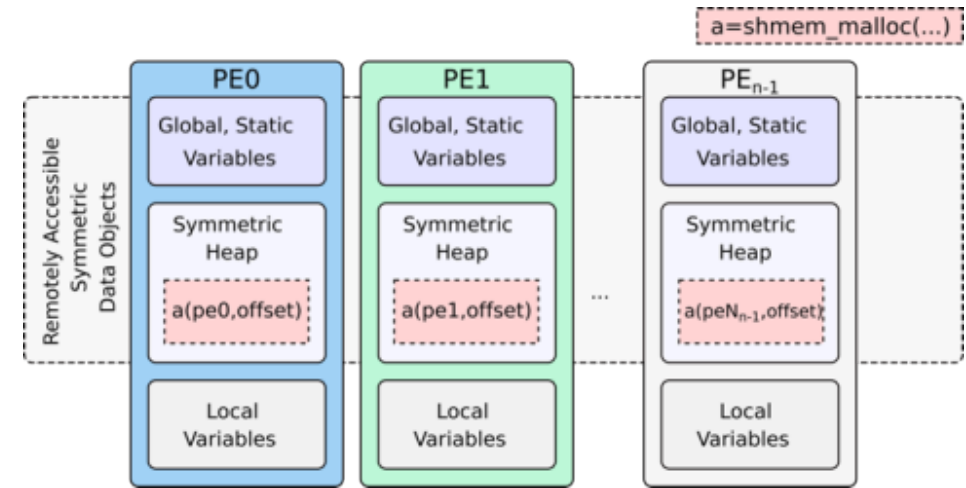
Adds distributed View support through templating on a remote memory space

Several remote memory space back-ends are support: MPI One-sided, SHMEM and NVSHMEM

Underlying implementation allocates data on the symmetric heap (all participants allocate the same size)

In development

- Data shaping (subviews), block transfers (local_deep_copy) and memory access traits
- Performance optimizations (aggregation and caching)



```
using RemoteSpace_t = Kokkos::Experimental::SHMEMSpace;  
Kokkos::View<T**, RemoteSpace_t> a("A", numPEs, myN);  
  
a(0,0) = 6; //Writes 6 to view a on PE 0 at offset 0  
a(1,8) = 3; //Writes 3 to view a on PE 1 at offset 8
```

Examples: First dimension describes PE*. Note: This is a current design choice but subject to change as the PE can be deduced at runtime (at a small cost).



Example: Sparse Matvec

$$y = A * x$$

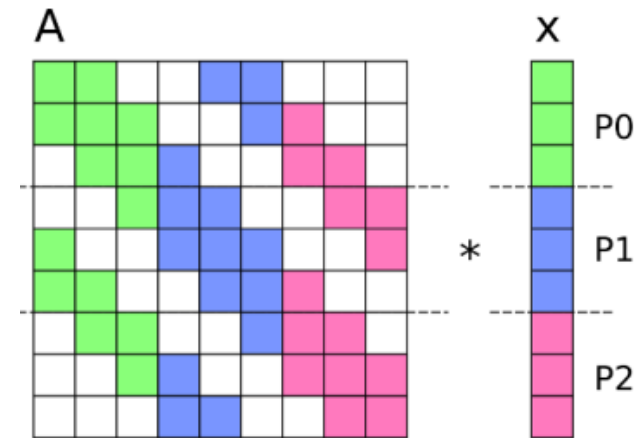
- A is a sparse matrix in Compressed Row Storage (CRS):
- Two implementations: Kokkos single- and multi memory spaces with KRS

Kokkos (single node)

```
1 Kokkos::View<double*> x = Kokkos::View("x", ncoeff);
2 ...
3 //SPMV: y=Ax (Kokkos)
4 Kokkos::parallel_for("SPMV", nrows, KOKKOS_LAMBDA(const int row){
5     double sum = 0.0;
6     for(i=A.rowOffset(row); i<A.rowOffset(row+1); ++i){
7         sum += A.vals(i) * x(A.colIdx(i));
8     }
9     y(row) = sum
10 });
```



```
1 using RemoteSpace_t = Kokkos::Experimental::DefaultRemoteMemorySpace;
2 Kokkos::View<double **, RemoteSpace_t> x = Kokkos::View("x", numRanks, ncoeff);
3 ...
4 //SPMV: y=Ax (Kokkos)
5 Kokkos::parallel_for("SPMV", nrows, KOKKOS_LAMBDA(const int row){
6     double sum = 0.0;
7     for(i=A.rowOffset(row); i<A.rowOffset(row+1); ++i){
8         int rank = A.colIdx(i) / nProc;
9         int offset = A.colIdx(i) % nProc;
10        sum += A.vals(i) * x(rank,offset);
11    }
12    y(row) = sum
13 });
```



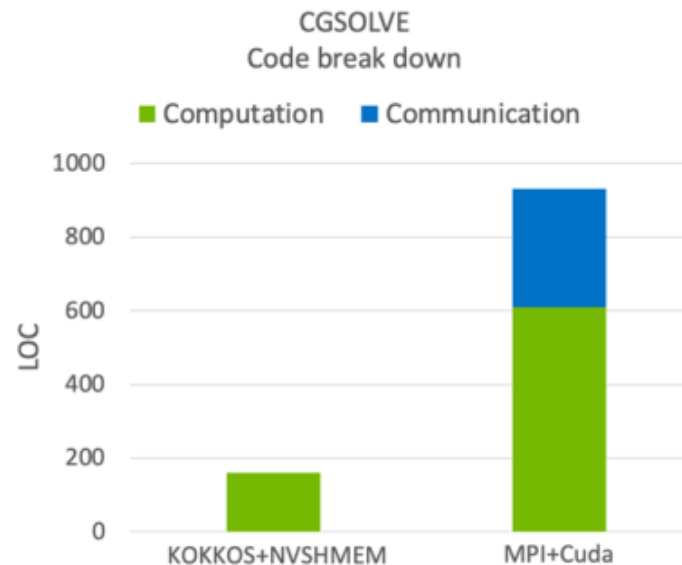
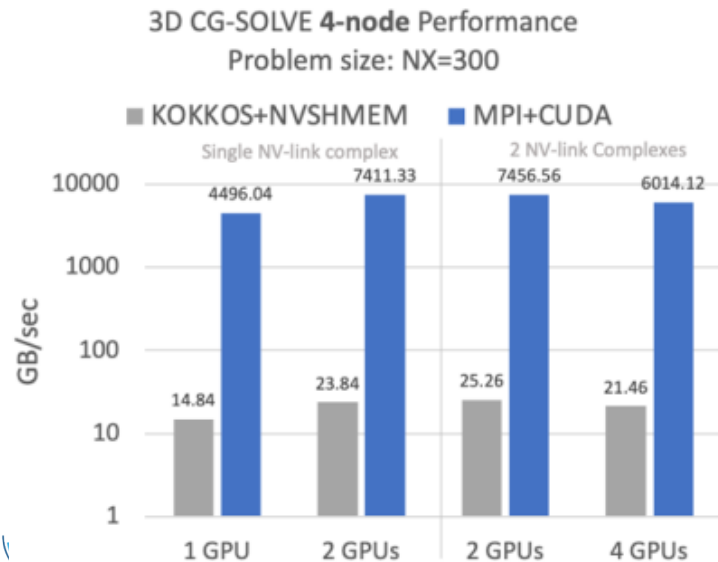
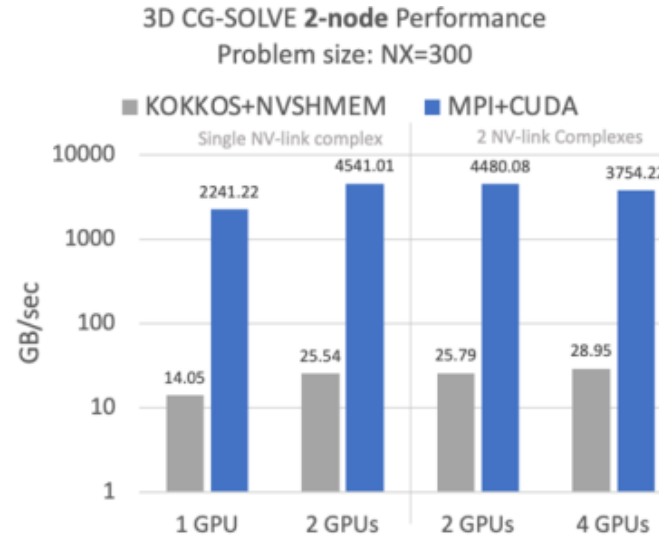
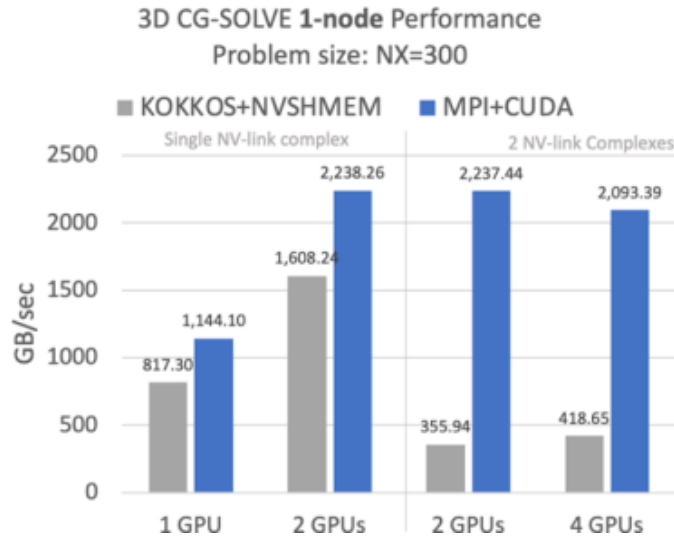
Coefficient matrix A
Vector x

<https://github.com/Mantevo/miniFE>

Q: How to support remote accesses to the vector x?



Example: CGSolve



Initial results with CGSolve: (relative to reference implementation with MPI+Cuda)

- 76% perf (1 NUMA)
- 20% perf (2 NUMA)
- Reduction in LOC: 5x

GPU can hide latency of fine-grained remote operations within a NUMA (GPU complex), but not across NIC/interconnect.

Important:

- **Packing, reuse and overlapping required**
- **Implicit Caching and Buffering**

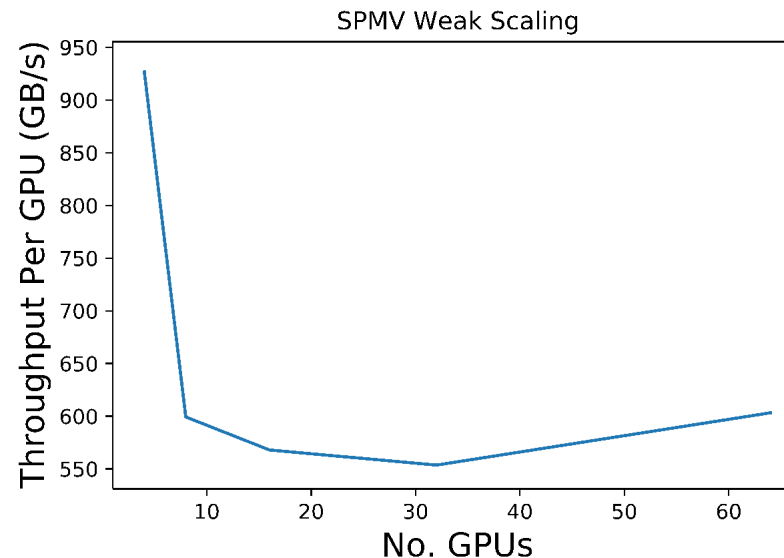
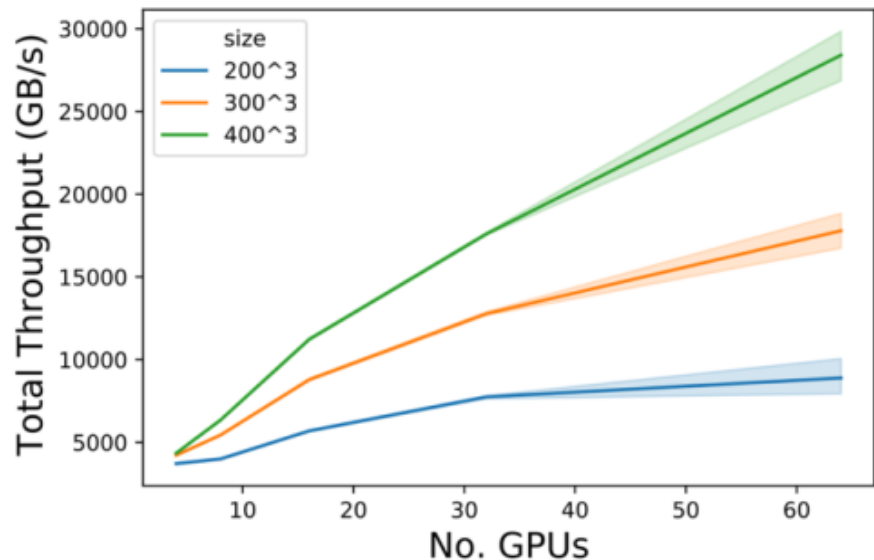
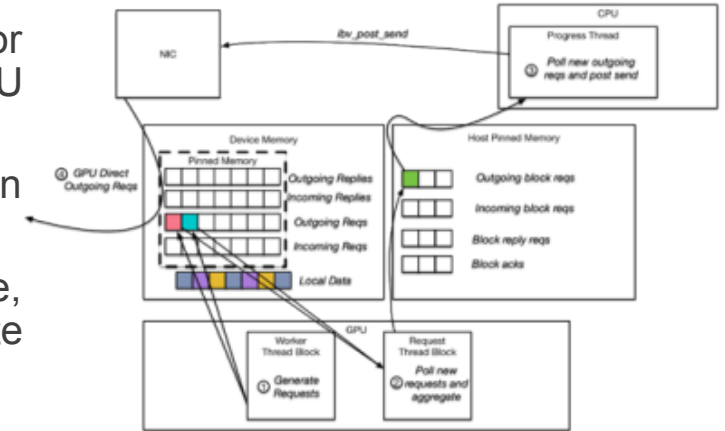




Caching and Buffering

RDMA Scatter-Gather Engine

- Problem: All threads participating in remote memory accesses eliminate potential for latency hiding and thus compromise a fundamental requirement for efficient GPU execution.
- Idea: The RDMA engine redirects accesses to caches. Cache misses result in submitting a request to a op-queue and block caller.
- A subset of resources process the op-queue and assembles ops into aggregate, coarse-grained RDMA accesses thus eliminating fine-grained, high latency remote accesses.
- New back-end called RDMA (not released yet)



**40X improvement
over original
NVSHMEM baseline**

**~60% of peak
memory throughput
for memory-bound
kernels**



Checkout more here

kokkos / kokkos-remote-spaces

< Code Issues 1 Pull requests Actions Projects Wiki Sec

main 4 branches 0 tags Go to file Add file Code

janciesko Merge pull request #15 from kokkos/dev... 526f750 on Sep 16 32 commits

examples	Merge pull request #13 from huanghua1994/devel...	2 months ago
modules	fix shmem build issues and documentation	2 months ago
src	fix shmem build issues and documentation	2 months ago
unit_tests	Additional examples.	2 months ago
CMakeLists.txt	fix shmem build issues and documentation	2 months ago
KokkosRemoteConfig.c...	Additional examples.	2 months ago
LICENSE	Add top level license file	5 months ago
README.md	fix shmem build issues and documentation	2 months ago

<https://github.com/kokkos/kokkos-remote-spaces>

The Kokkos Lecture Series, 2020

Kokkos Remote Spaces: Support for PGAS in Kokkos

How to write a PGAS application with Kokkos.

Learning objectives:

- ▶ How to create global Views.
- ▶ How access global data.
- ▶ Taking a closer look at SPMV (CG).

October 6, 2020 51/67

<https://bit.ly/kokkos-lecture-wiki>
<https://www.youtube.com/watch?v=1J3JQ3d3cRc>