SAND2018-2184C

Leveraging Kokkos/Tpetra for Performance Portability in the Thyra Abstraction Layer

PRESENTED BY

Alex Toth, Roger Pawlowski, Ross Bartlett

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-NA0003525.

Outline

- Performance portability in Trilinos
  - Kokkos programming model
  - Tpetra sparse linear algebra library
- Thyra abstraction layer
  - Usage in abstract numerical algorithms
  - Extensible vector operations using RTOp
  - RTOp and GPUs – problem and current solution
- Numerical Tests
  - Vector operations
  - 3D Poisson problem
- Conclusions and Future Plans

Performance Portability in Trilinos

- Performance portability – write code once and run well on several different backends
- Numerical software may be required to support several architectures
  - Multicore CPUs
  - Manycore CPUs
  - GPUs
- Performance portability goals:
  - Code compiles and runs on multiple architectures
  - Obtains performant memory access patterns
  - Utilization of architecture-specific features
- The Kokkos and Tpetra packages are the foundation of the performance portability effort in Trilinos

Kokkos

- Kokkos is a programming model for thread parallelism
  - Abstracts details of memory and execution spaces away from computational work units (functors or lambdas)
- Important features
  - Kokkos::View – Multidimensional array with architecture-dependent layout
  - Kokkos::parallel_{for, reduce, scan} – Maps work units to threads in an architecture-aware manner
- Exposes multiple levels of parallelism
  - Flat
  - Thread teams
  - Task based
- Current computational back-ends – Serial, OpenMP, Pthreads, Cuda
- See MS36 & MS48 "Kokkos Enables Productivity and Performance Portability" Thursday afternoon for more information on Kokkos

Tpetra

- Tpetra is a sparse linear algebra library which implements:
  - Sparse graphs, sparse matrices, multivectors, vectors
  - MPI communication and redistribution patterns
  - MPI + X (threads) matrix/vector operations
- Vector and matrix types are templated on four parameter types:
  - Scalar type
  - Local ordinal type
  - Global ordinal type
  - Kokkos device type – provides details about memory and execution spaces
- Current issue when executing on GPU – requires Cuda Unified Virtual Memory (UVM)
  - UVM is managed memory which has a single pointer which can be accessed either from the host or device
  - Concurrent access from host and device is not allowed, so Tpetra requires CUDA_LAUNCH_BLOCKING enabled.

Thyra

- Thyra is a a collection of interfaces to support interoperability between abstract numerical algorithms (ANA), linear algebra libraries (LAL), and applications
- Fundamental Operator/Vector interfaces
  - VectorSpaceBase
  - LinearOpBase
  - MultiVectorBase
  - VectorBase
- Fundamental Operator Solve Interfaces
  - LinearOpWithSolveFactoryBase
  - LinearOpWithSolveBase
- Fundamental Nonlinear interfaces
  - ModelEvaluator
  - NonlinearSolverBase

Thyra Vector Operations

- Thyra includes an implementation of the operator/vector interfaces for Tpetra
- In order to provide maximum extensibility, Thyra previously performed all element-wise vector operations using RTOp (reduction/transformation operators) objects
- RTOp uses a visitor design pattern to perform element-wise operations of the form:

$$op(a, b, v^0_{a:b} \dots v^{p-1}_{a:b}, z^0_{a:b} \dots z^{q-1}_{a:b}, \beta) \rightarrow z^0_{a:b} \dots z^{q-1}_{a:b}, \beta$$

- A concrete implementation of the Thyra (Multi)VectorBase interface is required to implement a single function "applyOp" which defines how an RTOp object visits the elements of the vector
- An ANA requiring a non-standard operation simply must create an RTOp to perform the operation, and it is automatically supported by the vector

Thyra ANA-LAL Interface

ANAs in Trilinos with Thyra implementations (e.g. Krylov solvers in Belos, Newton-based solvers in NOX) operate on linear algebra objects using a collection of non-member functions, which previously called "applyOp" with the appropriate RTOp, to perform vector operations:

```
template<Scalar>
void Belos::MultiVecTraits<Scalar, Thyra::MultiVectorBase<Scalar> >::
MvScale( Thyra::MultiVectorBase<Scalar>& mv, const Scalar alpha )
    { Thyra::scale(alpha, Teuchos::inoutArg(mv)); }


NOX::Abstract::Vector&
NOX::Thyra::Vector::scale(double alpha)
{
  Thyra::scale(alpha, Teuchos::inoutArg(*thyraVec));
  return *this;
}
```

Anatomy of an RTOp

Concrete RTOp objects inherit from the abstract base class RTOpPack::RTOpT, and must implement the virtual function "apply_op_impl":

```
template<class Scalar>
void ConcreteRTOp<Scalar>:: apply_op_impl(
    const Teuchos::ArrayView<const RTOpPack::ConstSubVectorView<Scalar>>& in_vecs,
    const Teuchos::ArrayView<const RTOpPack::SubVectorView<Scalar>>& out_vecs,
    const Teuchos::Ptr<RTOpPack::ReductTarget>& reduct_obj_inout
{
  …
  for (int i = 0; i < length; i += stride) {
    applyKernel(in_vecs[0][i], …, out_vecs[0][i], …, reduct_obj_inout);
  }
…
}
```

RTOp and GPUs

- With this design, RTOp vector operations cannot run on GPU
  - Dynamic dispatch for "apply_op_impl" requires it to run on host
  - Dereferencing the (Const)SubVectorView arguments requires data accessible from host
  - Possible when using UVM, but requires migration of data between host and device
- While some ANAs may require exotic vector operations, many need only a small set of standard operations
- Solution: define a set of fundamental vector operations and add virtual interface functions to (Multi)VectorBase
  - Allows concrete Thyra implementations (namely the Tpetra implementation) to avoid RTOp calls for this set of operations
  - RTOp still supported for other vector operations, but not performance portable and require UVM when running on GPU

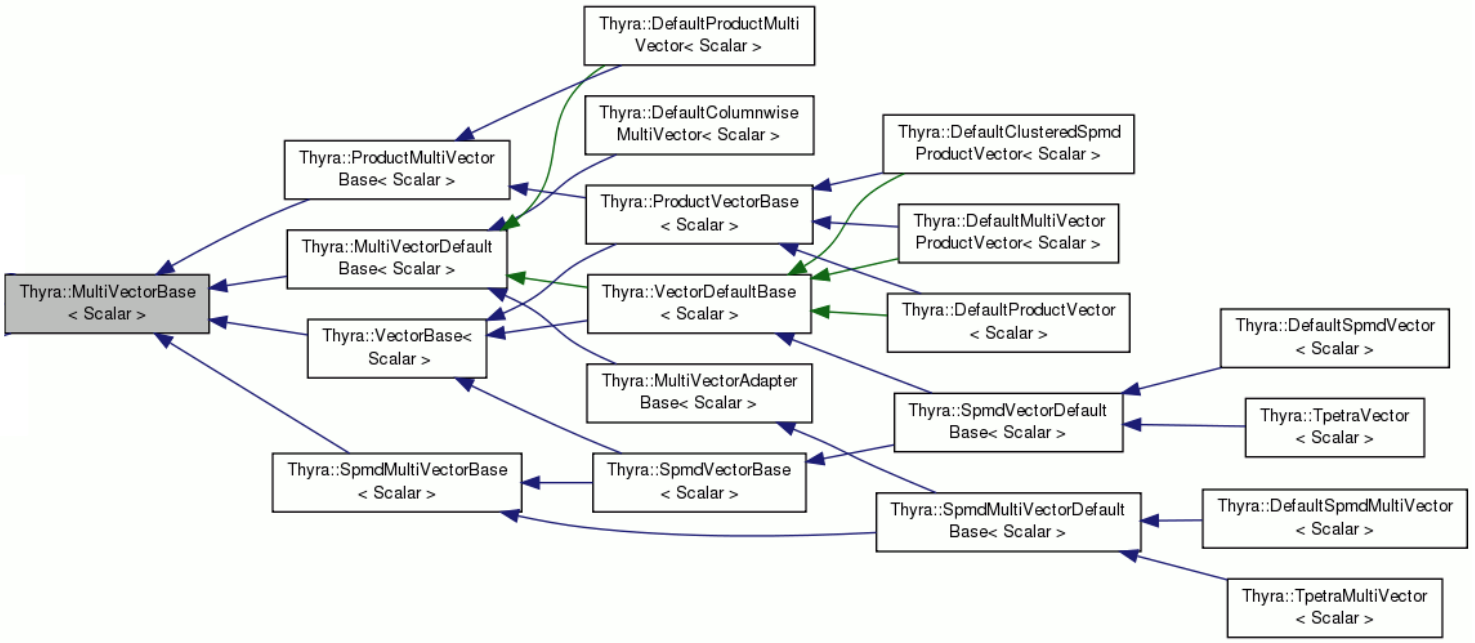
New Interface Functions for Thyra Vector Operations

- assign (Scalar)
- assign (MultiVector)
- scale
- linear_combination
- dots
- norms_1
- norms_2
- norms_inf
- randomize
- abs
- ele_wise_scale
- reciprocal
- dot
- norm_1
- norm_2
- norm_2 (weighted)
- norm_inf

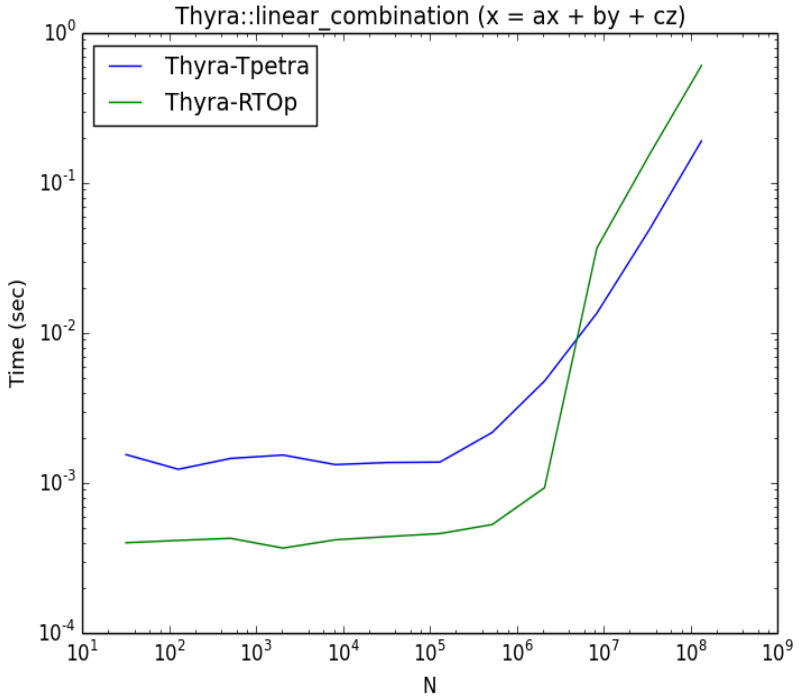
Adding New Interface Functions to Thyra

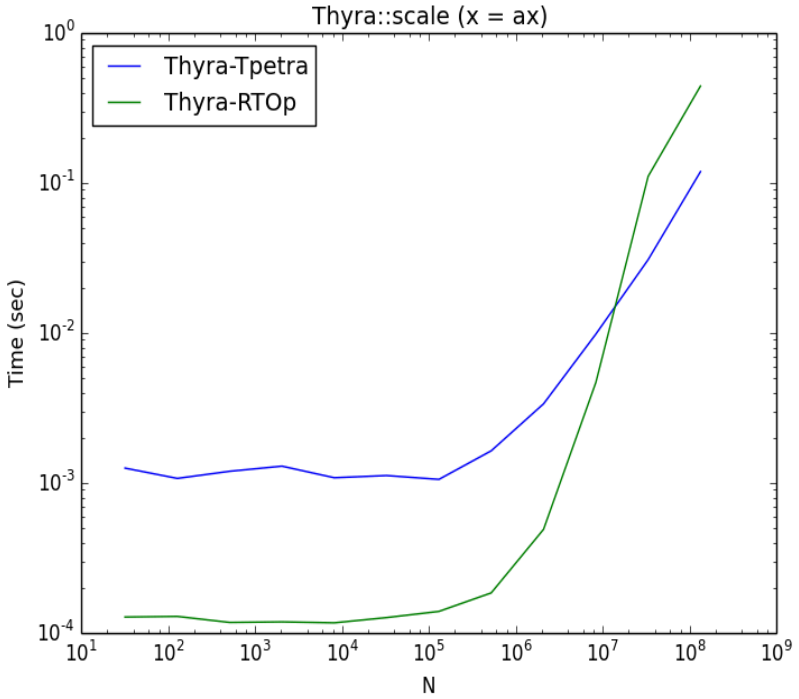
A new vector operation is added to Thyra through the following process:

1. Add new interface function to (Multi)VectorBase
2. Redirect nonmember vector operation to call new interface function
3. Provide a default implementation using RTOps in (Multi)VectorDefaultBase
4. Override where appropriate in the MultiVectorBase inheritance hierarchy



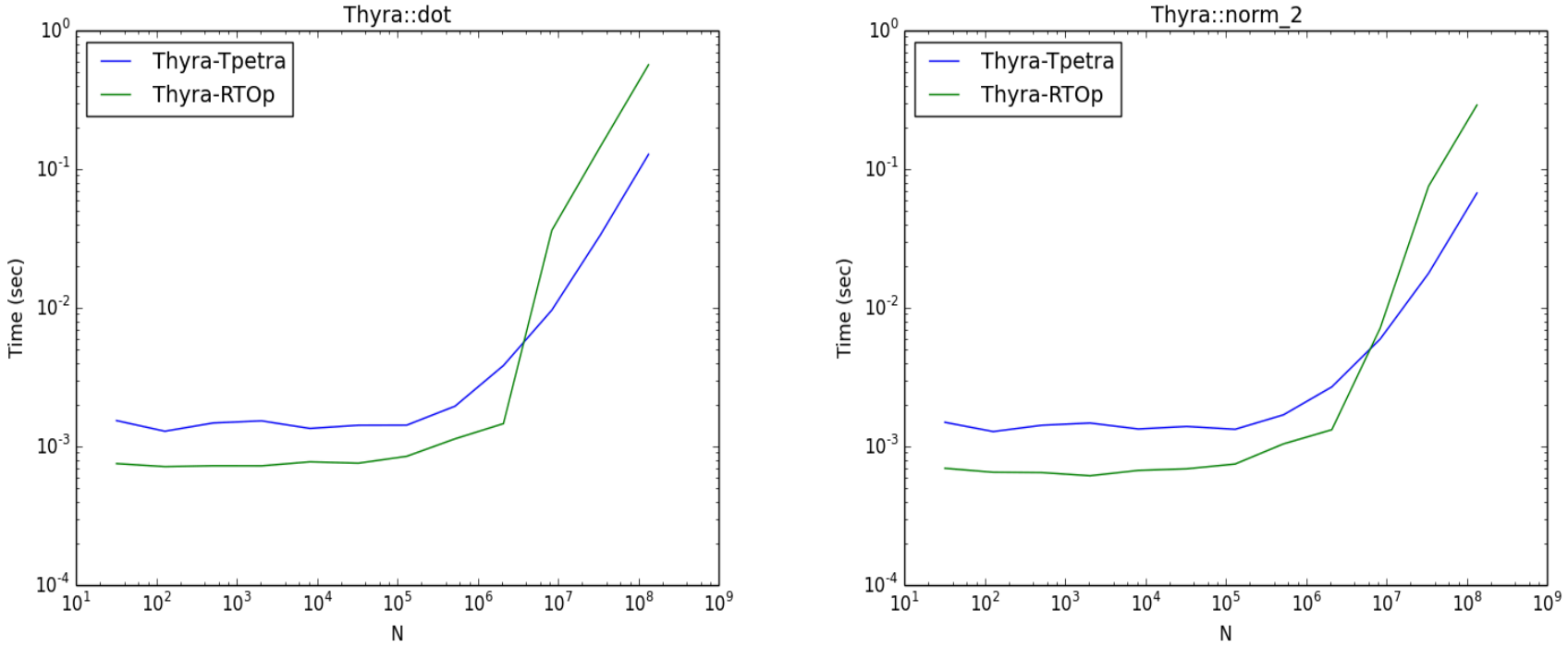
Transformation Timings





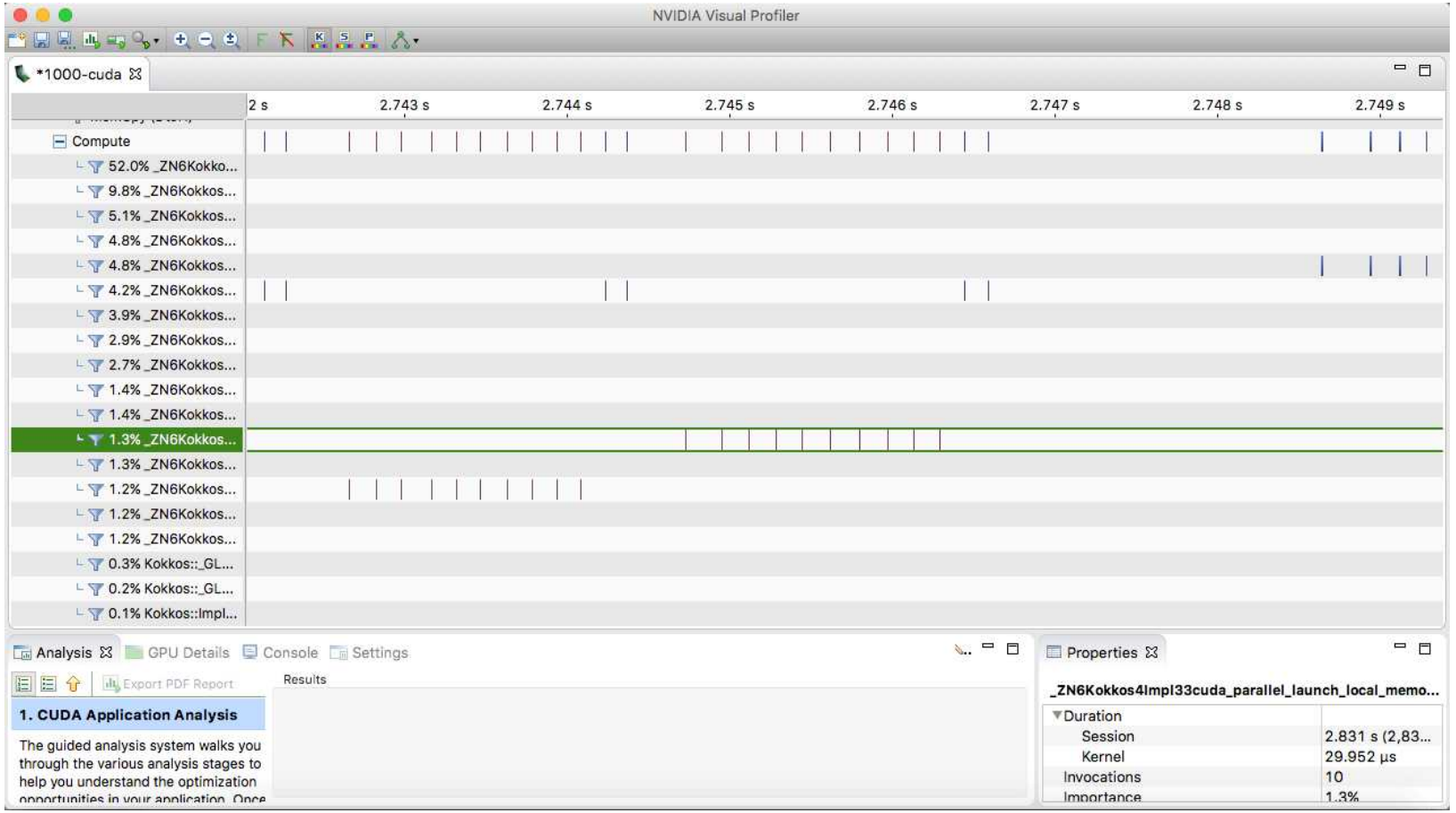
Transformation operation timings (10 call average).
Tpetra operations performed on GPU, RTOp on CPU

Reduction Timings



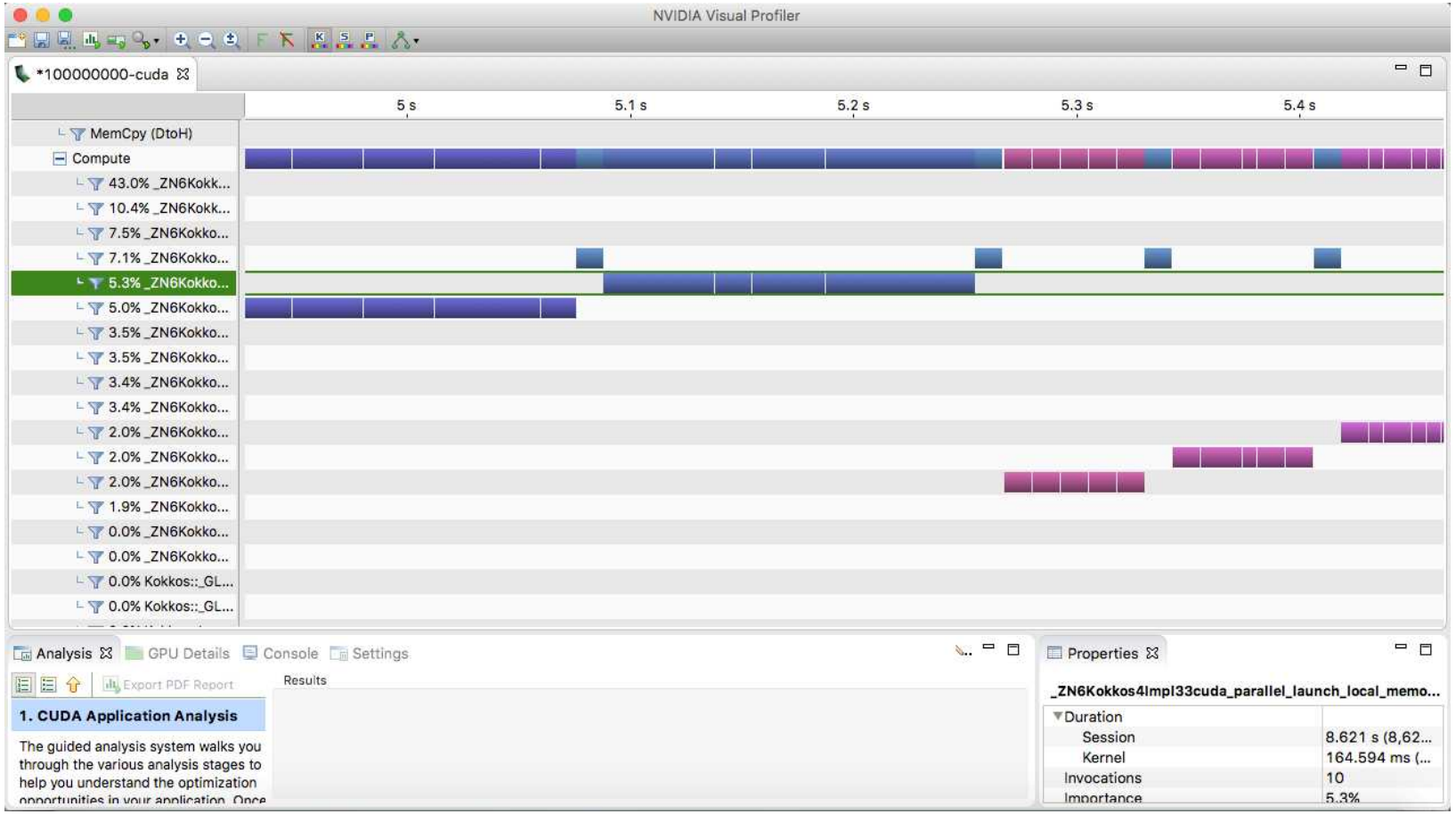
Reduction operation timings (10 call average). Tpetra operations performed on GPU, RTOp on CPU

Visual Profiler Results



Vector Size = 1000

Visual Profiler Results

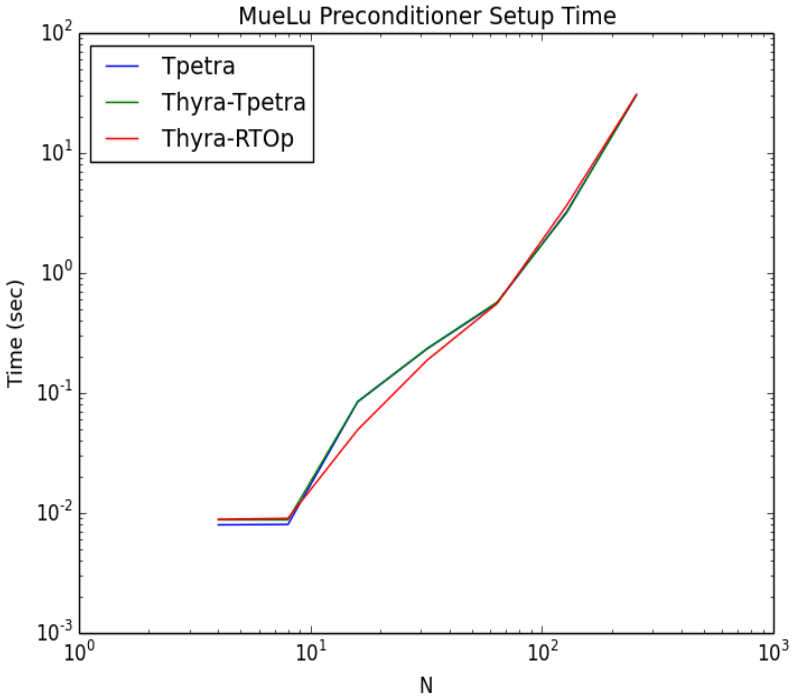


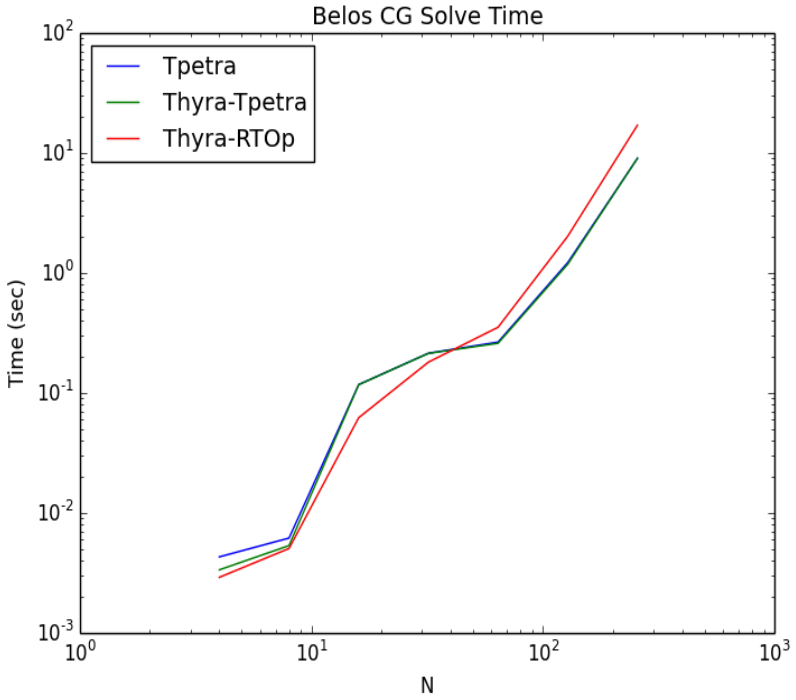
Vector Size = 100000000

3D Poisson Equation

- Solving $\Delta u = f$ on $N \times N \times N$ grid with random RHS
- Linear solver – Belos conjugate gradient solver
- Preconditioner – MueLu multigrid preconditioner
  - V-cycle
  - Max coarse size = 1000
  - Chebyshev pre/post-smoother
  - Direct solve on coarsest grid
- Belos includes direct Tpetra implementation, so we compare direct Tpetra against Thyra-wrapped Tpetra with both RTOp and native Tpetra vector operations

Poisson Results





Preconditioner setup and linear system solve times.
Tpetra operations on GPU, RTOp on CPU

MueLu Setup UVM Profiling

	N =	4	8	16	32	64	128	256
Tpetra	HtoD Size	4.80KB	5.20KB	2.21MB	5.83MB	18.6MB	121MB	927MB
	HtoD Time	3.79us	3.97us	546us	1.27ms	3.11ms	19.1ms	139ms
	DtoH Size	131KB	131KB	12.1MB	43.4MB	177MB	1.28GB	9.98GB
	DtoH Time	29.2us	29.4us	2.43ms	8.18ms	29.9ms	206ms	1.60s
Thyra-Tpetra	HtoD Size	8.80KB	8.80KB	2.21MB	5.81MB	18.4MB	120MB	915MB
	HtoD Time	7.08us	7.07us	535us	1.29ms	3.22ms	18.7ms	135ms
	DtoH Size	206KB	206KB	12.0MB	44.4MB	177MB	1.28GB	9.98GB
	DtoH Time	47.2us	45.9us	2.47ms	8.18ms	29.4ms	207ms	1.61s
Thyra-RTOp	HtoD Size	8.80KB	8.80KB	2.21MB	5.81MB	18.4MB	120MB	915MB
	HtoD Time	7.02us	7.34us	531us	1.29ms	3.10ms	18.4ms	134ms
	DtoH Size	206KB	206KB	12.0MB	44.4MB	177MB	1.28GB	9.98GB
	DtoH Time	45.9us	46.2us	2.44ms	8.14ms	30.1ms	207ms	1.62s

Belos CG Solve UVM Profiling

	N =	4	8	16	32	64	128	256
Tpetra	HtoD Size	48.4KB	57.2KB	752KB	1.77MB	1.82MB	3.97MB	4.28MB
	HtoD Time	25.7us	32.4us	487us	1.20ms	1.26ms	2.95ms	3.32ms
	DtoH Size	481KB	688KB	9.14MB	24.3MB	25.9MB	57.7MB	62.1MB
	DtoH Time	105us	151us	2.02ms	5.45ms	5.37ms	12.1ms	12.8ms
Thyra-Tpetra	HtoD Size	48.4KB	57.2KB	752KB	1.77MB	1.82MB	3.97MB	4.28MB
	HtoD Time	27.9us	33.0us	501us	1.23ms	1.32ms	2.93ms	3.74ms
	DtoH Size	550KB	894KB	10.5MB	24.2MB	27.4MB	58.1MB	64.8MB
	DtoH Time	119us	195us	2.29ms	5.29ms	5.87ms	12.1ms	13.5ms
Thyra-RTOp	HtoD Size	57.2KB	83.6KB	2.10MB	12.4MB	85.6MB	812MB	6.88GB
	HtoD Time	30.5us	36.8us	789us	2.87ms	14.0ms	127ms	1.04s
	DtoH Size	619KB	1.31MB	18.8MB	70.9MB	233MB	2.00GB	16.4GB
	DtoH Time	134us	269us	4.04ms	13.2ms	38.9ms	317ms	2.58s
CG Iters		1	1	11	12	12	15	16

Future Plans

Possibilities for future of Thyra vector operations:

- Remain in current state
  - Virtual interface functions for fundamental operations
  - Non-performance portable RTOp support for custom user operations
- Convert RTOp to operate with Kokkos directly
  - Replace (Const)SubVectorView with Kokkos::View utilize Kokkos::parallel_for to apply operations
  - Maintains extensibility by allowing for performance portable custom user operations
  - Less flexible as this adds a requirement of a specific linear algebra library
- Remove RTOp support
  - Significant loss of extensibility and flexibility

Conclusions

- Currently significant focus on performance portability in Trilinos
- Thyra had previously not benefited from this effort
  - RTOp vector operations provide significant benefit in extensibility, but current design is not performance portable
  - RTOp operations for data residing on GPU requires UVM and potentially leads to significant migration of data between host and device, and additionally requires Cuda kernels to be launched synchronously to avoid concurrent access
- By providing a small set of fundamental vector operations in the Thyra (Multi)VectorBase class, many ANAs can completely bypass RTOp calls
- Numerical tests display a significant reduction in UVM data migration as a result of these changes

Acknowledgements

This research was supported by the Exascale Computing Project (17-SC-20-SC), a collaborative effort of two U.S. Department of Energy organizations (Office of Science and the National Nuclear Security Administration) responsible for the planning and preparation of a capable exascale ecosystem, including software, applications, hardware, advanced system engineering, and early testbed platforms, in support of the nations exascale computing imperative.