This paper describes objective technical results and analysis. Any subjective views or opinions that might be expressed in the paper do not necessarily represent the views of the U.S. Department of Energy or the United States Government.

SAND2022-11815C

Sandia National Laboratories

What's new in Tpetra & Data Services?

TRILINOS

Presented by: Chris Siefert, Tpetra Package Lead

EuroTUG 2022, September 13, 2022

U.S. DEPARTMENT OF ENERGY NNSA National Nuclear Security Administration

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.

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.

Outline

- What’s new in Zoltan2?
- What’s new in SEACAS/IOSS/Exodus?
- What’s new in STK?
- What’s new in Tpetra?
- Other new developments!

Note: I will be repeating “new” developments from TUG ’21, more detail on those can be found here: https://trilinos.github.io/trilinos_user-developer_group_meeting_2021.html

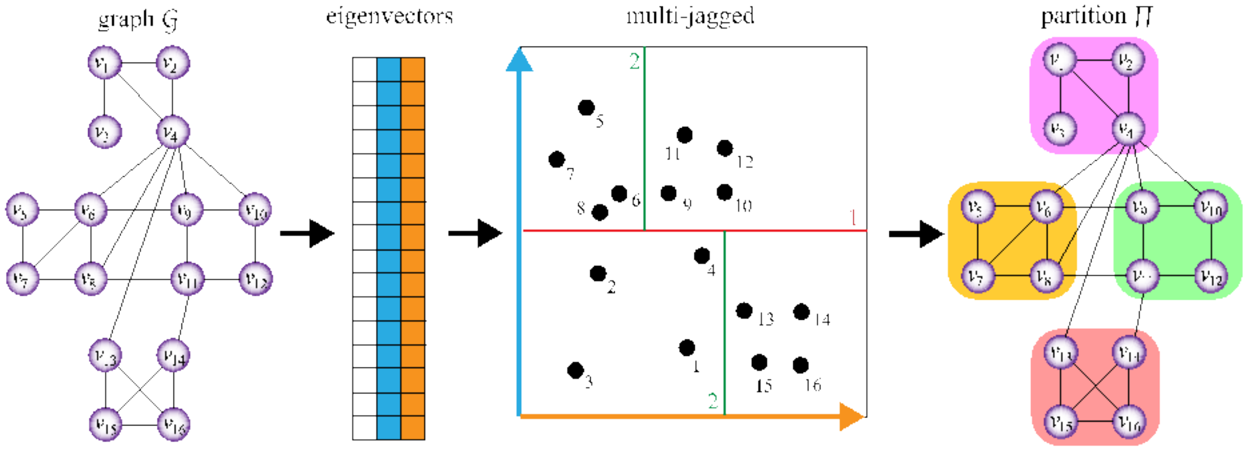
What’s new in Zoltan2? [TUG ‘21 and newer]

- Hybrid distributed/shared memory graph coloring
  - Uses MPI + KokkosKernels coloring
  - Runs on CPU and GPU (Nvidia/CUDA and AMD/HIP)
  - Supports dist-1, dist-2, and partial dist-2 coloring

- Sphynx: New graph partitioner
  - Algorithm based on spectral partitioning
  - Runs on both CPU and GPU (via Kokkos)
  - First multi-GPU graph partitioner!

- In progress:
  - Kokkos-based API for input adapters so users can provide data on either host or device
  - Multilevel graph partitioner for GPU

Sphynx partitioner pipeline:


graph G
eigenvectors
multi-jagged
partition Π
v_1 v_2 v_3 v_4 v_5 v_6 v_7 v_8 v_9 v_{10} v_{11} v_{12} v_{13} v_{14} v_{15} v_{16}
1
2
2
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16

Slide courtesy of Erik Boman, Zoltan2 lead.

What’s new in SEACAS/IOSS/Exodus? [TUG ‘21]

- POC: Greg Sjaardema
- New Features
  - Assemblies – hierarchical groups of blocks/sets/assemblies
  - Blobs -- store arbitrarily-sized objects in an exodus file
  - Entity Attributes -- “provenance” or annotation data on entities and fields
  - Aprepro – Arrays, Exodus integration
  - Exodus.py – python3, improved capabilities, testing
- New Integrations – FAODEL, Catalyst2, ADIOS2, TextMesh
- In progress:
  - Discontinuous Galerkin Fields
  - HDF5 VOL
  - Compression (lossy and lossless)
- Others: Windows, scalability, code quality

What’s new in STK? [TUG ‘21]

- POC: Alan Williams

- GPU: Improving the performance of synchronizing Fields between CPU and GPU memory spaces.
  - Primarily for Sierra SM
- AMD/HIP: stk-mesh unit-tests now build and run on AMD platforms, using ROCM 4.3.
  - Primarily for ExaWind
- STK Balance: improving work-flow and performance of Balance and BalanceM2N coming soon.

What's new in Tpetra?

- A *lot* has changed since EuroTUG 2019!
- The *big* ones are that Tpetra (and thus the derived linear solver stack) now supports…
  - NVIDIA CUDA w/o UVM.
  - AMD GPUs w/ HIP.
  - Intel GPUs w/ SYCL (Warning: Not yet regularly tested.)
- As of Trilinos 13.4 over 27,000 lines of deprecated code/interfaces were removed.
- There are other new features (on-node graph assembly, simultaneous communications, BlockCrs capabiltities, etc.) as well. We'll get to those in time.

Dynamic Profile Removal [TUG ‘19]

- For better portability to GPUs, we have removed the `DynamicProfile` option for matrix/graph assembly.

- You now need at least an upper bound on storage to build the Graph (like the old `StaticProfile` option).

- Off-processor assembly is still supported (and there is some resizing support for off-rank imports).

FECrs[Graph|Matrix|Vector] [TUG '19]

- For folks interested in finite elements, we have a FE-centric assembly layer.

- Much better performance than using off-rank calls to `insertGlobalEntries`.

- Does *not* require ghosted elements
  - Key assumption: If you own an element, you own at least one dof associated with that element.
  - Requires an `ownedRowMap` and an `ownedPlusSharedRowMap` (any dof into which you will be inserting entries. Not quite the column map).

- All indexing can be done locally and both owned and ghost rows are pre-allocated.

WrappedDualView and UVM-free Code [TUG'21]

- UVM = CUDA Vinified Memory (can be addressed both on Host & GPU)
- Tpetra has Kokkos::DualViews of matrix and vector data
- Kokkos::DualView provides the *means* for tracking host/device views.
  - Sync/modify mechanics.
  - Correct use has to be enforced by the user.
- Tpetra::WrappedDualView manages the sync / modify flags between host and device
  - A little like SYCL buffers.
  - Users no longer sync / modify explicitly.
  - Users cannot hold both host and device pointers concurrently.
  - Affects MultiVector, CrsMatrix, CrsGraph, and Block variants.

Example: Vector fill with UVM is straightforward [TUG ‘21]

```
// Without UVM, this code will fail
multivector_t mv(…);
auto mvData =
     mv.getLocalViewHost();



for (j = 0; j < numData; j++)
   mvData(j,0) = rhs(j);


myDeviceFunction(mv);
```

Code worked with UVM

but failed without UVM

Non-UVM requires careful management of host and device views [TUG ‘21]

Without UVM, explicit modify/syncs were needed – messy and error-prone

```
multivector_t mv(…);
auto mvData =
     mv.getLocalViewHost();
mv.clear_sync_state();
mv.modify_host();
for (j = 0; j < numData; j++)
   mvData(j,0) = rhs(j);
mv.sync_device();
myDeviceFunction(mv);
```

Tpetra host/device management issues easier [TUG '21]

Without UVM, explicit modify/syncs were needed – messy and error-prone

```
multivector_t mv(…);
auto mvData =
     mv.getLocalViewHost();
mv.clear_sync_state();
mv.modify_host();
for (j = 0; j < numData; j++)
   mvData(j,0) = rhs(j);
mv.sync_device();
myDeviceFunction(mv);
```

Tpetra now manages the sync/modify state for users

```
multivector_t mv(…);
{ auto mvData =
       mv.getLocalViewHost(
          Tpetra::Access::OverwriteAll);

  for (j = 0; j < numData; j++)
    mvData(j,0) = rhs(j);
}
myDeviceFunction(mv);
```

Key changes for Tpetra::MultiVector users [TUG '21]

1. Capture host and device views in separate scopes
   - Don't hold raw pointers to multivector's data
   - Let views go out of scope as soon as you're done working with them

2. Separate scope for local operations and Trilinos operations on an object
   - Trilinos operations can choose where to access data

3. Indicate intended usage of views
   - ReadOnly, ReadWrite, OverwriteAll

4. Reduce switching between host and device accesses
   - Be aware of data synchronization

Key changes for Tpetra::CrsGraph/CrsMatrix users [TUG '21]

Same as MultiVector

1. Capture host and device views in separate scopes
   - Don't hold raw pointers to data
   - Let views go out of scope as soon as you're done working with them
2. Separate scope for local operations and Trilinos operations on an object
   - Trilinos operations can choose where to access data
3. Indicate intended usage of views
   - ReadOnly, ReadWrite, OverwriteAll
4. Reduce switching between host and device accesses
   - Be aware of data synchronization
5. getLocalMatrix*() and getLocalGraph*() build Kokkos' matrix and graph ON DEMAND now (rather than returning stored data structures); use wisely
6. Functions returning Teuchos::ArrayView of CrsMatrix/CrsGraph data are dangerous and <u>have been removed</u>.
7. Functions returning raw pointers to CrsMatrix/CrsGraph data are dangerous and <u>have been removed</u>.

Indicate intended usage of views [TUG '21]

Tpetra syncs as needed for type of access

- Tpetra::Access::ReadOnly
  - Tpetra syncs if needed
- Tpetra::Access::ReadWrite
  - Tpetra syncs if needed
  - Tpetra marks modified
- Tpetra::Access::OverwriteAll
  - Tpetra syncs only if view is a subview
  - Tpetra marks modified
  - Use only if writing ALL entries of view

```
// Use access tags to indicate intent
{
  auto read_h =
       mv.getLocalViewHost(
            Tpetra::Access::ReadOnly);
  auto readwrite_h =
       mv.getLocalViewHost(
            Tpetra::Access::ReadWrite);
  auto write_h =
       mv.getLocalViewHost(
            Tpetra::Access::OverwriteAll);

}
```

Access tags allow Tpetra to manage sync/modify status for users

MultiVector: Update code to remove old interfaces [TUG '21]

For now, most interfaces remain

- Get an ArrayRCP (1D or 2D):
  - `getData, getDataNonConst`
  - `get1dView, get1dViewNonConst`
  - `get2dView, get2dViewNonConst`
- Get a single column as Vector:
  - `getVector, getVectorNonConst`

Removed before Trilinos 13.4

- `Tpetra::withLocalAccess`
- `Tpetra::for_each`
- `Tpetra::transform`

Removed by Trilinos 13.4

- Accessors without Access tags
  - `getLocalViewHost()`
  - `getLocalViewDevice()`
  - `getLocalView<>()`
  - `getLocalBlock()`
- Sync/modify now handled by MultiVector
  - `mv.sync_host(), mv.sync_device(), mv.sync<>()`
  - `mv.modify_host(), mv.modify_device(), mv.modi()`
  - `mv.clear_sync_state()`

Asynchronous Import/Export [NEW]

- Motivation
  - Import/Export transfer data from one distributed object (`Tpetra::DistObject`) to another
  - Let's say you have many MultiVectors to do import on …
  - What if you want to overlap communication?
    - Launch sends for multiple DistObjects simultaneously
    - Launch sends and do some other computation while you wait
- Synchronous API
  - Do the complete import, don't return until it's finished: `DistObject::doImport`
- New asynchronous API
  - Pack data and kick off sends: `DistObject::beginImport`
  - (Optionally) check if data has arrived and is ready to unpack: `DistObject::transferArrived`
  - Unpack and combine data: `DistObject::endImport`
- Backend improvements mean each DistObject handles communication separately
  - BUT, can still share the same communication plan from the importer (expensive to create)

Lead developer: Timothy Smith

Prototype: On-node graph assembly [NEW]

- For on-node matrix assembly, we've had an interface for quite some time…
    - Grab the Kokkos::SparseCrsMatrix and wok on that directly.

- But how do you assembly a *Graph* on-node?
    - For many apps, host-assembly suffices --- the connectivity never changes.
    - But some apps have Graphs that change over time.

- Brian Kelley has been working on a FEM-centric prototype for graph assembly:

```
RCP<CrsGraph> Tpetra::assembleFEGraph(
    RCP<Map> rowMap,
    View<GO**, Node::memory_space> ownedElements,
    View<GO**, Node::memory_space> ghostElements);
```

- Still in development: Watch for more info at next EuroTUG.

Lead developer: Brian Kelley

Improved BlockCrsMatrix Support [NEW]

- Tpetra::BlockCrsMatrix was designed to support fixed-sized, small, blocks, e.g., 5x5.

- Uses a CrsGraph on *nodes* (groups of dofs) for the blocked problem --- less pointer chasing than CrsGraph for each individual dof.

- New features
  - Transpose operation.
  - Sparse matrix-matrix multiplication.

- Enables blocks-through-the-whole-hierarchy in certain MueLu code-paths.

- Still in development: Should be in Trilinos/develop by end of CY22.

Lead developer: Conrad Clevenger

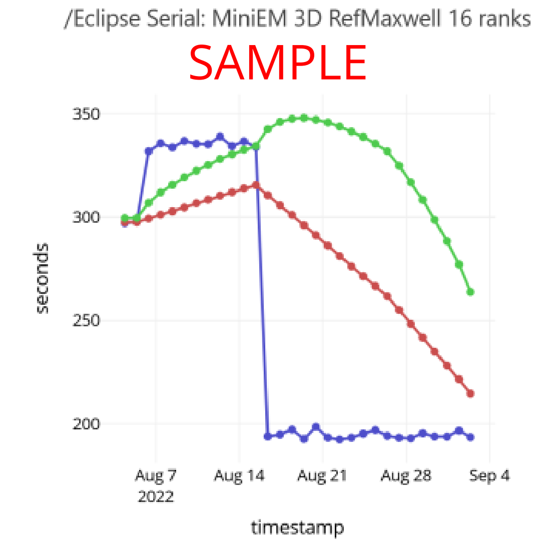
Performance Monitoring [NEW]

- Nightly performance testing on: Intel CPU, ARM CPU, Power9/A100 (NVIDIA), EPYC/MI250 (AMD).

- Performance tests:
  - Tpetra SpMV.
  - Tpetra FE assembly.
  - MiniEM (Maxwell CG+MueLu).
  - Abnormal Energy (GMRES + ILU(3) w/ overlap 2).

- Checked by humans every Tuesday.

- Goals: Work towards automatic changepoint detection, more app-relevant tests.


/Eclipse Serial: MiniEM 3D RefMaxwell 16 ranks

SAMPLE

Lead developers: Brian Kelley / Jonathan Hu

Thank you for your time!

- The last few years in Tpetra have been full of new developments!

- New architectures, UVM-free Cuda, overlapping halo exchanges and more!

- Is there something *you* want to see in Tpetra & Data services? No guarantees, but please feel free to ask (or submit a patch)!