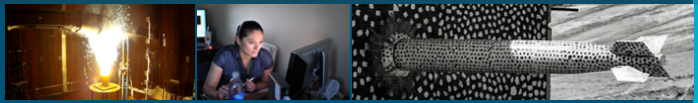
Sandia National Laboratories

Epetra to Tpetra Transition in ALEGRA



Presented by:

Tim Fuller

November 1, 2023

UNCLASSIFIED UNLIMITED RELEASE

Requirements

- Transition Epetra/AztecOO/ML to Tpetra/Belos/MueLu
- Enable 64 bit global indices to enable 2^{32} (or more) unknowns
- Ensure same results using Tpetra stack as Epetra stack

Strategy

- Support both Epetra and Tpetra via runtime switch
- Work with Trilinos developers to develop missing functionality in Tpetra stack
- Develop IO capabilities to compare results between Epetra/Teptra stacks
- Nightly regression testing of both stacks

Epetra to Tpetra transition: Map

```
Epetra_Map* map = new Epetra_Map(
  -1, num_owned, &(indices.front()), 0, Comm
);
```

```
using LO = Tpetra::Map<>::local_ordinal_type;
using GO = Tpetra::Map<>::global_ordinal_type;
using NT = Tpetra::Map<>::node_type;
const Tpetra::global_size_t invalid =
  Teuchos::OrdinalTraits<Tpetra::global_size_t>::invalid();
using map_type = Tpetra::Map<LO, GO, NT>;
Teuchos::RCP map = Teuchos::rcp(
  new map_type(invalid, indices(), 0, comm)
);
Teuchos::RCP map_owned_plus_shared = Teuchos::rcp(
  new map_type(invalid, indices_owned_plus_shared(), 0, comm)
);
```

Epetra to Tpetra transition: FEVector

```
Epetra_FEVector* vec = new Epetra_FEVector(*map);
for (int i=0; i<vec->MyLength(); i++) {
  (*vec)[0][i] = ...;
}
```

```
using mv_type = Tpetra::FEMultiVector<LO, GO, NT, SC>;
Teuchos::RCP<mv_type> vec = Teuchos::rcp(
  new mv_type(map, graph->getImporter(), 1))
);
size_t len = vec->getMap()->getLocalNumElements();
auto data = vec->get1dViewNonConst();
for (size_t i=0;i<local_length;i++) {
  vec[i] = ...;
}
```

Epetra to Tpetra transition: Graph

```
std::vector<int> nnz(num_owned);
...; // fill nnz;
Epetra_FECrsGraph* graph = new Epetra_FECrsGraph(
  Copy, *map, &(nnz.front())
);
for(auto item : items) {
  ...; // create connections a;
  auto ig = item->globalIndex().getGOValue();
  std::vector<GO> x(a.begin(), a.end());
  graph->InsertGlobalIndices(1, &ig, x.size(), &(x.front()));
}
graph->GlobalAssemble();
graph->OptimizeStorage();
```

```
using graph_type = Tpetra::FECrsGraph<LO, GO, NT>;
std::vector<int> nnz(num_owned);
...; // fill nnz;
Teuchos::RCP<graph_type> graph = Teuchos::rcp(
  new graph_type(
    map, map_owned_plus_shared, MAX_NODE_ENTRIES_PER_ROW
  )
);
Tpetra::beginAssembly(*graph);
for(auto item : items) {
  ...; // create connections a;
  GO ig = item->globalIndex().getGOValue();
  Teuchos::Array<GO> x(a.begin(), a.end());
  graph->insertGlobalIndices(ig, x());
}
Tpetra::endAssembly(*graph);
```

Epetra to Tpetra transition: Matrix

```
int num_entries;
double * values;
Epetra_FECrsMatrix* matrix = new Epetra_FECrsMatrix(Copy, *graph);
for(unsigned int i=0; i<owned.size(); i++){
  auto ig = owned[i]->globalIndex().getGOValue();
  matrix->ExtractGlobalRowView(ig, num_entries, values);
  for(int i=0; i<num_entries; i++) {
    values[i] = ...;
  }
}
matrix->OptimizeStorage();
```

```
using matrix_type = Tpetra::FECrsMatrix<LO, GO, NT, SC>;
using local_indices_type =
  typename matrix_type::local_inds_host_view_type;
using values_type =
  typename matrix_type::values_host_view_type;
using nonconst_values_type =
  typename matrix_type::nonconst_values_host_view_type;
Teuchos::RCP<matrix_type> matrix =
  Teuchos::rcp(new matrix_type(graph)));
Tpetra::beginAssembly(*matrix);
local_indices_type indices;
values_type values;
for(unsigned int i=0; i<owned.size();i++){
  GO ig = owned[i]->globalIndex().getGOValue();
  LO lid = matrix->getRowMap()->getLocalElement(ig);
  matrix->getLocalRowView(lid, indices, values);
  auto num_entries = values.size();
  nonconst_values_type updated("updated", values.size());
  for(size_t i=0; i<indices.size(); i++){
    updated[i] = ...;
  }
  matrix->replaceLocalValues(lid, indices, updated);
}
Tpetra::endAssembly(*matrix);
```

Lessons Learned

- Epetra does a lot of work under the hood that users of Tpetra must do themselves
- `Tpetra_FECrsGraph`, `Tpetra_FECrsMatrix`, and `Tpetra_FEMultiVector` had a non-intuitive interface (fixed)
- Belos and MueLu lack(ed) implementations for several solvers and preconditioners implemented in AztecOO/ML that users of ALEGRA depend on