STK: Beyond stk-mesh
(what else is in STK?)

STK Tutorial

Carter Edwards, Todd Coffey, Dan Sunderland, Alan Williams

Review & Approval: SAND2011-0814C
Algorithm-Support (AlgSup)
- Multi-threaded execution of bucket-loop algorithms

Search
- Proximity, mesh independent

Linsys, IO, Rebalance
- Bridges from mesh-data to external capabilities
- Built optionally

Util
- Everything depends on util directly or indirectly

Transfer
- Not yet implemented

Dependency diagram:
- Arrows point towards a module that is used (depended on) by another module.
STK linsys – bridge between mesh-data and linear-systems

Stk_linsys provides helpers for assembling linear-systems
STK IO – Bridge between mesh-data and disk IO

IOSS: Abstract Finite Element mesh interface
• Object-Oriented
• Database-Independent
• ExodusII is primary output format
• Also supports XDMF, heartbeat, history, and pamgen

Stk_io is a Bridge layer that understands the interfaces of stk_mesh and loss.
• Provides functions for moving data to/from stk_io and loss.
• Provides high-level functions for creating a mesh input and/or results output with minimal code.
• Lower-level functions also provided for more control of the data movement.
Recall the gears demo had loops for updating coordinates, etc.

In general, many algorithms on mesh data can be described as:
- create selector for desired parts
- get buckets
- for each bucket:
  - get field-data
  - perform computations

The bucket-loop is often compute-intensive and can benefit from thread-parallel (or GPU) execution.
for (BucketVector::iterator b_itr = selected_node_buckets.begin();
    b_itr != selected_node_buckets.end(); ++b_itr) {
    Bucket & b = **b_itr;

    const BucketArray<CartesianField> old_coord_data(cartesian_coord_field, b);
    BucketArray<CartesianField> displ_data(displacement_field.field_of_state(StateNew), b);
    ...

    for (size_t node_index = 0; node_index < b.size(); ++index) {
        // Compute new and old coordinate data and assign to displacements:

        displ_data(0,index) = new_coord_data[0] - old_coord_data(0,index);
        ...
    }
}
An Algorithm can be any class that has an ‘apply’ method...
(and the apply method contains the body of your bucket-loop)

```cpp
struct My_Algorithm {

    void apply(stk::mesh::Bucket::iterator begin,
                stk::mesh::Bucket::iterator end) const
    {
        size_t num_nodes = std::distance(end - begin);

        const BucketArray<CartesianField> old_coord_data(cartesian_coord_field, begin, end);
        BucketArray<CartesianField> displ_data(displacement_field.field_of_state(StateNew), begin, end);

        for (size_t node_index = 0; node_index < num_nodes; ++node_index) {
            // Compute new and old coordinate data and assign to displacements:
            displ_data(0,node_index) = new_coord_data[0] - old_coord_data(0,node_index);
        }
    }

};
```

Example using Intel Thread Building Blocks (TBB)

```cpp
stk::AlgorithmRunnerInterface*
    alg_runner = stk::algorithm_runner_tbb(num_threads);

stk::mesh::Part& gear_part = ...
stk::mesh::Selector select_nodes = gear_part & meta_data.locally_owned_part();

My_Algorithm alg;

alg_runner->run(select_nodes, ..., bulk_data.buckets(NODE_RANK), alg);
```

The algorithm-runner implementation can launch buckets on different threads.

The goal is for application code (My_Algorithm) to be unaware of threading, as much as possible. (Of course thread-safety issues must still be kept in mind...)
STK_Search, STK_Transfer

• Geometric proximity searches including: OctTree, BIH

• Proximity search doesn’t depend on stk_mesh
  • Users provide lists of bounding boxes or bounding spheres

• Stk_transfer will use stk_search, and will depend on stk_mesh
• …but there will also be stk_mesh – to – non-stk_mesh transfers
• **Stk_percept**
  • Verification tools, including:
    • Postprocessing
    • Error metrics (e.g. norms)
    • Manufactured solutions

• **Stk_adapt**
  • Extend/modify the discretization
    • Refine elements
    • Increase polynomial order
    • Convert topology (e.g. hex to tet)
STK_util - Utilities

STK_util doesn’t depend on any other STK module

STK_util contains several subdirectories of related tools

- **Diag**
  - Timers
  - Ostream manipulators

- **Environment**
  - Error Reporting
  - Output Logging

- **Parallel**
  - Comm utils
  - Distributed-Index

- **Util**
  - String comparisons
  - Array Ops
  - etc
Future Modules

- **Stk_transfer:**
  - Mesh to mesh transfers
  - Including stk-mesh – to – non-stk-mesh transfers

- **Stk_parser:**
  - Input deck parsing
  - Sierra SDDM

- **Stk_coupling:**
  - Code coupling layer
Goals of the Toolkit Team

- Clarity of code
- Use-case driven API & implementation
- Capability/Flexibility/Performance
  - Growing collection of performance tests, use-cases, ...
- Modularity
- Highly unit-tested code
  
<table>
<thead>
<tr>
<th>Path</th>
<th>Coverage</th>
<th>Lines (Total)</th>
</tr>
</thead>
<tbody>
<tr>
<td>stk_mesh stk mesh base</td>
<td>90.1 %</td>
<td>3272 / 3631</td>
</tr>
<tr>
<td>stk_mesh stk mesh baseImpl</td>
<td>93.8 %</td>
<td>1099 / 1172</td>
</tr>
</tbody>
</table>

- Threading/GPU support
- Facilitate external collaborations with Sierra
- Agile development
Current Users

- Sierra (suite of finite-element analysis applications), SNL
- Cubit (Mesh generation)
  Steve Owen/Matt Staten, SNL
- Albany (Prototyping rapid PDE-application development)
  Andy Salinger, SNL
- Adaptive Mesh Refinement
  Ulrich Küttler, Technische Universität München
- Superconductivity simulations
  Nico Schloemer, University of Antwerp
- Fracture mechanics
  Joe Bishop, Fracture LDRD, SNL
- Charon (planned for FY11), SNL
STK Capability in Albany

Exodus → Pamgen → STK_IO → STK Mesh → STK_Rebalance

Hand-Coded:  

Cubit  

Abstract Discretization
Coordinates
Fields
Connectivity
Vector Maps
- Solution (DOF)
- Coordinate
- Node
Jacobian Graph
Node Sets

output
Thank you!

- For more information:
  http://trilinos.sandia.gov/packages/stk/
stk-users@software.sandia.gov email list

- We want more feedback!
  - Good clean clear APIs and code can’t be achieved without user feedback!

- Thanks for attending this tutorial!

- Developers:
  Dave Baur, Todd Coffey, Carter Edwards, Jim Foucar, Russell Hooper, James Overfelt, Greg Sjaardema, Dan Sunderland, Alan Williams