SAND 2014-18329 C

Implementation of Contact Global Search using Kokkos

Glen Hansen, Samuel P. Mish and Patrick G. Xavier

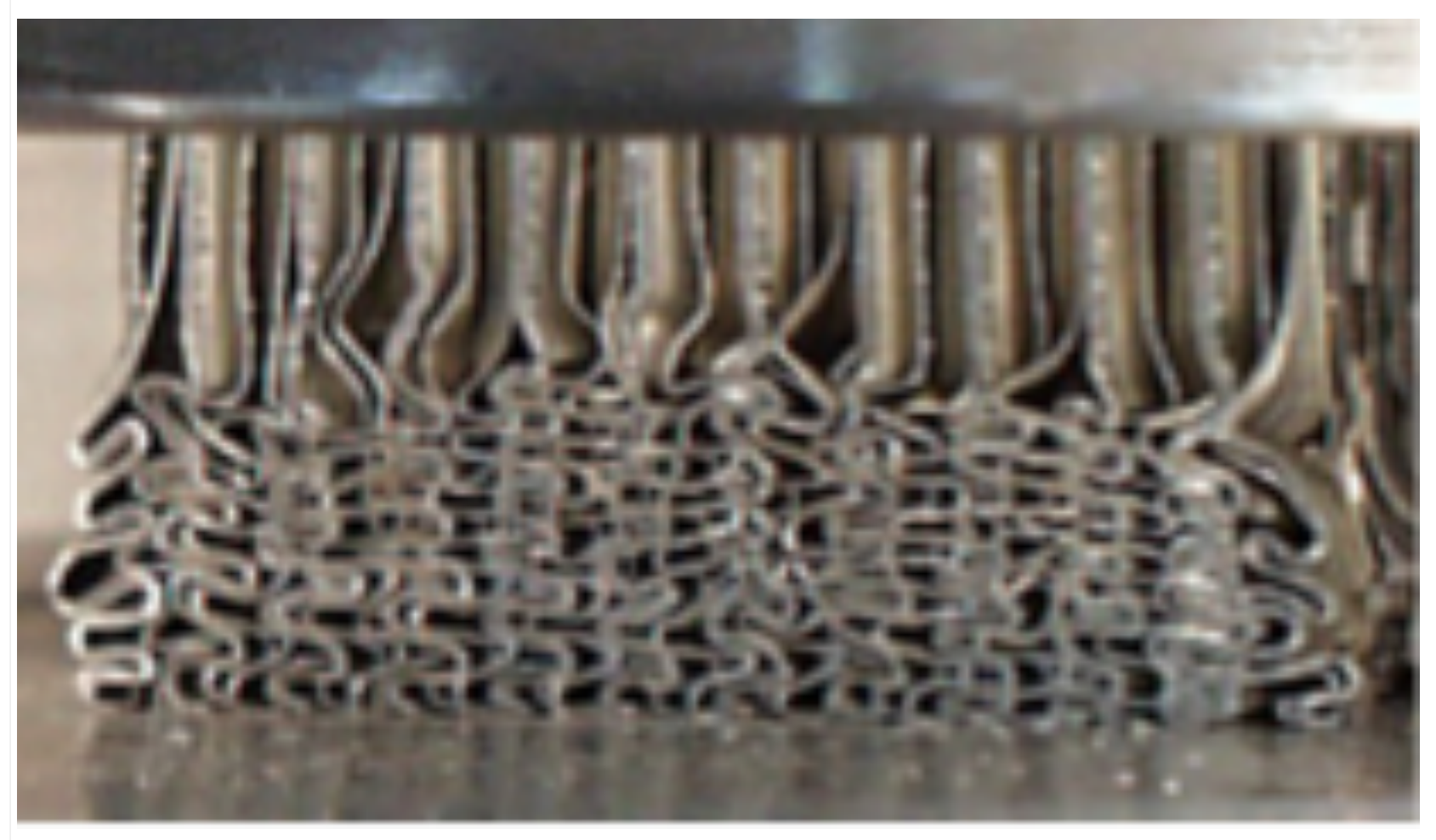
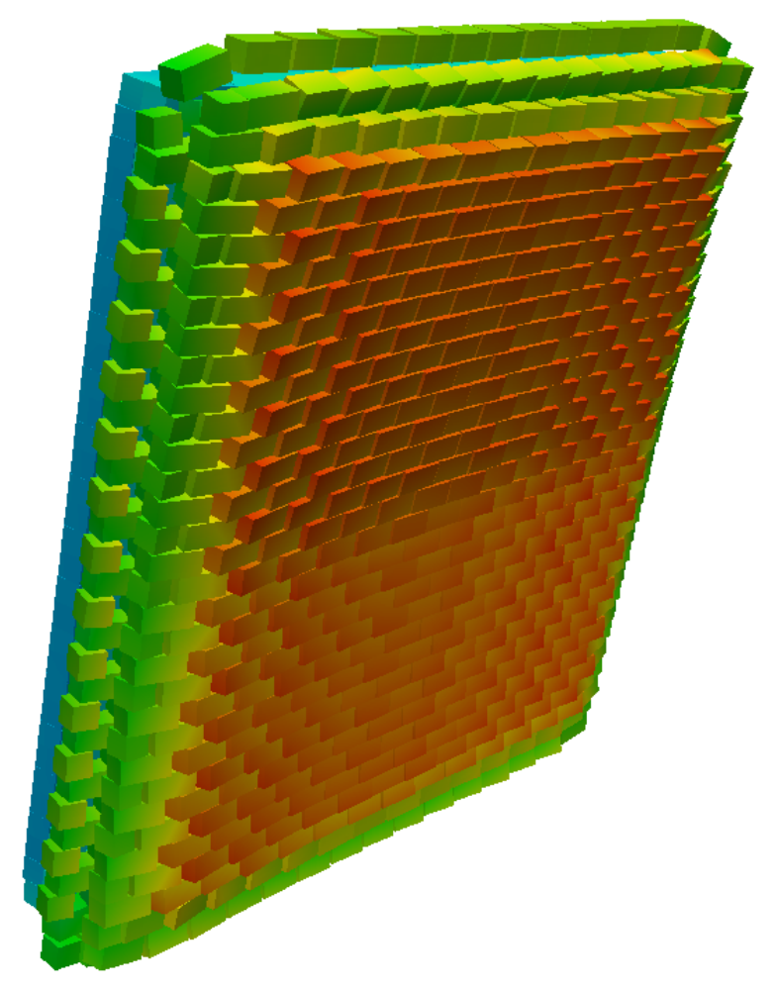
Computational Multiphysics & Simulation Modeling Sciences Departments, Sandia National Laboratories

Overview

We focus on how one might pursue reworking an MPI-parallel code to achieve high performance on MPI+X computer architectures. We present an approach that was applied to contact simulation, specifically considering the expensive global contact search operation. Using Sandia's Algorithms for Modeling Contact in a Multi-physics Environment (ACME) library as the reference implementation, we developed a new approach for global search that employed a manycore search algorithm based on a Morton-code linearized Bounding Volume Hierarchy (BVH), developed by NVidia for use on GPU co-processors.

Modeling Contact with Parallel Computation

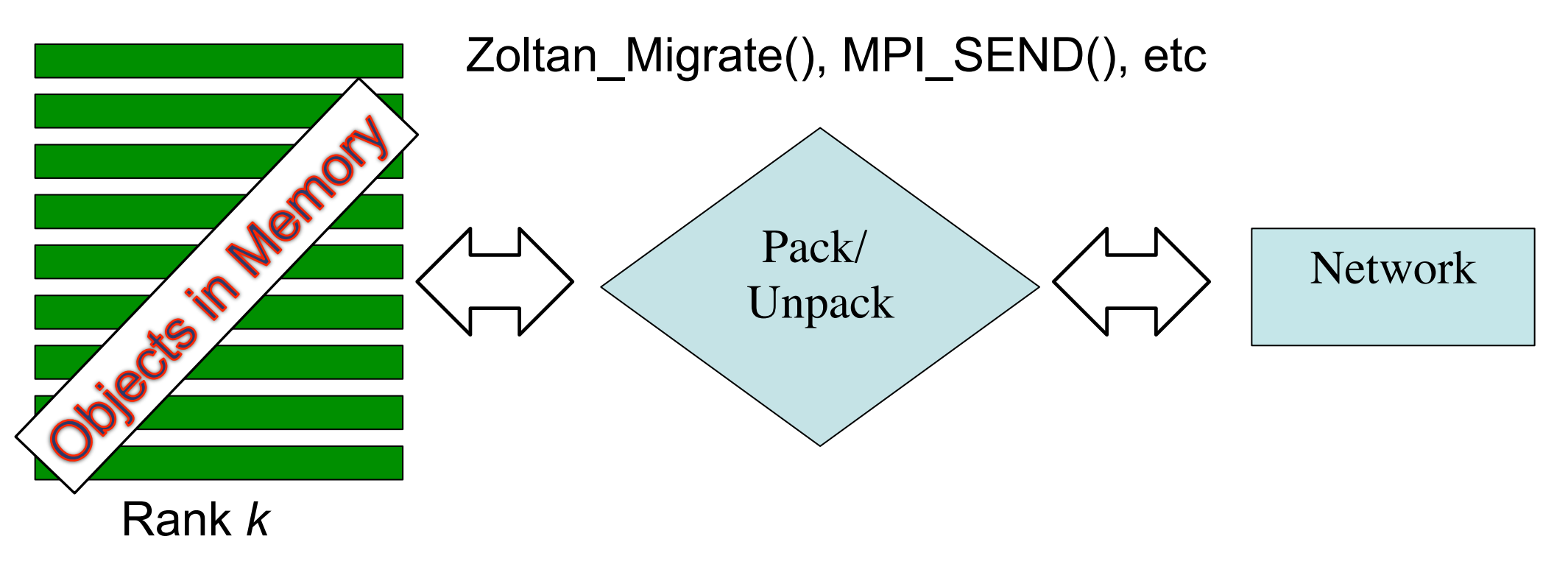
The accurate and efficient modeling of the behavior of contacting material surfaces and bodies during a transient computational mechanics simulation is one of the more time consuming activities in a typical engineering analysis calculation. Within a general calculation, one is often challenged with the need to analyze the behavior of multiple bodies being deformed or in relative motion with respect to each other, such as modeling individual interactions within a set of billiard balls, as well as problems where only a single external surface is of interest but the deformation of the surface is such that self-contact occurs.



Self-contact example (L) and the “Brick Wall” test problem (R).

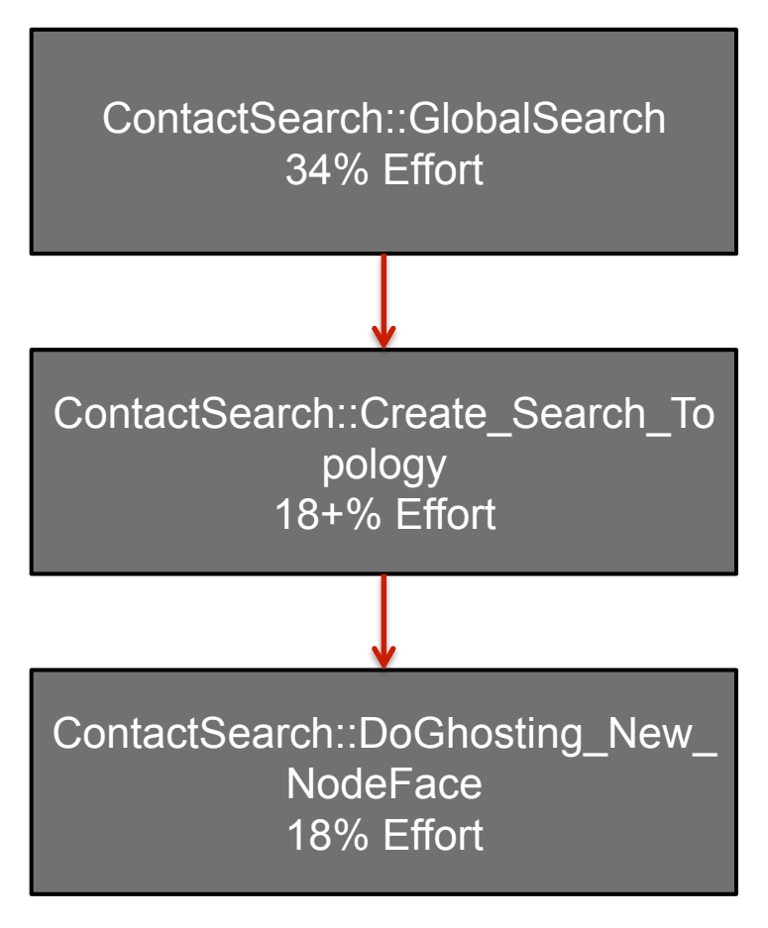
ACME is a mature, widely used multibody contact library developed by Sandia National Laboratories. The library implements various algorithms for:

- Contact search: find potential interactions between body surfaces
  - Global search: calculate relationships between sets of surface entities
  - Local search: determine possibly intersecting pairs of discrete entities
- Contact enforcement: calculate interaction forces between entities to prevent penetration or other violations of surface integrity.

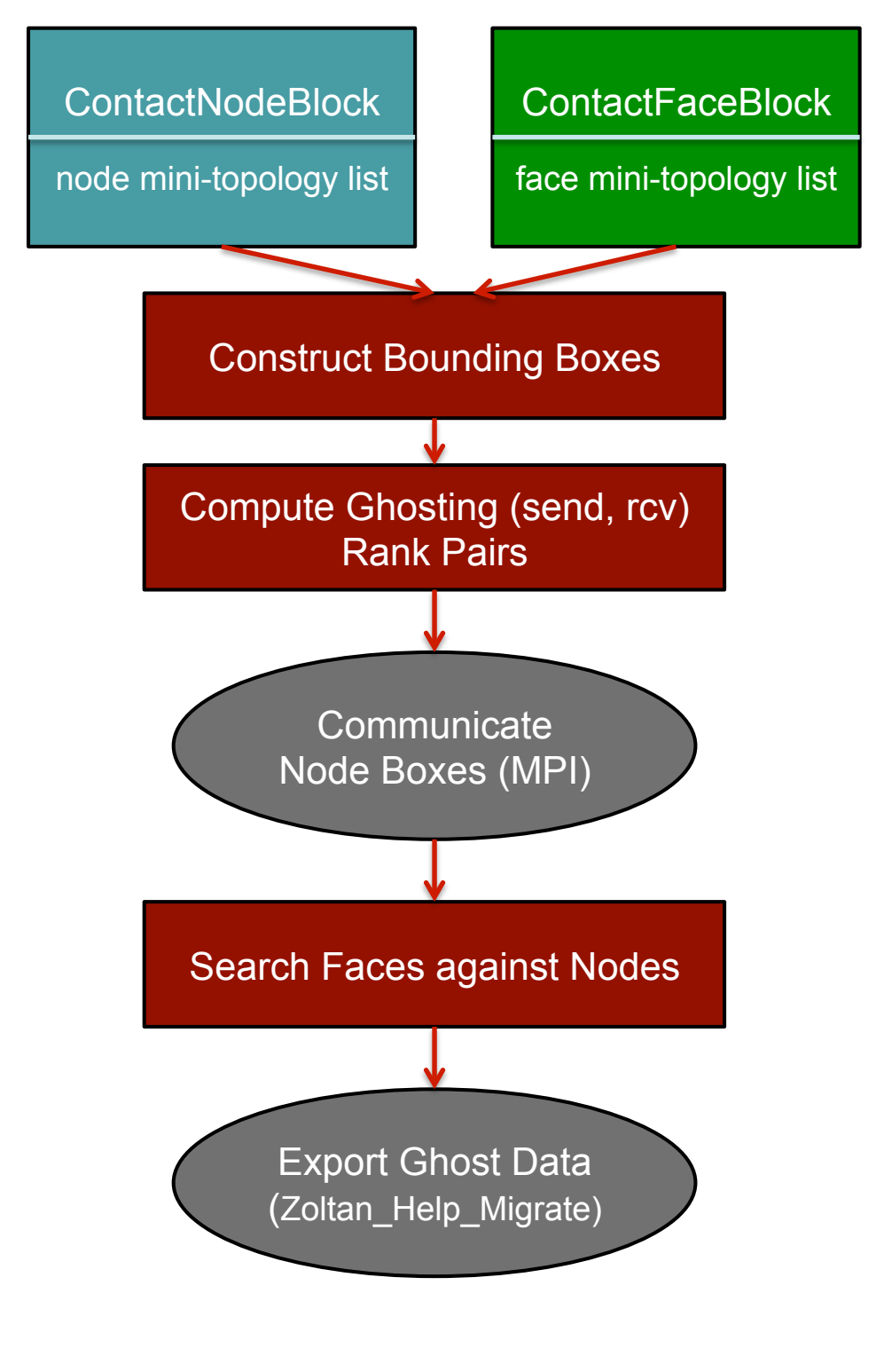
The purpose of the ACME library is to develop the mathematical terms and constraints used to augment the equilibrium equations that describe the forces that govern bodies in contact. ACME contains approximately 90,000 lines of C++ code, is an MPI-parallel code, and has a test suite that was used to ensure the functionality and relevance of the final result.



Profiling. Running an ALEGRA+ACME test program, we identified the operations that were executed each timestep in a simulation of the Brick Wall problem. Profiling showed that the global search and ghosting operations needed to implement it dominated the contact workload.


ContactSearch::GlobalSearch 34% Effort
ContactSearch::Create_Search_Topology 18+% Effort
ContactSearch::DoGhosting_New_NodeFace 18% Effort

Ghosting Algorithm


ContactNodeBlock
node mini-topology list
ContactFaceBlock
face mini-topology list
Construct Bounding Boxes
Compute Ghosting (send, rcv) Rank Pairs
Communicate Node Boxes (MPI)
Search Faces against Nodes
Export Ghost Data (Zoltan_Help_Migrate)

Re-Engineering for MPI+X using Kokkos

Kokkos Features

- Collection of templated C++ components designed to address code portability across multiple programming, memory, execution, and hardware models
- Includes parallel for, scan, and reduce patterns implemented using functors
- Containers: *View* (references to multidimensional arrays), *vector*, *unordered map*, *CRS graph*
- A *View* provides a default data layout according to the memory model; it can be overridden
- A *DualView* provides convenient synchronization of data between the *host* (traditional CPU) and the *device* (e.g., GPU) copies of multidimensional arrays.
- Template specialization enables platform-specific implementations, e.g., for optimization.

[For more, see *www.vecpar.org/slides/**Kokkos**-overview-GTC2014.pptx*.]

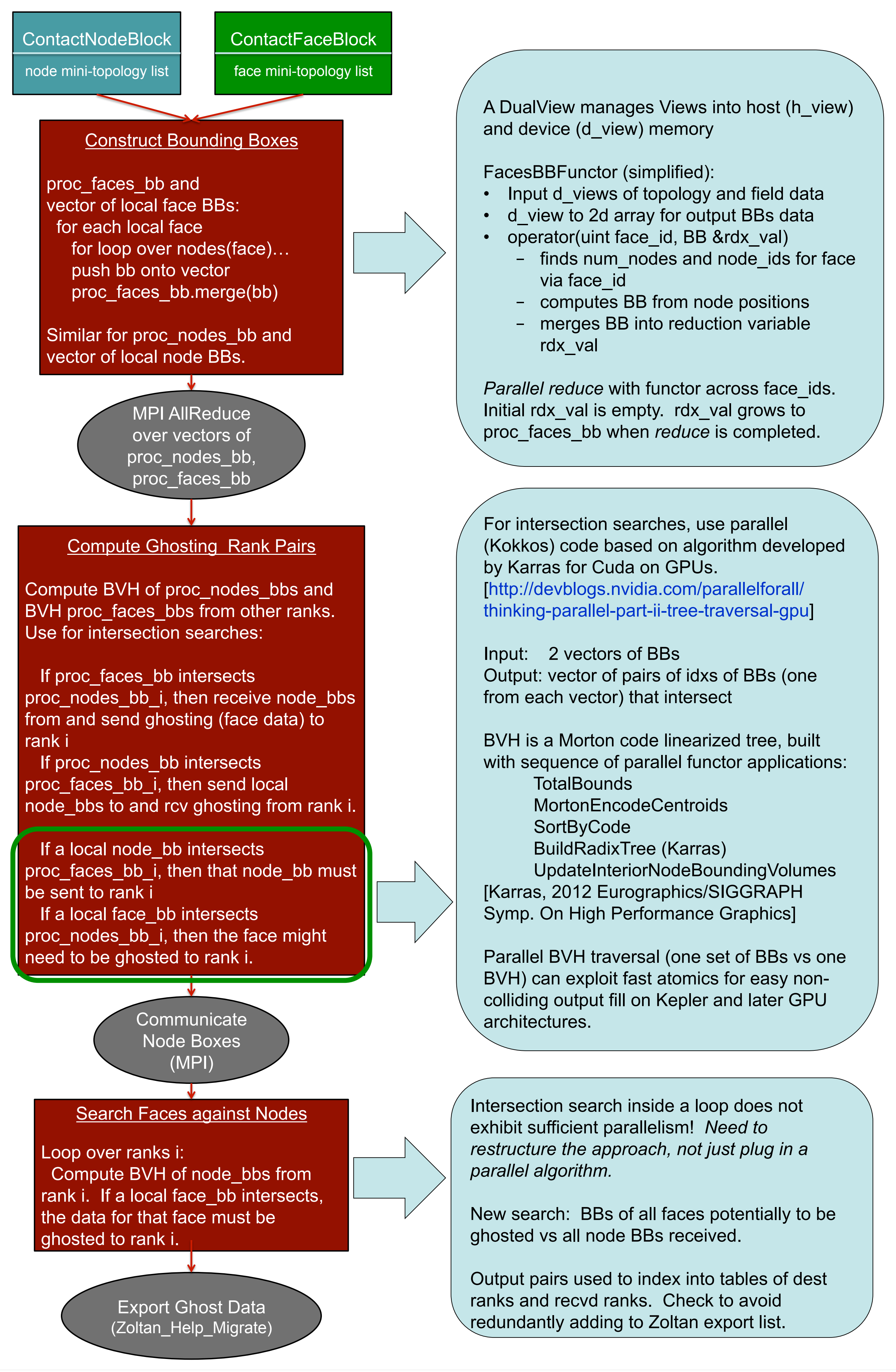
Kokkos::DualViews Our goal is re-engineering of the code for MPI+X platforms; we converted the code to store data in *DualViews* instead of the object-oriented pointer-based data structures used in the original code. These modifications were performed everywhere we needed to transfer data between the host and device. Data types included:

- Field data, such as POSITION (displacements, etc)
- Topology data, such as the number of nodes of a face or the index of where the set of node identifiers that describe a particular face begins in an array.

ACME MiniContact Project “+X” Targets (today)

- Cuda: obtained results on “Curie”, a Cray XK7 w/ Nvidia K20x GPUs, using one GPU per MPI rank
- Pthreads: on conventional multicore CPUs, single and multiple socket machines
- OpenMP

Re-Engineering Contact Entity Ghosting



Construct Bounding Boxes

```
proc_faces_bb and
vector of local face BBs:
  for each local face
     for loop over nodes(face)…
     push bb onto vector
     proc_faces_bb.merge(bb)

Similar for proc_nodes_bb and
vector of local node BBs.
```

A DualView manages Views into host (h_view) and device (d_view) memory

FacesBBFunctor (simplified):

- Input d_views of topology and field data
- d_view to 2d array for output BBs data
- operator(uint face_id, BB &rdx_val)
  - finds num_nodes and node_ids for face via face_id
  - computes BB from node positions
  - merges BB into reduction variable rdx_val

Parallel reduce with functor across face_ids. Initial rdx_val is empty. rdx_val grows to proc_faces_bb when *reduce* is completed.

MPI AllReduce over vectors of proc_nodes_bb, proc_faces_bb

Compute Ghosting Rank Pairs

Compute BVH of proc_nodes_bbs and BVH proc_faces_bbs from other ranks. Use for intersection searches:

If proc_faces_bb intersects proc_nodes_bb_i, then receive node_bbs from and send ghosting (face data) to rank i

If proc_nodes_bb intersects proc_faces_bb_i, then send local node_bbs to and rcv ghosting from rank i.

If a local node_bb intersects proc_faces_bb_i, then that node_bb must be sent to rank i

If a local face_bb intersects proc_nodes_bb_i, then the face might need to be ghosted to rank i.

For intersection searches, use parallel (Kokkos) code based on algorithm developed by Karras for Cuda on GPUs. [http://devblogs.nvidia.com/parallelforall/thinking-parallel-part-ii-tree-traversal-gpu]

Input: 2 vectors of BBs
Output: vector of pairs of idxs of BBs (one from each vector) that intersect

BVH is a Morton code linearized tree, built with sequence of parallel functor applications:

```
     TotalBounds
     MortonEncodeCentroids
     SortByCode
     BuildRadixTree (Karras)
     UpdateInteriorNodeBoundingVolumes
```

[Karras, 2012 Eurographics/SIGGRAPH Symp. On High Performance Graphics]

Parallel BVH traversal (one set of BBs vs one BVH) can exploit fast atomics for easy non-colliding output fill on Kepler and later GPU architectures.

Communicate Node Boxes (MPI)

Search Faces against Nodes

```
Loop over ranks i:
  Compute BVH of node_bbs from
  rank i.  If a local face_bb intersects,
  the data for that face must be
  ghosted to rank i.
```

Intersection search inside a loop does not exhibit sufficient parallelism! *Need to restructure the approach, not just plug in a parallel algorithm.*

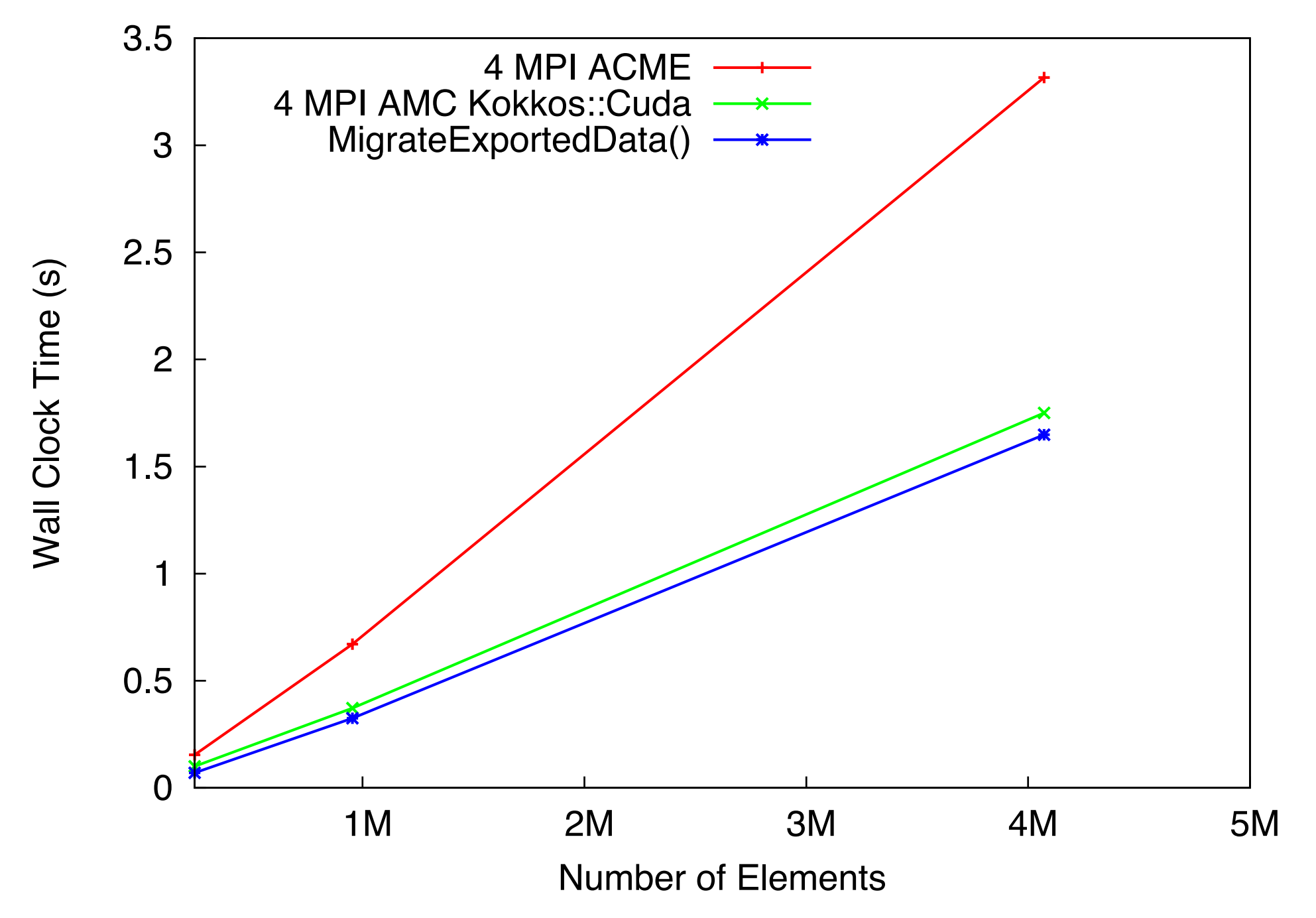
New search: BBs of all faces potentially to be ghosted vs all node BBs received.

Output pairs used to index into tables of dest ranks and recvd ranks. Check to avoid redundantly adding to Zoltan export list.

Export Ghost Data (Zoltan_Help_Migrate)

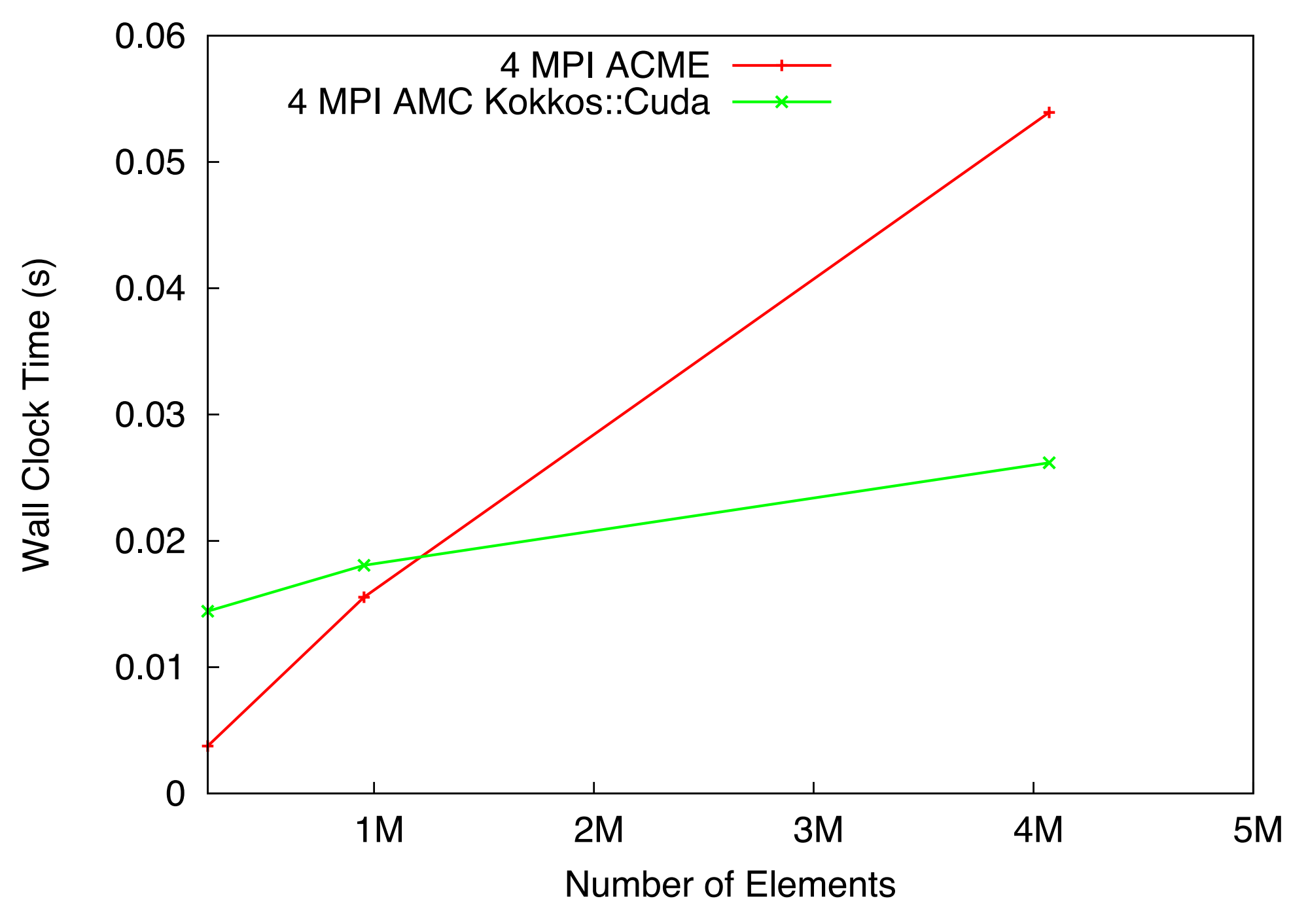
Performance Results

We compared the performance of the new ghosting function against the reference ACME version on Brick Wall test problems employing 243K, 954K, and 4072K elements. We ran the code on 4 MPI ranks on Curie, with one MPI rank per compute node. We used one regular processor core and one GPU per MPI rank. In the graph below, we show the time spent in the ghosting function in the reference version of the code (denoted “4 MPI ACME”) and the new version (“4 MPI AMC Kokkos::Cuda”). We also show the time spent using Zoltan to export the data to be ghosted between ranks once the communication pattern was determined within the global search.



Note that the MPI+X version is considerably faster (14x for the 4M element brick problem) at computing the problem geometry to be ghosted. Further, the migrate operation is an MPI communication process that is common to both approaches. This fixed cost baseline is the blue curve in the above.

Below, we show the scaling of the bounding-box intersection search algorithm. For the MPI+X version, we include the time required to sync data between the host and GPU.



While the scaling of the MPI+X version is superior to the reference, there is a high fixed cost. Examining additional profiling data, we have discovered that 2/3 of this may be traced to our use of a naïve parallel sorting implementation and 64-bit Morton codes. We will address these issues in future work.

Conclusion

To address the challenges and opportunities of MPI+X, we implemented a staged re-engineering approach using ACME; a 90K line library.

- To select what sections of code to address, we first used profiling to identify where to focus. We then looked for the top time consuming sections while considering dependencies and the existing code structure.
- We have completed the rework of the ghosting function for node-face contact models. Changes in the new version include: a) implementation of the parallel Morton code algorithm using Kokkos, and b) calling the parallel search methods using generic DualView interfaces and functors.
- Kokkos has enabled code portability and convenience, with good performance on Cuda.

Future work includes results on the Xeon Phi, re-engineering local contact search, developing new contact enforcement methods, and improving performance of our intersection search implementation.

Acknowledgements:

- We would like to thank Tero Karras of Nvidia for his Cuda code and suggestions.
- Sandia National Laboratories is a multi-program laboratory managed and operated by Sandia Corporation, a wholly owned subsidiary of Lockheed Martin Corporation, for the U.S. Department of Energy’s National Nuclear Security Administration under contract DE-AC04-94AL85000.

Los Alamos NATIONAL LABORATORY EST. 1943

Sandia National Laboratories

National Nuclear Security Administration