SAND2015-xxxx

SAND2015-1628C

Development of a Contact MiniApplication Using Kokkos

Glen Hansen

Patrick Xavier

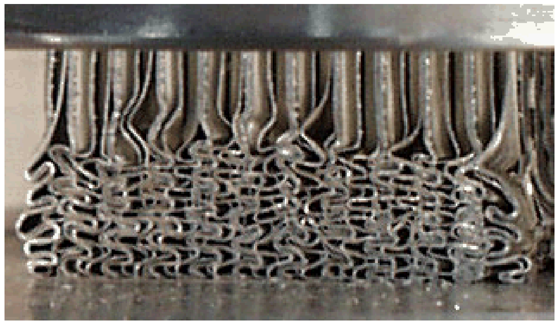
Sam Mish

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.

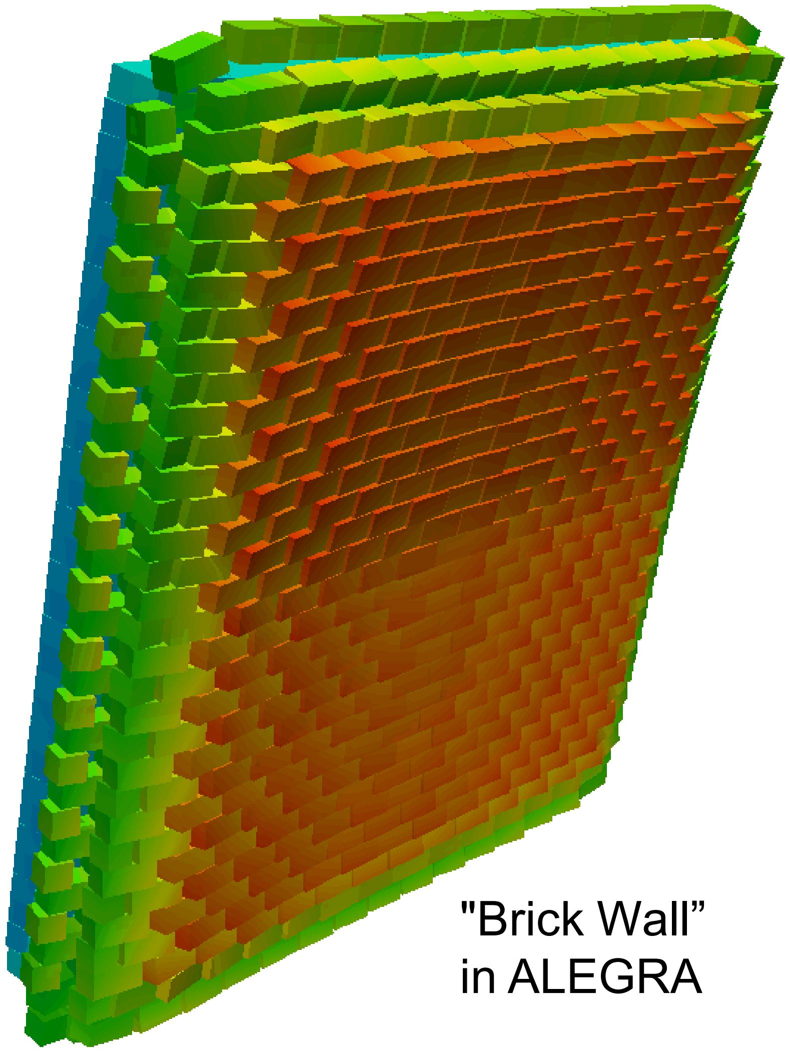
National Nuclear Security Administration

Sandia National Laboratories

Contact mechanics



- **Global search**
  - Repartition to load balance contact operations
  - “Ghosting” is used to ensure potentially-contacting entities are visible on a given processor
- **Local search and imprinting**
  - On each processor, “imprint” contacting entities
- **Enforcement**
  - Assemble terms that describe physics interaction between entities



"Brick Wall” in ALEGRA

Sandia National Laboratories

Primary decomposition-based search

- **Contact calculations are performed using the same decomposition used for the physics solution**
  - The surface entities that are involved in contact are not typically balanced across the available processors
  - Indeed, some processors might not contain any surface entities and will idle during contact operations
- **Use a “ghosting search” to migrate possibly contacting entities so they are visible to each other on the processor**

Requires only a limited amount of entity communication but the load balance is generally poor.

Secondary decomposition-based search

- A secondary decomposition is used to evenly spread the contacting surface entities across the available processors
- Use of either an inertial or RCB decomposition ensures most of the possibly contacting entities are on the same processor. A “ghosting search” might again be used to migrate entities to eliminate further communications during the remaining contact operations

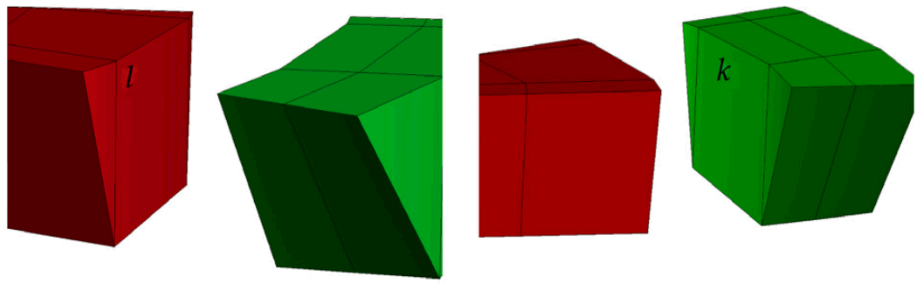
A large amount of communication is needed to evenly distribute the surface entities, and again to recover the primary decomposition at the completion of the contact operation. Load balance is good, however.

The “fastest” approach is problem dependent. We consider only the ghosting search in this study.

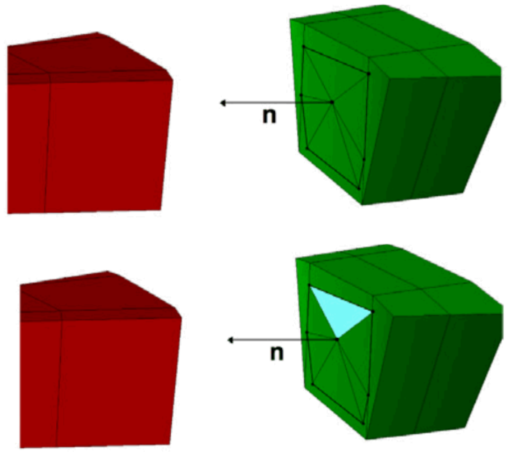
Sandia National Laboratories

Local search and imprinting

- Search to determine possibly contacting "pairs" or subsets of entities between surfaces



- "Imprinting" establishes the geometric relationships between entities via projection, to support finite element enforcement operations



$$D \leftarrow \sum_t A^t \sum_g N_\lambda N_s ,$$

$$M \leftarrow \sum_t A^t \sum_g N_\lambda N_m$$

Enforcement

- **"Augment" the physics problem with terms that represent the contact interaction, using multi-point constraints (MPC), penalty methods, or Lagrange multipliers**

$$\Pi_T = \int_{\Gamma^C} q\left(T^s - T^m - \frac{q}{U}\right) d\Gamma^C$$

$$a_T^h(T, v) + c_T^h(v, \lambda_T) + c_T^h(T, \mu_T) = \begin{pmatrix} \mathbf{T}_i^T & \mathbf{T}_m^T & \mathbf{T}_s^T & \lambda_{\mathbf{T}}^T \end{pmatrix} \begin{pmatrix} A_{ii} & A_{im} & A_{is} & 0 \\ A_{mi} & A_{mm} & 0 & M \\ A_{si} & 0 & A_{ss} & D \\ 0 & M^T & D^T & \frac{2}{\mathbf{U}} \end{pmatrix} \begin{pmatrix} \mathbf{v}_i \\ \mathbf{v}_m \\ \mathbf{v}_s \\ \boldsymbol{\mu}_{\mathbf{T}} \end{pmatrix}$$

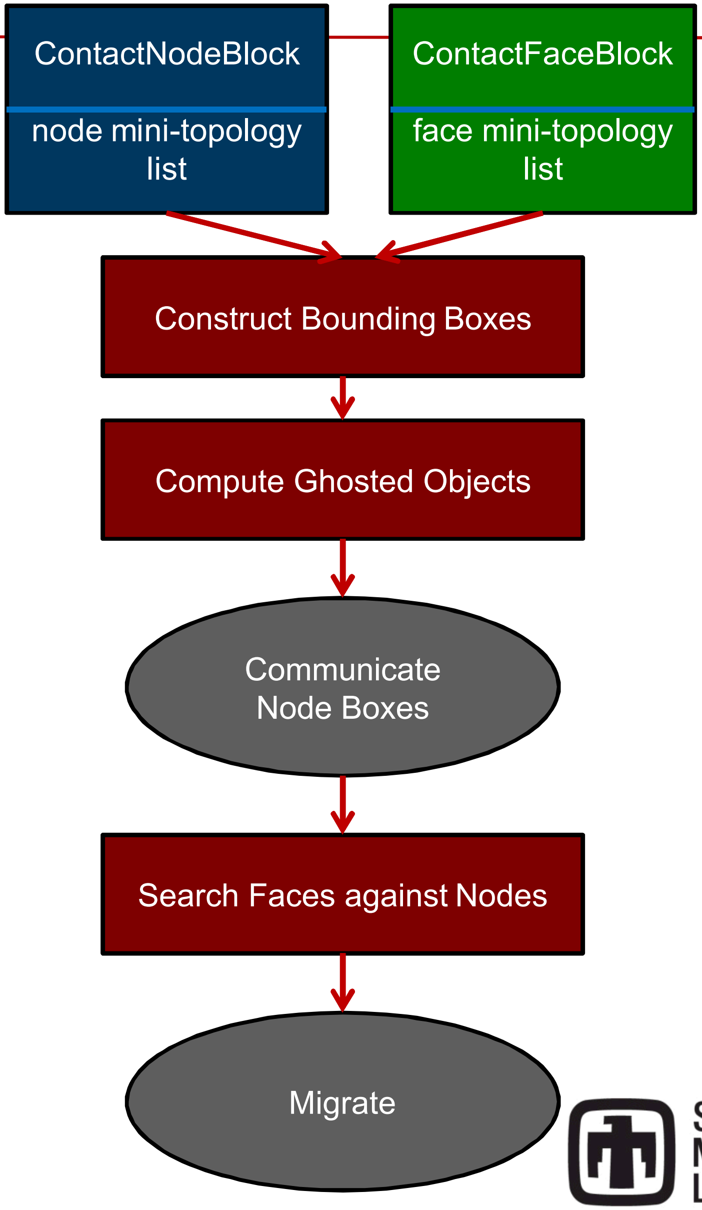
$$\Pi_{\mathbf{u}} = \int_{\Gamma^C} t_n \left(g_n - \frac{t_n}{P_c}\right) d\Gamma^C$$

$$a_{\mathbf{u}}^h(\mathbf{u}, \mathbf{w}) + c_{\mathbf{u}}^h(\mathbf{w}, \boldsymbol{\lambda}_{\mathbf{u}}) + c_{\mathbf{u}}^h(\mathbf{u}, \boldsymbol{\mu}_{\mathbf{u}}) = \begin{pmatrix} \mathbf{u}_i^T & \mathbf{u}_m^T & \mathbf{u}_s^T & \lambda_{\mathbf{u}}^T \end{pmatrix} \begin{pmatrix} A_{ii} & A_{im} & A_{is} & 0 \\ A_{mi} & A_{mm} & 0 & M \\ A_{si} & 0 & A_{ss} & D \\ 0 & M^T & D^T & \frac{2}{P_c} \end{pmatrix} \begin{pmatrix} \mathbf{w}_i \\ \mathbf{w}_m \\ \mathbf{w}_s \\ \boldsymbol{\mu}_{\mathbf{u}} \end{pmatrix}$$

Sandia National Laboratories

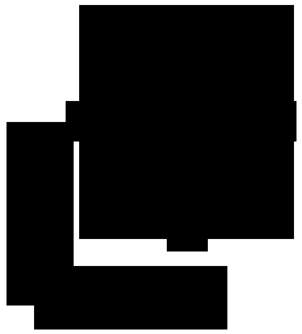
Contact search – MPI+X strategy

- Transfer node and face lists to the coprocessors
- Compute potential objects to ghost
- Transfer node boxes to host, communicate with MPI
- Communicate incoming node boxes to coprocessors, search against faces
- Construct export buffers that describe entities to be ghosted
- Zoltan migrate ghosted entities using MPI



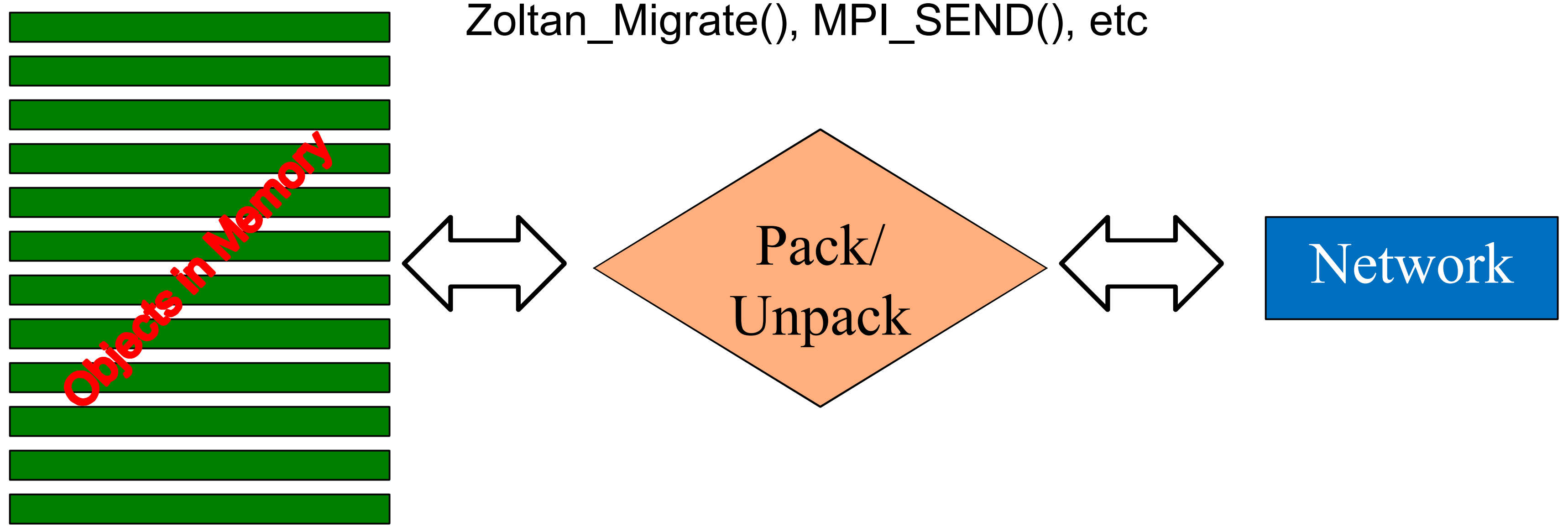
Parallel search algorithm

- **Epsilon-inflated axis-aligned bounding boxes (AABBs) wrap the entities that could come in contact**
- **Need both the tree traversal and construction to be parallel with sufficient *occupancy* on coprocessors**
- **Employ linear BVH tree designed to be constructed in parallel**
  - sort along Z-order curve defined by Morton codes
  - number the entities in a careful way to break the dependence on parent nodes in the tree (Karras, HPG 2012)
  - unfortunately, new approach requires 3 binary searches per entity instead of one and will run slower on a single thread than an optimal serial algorithm



Data movement: original MPI model

Zoltan_Migrate(), MPI_SEND(), etc

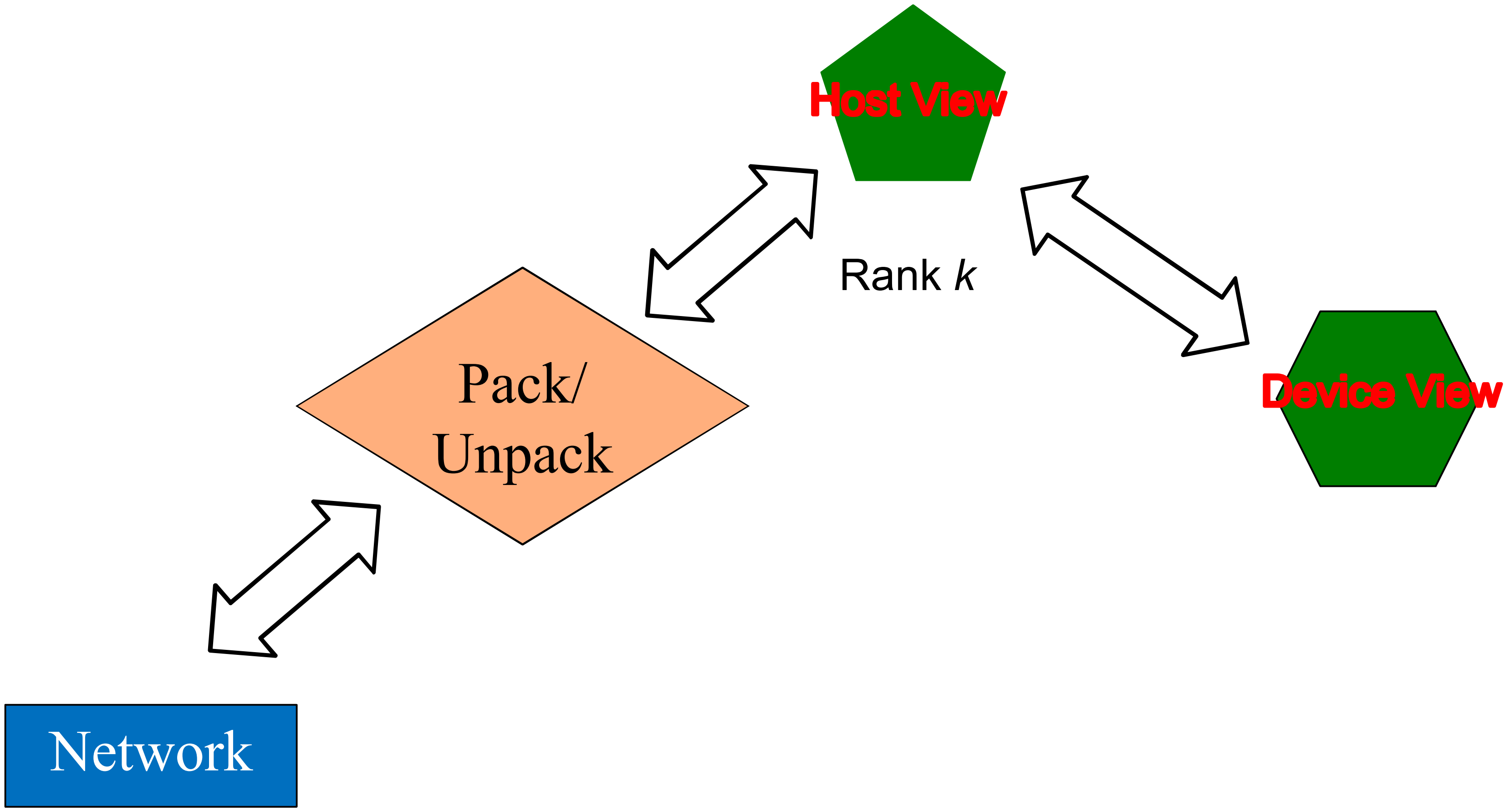


Rank *k*

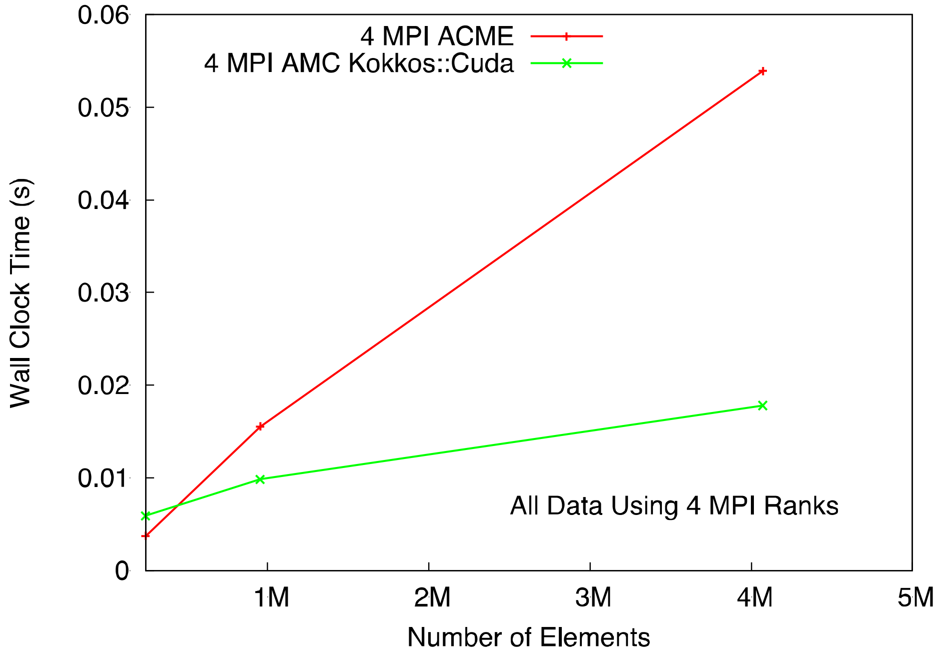
Design optimized for MPI performance/scalability

Data movement: current GPU implementation

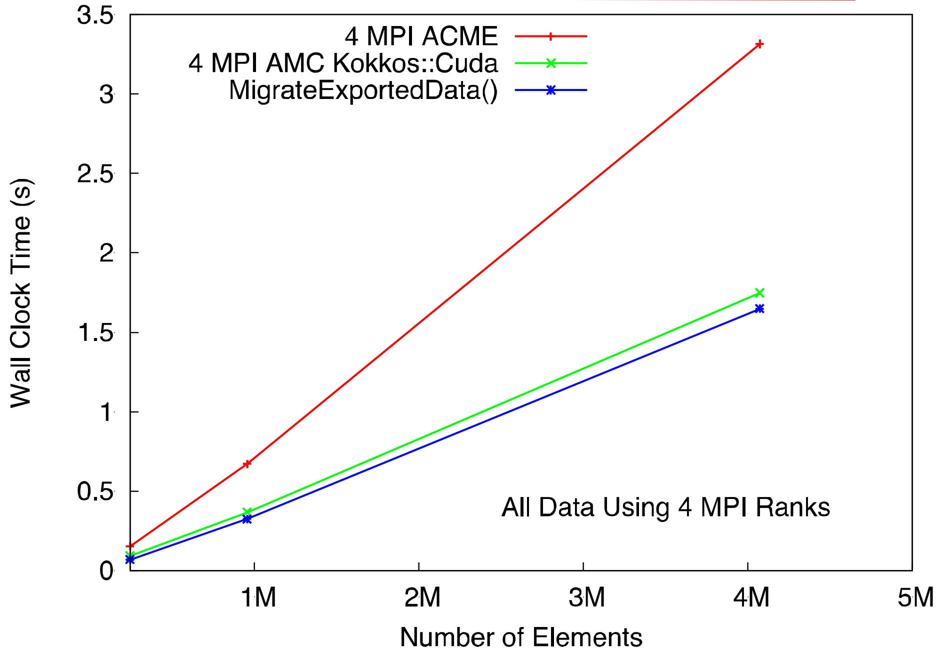
Design balanced for performance on both sides



Global ghosting search using Kokkos: MPI+GPU



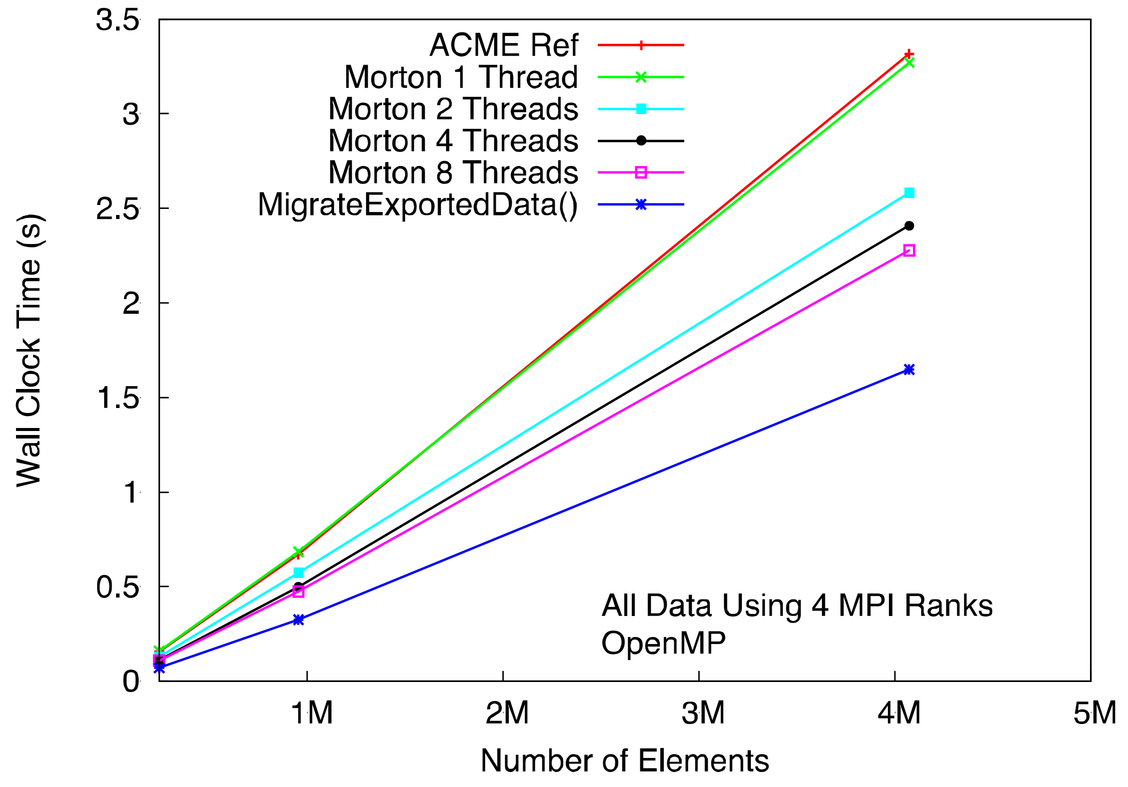
Global Search



Overall Scaling of Ghosting Function

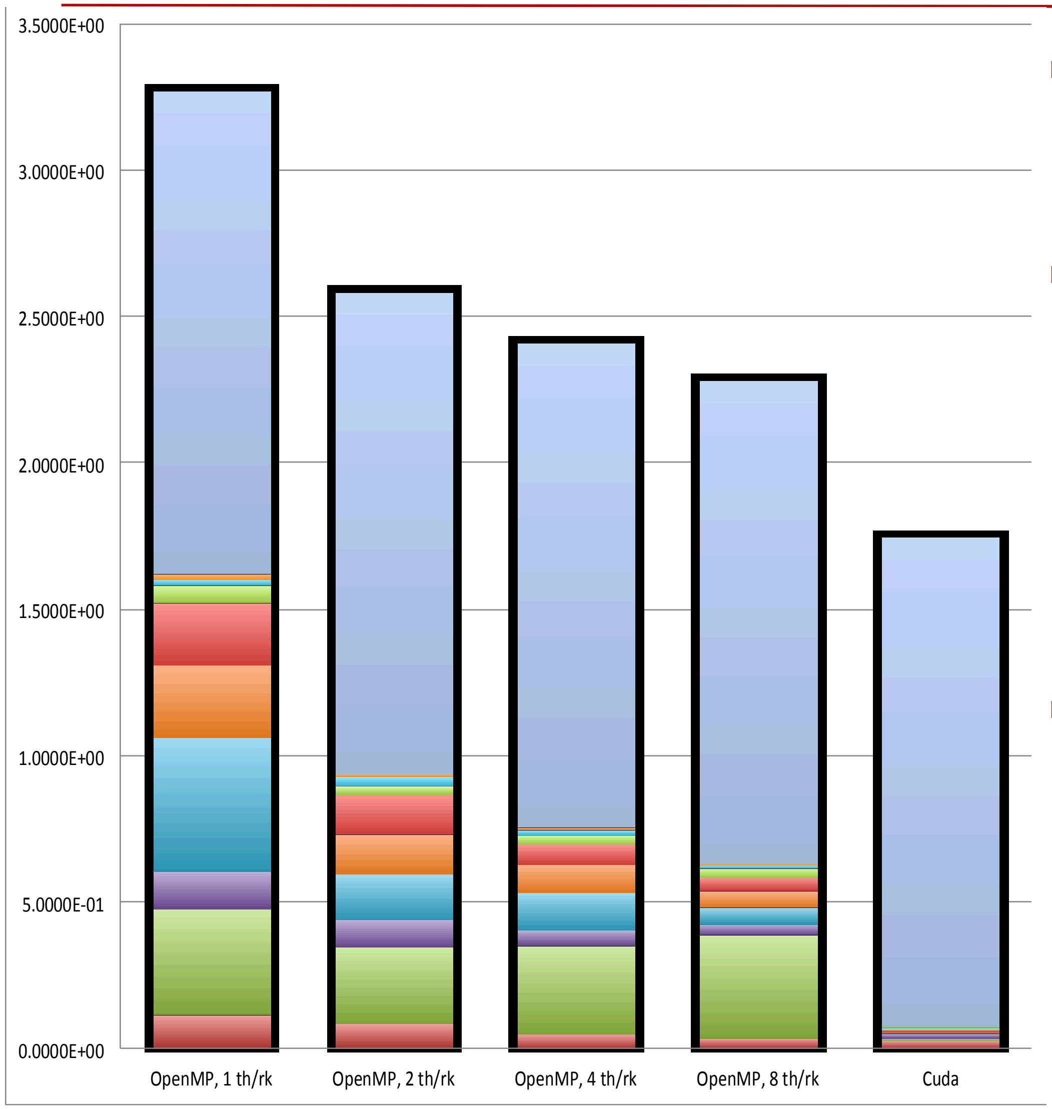
- **Results on "Curie" (one NVIDIA KX20 / node)**
- **MAS = Morton-code accelerated search**
- **MigrateExportedData() = Zoltan_Migrate of ghost nodes and faces**

Global ghosting search using Kokkos MPI+OpenMP



- **Results on "Curie" (16-core Opteron / node)**
- **MAS = Morton-code accelerated search**
- **MigrateExportedData() = Zoltan_Migrate of ghost nodes and faces**

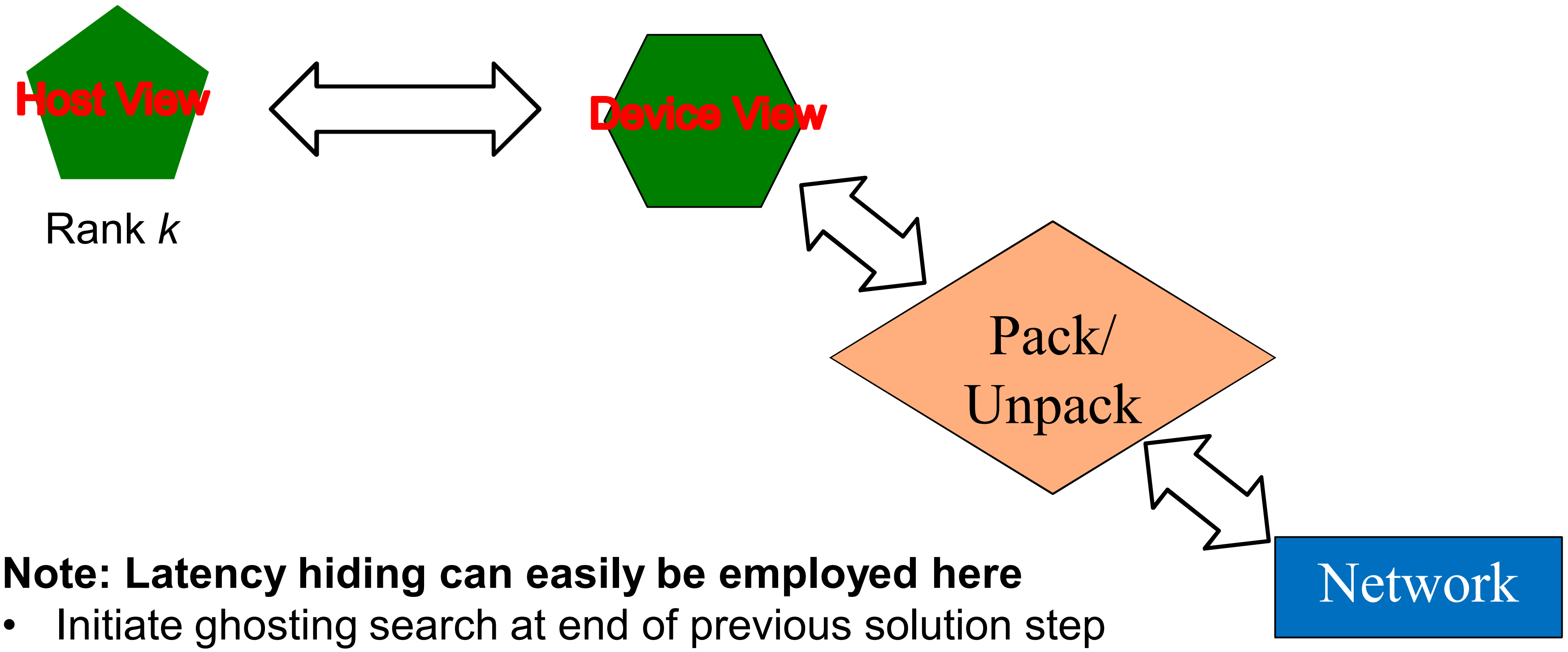
Scalability on multicore and GPU architectures



- **Compare Kokkos/OpenMP 1,2,4,8 threads and Kokkos/Cuda**
- **Diversity in the performance of Kokkos::atomic_fetch_and_add**
  - Fast on Kepler
  - Reconsider approach on Opteron
- **Ongoing work:**
  - try “count-allocate-fill” pattern, at least for OpenMP
  - Experiments on Xeon/Phi

Need to address MPI communications model

Pack and unpack on device buffer, then MPI_SEND() using device buffer handle



Note: Latency hiding can easily be employed here

- Initiate ghosting search at end of previous solution step
- Assemble remainder of finite element physics problem while waiting for ghosting migration to complete
- Perform local search and imprinting – proceed to assembly of contact contributions

Conclusions

- **A tradeoff exists between possible performance gains with MPI parallelism and on-node multicore parallelism**
- **With the contact problem, the physics solution and decomposition will enforce a given MPI rank structure and data layout**
- **Within that context, one must weigh entity communications costs and load balancing issues**
  - if "FLOPS are free," load imbalance may not be a large concern
  - However, there is a limited amount of memory on the GPU, may need to rebalance to spread the contact operations across available GPU memory
- **It can be the case that an implementation designed for one coprocessor (GPU) may not perform well on another (Phi/multicore)**

References

- G. Hansen, P. Xavier, S. Mish, T. Voth, M. Heinstein, M. Glass, "An MPI+X Implementation of Contact Global Search Using Kokkos," *Engineering with Computers*, submitted.
- T. Karras, "Maximizing Parallelism in the Construction of BVHs, Octrees, and *k*-d Trees, *High Performance Graphics*, 2012.
- T. Karras, "Thinking Parallel, Part III: Tree Construction on the GPU," http://devblogs.nvidia.com/parallelforall/thinking-parallel-part-iii-tree-construction-gpu/ (2012)

Sandia National Laboratories