

FASTMath Team Members: Karen Devine, Vitus Leung, and Siva Rajamanickam (SNL); Onkar Sahni, Mark Shephard, and Cameron Smith (RPI)

The FASTMath institute provides data-partitioning and load-balancing expertise and software for use in a wide range of scientific applications. We are addressing the challenges of heterogeneous architectures and dynamic, unstructured applications through research into new hierarchical and predictive partitioning strategies.

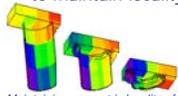
Zoltan: Load Balancing, Partitioning and Ordering

Zoltan provides parallel combinatorial algorithms important to scientific computing.

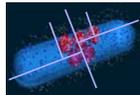
- *Static partitioning* creates initial distributions with balanced loads and low data movement.
- *Dynamic load balancing* maintains balance and locality in adaptive or evolving applications.
- *Data ordering* improves data locality and cache performance.

Zoltan supports a variety of applications with differing data structures and computational requirements.

- *Geometric methods* use physical coordinates to maintain locality of data.
- *Topology-based methods* use data dependencies to reduce communication and data movement.



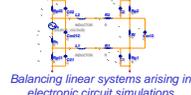
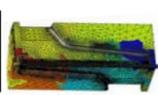
Maintaining geometric locality of surfaces for contact detection and crash simulations



Maintaining balance and geometric locality in a particle-based cell simulation



Load balancing of adaptive mesh refinement methods for safety analysis, shock physics, CFD, etc.



Balancing linear systems arising in electronic circuit simulations

- Common interface to all methods allows experimentation within applications to find the most effective strategy.

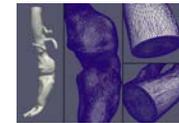
Using Zoltan in new or existing applications is easy.

- Native Zoltan interface uses callback functions for applications to describe their data.
- The iZoltan service accesses Zoltan using ITAPS-compliant mesh interfaces (iMesh, iMeshP).
- Integration with Trilinos Epetra classes is included through Trilinos/Isorropia.

ParMA: Partition improvement using Mesh Adjacencies

Mesh adjacencies represent application data more completely than standard graph-partitioning models.

- All mesh entities can be considered, while graph-partitioning models use only a subset of mesh adjacency information.
- Any adjacency can be obtained in $O(1)$ time (assuming use of a complete mesh adjacency structure).
- Partition model supports efficient part adjacency queries. (See FASTMath's Unstructured Mesh poster for more details.)



Geometry and mesh for simulations of blood flow in an abdominal aortic aneurysm (AAA) using the PHASTA CFD code

Partition Improvement with Multiple Criteria

- Diffusive procedure driven by application-defined priority list of mesh entity types to balance. Iteration over three stages:
 - Schedule load transfer from heavy parts to light parts
 - Select mesh elements on the part boundaries that will smooth the part boundary
 - Migrate selected mesh elements according to schedule



Regions selected for migration have at least as many faces on the part boundary (translucent) as in the part interior (shaded).

| Original | Zoltan's Hypergraph |
|----------|---------------------|
| ParMA #1 | Vtx > Rgn |
| ParMA #2 | Vtx = Edge > Rgn |
| ParMA #3 | Edge > Rgn |
| ParMA #4 | Edge = Face > Rgn |

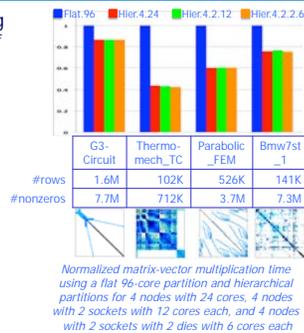
| AAA TEST | time(s) | MeanRgn | Rgn Imb. | MeanFace | Face Imb. | MeanEdge | Edge Imb. | MeanVtx | Vtx Imb. |
|----------|---------|---------|----------|----------|-----------|----------|-----------|---------|----------|
| Original | 249 | 8177 | 4.8% | 17,315 | 5.30% | 11,023 | 9.07% | 1886 | 19.41% |
| ParMA #1 | 6.6 | 8177 | 4.90% | - | - | - | - | 1865 | 4.96% |
| ParMA #2 | 8.8 | 8177 | 5.90% | - | - | 10,973 | 4.91% | 1870 | 4.96% |
| ParMA #3 | 5.5 | 8177 | 5.98% | - | - | 11,013 | 4.96% | - | - |
| ParMA #4 | 5.5 | 8177 | 5.93% | 17,309 | 4.97% | 11,014 | 4.96% | - | - |

Partition-improvement time and entity imbalances when applying ParMA to a hypergraph-based partition of a 133M region AAA mesh with 16,384 parts on 512 cores of Jaguar

- With ParMA Vtx>Rgn, PHASTA CFD strong scaling improves from 0.88 to 0.95 on 288k cores, JuGene BG/P, with a 1B-region mesh.

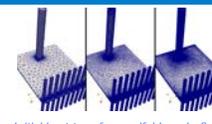
Hierarchical Partitioning for Multicore Computers

- Zoltan's hierarchical partitioning distributes data across levels of computers' hardware hierarchy (nodes, sockets, dies, cores).
- Improves data locality within hardware components
- Experiments on Magny-Cours (NERSC's Hopper) showed greatest benefit from partitioning first with respect to nodes and then with respect to cores (Hier.4.24).

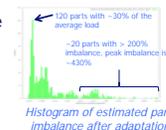


Predictive Load Balancing Research

- Parallel unstructured mesh adaptation can generate parts with up to 400% imbalance on non-trivial geometries due to local coarsening & refinement.
- Refining before repartitioning can exceed available memory in some processes, even if the system's total memory is sufficient.
- Solution: Redistribute mesh *before* adapting.
 - Merge parts that will be coarsened to create some empty parts.
 - Split parts with heavy refinement into the empty parts to remove imbalance spikes.
- Refine/coarsen the mesh.
- Apply ParMA's diffusive partition improvement.



Initial heat transfer manifold mesh, first adapted, and second adapted mesh



Histogram of estimated part imbalance after adaptation

Current Research and Development

- Integration of Zoltan and ParMA in iZoltan
- MPI+threads implementation of partitioners
- Communication reduction via asynchronous unstructured mesh migration procedures
- Scalable, effective partitioning for high core counts
- Hypergraph partitioning and ordering for PDSLin

Collaborators:
 Erik Boman (SNL), Ümit Catalyürek (Ohio St.), Mehmet Deveci (Ohio St.), Ken Jansen (U. Colorado), Lee Ann Riesen (SNL), Seegyoung Seol (RPI)
 Images courtesy of:
 Steve Attaway (SNL), Tim Davis (U. Florida), Carter Edwards (SNL), Scott Hutchinson (SNL), Alexander Ovcharenko (RPI), Steve Plimpton (SNL), Lee Ann Riesen (SNL), Min Zhou (RPI)

More Information: <http://www.fastmath-scidac.org> or contact any of the team members identified above.