Tutorial: The Zoltan Toolkit

Cedric Chevalier

Erik Boman, Karen Devine, Vitus Leung, Lee Ann Riesen

Sandia National Laboratories, NM

Umit Çatalyürek, Doruk Bozdag

Ohio State University

ACTS Workshop, August 2009

Sandia is a multiprogram laboratory operated by Sandia Corporation, a Lockheed Martin Company, for the United States Department of Energy's National Nuclear Security Administration under contract DE-AC04-94AL85000.

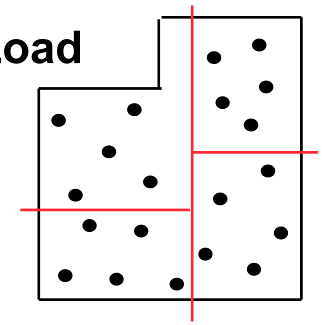
Outline

- High-level view of Zoltan
- Requirements, data models, and interface
- Dynamic Load Balancing and Partitioning
- Matrix Ordering
- Graph Coloring
- Utilities
- Alternate Interfaces
- Future Directions

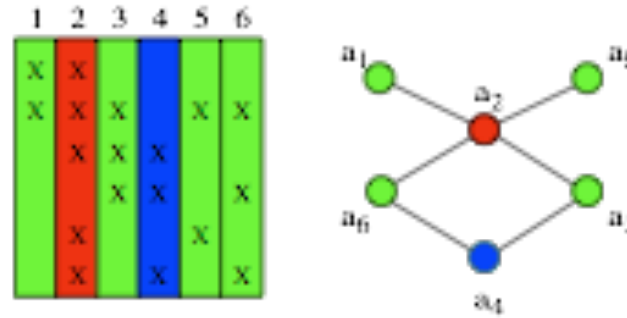
The Zoltan Toolkit

- Library of data management services for unstructured, dynamic and/or adaptive computations.

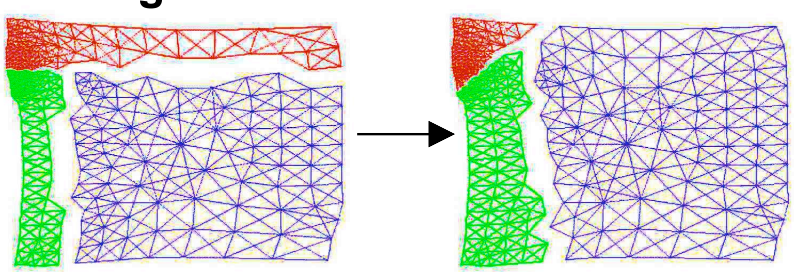
Dynamic Load Balancing



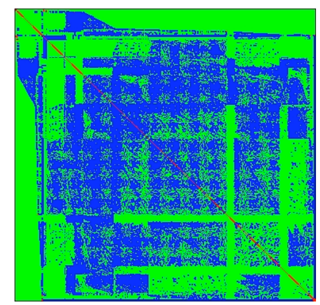
Graph Coloring



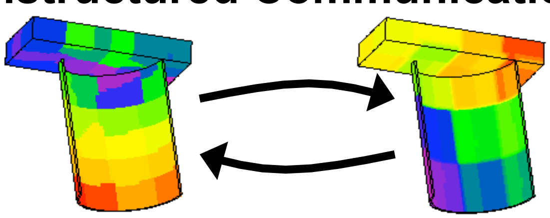
Data Migration



Matrix Ordering



Unstructured Communication



Distributed Data Directories

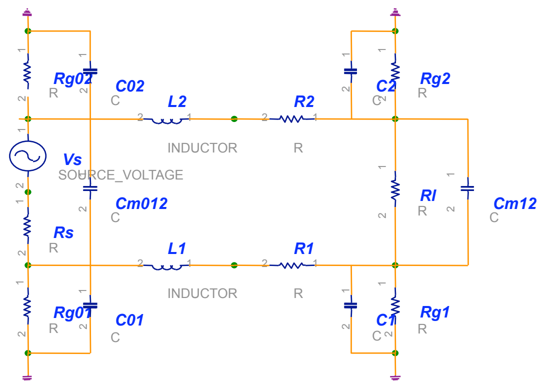
A	B	C	D	E	F	G	H	I
0	1	0	2	1	0	1	2	1

Zoltan System Assumptions

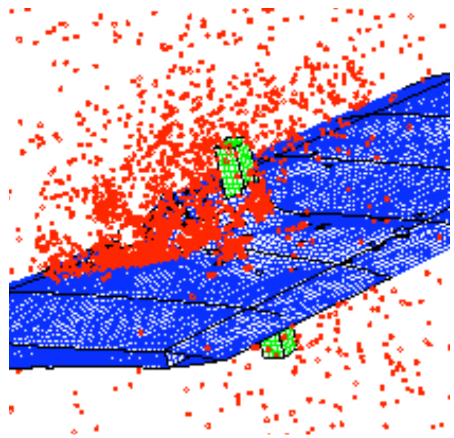
- Assume distributed memory model.
- Data decomposition + "Owner computes":
  - The data is distributed among the processors.
  - The owner performs all computation on its data.
  - Data distribution defines work assignment.
  - Data dependencies among data items owned by different processors incur communication.
- Zoltan is available in Trilinos since version 9.0
- Requirements:
  - MPI
  - C compiler
  - Autotools or CMake.

Zoltan Supports Many Applications

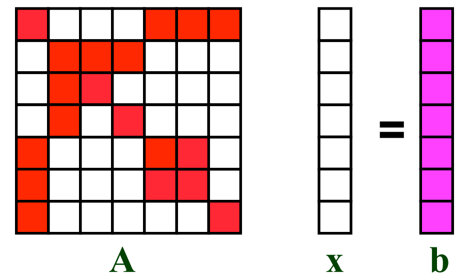
- Different applications, requirements, data structures.



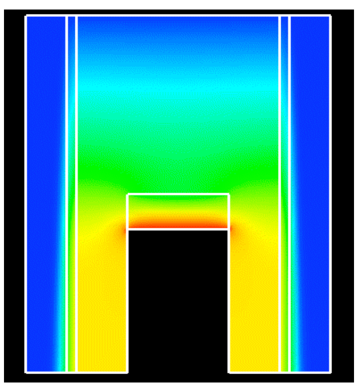
Parallel electronics networks



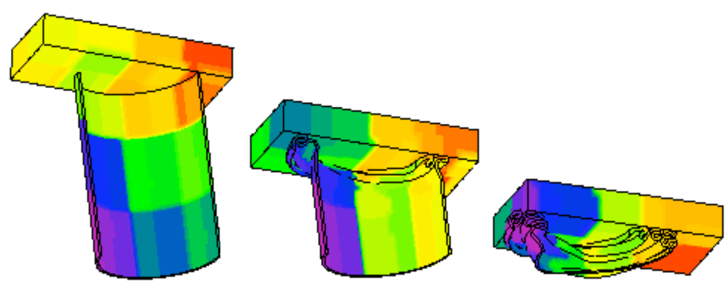
Particle methods



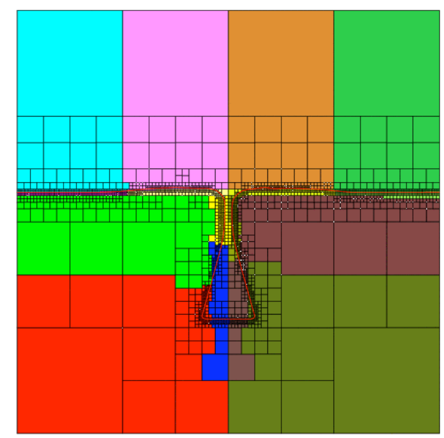
Linear solvers & preconditioners



Multiphysics simulations



Crash simulations



Adaptive mesh refinement

Zoltan Interface Design

- **Common interface to each class of tools.**
- **Tool/method specified with user parameters.**

- **Data-structure neutral design.**
  - **Supports wide range of applications and data structures.**
  - **Imposes no restrictions on application's data structures.**
  - **Application does not have to build Zoltan's data structures.**

Sandia National Laboratories

Zoltan Interface

- Simple, easy-to-use interface.
  - Small number of callable Zoltan functions.
  - Callable from C, C++, Fortran.
- Requirement: Unique global IDs for objects to be partitioned/ordered/colored. For example:
  - Global element number.
  - Global matrix row number.
  - (Processor number, local element number)
  - (Processor number, local particle number)

Zoltan Application Interface

- Application interface:
  - Zoltan queries the application for needed info.
    - IDs of objects, coordinates, relationships to other objects.
  - Application provides simple functions to answer queries.
  - Small extra costs in memory and function-call overhead.
- Query mechanism supports…
  - Geometric algorithms
    - Queries for dimensions, coordinates, etc.
  - Hypergraph- and graph-based algorithms
    - Queries for edge lists, edge weights, etc.
  - Tree-based algorithms
    - Queries for parent/child relationships, etc.
- Once query functions are implemented, application can access all Zoltan functionality.
  - Can switch between algorithms by setting parameters.

Sandia National Laboratories

Zoltan Application Interface

APPLICATION

Initialize Zoltan
(Zoltan_Initialize, Zoltan_Create)

↓

Select Method and Parameters
(Zoltan_Set_Params)

↓

Register query functions
(Zoltan_Set_Fn)

→

(Re)partition
(Zoltan_LB_Partition)

↓

Move data
(Zoltan_Migrate)

↓

COMPUTE

↓

Clean up
(Zoltan_Destroy)

ZOLTAN

Zoltan_LB_Partition:

- **Call query functions.**
- **Build data structures.**
- **Compute new decomposition.**
- **Return import/export lists.**

Zoltan_Migrate:

- **Call packing query functions for exports.**
- **Send exports.**
- **Receive imports.**
- **Call unpacking query functions for imports.**

Zoltan Query Functions

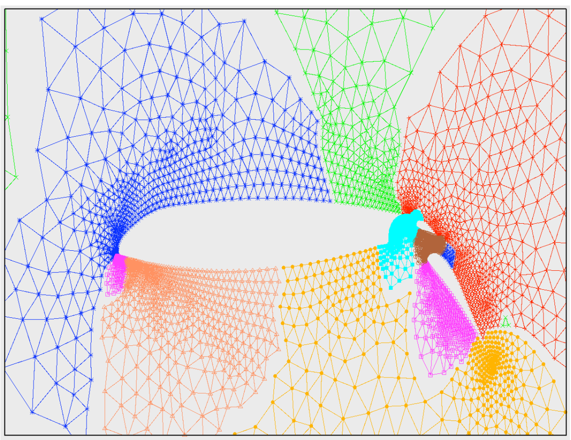
General Query Functions	
`ZOLTAN_NUM_OBJ_FN`	Number of items on processor
`ZOLTAN_OBJ_LIST_FN`	List of item IDs and weights.
Geometric Query Functions	
`ZOLTAN_NUM_GEOM_FN`	Dimensionality of domain.
`ZOLTAN_GEOM_FN`	Coordinates of items.
Hypergraph Query Functions	
`ZOLTAN_HG_SIZE_CS_FN`	Number of hyperedge pins.
`ZOLTAN_HG_CS_FN`	List of hyperedge pins.
`ZOLTAN_HG_SIZE_EDGE_WTS_FN`	Number of hyperedge weights.
`ZOLTAN_HG_EDGE_WTS_FN`	List of hyperedge weights.
Graph Query Functions	
`ZOLTAN_NUM_EDGE_FN`	Number of graph edges.
`ZOLTAN_EDGE_LIST_FN`	List of graph edges and weights.

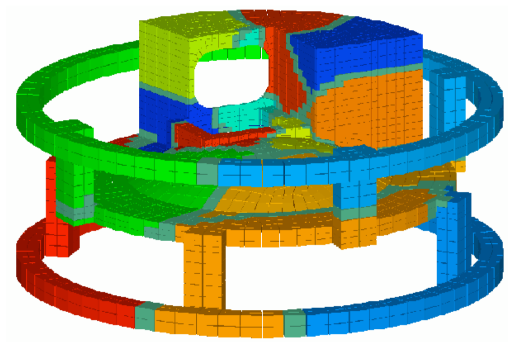
Using Zoltan in Your Application

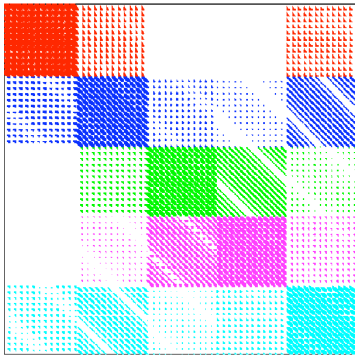
1. Decide what your objects are.
   - Elements? Grid points? Matrix rows? Particles?
2. Decide which tools (partitioning/ordering/coloring/utilities) and class of method (geometric/graph/hypergraph) to use.
3. Download Zoltan.
   - http://www.cs.sandia.gov/Zoltan (or http://trilinos.sandia.gov)
4. Write required query functions for your application.
   - Required functions are listed with each method in Zoltan User's Guide.
5. Call Zoltan from your application.
6. #include "zoltan.h" in files calling Zoltan.
7. Configure and build Zoltan.
8. Compile application; link with libzoltan.a.
   - mpicc application.c -lzoltan

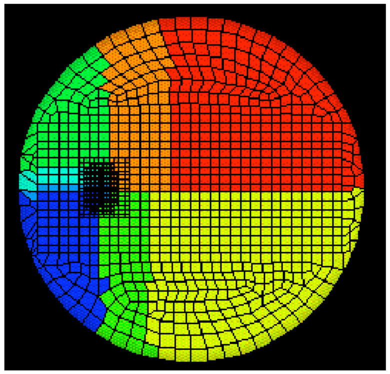
Partitioning and Load Balancing

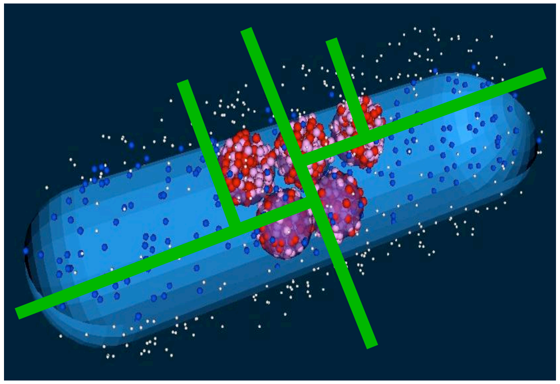
- Assignment of application data to processors for parallel computation.
- Applied to grid points, elements, matrix rows, particles, ….











Partitioning Interface

Zoltan computes the difference (Δ) from current distribution

Choose between:

a) Import lists (data to import from other procs)

b) Export lists (data to export to other procs)

c) Both (the default)

```
err = Zoltan_LB_Partition(zz,
    &changes, /* Flag indicating whether partition changed */
    &numGidEntries, &numLidEntries,
    &numImport, /* objects to be imported to new part */
    &importGlobalGids, &importLocalGids, &importProcs, &importToPart,
    &numExport, /* # objects to be exported from old part */
    &exportGlobalGids, &exportLocalGids, &exportProcs, &exportToPart);
```

Static Partitioning

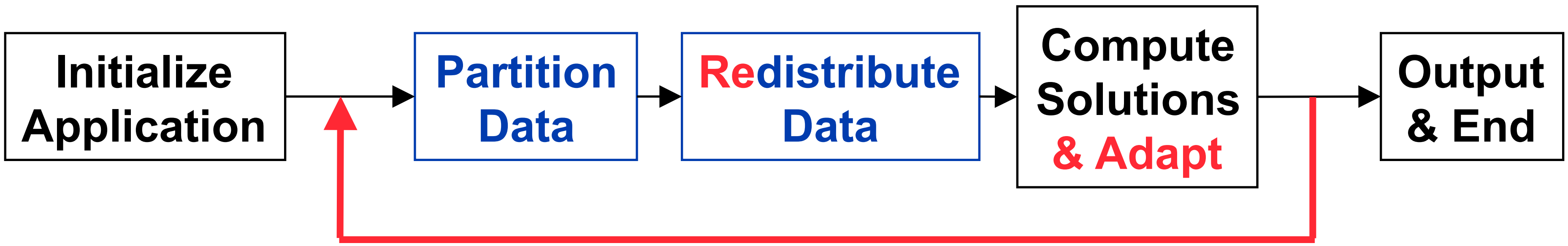
Initialize Application → Partition Data → Distribute Data → Compute Solutions → Output & End

- **Static partitioning in an application:**
  - **Data partition is computed.**
  - **Data are distributed according to partition map.**
  - **Application computes.**

- **Ideal partition:**
  - **Processor idle time is minimized.**
  - **Inter-processor communication costs are kept low.**

- **Zoltan_Set_Param(zz, “LB_APPROACH”, “PARTITION”);**

Dynamic Repartitioning (a.k.a. Dynamic Load Balancing)



- **Dynamic repartitioning (load balancing) in an application:**
  - **Data partition is computed.**
  - **Data are distributed according to partition map.**
  - **Application computes and, perhaps, adapts.**
  - **Process repeats until the application is done.**

- **Ideal partition:**
  - **Processor idle time is minimized.**
  - **Inter-processor communication costs are kept low.**
  - **Cost to redistribute data is also kept low.**

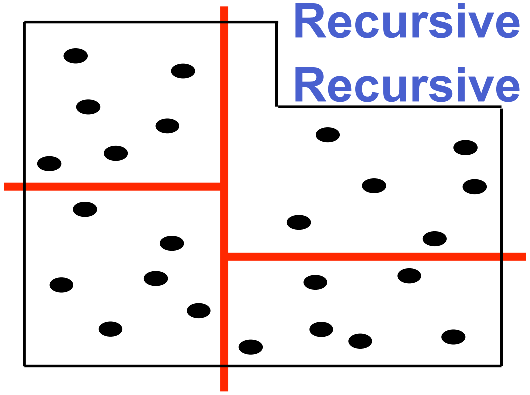
- **Zoltan_Set_Param(zz, “LB_APPROACH”, “REPARTITION”);**

Zoltan Toolkit: Suite of Partitioners

- No single partitioner works best for all applications.
  - Trade-offs:
    - Quality vs. speed.
    - Geometric locality vs. data dependencies.
    - High-data movement costs vs. tolerance for remapping.
- Application developers may not know which partitioner is best for application.
- Zoltan contains suite of partitioning methods.
  - Application changes only one parameter to switch methods.
    - Zoltan_Set_Param(zz, “LB_METHOD”, “new_method_name”);
  - Allows experimentation/comparisons to find most effective partitioner for application.

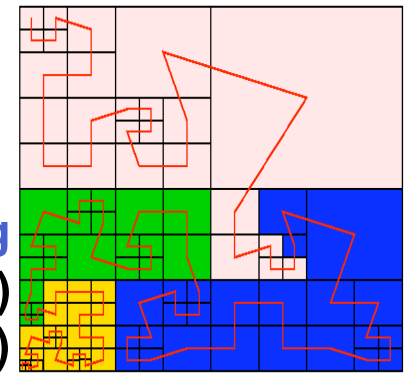
Partitioning Algorithms in the Zoltan Toolkit

Geometric (coordinate-based) methods



Recursive Coordinate Bisection **(Berger, Bokhari)**

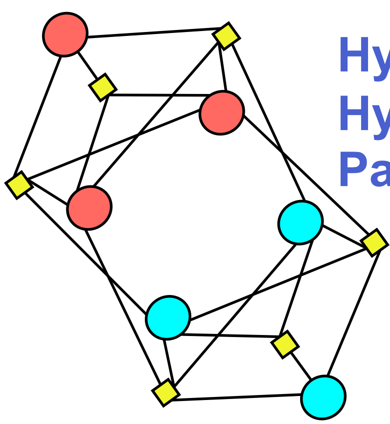
Recursive Inertial Bisection **(Taylor, Nour-Omid)**



Space Filling Curve Partitioning **(Warren&Salmon, et al.)**

Refinement-tree Partitioning **(Mitchell)**

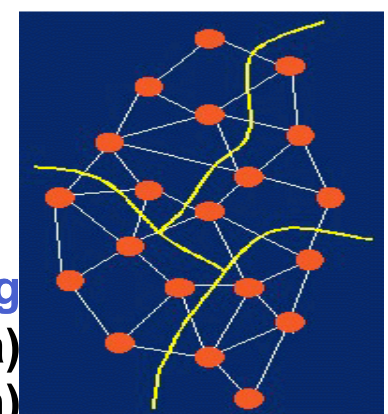
Combinatorial (topology-based) methods



Hypergraph Partitioning

Hypergraph Repartitioning

PaToH **(Catalyurek & Aykanat)**



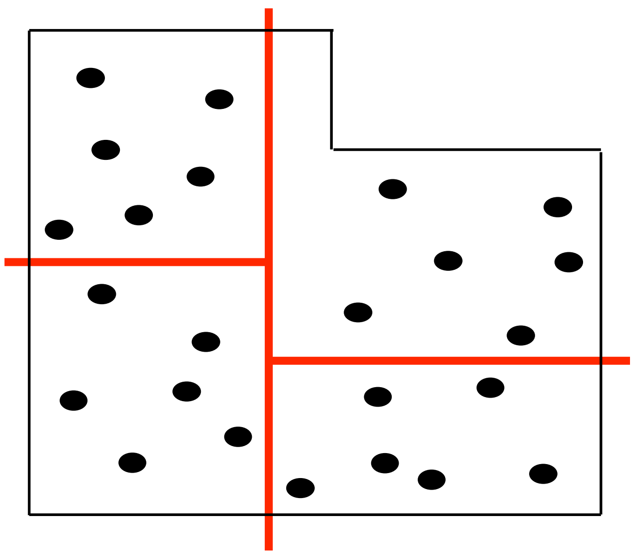
Zoltan Graph Partitioning

ParMETIS **(U. Minnesota)**

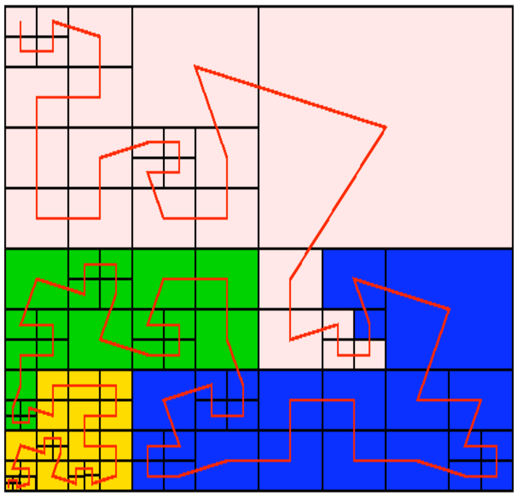
Jostle **(U. Greenwich)**

Geometric Partitioning

- Zoltan_Set_Param(zz, “LB_METHOD”, “RCB”);
  Zoltan_Set_Param(zz, “LB_METHOD”, “RIB”);
  Zoltan_Set_Param(zz, “LB_METHOD”, “HSFC”);
- Partition based on geometric locality.
  - Assign physically close objects to the same processor.



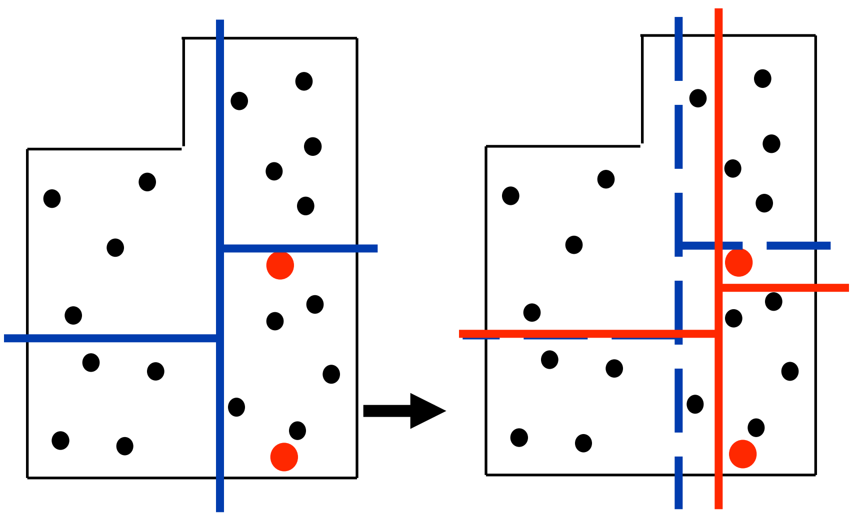
Recursive Coordinate Bisection (RCB)
Berger & Bokhari, 1987



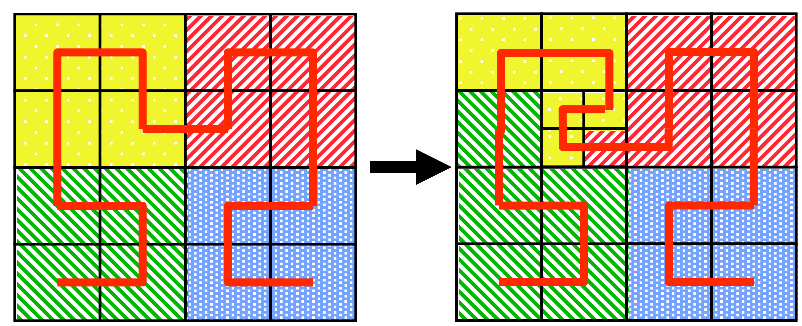
Space Filling Curve Partitioning (HSFC)
Warren & Salmon, 1993;
Pilkington & Baden, 1994; Patra & Oden, 1995

Geometric Repartitioning

- **No explicit control of migration costs, but…**
- **Implicitly achieves low data redistribution costs.**
- **For small changes in data, cuts move only slightly, resulting in little data redistribution.**

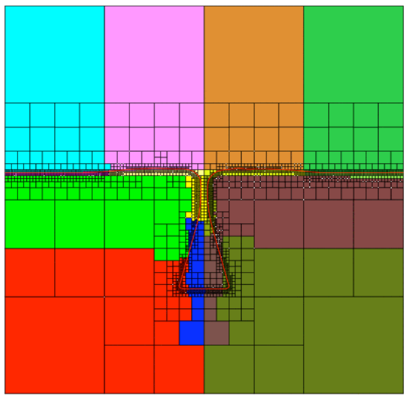


Recursive Coordinate Bisection (RCB)

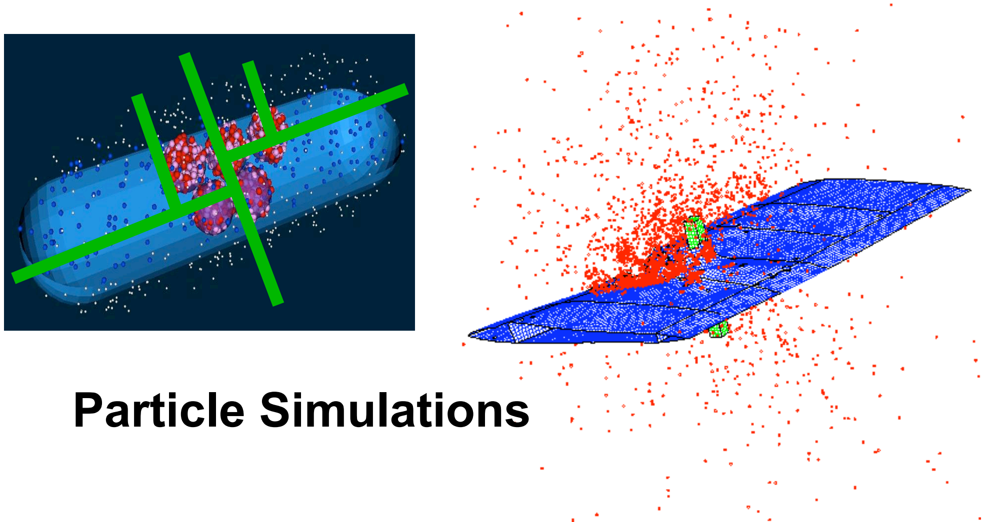


Space Filling Curve Partitioning (HSFC)

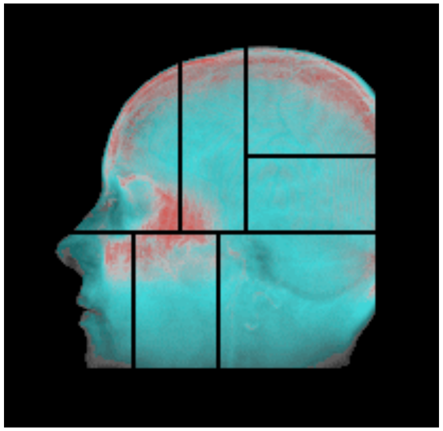
Applications of Geometric Partitioners



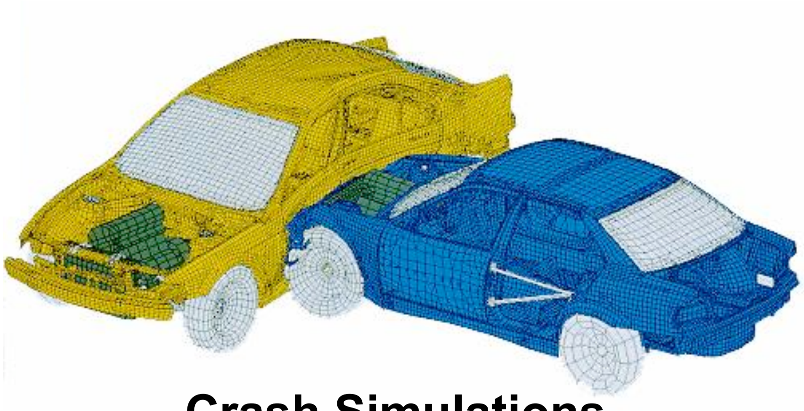
Adaptive Mesh Refinement



Particle Simulations



Parallel Volume Rendering



Crash Simulations
and Contact Detection

Geometric Methods: Advantages and Disadvantages

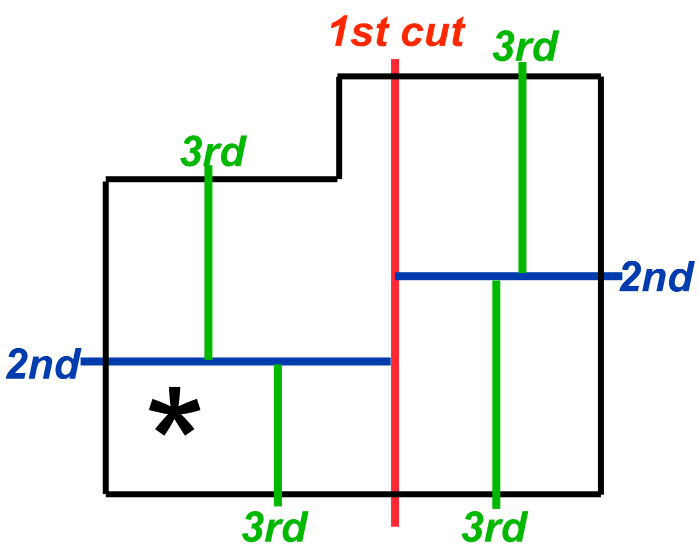
- Advantages:
  - Easiest partitioners to use.
  - Conceptually simple; fast and inexpensive.
  - All processors can inexpensively know entire partition (e.g., for global search in contact detection).
  - No connectivity info needed (e.g., particle methods).
  - Good on specialized geometries.

SLAC'S 55-cell Linear Accelerator with couplers: One-dimensional RCB partition reduced runtime up to 68% on 512 processor IBM SP3. (Wolf, Ko)

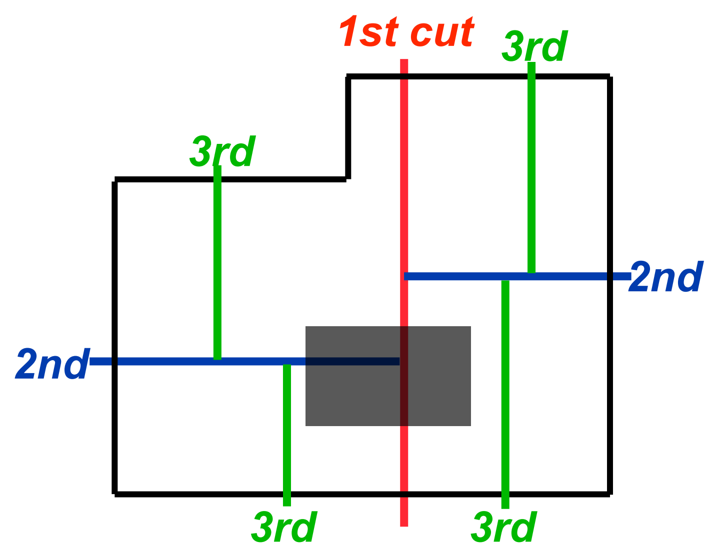
- Disadvantages:
  - No explicit control of communication costs.
  - Mediocre partition quality.
  - Can generate disconnected subdomains for complex geometries.
  - Need coordinate information.

Auxiliary Capabilities for Geometric Methods

- Zoltan can store cuts from RCB, RIB, and HSFC inexpensively in each processor.
  – Zoltan_Set_Param(zz, “KEEP_CUTS”, “1”);
- Enables parallel geometric search without communication.
  – Useful for contact detection, particle methods, rendering.



Determine the part/processor owning region with a given point.
Zoltan_LB_Point_PP_Assign



Determine all parts/processors overlapping a given region.
Zoltan_LB_Box_PP_Assign

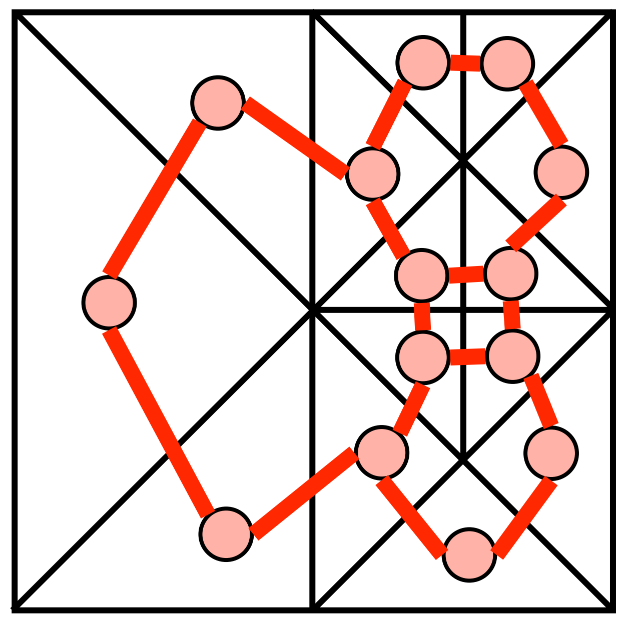
For geometric partitioning (RCB, RIB, HSFC), use …

General Query Functions	
`ZOLTAN_NUM_OBJ_FN`	Number of items on processor
`ZOLTAN_OBJ_LIST_FN`	List of item IDs and weights.
Geometric Query Functions	
`ZOLTAN_NUM_GEOM_FN`	Dimensionality of domain.
`ZOLTAN_GEOM_FN`	Coordinates of items.
Hypergraph Query Functions	
`ZOLTAN_HG_SIZE_CS_FN`	Number of hyperedge pins.
`ZOLTAN_HG_CS_FN`	List of hyperedge pins.
`ZOLTAN_HG_SIZE_EDGE_WTS_FN`	Number of hyperedge weights.
`ZOLTAN_HG_EDGE_WTS_FN`	List of hyperedge weights.
Graph Query Functions	
`ZOLTAN_NUM_EDGE_FN`	Number of graph edges.
`ZOLTAN_EDGE_LIST_FN`	List of graph edges and weights.

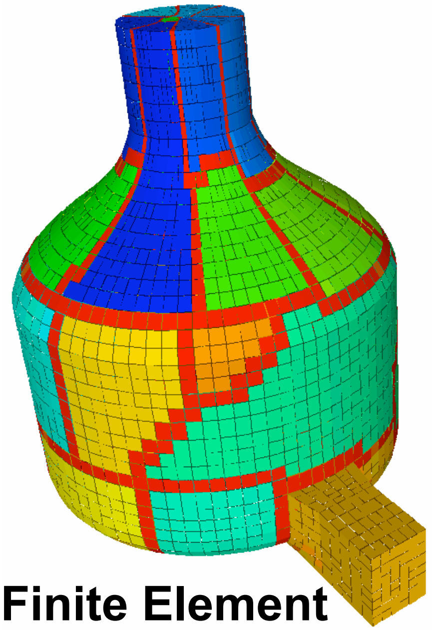
Graph Partitioning

- Zoltan_Set_Param(zz, “LB_METHOD”, “GRAPH”);
- Zoltan_Set_Param(zz, “GRAPH_PACKAGE”, “ZOLTAN”); or
  Zoltan_Set_Param(zz, “GRAPH_PACKAGE”, “PARMETIS”); or
  Zoltan_Set_Param(zz, “GRAPH_PACKAGE”, “SCOTCH”);

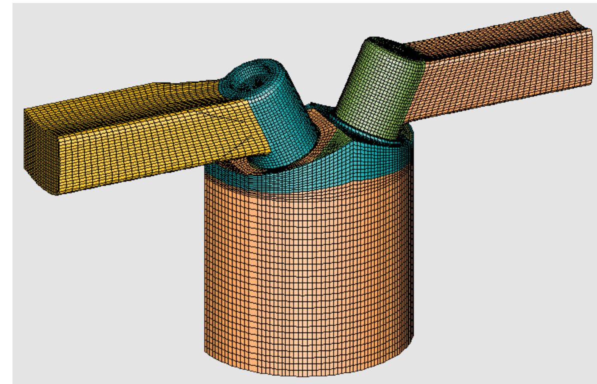
- Kernighan, Lin, Schweikert, Fiduccia, Mattheyes, Simon, Hendrickson, Leland, Kumar, Karypis, et al.
- Represent problem as a weighted graph.
  - Vertices = objects to be partitioned.
  - Edges = dependencies between two objects.
  - Weights = work load or amount of dependency.
- Partition graph so that …
  - Parts have equal vertex weight.
  - Weight of edges cut by part boundaries is small.



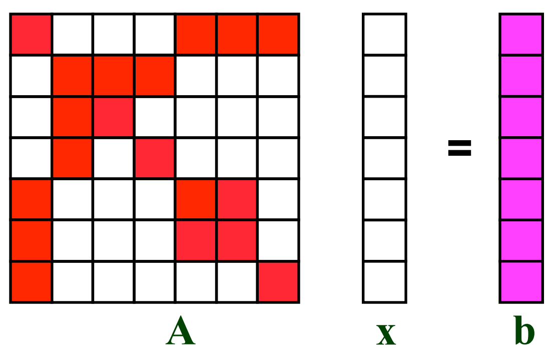
Applications using Graph Partitioning



Finite Element Analysis



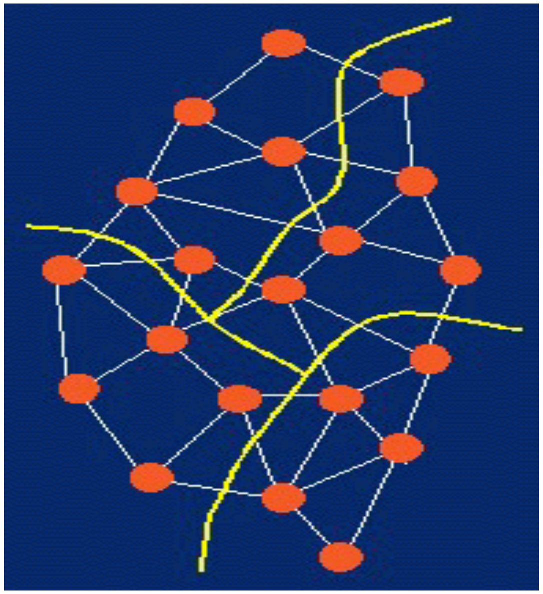
Multiphysics and multiphase simulations



Linear solvers & preconditioners
(square, structurally symmetric systems)

Graph Partitioning: Advantages and Disadvantages

- Advantages:
  - Highly successful model for mesh-based PDE problems.
  - Explicit control of communication volume gives higher partition quality than geometric methods.
  - Excellent software available.
    - Serial: Chaco (SNL)
Jostle (U. Greenwich)
METIS (U. Minn.)
Party (U. Paderborn)
Scotch (U. Bordeaux)
    - Parallel: Zoltan (SNL)
ParMETIS (U. Minn.)
PJostle (U. Greenwich)
PTScotch (U. Bordeaux)



- Disadvantages:
  - More expensive than geometric methods.
  - Edge-cut model only approximates communication volume.

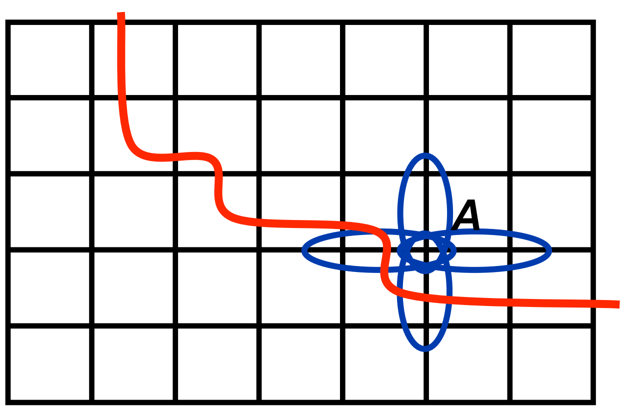
For graph partitioning, coloring & ordering, use …

General Query Functions	
`ZOLTAN_NUM_OBJ_FN`	Number of items on processor
`ZOLTAN_OBJ_LIST_FN`	List of item IDs and weights.
Geometric Query Functions	
`ZOLTAN_NUM_GEOM_FN`	Dimensionality of domain.
`ZOLTAN_GEOM_FN`	Coordinates of items.
Hypergraph Query Functions	
`ZOLTAN_HG_SIZE_CS_FN`	Number of hyperedge pins.
`ZOLTAN_HG_CS_FN`	List of hyperedge pins.
`ZOLTAN_HG_SIZE_EDGE_WTS_FN`	Number of hyperedge weights.
`ZOLTAN_HG_EDGE_WTS_FN`	List of hyperedge weights.
Graph Query Functions	
`ZOLTAN_NUM_EDGE_FN`	Number of graph edges.
`ZOLTAN_EDGE_LIST_FN`	List of graph edges and weights.

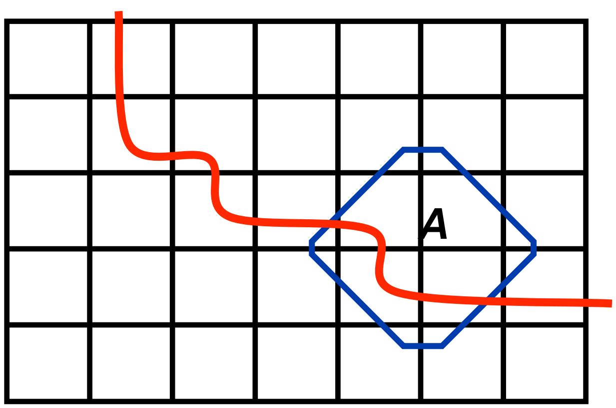
Hypergraph Partitioning

- Zoltan_Set_Param(zz, “LB_METHOD”, “HYPERGRAPH”);
- Zoltan_Set_Param(zz, “HYPERGRAPH_PACKAGE”, “ZOLTAN”); or Zoltan_Set_Param(zz, “HYPERGRAPH_PACKAGE”, “PATOH”);

- Alpert, Kahng, Hauck, Borriello, Çatalyürek, Aykanat, Karypis, et al.
- Hypergraph model:
  - Vertices = objects to be partitioned.
  - Hyperedges = dependencies between two or more objects.
- Partitioning goal: Assign equal vertex weight while minimizing hyperedge cut weight.



Graph Partitioning Model



Hypergraph Partitioning Model

Hypergraph Repartitioning

- Augment hypergraph with data redistribution costs.
  - Account for data's current processor assignments.
  - Weight dependencies by their size and frequency of use.
- Partitioning then tries to minimize total communication volume:

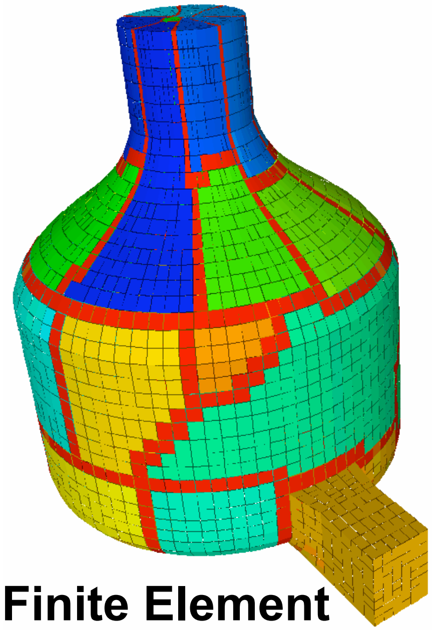
  Data redistribution volume

  + Application communication volume

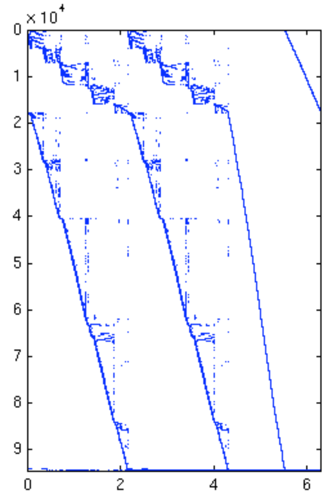
  Total communication volume
- Data redistribution volume: callback returns data sizes.
  - Zoltan_Set_Fn(zz, ZOLTAN_OBJ_SIZE_MULTI_FN_TYPE, myObjSizeFn, 0);
- Application communication volume = Hyperedge cuts * Number of times the communication is done between repartitionings.
  - Zoltan_Set_Param(zz, "PHG_REPART_MULTIPLIER", "100");

Best Algorithms Paper Award at IPDPS07
"Hypergraph-based Dynamic Load Balancing for Adaptive Scientific Computations"
Çatalyürek, Boman, Devine, Bozdag, Heaphy, & Riesen

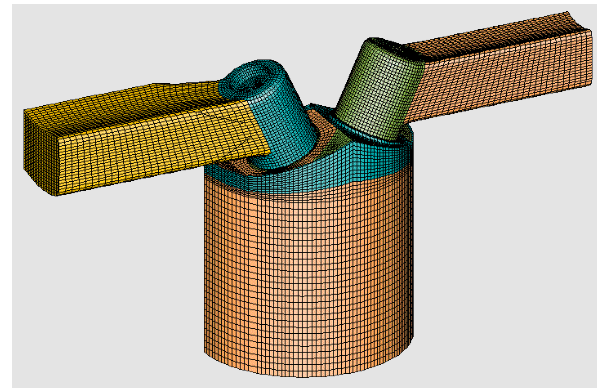
Hypergraph Applications



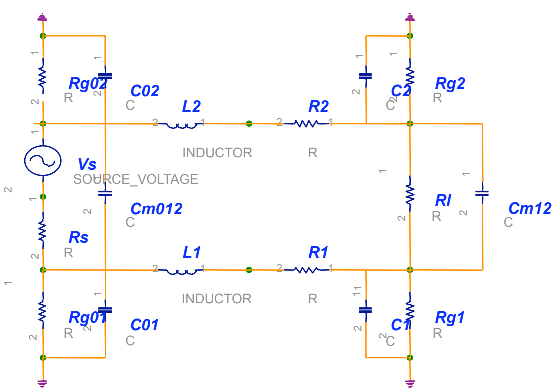
Finite Element Analysis



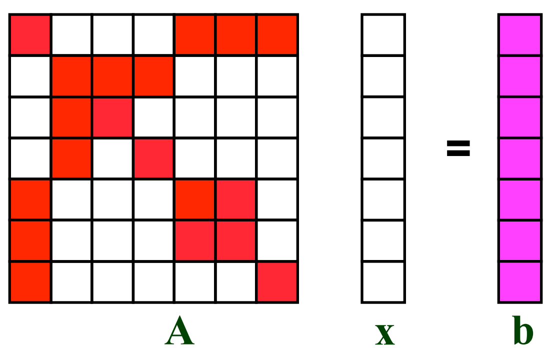
Linear programming for sensor placement



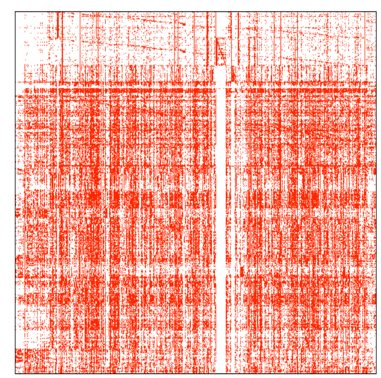
Multiphysics and multiphase simulations



Circuit Simulations



Linear solvers & preconditioners
(no restrictions on matrix structure)



Data Mining

Hypergraph Partitioning: Advantages and Disadvantages

- **Advantages:**
  - Communication volume reduced 30-38% on average over graph partitioning (Catalyurek & Aykanat).
    - 5-15% reduction for mesh-based applications.
  - More accurate communication model than graph partitioning.
    - Better representation of highly connected and/or non-homogeneous systems.
  - Greater applicability than graph model.
    - Can represent rectangular systems and non-symmetric dependencies.
- **Disadvantages:**
  - Usually more expensive than graph partitioning.

For hypergraph partitioning and repartitioning, use …

General Query Functions	
`ZOLTAN_NUM_OBJ_FN`	Number of items on processor
`ZOLTAN_OBJ_LIST_FN`	List of item IDs and weights.
Geometric Query Functions	
`ZOLTAN_NUM_GEOM_FN`	Dimensionality of domain.
`ZOLTAN_GEOM_FN`	Coordinates of items.
Hypergraph Query Functions	
`ZOLTAN_HG_SIZE_CS_FN`	Number of hyperedge pins.
`ZOLTAN_HG_CS_FN`	List of hyperedge pins.
`ZOLTAN_HG_SIZE_EDGE_WTS_FN`	Number of hyperedge weights.
`ZOLTAN_HG_EDGE_WTS_FN`	List of hyperedge weights.
Graph Query Functions	
`ZOLTAN_NUM_EDGE_FN`	Number of graph edges.
`ZOLTAN_EDGE_LIST_FN`	List of graph edges and weights.

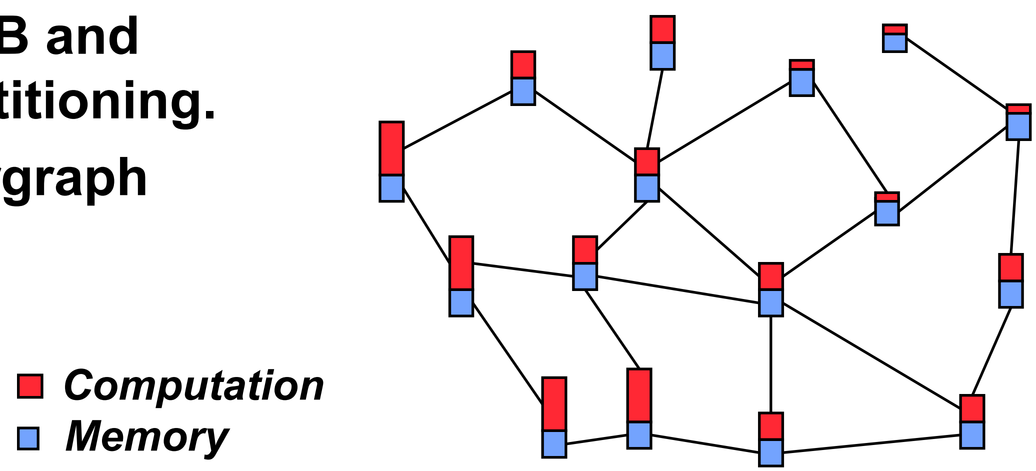
Or can use graph queries to build hypergraph.

General Query Functions	
ZOLTAN_NUM_OBJ_FN	Number of items on processor
ZOLTAN_OBJ_LIST_FN	List of item IDs and weights.
Geometric Query Functions	
ZOLTAN_NUM_GEOM_FN	Dimensionality of domain.
ZOLTAN_GEOM_FN	Coordinates of items.
Hypergraph Query Functions	
ZOLTAN_HG_SIZE_CS_FN	Number of hyperedge pins.
ZOLTAN_HG_CS_FN	List of hyperedge pins.
ZOLTAN_HG_SIZE_EDGE_WTS_FN	Number of hyperedge weights.
ZOLTAN_HG_EDGE_WTS_FN	List of hyperedge weights.
Graph Query Functions	
ZOLTAN_NUM_EDGE_FN	Number of graph edges.
ZOLTAN_EDGE_LIST_FN	List of graph edges and weights.

Sandia National Laboratories

Multi-criteria Load-balancing

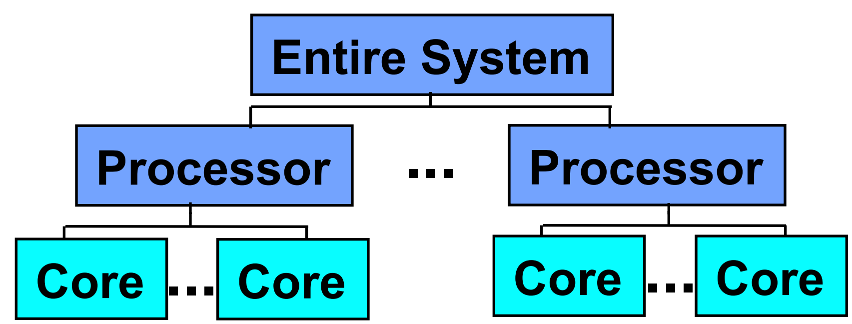
- Multiple constraints or objectives
  - Compute a single partition that is good with respect to multiple factors.
    - Balance both computation and memory.
    - Balance meshes in loosely coupled physics.
    - Balance multi-phase simulations.
  - Extend algorithms to multiple weights
    - Difficult. No guarantee good solution exists.
- Zoltan_Set_Param(zz, "OBJ_WEIGHT_DIM", "2");
  - Available in RCB, RIB and ParMETIS graph partitioning.
  - In progress in Hypergraph partitioning.



Heterogeneous Architectures

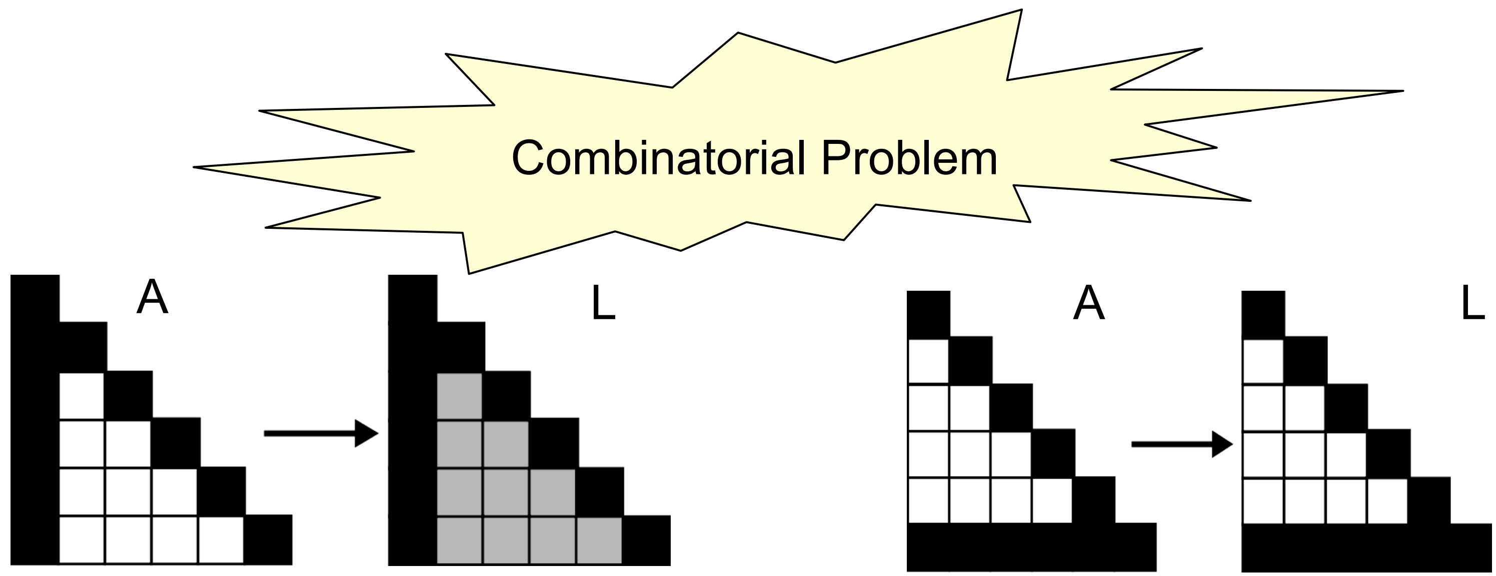
- Clusters may have different types of processors.
- Assign "capacity" weights to processors.
  - E.g., Compute power (speed).
  - Zoltan_LB_Set_Part_Sizes(…);
    - Note: Can use this function to specify part sizes for any purpose.
- Balance with respect to processor capacity.

- Hierarchical partitioning: Allows different partitioners at different architecture levels.
  - Zoltan_Set_Param(zz, "LB_METHOD", "HIER");
  - Requires three additional callbacks to describe architecture hierarchy.
    - ZOLTAN_HIER_NUM_LEVELS_FN
    - ZOLTAN_HIER_PARTITION_FN
    - ZOLTAN_HIER_METHOD_FN



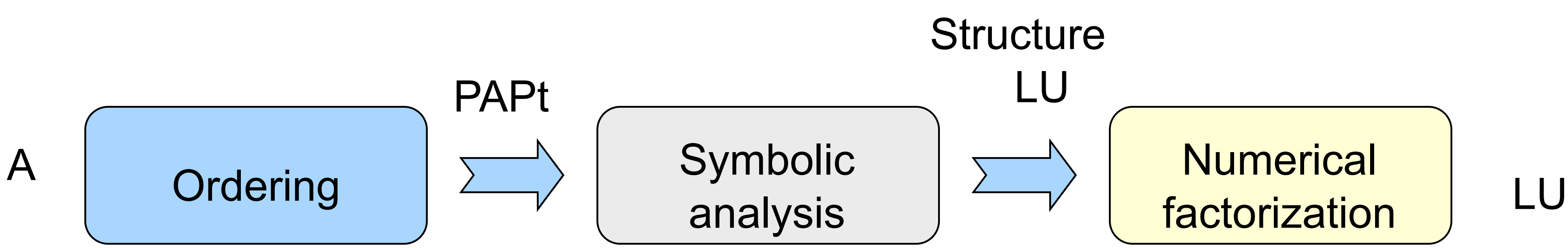
Sparse Matrix Ordering problem

- **When solving sparse linear systems with direct methods, nonzero terms are created during the factorization process (A→LU or LDL^t or LL^t)**
- **Fill-in depends on the order of the unknowns.**
  - **Need to provide fill-reducing ordering**



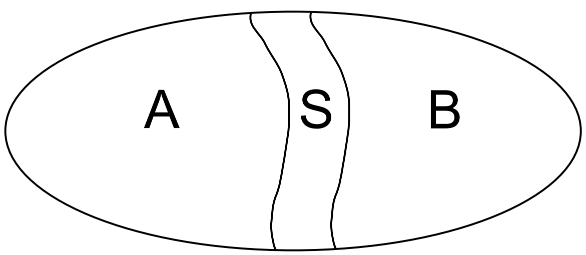
Sparse LU (or Cholesky) factorization framework

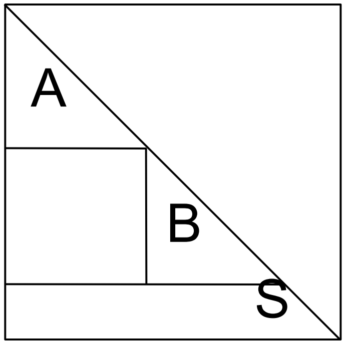
- 3 essential steps to factorize A in LU:
  - Order A:
    - Ordering
    - minimize the fill-in in L and U
  - Compute the structure of L and U:
    - Symbolic factorization
    - Schedule the numerical factorization
  - Compute the values of L and U:
    - Numerical factorization

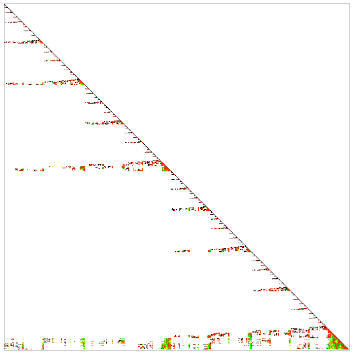


Nested dissection

- **Principle [George 1973]**
  - **Find a vertex separator *S* in graph.**
  - **Order vertices of *S* with highest available indices.**
  - **Recursively apply the algorithm to the two separated subgraphs *A* and *B.***

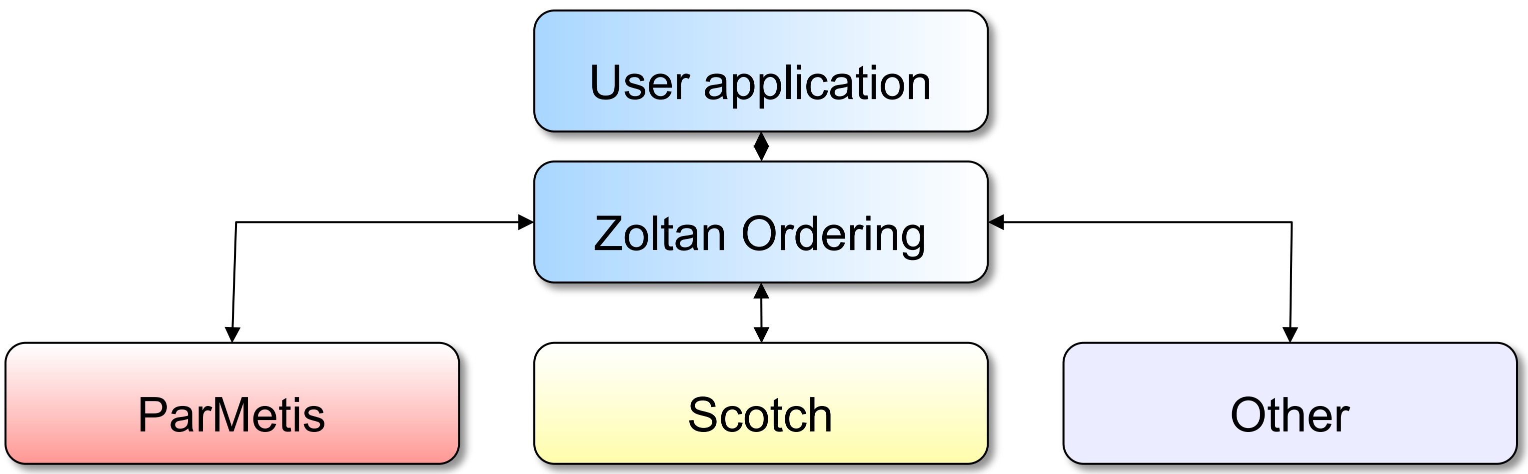






Matrix Ordering with Zoltan

- Computed by third party libraries:
  - ParMetis (U. Minnesota)
  - Scotch (INRIA Bordeaux)
  - Easy to add another
- The calls to third party libraries are transparent to the user, thus Zoltan's calls can be a standard way to compute ordering



Ordering interface in Zoltan

- Compute ordering with one function: Zoltan_Order
- Output provided:
  - New order of the unknowns (direct permutation)
  - Access to elimination tree, “block” view of the ordering
    - Suitable for parallel symbolic factorization
- Ordering can also be used through Isorropia for Trilinos User:
  - Direct support for Epetra matrices

Zoltan Graph Coloring

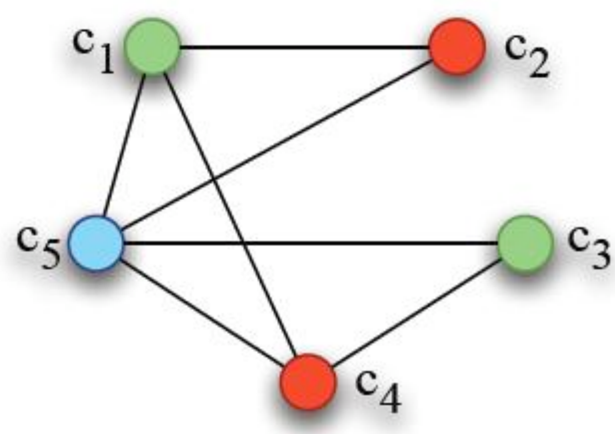
- **Parallel distance-1 and distance-2 graph coloring.**
- **Graph built using same application interface and code as graph partitioners.**
- **Generic coloring interface; easy to add new coloring algorithms.**
- **Algorithms**
  - **Distance-1 coloring: Bozdag, Gebremedhin, Manne, Boman, Catalyurek, *EuroPar'05, JPDC'08*.**
  - **Distance-2 coloring: Bozdag, Catalyurek, Gebremedhin, Manne, Boman, Ozguner, *HPCC'05, SISC'08* (in submission).**

Distance-1 Graph Coloring

- **Problem (NP-hard)**

  Color the vertices of a graph with as few colors as possible such that no two adjacent vertices receive the same color.

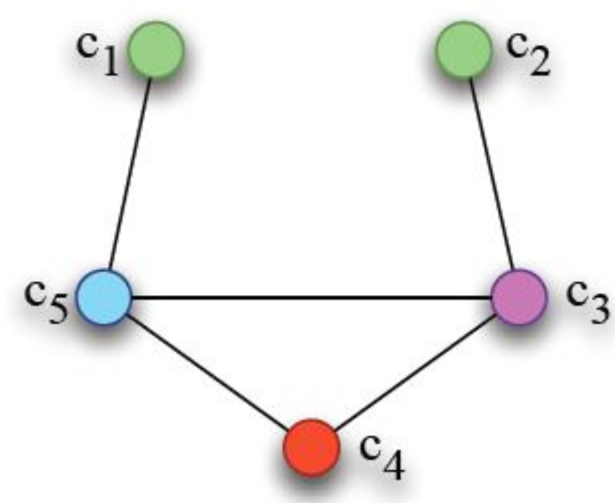
- **Applications**
  - **Iterative solution of sparse linear systems**
  - **Preconditioners**
  - **Sparse tiling**
  - **Eigenvalue computation**
  - **Parallel graph partitioning**



Distance-2 Graph Coloring

- **Problem (NP-hard)**
  Color the vertices of a graph with as few colors as possible such that a pair of vertices connected by a path on two or less edges receives different colors.

- **Applications**
  - **Derivative matrix computation in numerical optimization**
  - **Channel assignment**
  - **Facility location**
- **Related problems**
  - **Partial distance-2 coloring**
  - **Star coloring**



Coloring Interface in Zoltan

- Both distance-1 and distance-2 coloring routines can be invoked by Zoltan_Color function.
- The colors assigned to the objects are returned in an array.

For graph partitioning, coloring & ordering, use …

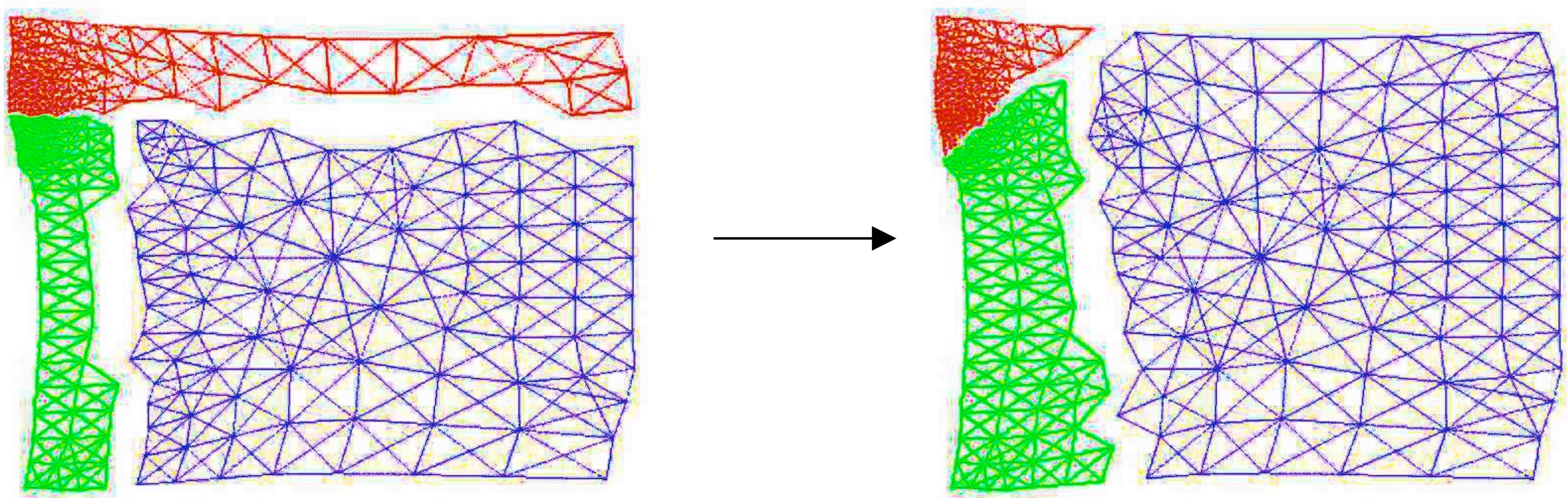
General Query Functions	
ZOLTAN_NUM_OBJ_FN	**Number of items on processor**
ZOLTAN_OBJ_LIST_FN	**List of item IDs and weights.**
Geometric Query Functions	
ZOLTAN_NUM_GEOM_FN	**Dimensionality of domain.**
ZOLTAN_GEOM_FN	**Coordinates of items.**
Hypergraph Query Functions	
ZOLTAN_HG_SIZE_CS_FN	**Number of hyperedge pins.**
ZOLTAN_HG_CS_FN	**List of hyperedge pins.**
ZOLTAN_HG_SIZE_EDGE_WTS_FN	**Number of hyperedge weights.**
ZOLTAN_HG_EDGE_WTS_FN	**List of hyperedge weights.**
Graph Query Functions	
ZOLTAN_NUM_EDGE_FN	**Number of graph edges.**
ZOLTAN_EDGE_LIST_FN	**List of graph edges and weights.**

Other Zoltan Functionality

- Tools needed when doing dynamic load balancing:
  - Data Migration
  - Unstructured Communication Primitives
  - Distributed Data Directories
- All functionalities are described in Zoltan User's Guide.
  - http://www.cs.sandia.gov/Zoltan/ug_html/ug.html

Zoltan Data Migration Tools

- **After partition is computed, data must be moved to new decomposition.**
  - **Depends strongly on application data structures.**
  - **Complicated communication patterns.**
- **Zoltan can help!**
  - **Application supplies query functions to pack/unpack data.**
  - **Zoltan does all communication to new processors.**



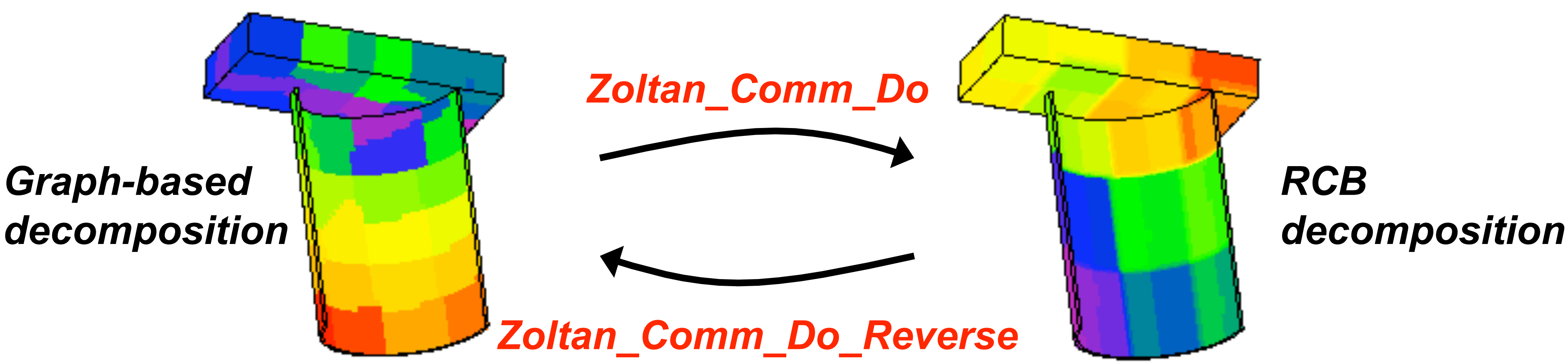
Using Zoltan's Data Migration Tools

- Required migration query functions:
  - ZOLTAN_OBJ_SIZE_MULTI_FN:
    - Returns size of data (in bytes) for each object to be exported to a new processor.
  - ZOLTAN_PACK_MULTI_FN:
    - Remove data from application data structure on old processor;
    - Copy data to Zoltan communication buffer.
  - ZOLTAN_UNPACK_MULTI_FN:
    - Copy data from Zoltan communication buffer into data structure on new processor.

-
```
int Zoltan_Migrate(struct Zoltan_Struct *zz,
        int num_import, ZOLTAN_ID_PTR import_global_ids,
        ZOLTAN_ID_PTR import_local_ids, int *import_procs,
        int *import_to_part,
        int num_export, ZOLTAN_ID_PTR export_global_ids,
        ZOLTAN_ID_PTR export_local_ids, int *export_procs,
        int *export_to_part);
```

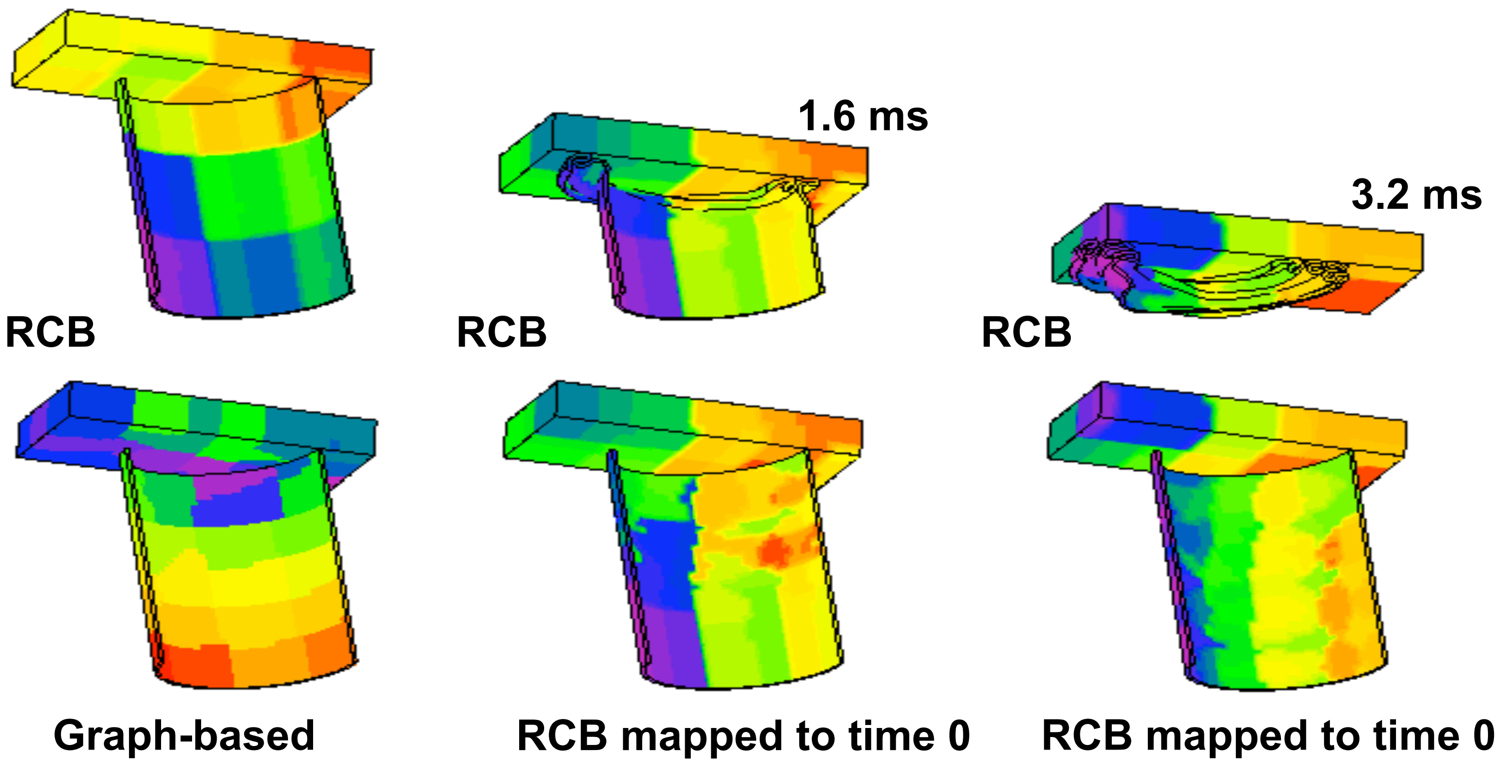
Zoltan Unstructured Communication Package

- Simple primitives for efficient irregular communication.
  - Zoltan_Comm_Create: Generates communication plan.
    - Processors and amount of data to send and receive.
  - Zoltan_Comm_Do: Send data using plan.
    - Can reuse plan. (Same plan, different data.)
  - Zoltan_Comm_Do_Reverse: Inverse communication.
- Used for most communication in Zoltan.



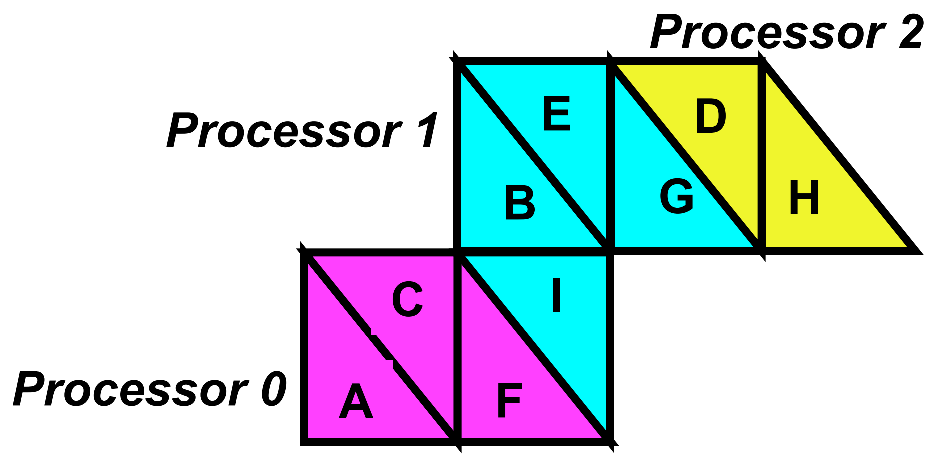
Example Application: Crash Simulations

- **Multiphase simulation:**
  - Graph-based decomposition of elements for finite element calculation.
  - Dynamic geometric decomposition of surfaces for contact detection.
  - Migration tools and Unstructured Communication package map between decompositions.



Zoltan Distributed Data Directory

- Helps applications locate off-processor data.
- Rendezvous algorithm (Pinar, 2001).
  - Directory distributed in known way (hashing) across processors.
  - Requests for object location sent to processor storing the object's directory entry.



Processor 0

Directory Index →	A	B	C
Location →	0	1	0

Processor 1

Directory Index →	D	E	F
Location →	2	1	0

Processor 2

Directory Index →	G	H	I
Location →	1	2	1

Alternate Interfaces to Zoltan

- **C, C++ and F90 interfaces in Zoltan.**
- **Matrix-based interface in Trilinos.**
- **Mesh-based interface in ITAPS.**

Isorropia: Trilinos Interface to Zoltan

- Trilinos Toolkit (M. Heroux, SNL, PI): Packages for …
  - Parallel matrix and vector classes (Epetra)
  - Linear, nonlinear and eigen solvers
  - Preconditioners
  - Matrix partitioning (Isorropia)
  - Time integration, discretizations, inline meshing, ….
- Epetra provides parallel matrix and vector classes.
- Isorropia uses Zoltan to repartition Epetra objects.
  - B = Isorropia::Epetra::createBalancedCopy(A, params); or
  - Partitioner, redistributor, and cost-evaluator classes.
- Trilinos v9.0 includes:
  - Zoltan in the Trilinos distribution and build environment.
  - Isorropia interfaces to matrix ordering and coloring.

(Member of SciDAC2 TOPS CET)

ITAPS Dynamic Services: Mesh-based Interface to Zoltan

- Interoperable Technologies for Advanced Petascale Simulations (L. Diachin, LLNL, PI)
  - SciDAC2 CET.
- ITAPS Goals:
  - Develop the next generation of meshing and geometry tools for petascale computing.
    - E.g., adaptive mesh refinement, shape optimization.
  - Improve applications' ability to use these tools.
    - "Standardization" of mesh interfaces.
- Dynamic Services toolkit:
  - ITAPS-compliant mesh interface to Zoltan tools.
  - Integration with ITAPS iMeshP parallel mesh interface to be released FY09.

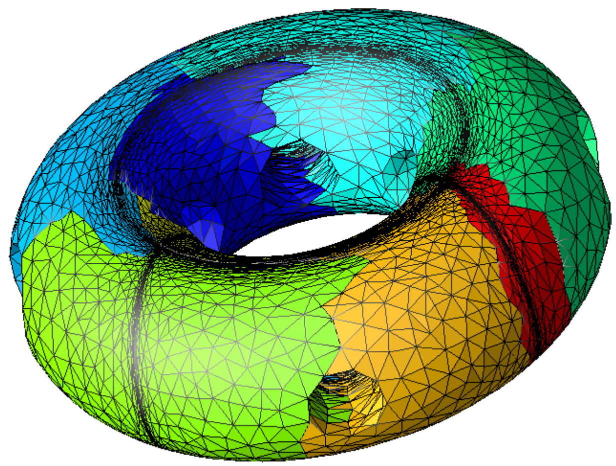


Image courtesy of M. Shephard, RPI

Current Work

- Two-dimensional matrix partitioning.
- Performance improvements for hypergraph partitioning.
- Multi-criteria hypergraph partitioning.
- May be available in Trilinos 10:
  - Non-symmetric matrix ordering.
  - Other coloring problems: partial distance 2, …

For More Information...

- Zoltan Home Page
  - http://www.cs.sandia.gov/Zoltan
  - User's and Developer's Guides
  - Tutorial: "Getting Started with Zoltan: A Short Tutorial"
  - Download Zoltan software under GNU LGPL.
- Trilinos Home Page
  - http://trilinos.sandia.gov
- ITAPS Home Page
  - http://www.itaps.org
- CSCAPES Home Page
  - http://www.cscapes.org
- Email:
  - zoltan-dev@software.sandia.gov
  - kddevin@sandia.gov