SAND2008-4024C

Tutorial: The Zoltan Toolkit

Karen Devine and Cedric Chevalier
Sandia National Laboratories, NM

Umit Catalyurek
Ohio State University

CSCAPES Workshop, June 2008

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.

Sandia National Laboratories

Outline

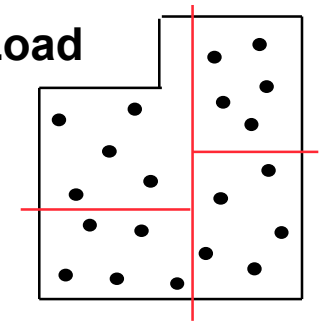
- High-level view of Zoltan
- Requirements, data models, and interface
- Dynamic Load Balancing and Partitioning
- Matrix Ordering
- Graph Coloring
- Utilities
- Alternate Interfaces
- Future Directions

Sandia National Laboratories

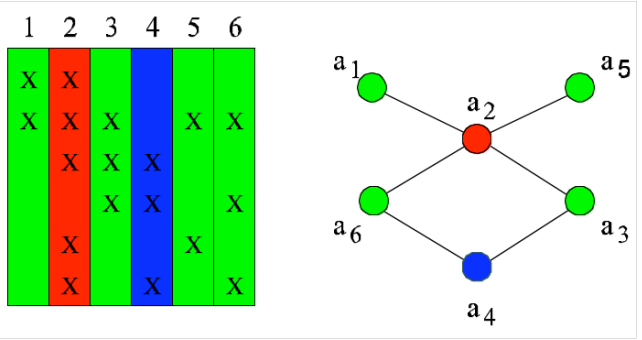
The Zoltan Toolkit

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

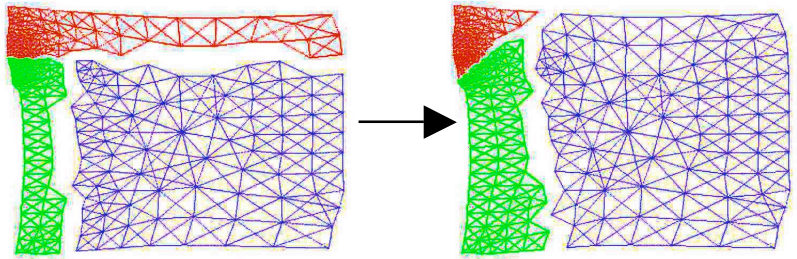
Dynamic Load Balancing



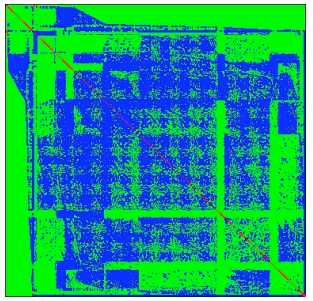
Graph Coloring



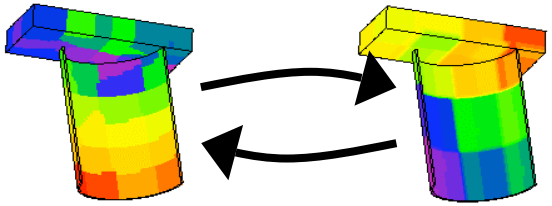
Data Migration



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

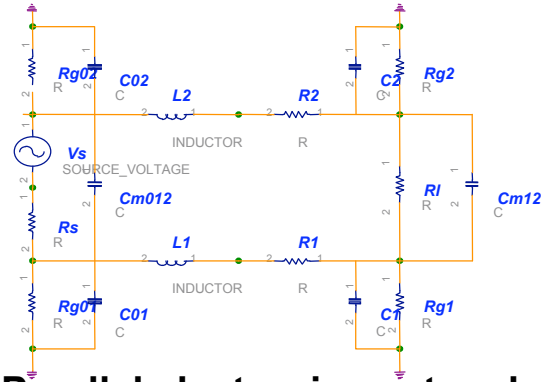
Sandia National Laboratories

- 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.
- Requirements:
  - MPI
  - C compiler
  - GNU Make (gmake)

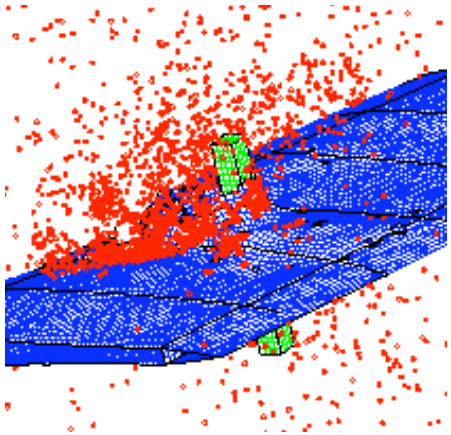
Zoltan Supports Many Applications

Sandia National Laboratories

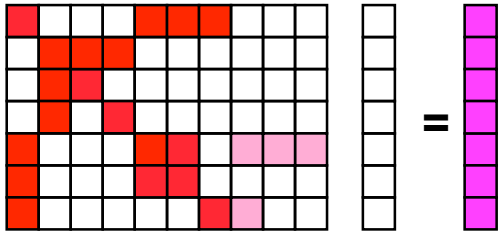
- **Different applications, requirements, data structures.**



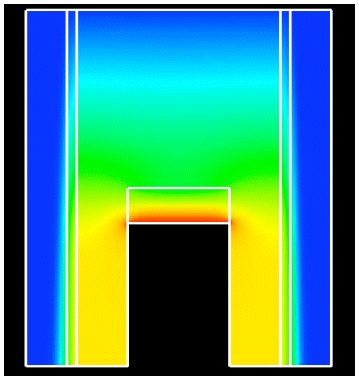
Parallel electronics networks



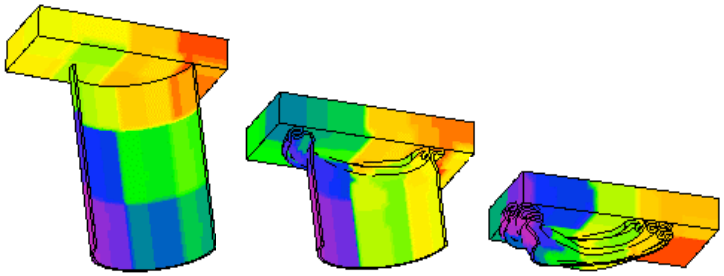
Particle methods



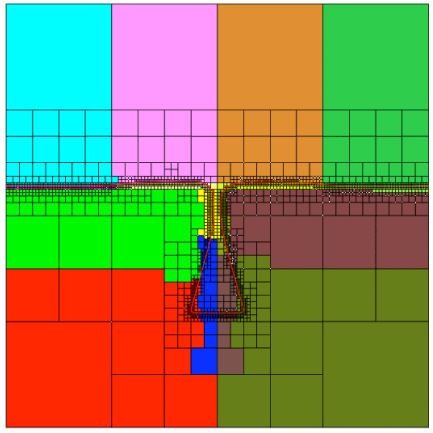
Linear solvers & preconditioners



Multiphysics simulations



Crash simulations



Adaptive mesh refinement

Zoltan Interface Design

Sandia National Laboratories

- 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.

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)**

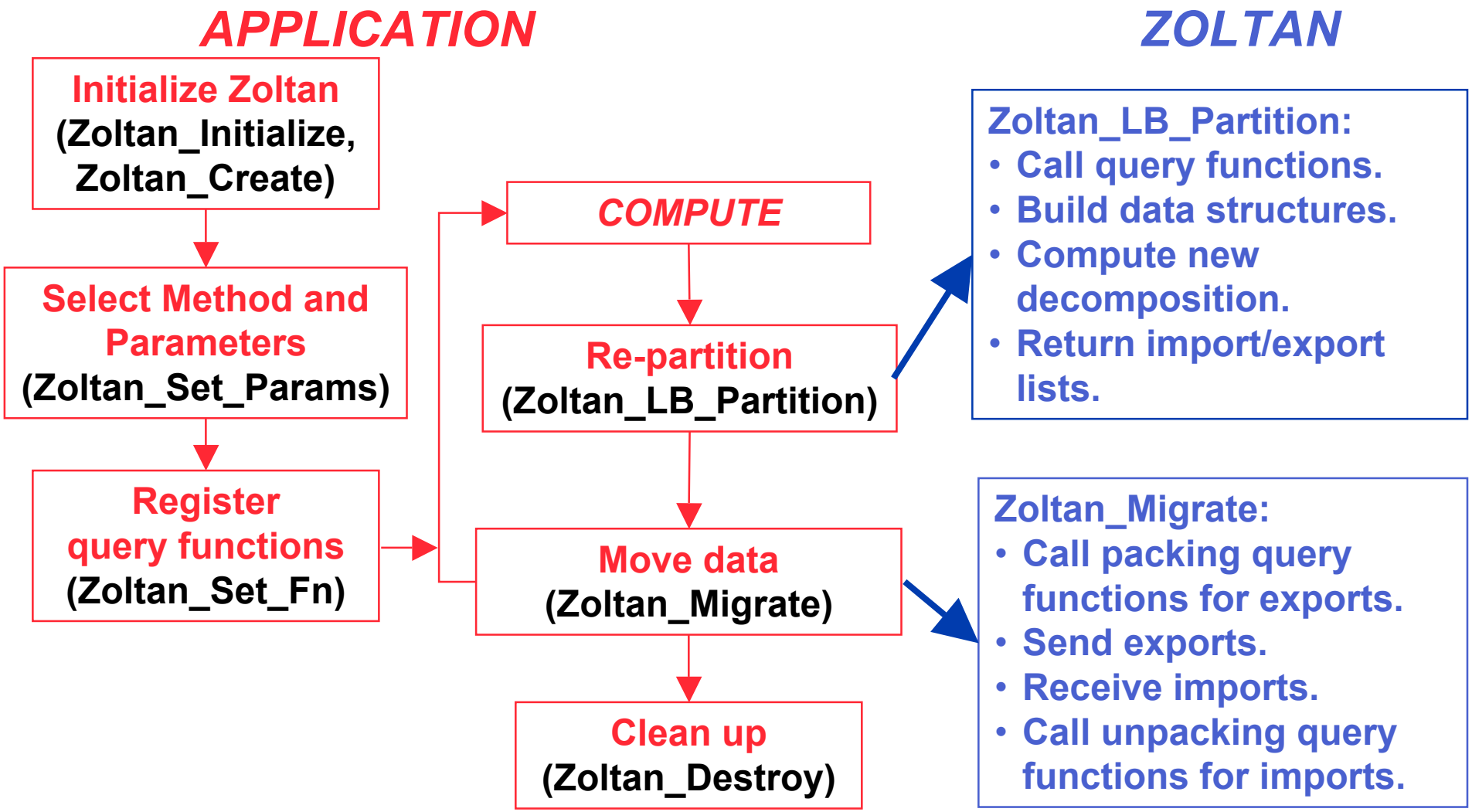
Zoltan Application Interface

Sandia National Laboratories

- **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



Zoltan Query Functions

Sandia National Laboratories

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.

Example zoltanSimple.c: ZOLTAN_OBJ_LIST_FN

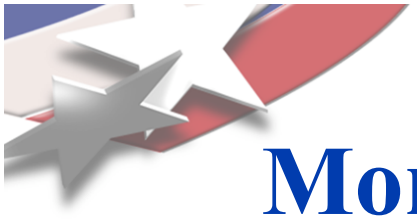
Sandia National Laboratories

```
void exGetObjectList(void *userDefinedData,
                     int numGlobalIds, int numLocalIds,
                     ZOLTAN_ID_PTR gids, ZOLTAN_ID_PTR lids,
                     int wgt_dim, float *obj_wgts,
                     int *err)
{
/* ZOLTAN_OBJ_LIST_FN callback function.
** Returns list of objects owned by this processor.
** lids[i] = local index of object in array.
*/
  int i;

  for (i=0; i<NumPoints; i++)
  {
    gids[i] = GlobalIds[i];
    lids[i] = i;
  }

  *err = 0;

  return;
}
```

Example zoltanSimple.c: ZOLTAN_GEOM_MULTI_FN

Sandia National Laboratories

```
void exGetObjectCoords(void *userDefinedData,
                       int numGlobalIds, int numLocalIds, int numObjs,
                       ZOLTAN_ID_PTR gids, ZOLTAN_ID_PTR lids,
                       int numDim, double *pts, int *err)
{
/* ZOLTAN_GEOM_MULTI_FN callback.
** Returns coordinates of objects listed in gids and lids.
*/
  int i, id, id3, next = 0;
  if (numDim != 3) {
    *err = 1; return;
  }
  for (i=0; i<numObjs; i++){
    id = lids[i];
    if ((id < 0) || (id >= NumPoints)) {
      *err = 1; return;
    }
    id3 = lids[i] * 3;
    pts[next++] = (double)(Points[id3]);
    pts[next++] = (double)(Points[id3 + 1]);
    pts[next++] = (double)(Points[id3 + 2]);
  }
}
```



More Details on Query Functions

Sandia National Laboratories

- **void* data pointer allows user data structures to be used in all query functions.**
  - **To use, cast the pointer to the application data type.**
- **Local IDs provided by application are returned by Zoltan to simplify access of application data.**
  - **E.g. Indices into local arrays of coordinates.**
- **`ZOLTAN_ID_PTR` is pointer to array of unsigned integers, allowing IDs to be more than one integer long.**
  - **E.g., (processor number, local element number) pair.**
  - **`numGlobalIds` and `numLocalIds` are lengths of each ID.**
- **All memory for query-function arguments is allocated in Zoltan.**

```
void ZOLTAN_GET_GEOM_MULTI_FN(void *userDefinedData,
                              int numGlobalIds, int numLocalIds, int numObjs,
                              ZOLTAN_ID_PTR gids, ZOLTAN_ID_PTR lids,
                              int numDim, double *pts, int *err)
```

Using Zoltan in Your Application

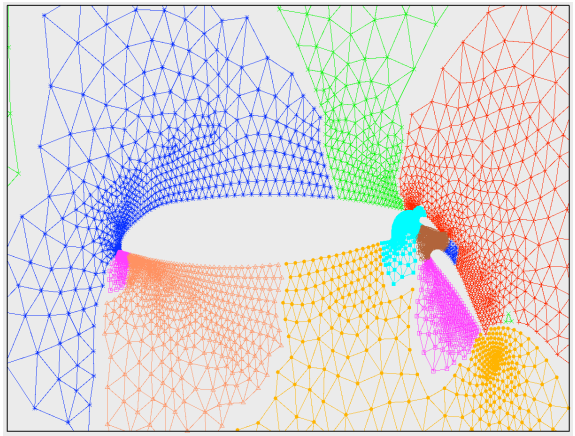
Sandia National Laboratories

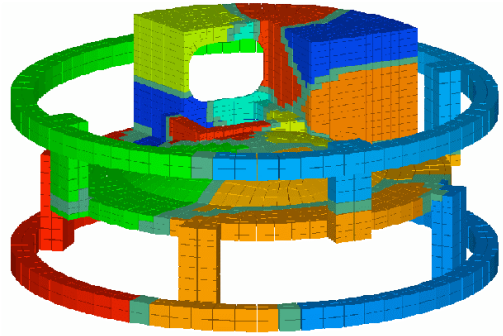
1. Decide what your objects are.
   - Elements? Grid points? Matrix rows? Particles?
2. Decide which tools (partitioning/ordering/coloring/utilities) and class of method (geometric/graph/hypergraph) to use.
3. Download Zoltan.
   - http://www.cs.sandia.gov/Zoltan
4. Write required query functions for 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. Edit Zoltan configuration file and build Zoltan.
8. Compile application; link with libzoltan.a.
   - mpicc application.c -lzoltan

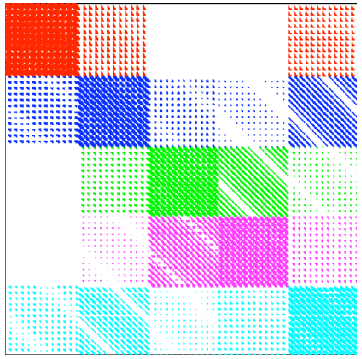
Sandia National Laboratories

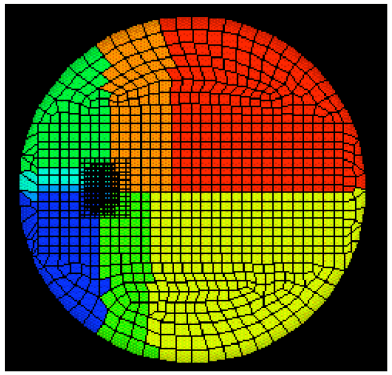
Partitioning and Load Balancing

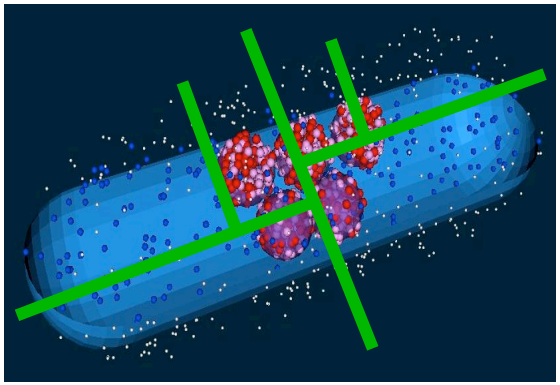
- Assignment of application data to processors for parallel computation.
- Applied to grid points, elements, matrix rows, particles, ….











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)

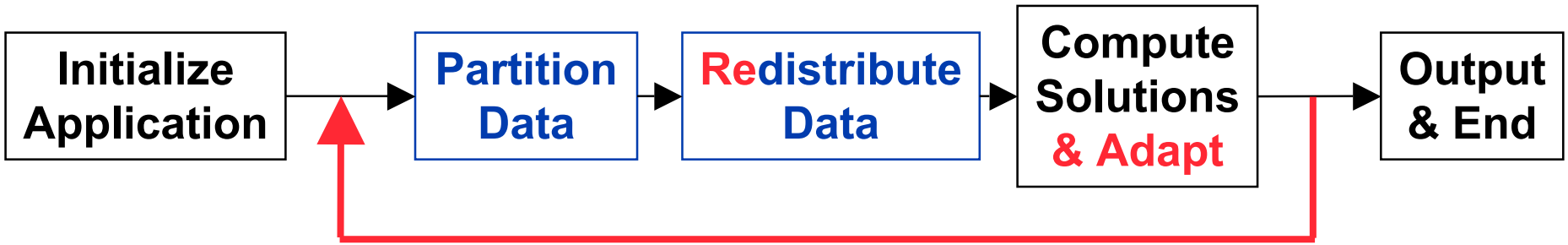
```
err = Zoltan_LB_Partition(zz,
    &changes, /* Flag indicating whether partition changed */
    &numGidEntries, &numLidEntries,
    &numImport, /* objects to be imported to new part */
    &importGlobalGids, &importLocalGids, &importProcs, &importToPart,
    &numExport, /* # objects to be exported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

Sandia National Laboratories

Static Partitioning

- 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

Sandia National Laboratories

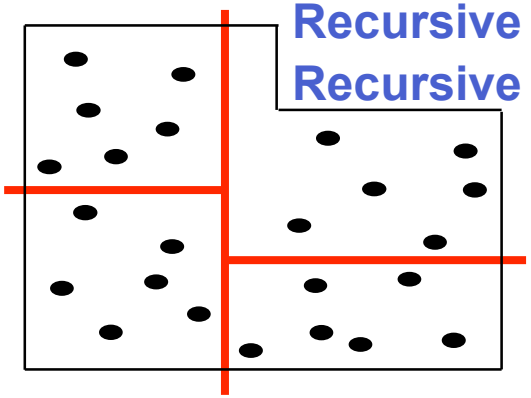
- No single partitioner works best for all applications.
  - Trade-offs:
    - Quality vs. speed.
    - Geometric locality vs. data dependencies.
    - High-data movement costs vs. tolerance for remapping.
- Application developers may not know which partitioner is best for application.

- 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

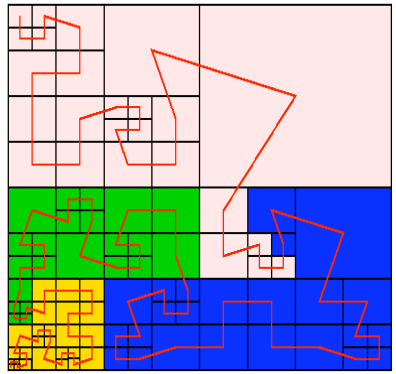
Sandia National Laboratories

Geometric (coordinate-based) methods



Recursive Coordinate Bisection (Berger, Bokhari)

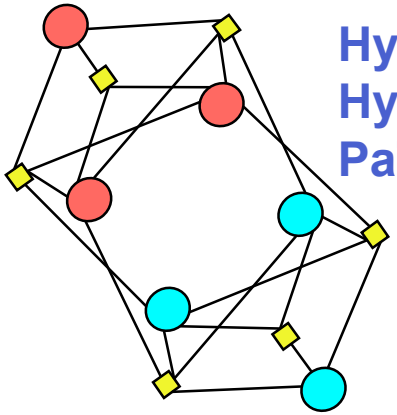
Recursive Inertial Bisection (Taylor, Nour-Omid)



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

Refinement-tree Partitioning (Mitchell)

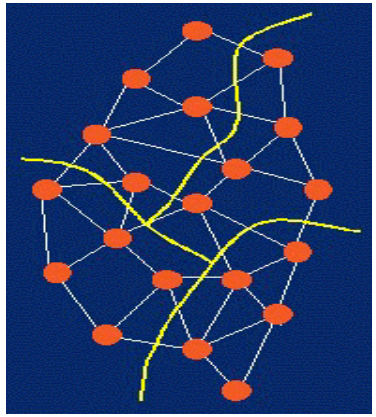
Combinatorial (topology-based) methods



Hypergraph Partitioning

Hypergraph Repartitioning

PaToH (Catalyurek & Aykanat)



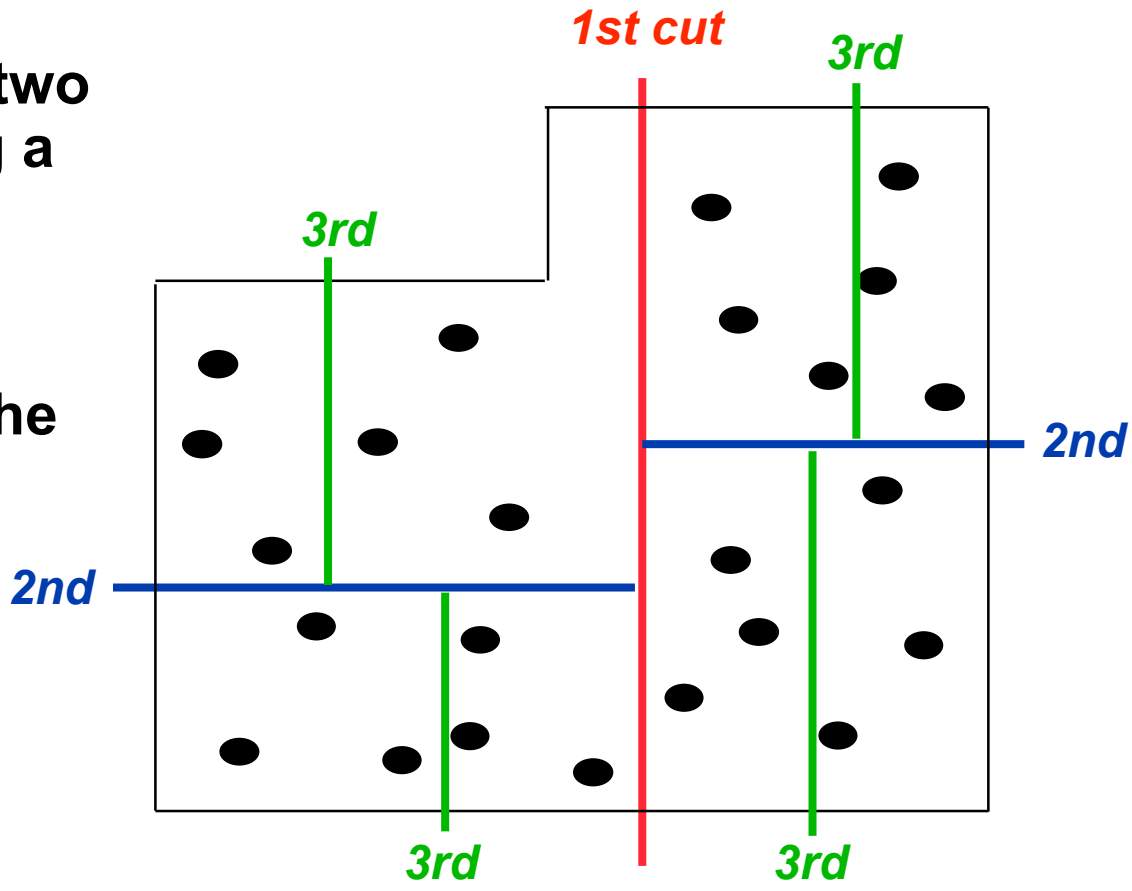
Zoltan Graph Partitioning

ParMETIS (U. Minnesota)

Jostle (U. Greenwich)

Recursive Coordinate Bisection

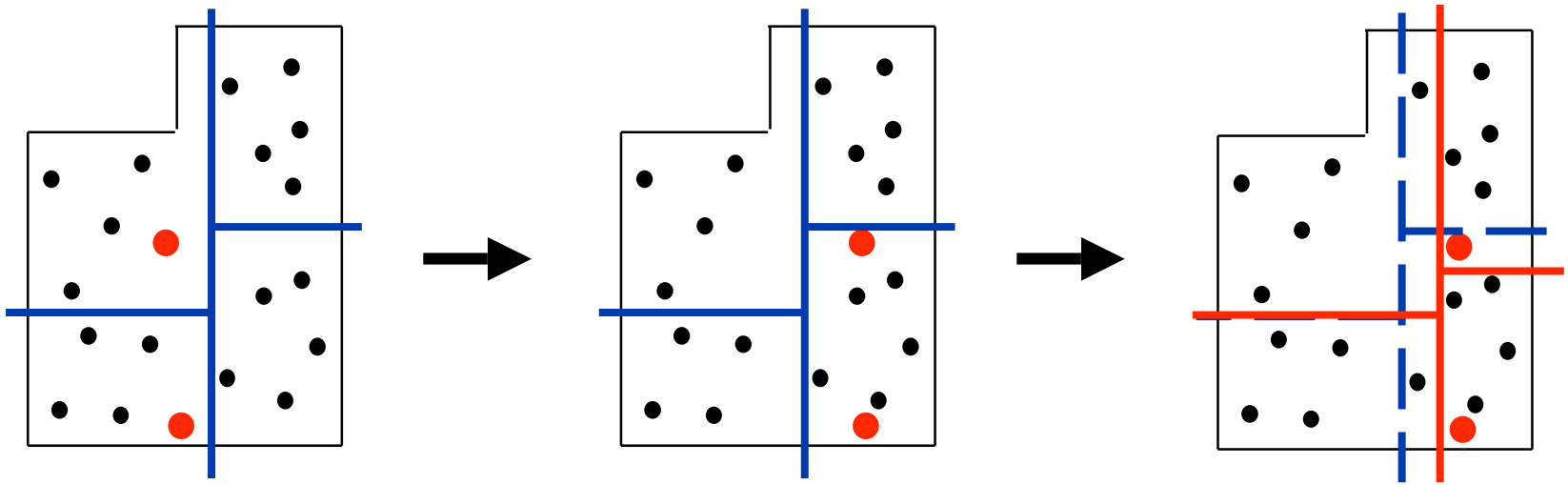
- Zoltan_Set_Param(zz, "LB_METHOD", "RCB");
- Berger & Bokhari (1987).
- Idea:
  - Divide work into two equal parts using a cutting plane orthogonal to a coordinate axis.
  - Recursively cut the resulting subdomains.



Sandia National Laboratories

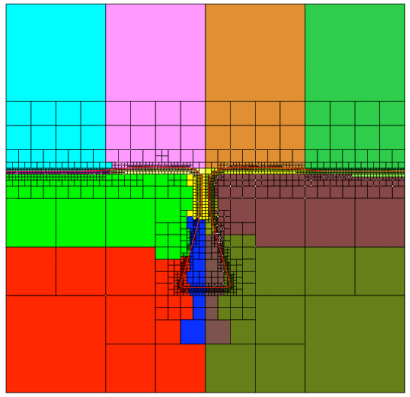
Geometric Repartitioning

- Implicitly achieves low data redistribution costs.
- For small changes in data, cuts move only slightly, resulting in little data redistribution.

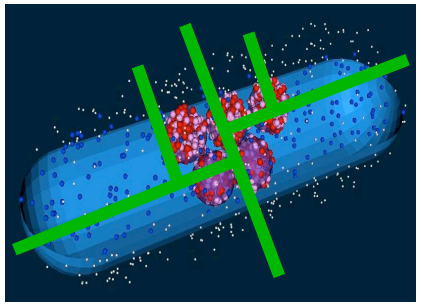


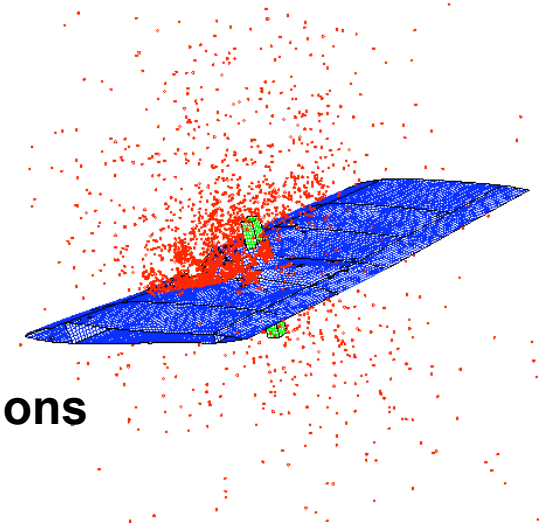
Sandia National Laboratories

Applications of Geometric Methods

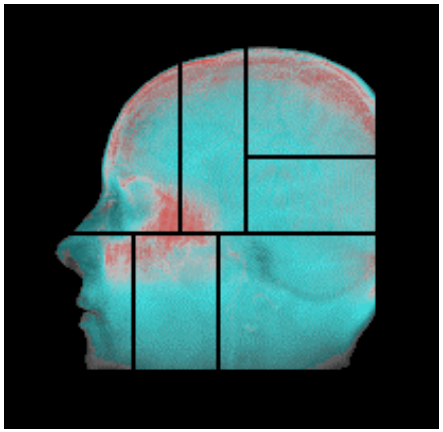


Adaptive Mesh Refinement

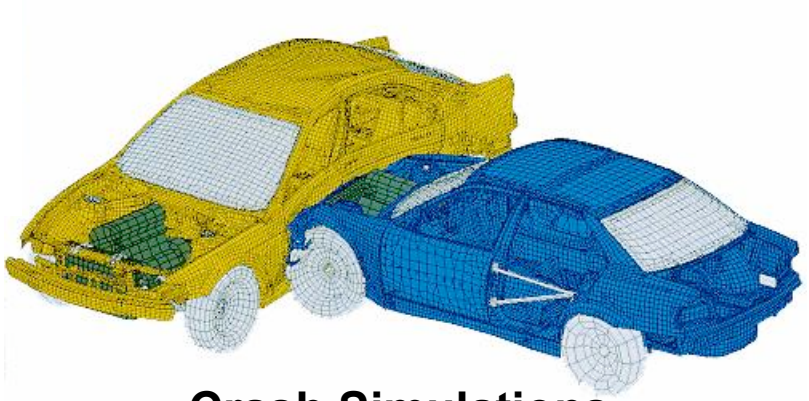




Particle Simulations



Parallel Volume Rendering



Crash Simulations
and Contact Detection

Sandia National Laboratories

RCB Advantages and Disadvantages

- **Advantages:**
  - **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.**

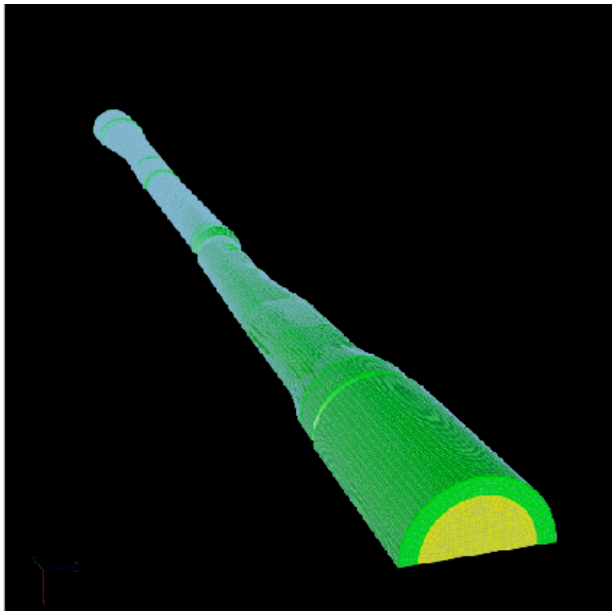
SLAC'S 55-cell Linear Accelerator with couplers: One-dimensional RCB partition reduced runtime up to 68% on 512 processor IBM SP3. (Wolf, Ko)

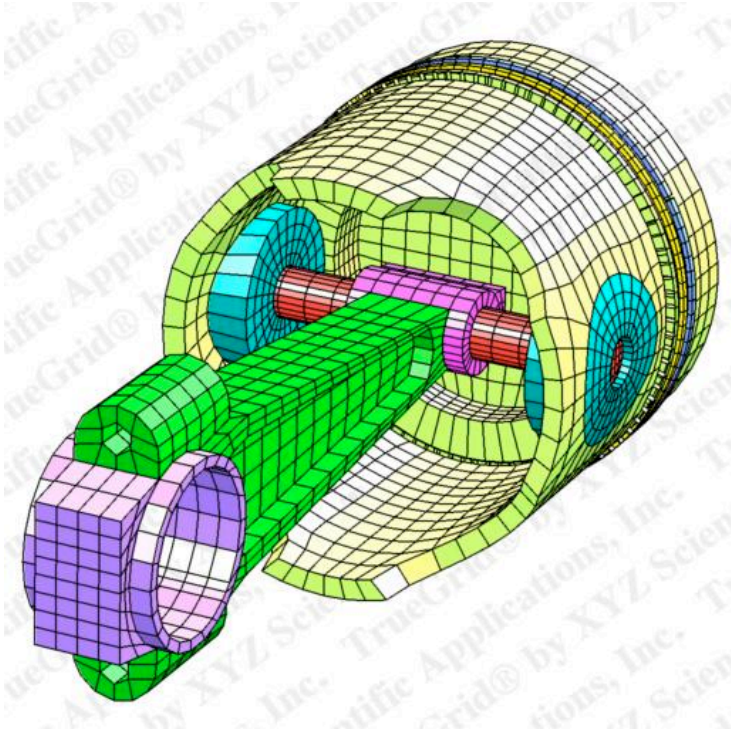
- **Disadvantages:**
  - **No explicit control of communication costs.**
  - **Mediocre partition quality.**
  - **Can generate disconnected subdomains for complex geometries.**
  - **Need coordinate information.**

Variations on RCB : Recursive Inertial Bisection

Sandia National Laboratories

- Zoltan_Set_Param(zz, “LB_METHOD”, “RIB”);
- Simon, Taylor, et al., 1991
- Cutting planes orthogonal to principle axes of geometry.
- Not incremental.

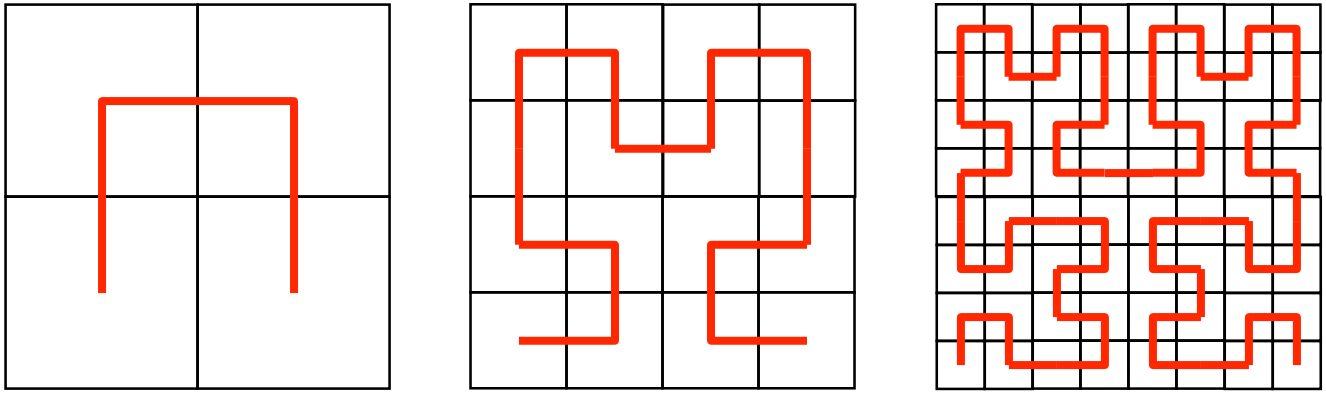




Space-Filling Curve Partitioning (SFC)

Sandia National Laboratories

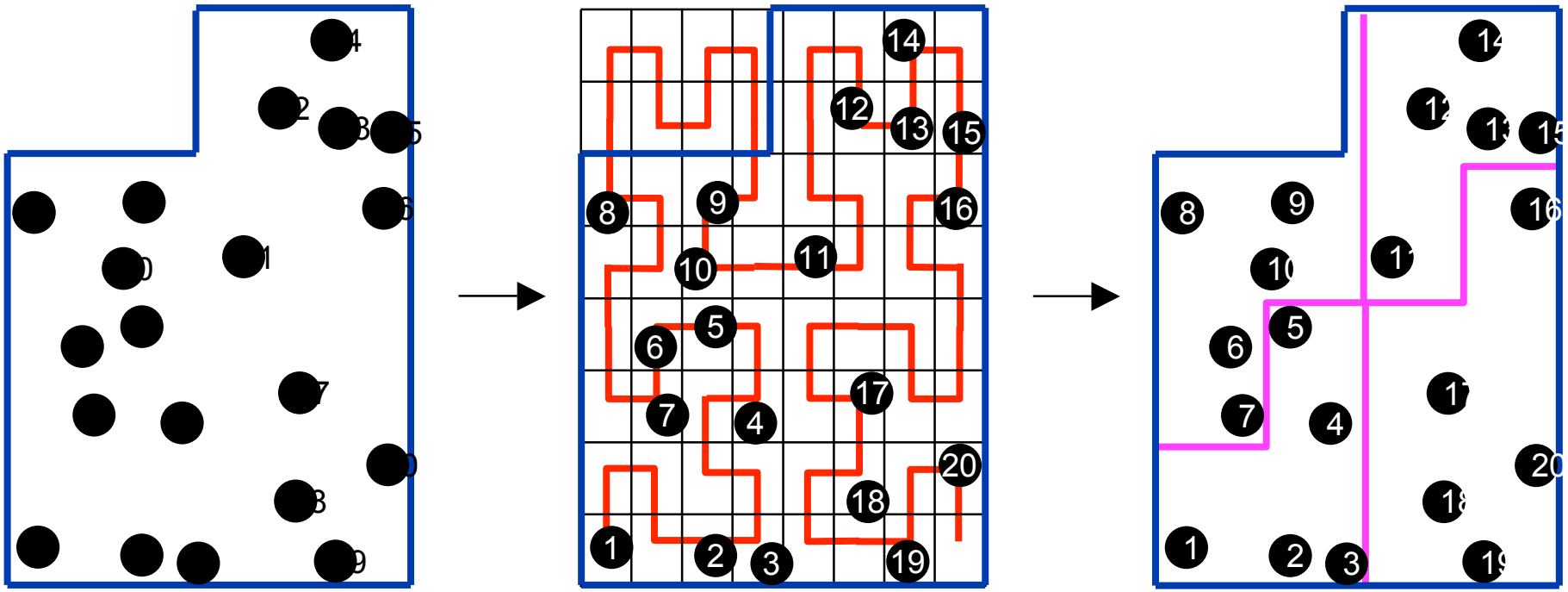
- Zoltan_Set_Param(zz, "LB_METHOD", "HSFC");
- Space-Filling Curve (Peano, 1890):
  - Mapping between R^3 to R^1 that completely fills a domain.
  - Applied recursively to obtain desired granularity.
- Used for partitioning by …
  - Warren and Salmon, 1993, gravitational simulations.
  - Pilkington and Baden, 1994, smoothed particle hydrodynamics.
  - Patra and Oden, 1995, adaptive mesh refinement.



Sandia National Laboratories

SFC Algorithm

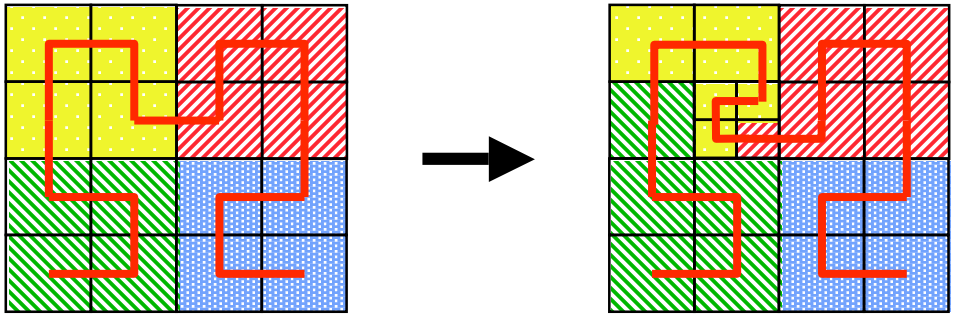
- Run space-filling curve through domain.
- Order objects according to position on curve.
- Perform 1-D partition of curve.



SFC Advantages and Disadvantages

Sandia National Laboratories

- **Advantages:**
  - **Simple, fast, inexpensive.**
  - **Maintains geometric locality of objects in processors.**
  - **Implicitly incremental for repartitioning.**

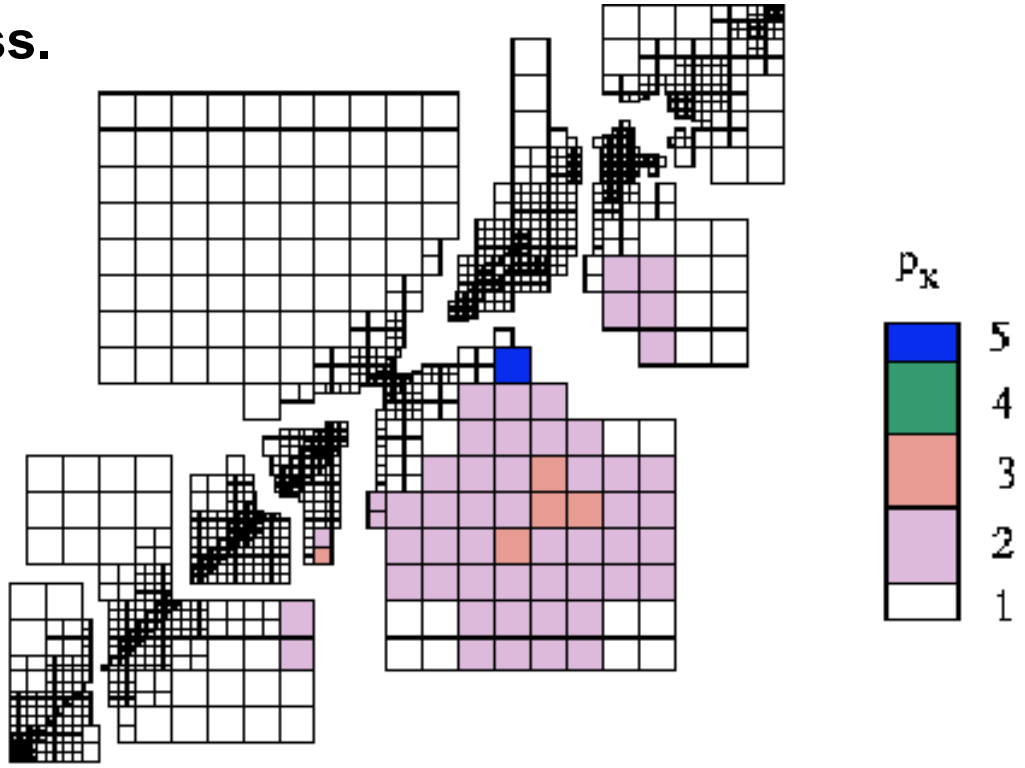


- **Disadvantages:**
  - **No explicit control of communication costs.**
  - **Can generate disconnected subdomains.**
  - **Often lower quality partitions than RCB.**
  - **Geometric coordinates needed.**

Sandia National Laboratories

Applications using SFC

- **Adaptive hp-refinement finite element methods.**
  - **Assigns physically close elements to same processor.**
  - **Inexpensive; incremental; fast.**
  - **Linear ordering can be used to order elements for efficient memory access.**



hp-refinement mesh; 8 processors.
Patra, et al. (SUNY-Buffalo)

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

Sandia National Laboratories

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.

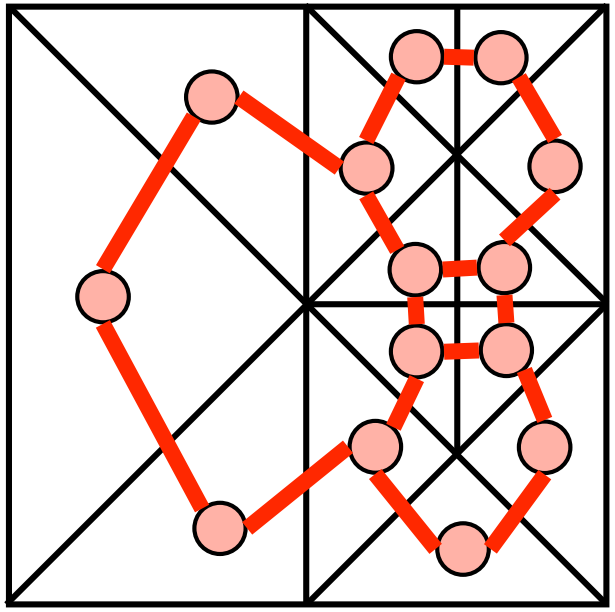
Sandia National Laboratories

Graph Partitioning

- Zoltan_Set_Param(zz, "LB_METHOD", "GRAPH");
- Zoltan_Set_Param(zz, "GRAPH_PACKAGE", "ZOLTAN"); or
  Zoltan_Set_Param(zz, "GRAPH_PACKAGE", "PARMETIS");

- Kernighan, Lin, Schweikert, Fiduccia, Mattheyes, Simon, Hendrickson, Leland, Kumar, Karypis, et al.

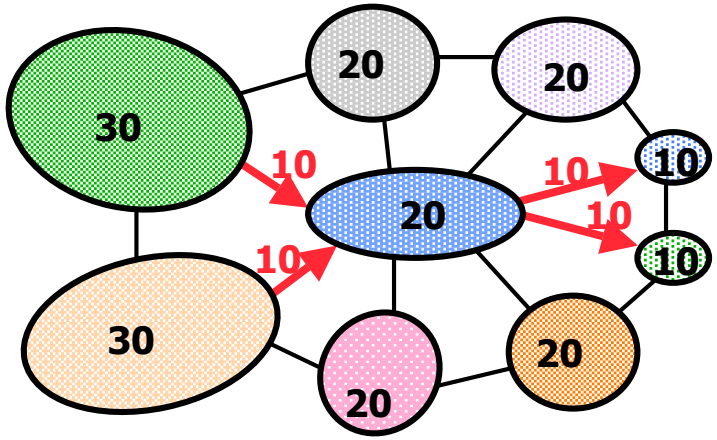
- Represent problem as a weighted graph.
  - Vertices = objects to be partitioned.
  - Edges = dependencies between two objects.
  - Weights = work load or amount of dependency.
- Partition graph so that …
  - Parts have equal vertex weight.
  - Weight of edges cut by part boundaries is small.



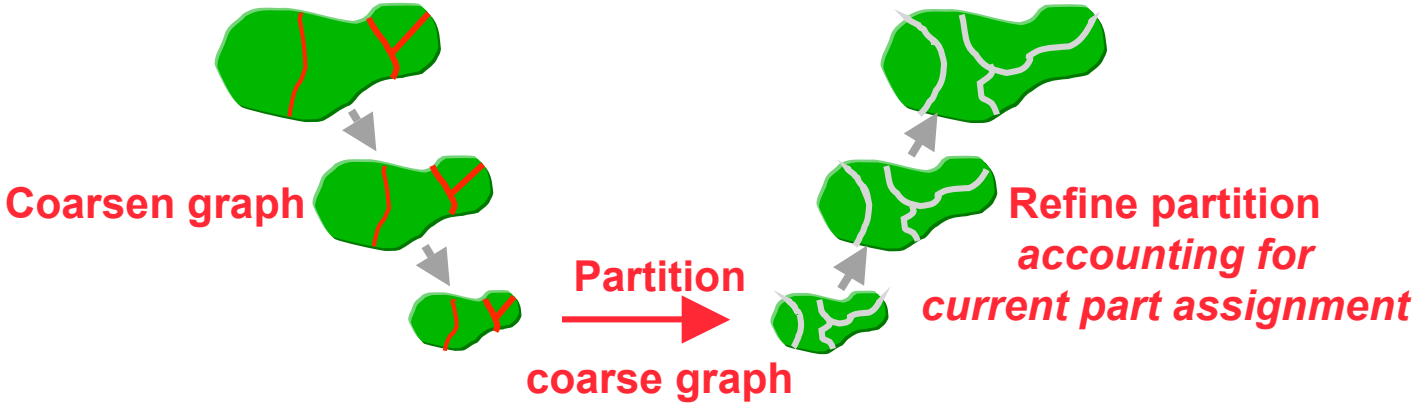
Sandia National Laboratories

Graph Repartitioning

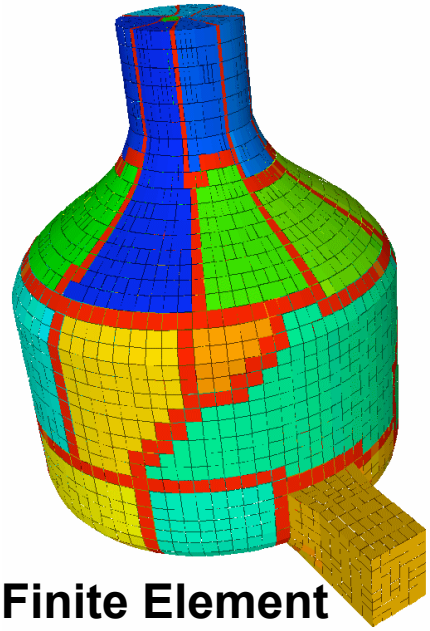
- **Diffusive strategies (Cybenko, Hu, Blake, Walshaw, Schloegel, et al.)**
  - **Shift work from highly loaded processors to less loaded neighbors.**
  - **Local communication keeps data redistribution costs low.**



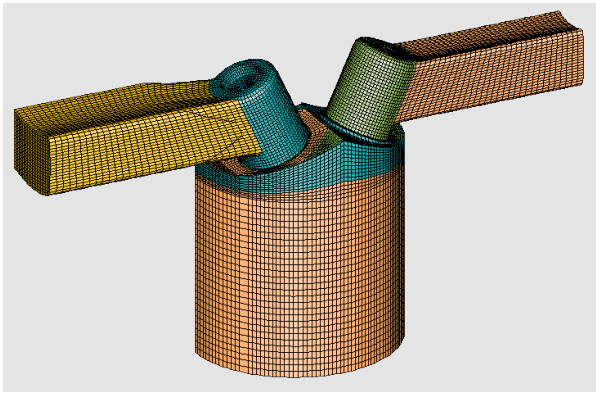
- **Multilevel partitioners that account for data redistribution costs in refining partitions (Schloegel, Karypis)**
  - **Parameter weights application communication vs. redistribution communication.**



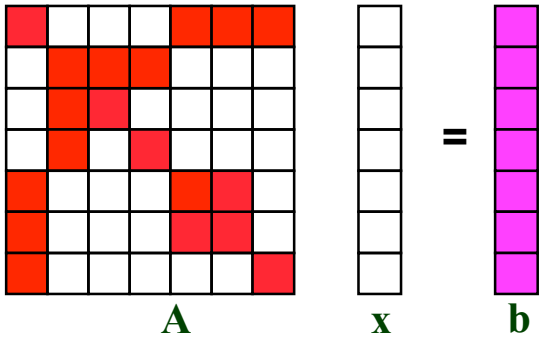
Applications using Graph Partitioning



Finite Element Analysis



Multiphysics and multiphase simulations

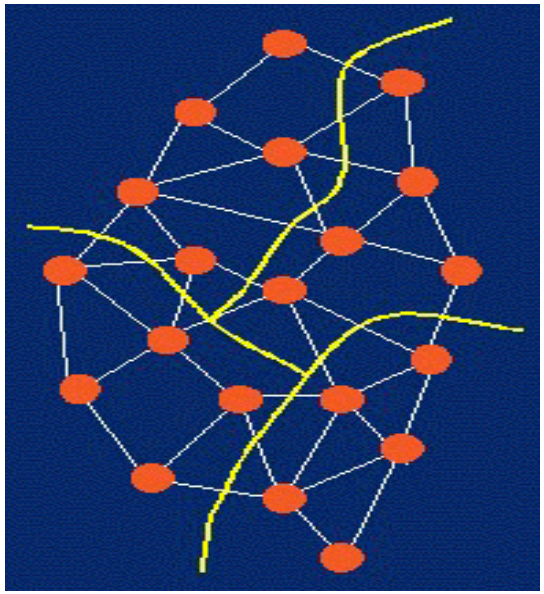


Linear solvers & preconditioners
(square, structurally symmetric systems)

Graph Partitioning: Advantages and Disadvantages

Sandia National Laboratories

- **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)



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

For graph partitioning, coloring & ordering, use …

Sandia National Laboratories

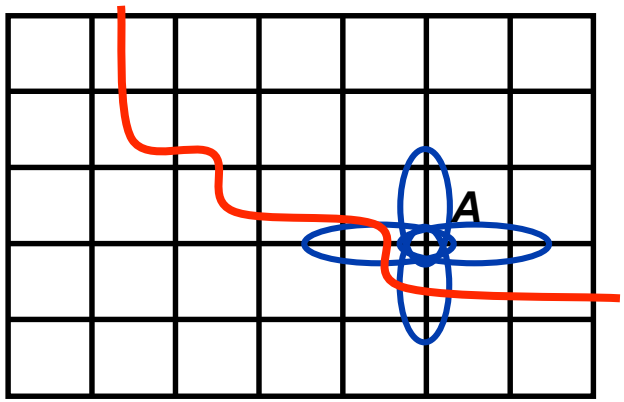
General Query Functions	
`ZOLTAN_NUM_OBJ_FN`	Number of items on processor
`ZOLTAN_OBJ_LIST_FN`	List of item IDs and weights.
Geometric Query Functions	
`ZOLTAN_NUM_GEOM_FN`	Dimensionality of domain.
`ZOLTAN_GEOM_FN`	Coordinates of items.
Hypergraph Query Functions	
`ZOLTAN_HG_SIZE_CS_FN`	Number of hyperedge 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

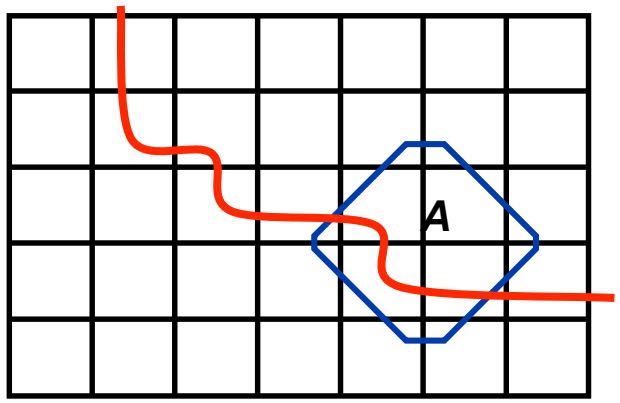
Hypergraph Partitioning

- 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

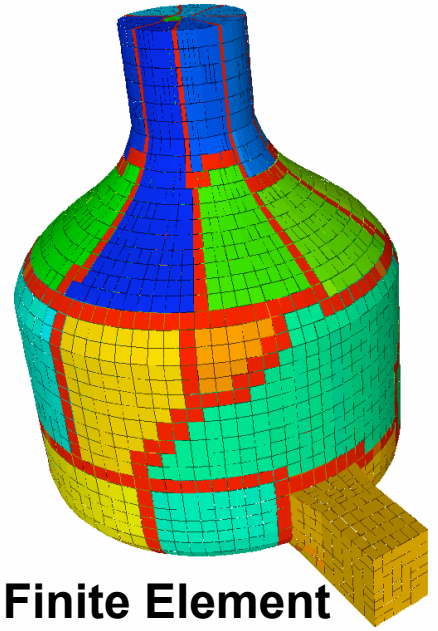
  + Application communication volume

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

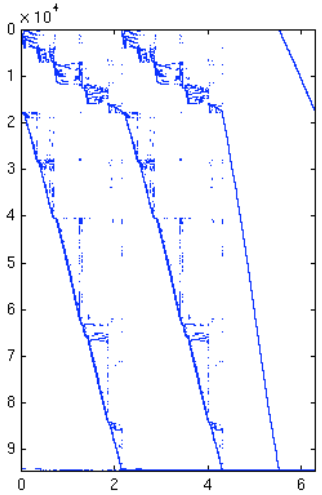
Best Algorithms Paper Award at IPDPS07
"Hypergraph-based Dynamic Load Balancing for Adaptive Scientific Computations"
Catalyurek, Boman, Devine, Bozdag, Heaphy, & Riesen

Sandia National Laboratories

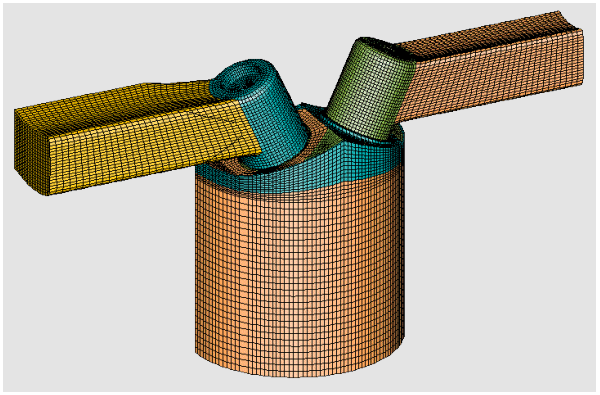
Hypergraph Applications



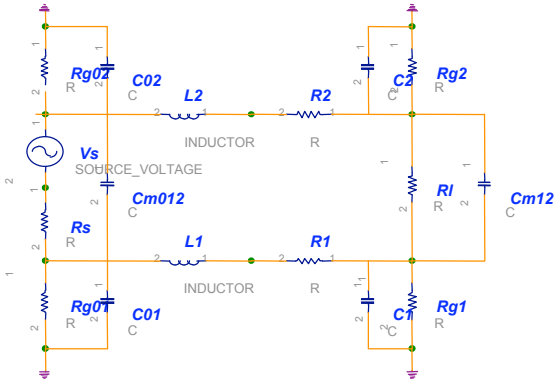
Finite Element Analysis



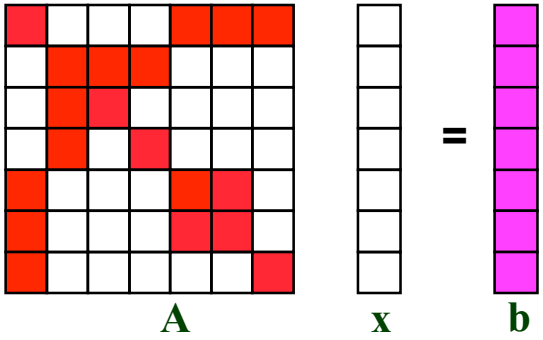
Linear programming for sensor placement



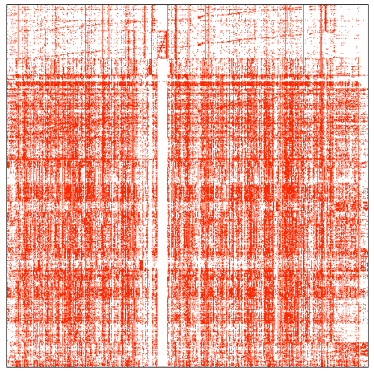
Multiphysics and multiphase simulations



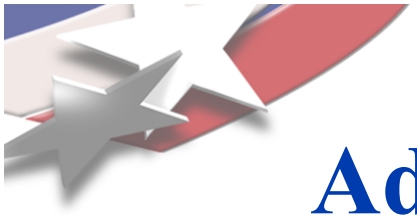
Circuit Simulations



Linear solvers & preconditioners
(no restrictions on matrix structure)



Data Mining



Hypergraph Partitioning: Advantages and Disadvantages

Sandia National Laboratories

- **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:**
  - **More expensive than graph partitioning.**

For hypergraph partitioning and repartitioning, use …

Sandia National Laboratories

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.

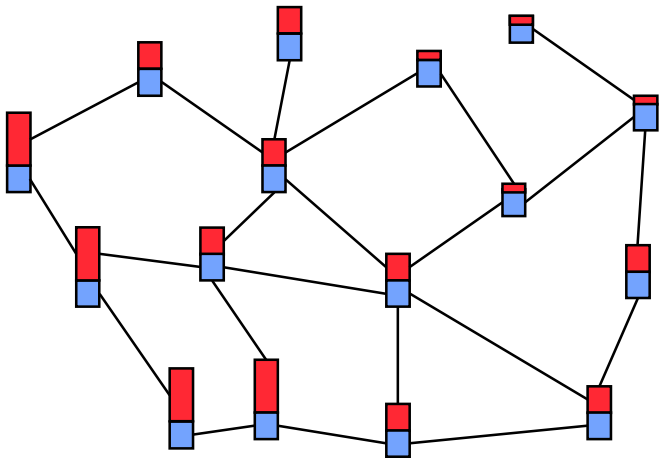
Or can use graph queries to build hypergraph.

Sandia National Laboratories

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.

Multi-criteria Load-balancing

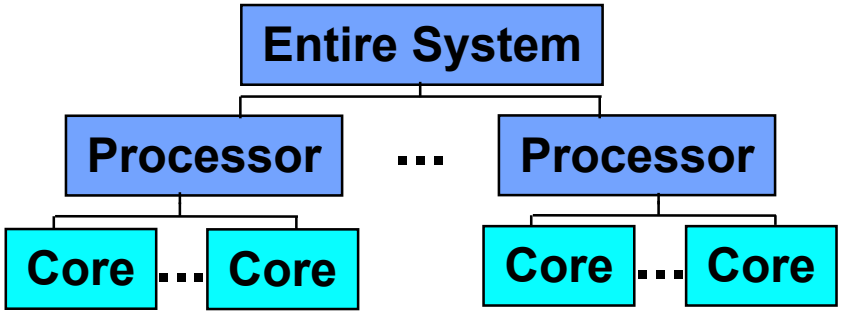
- Multiple constraints or objectives
  - Compute a single partition that is good with respect to multiple factors.
    - Balance both computation and memory.
    - Balance meshes in loosely coupled physics.
    - Balance multi-phase simulations.
  - Extend algorithms to multiple weights
    - Difficult.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

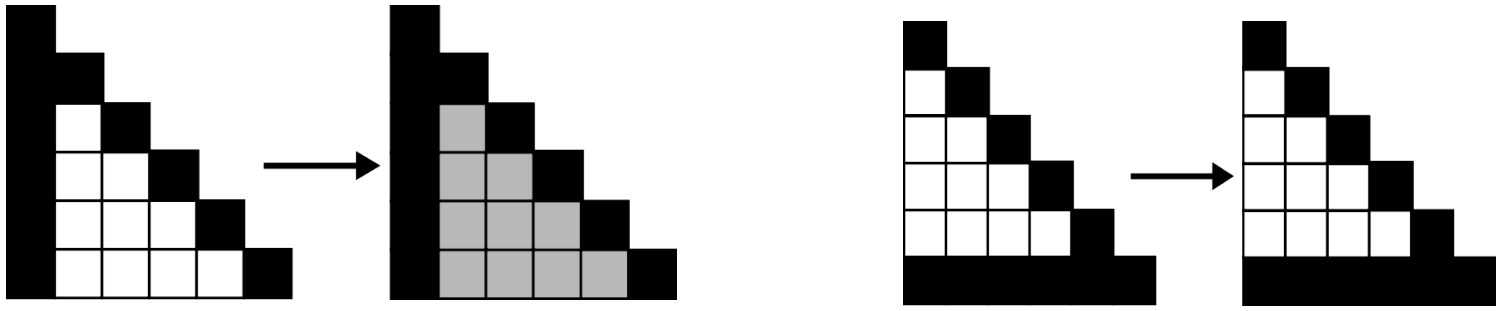
- Hierarchical partitioning: Allows different partitioners at different architecture levels.
  - Zoltan_Set_Param(zz, "LB_METHOD", "HIER");



Sandia National Laboratories

Sparse Matrix Ordering problem

- When solving sparse linear systems with direct methods, non-zero terms are created during the factorization process ($A \rightarrow LL^t$, $A \rightarrow LDL^t$ or $A \rightarrow LU$).
- Fill-in depends on the order of the unknowns.
  - Need to provide fill-reducing orderings.



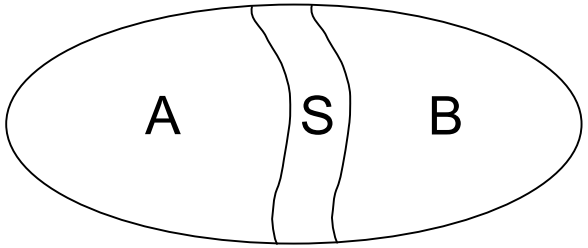
Sandia National Laboratories

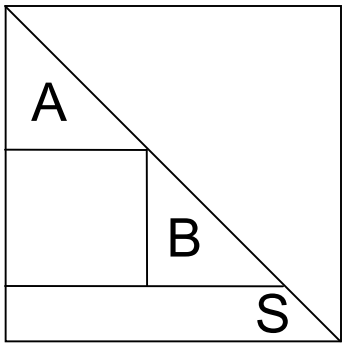
Fill Reducing ordering

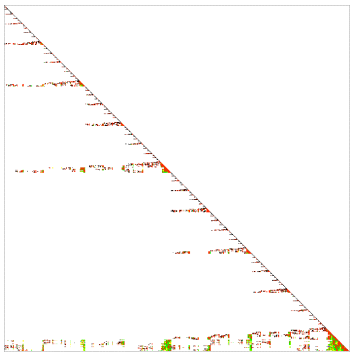
- **Combinatorial problem, depending on only the structure of the matrix *A*:**
  - **We can work on the graph associated with *A*.**
- **NP-Complete, thus we deal only with heuristics.**
- **Most popular heuristics:**
  - **Minimum Degree algorithms (AMD, MMD, AMF …)**
  - **Nested Dissection**

Nested dissection (1)

- **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*.**







Sandia National Laboratories

Nested dissection (2)

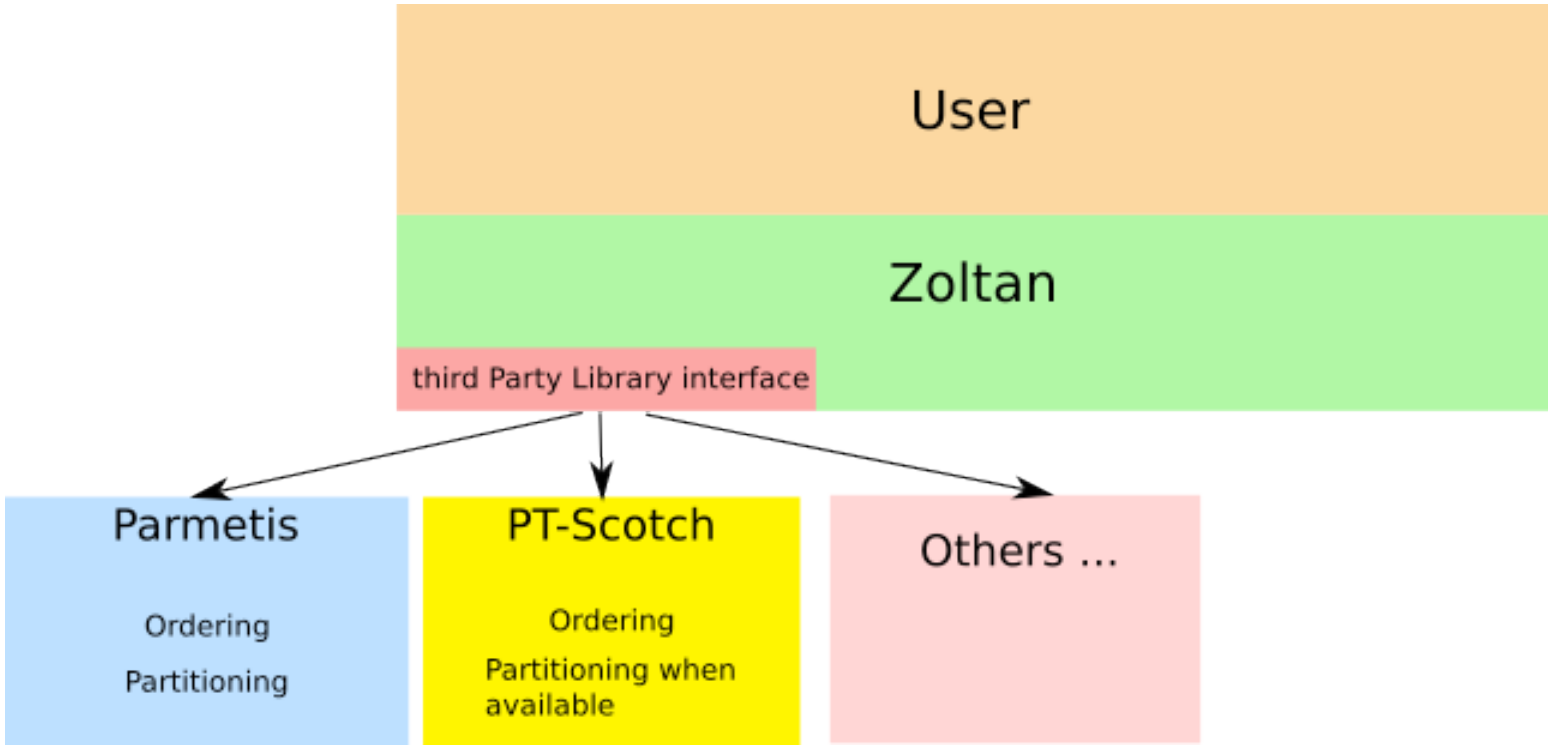
- **Advantages:**
  - **Induces high quality block decompositions.**
    - **Suitable for block BLAS 3 computations.**
  - **Increases the concurrency of computations.**
    - **Compared to minimum degree algorithms.**
    - **Very suitable for parallel factorization.**
      - **It's the scope here: parallel ordering is for parallel factorization.**

Matrix ordering within Zoltan

- Computed by third party libraries:
  - ParMETIS
  - Scotch (more specifically PT-Scotch, the parallel part)
  - Easy to add another one.
- The calls to the external ordering library are transparent for the user, and thus Zoltan's call can be a standard way to compute ordering.

Zoltan ordering architecture

Sandia National Laboratories



Ordering interface in Zoltan

- Compute ordering with one function:

  Zoltan_Order

- Output provided:
  - New order of the unknowns (direct permutation), available in two forms:
    - one is the new number in the interval [0,N-1];
    - the other is the new order of Global IDs.
  - Access to elimination tree, “block” view of the ordering.

Sandia National Laboratories

Experimental results (1)

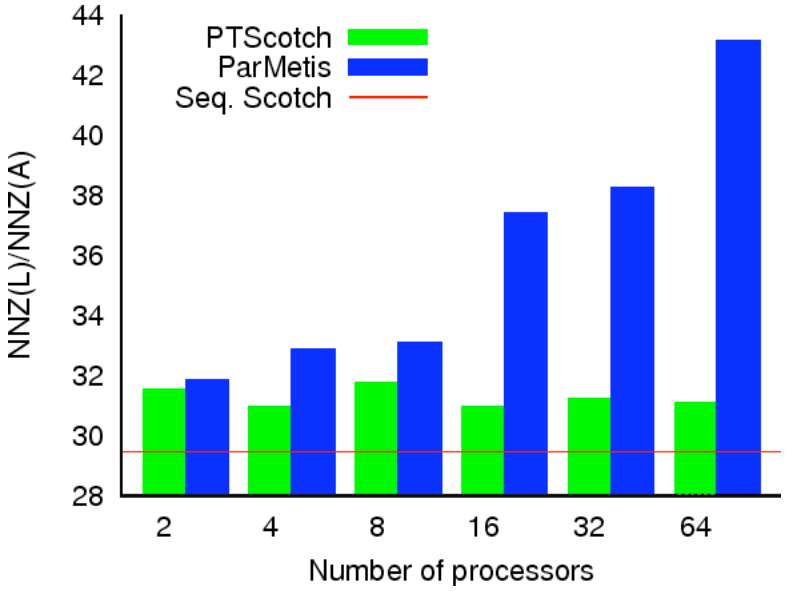
- **Metric is OPC, the operation count of Cholesky factorization.**
- **Largest matrix ordered by PT-Scotch: 83 millions of unknowns on 256 processors (CEA/CESTA).**
- **Some of our largest test graphs.**

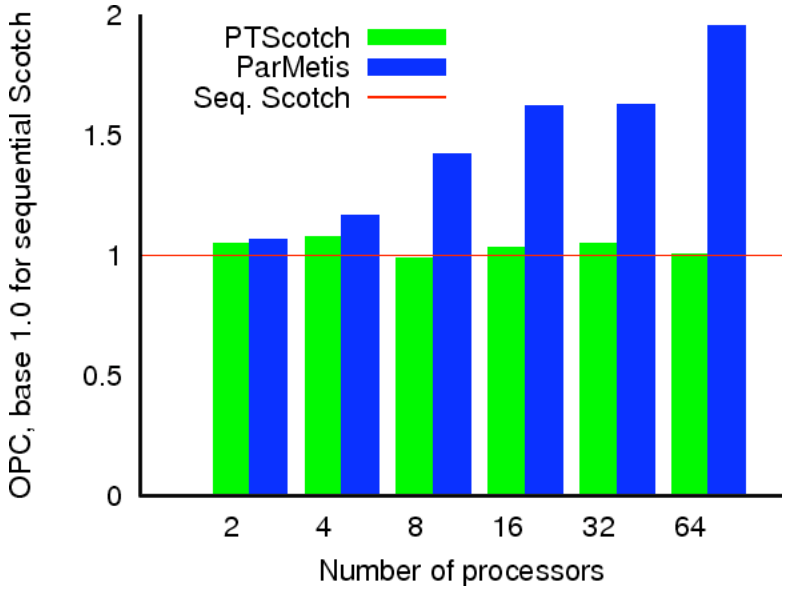
Graph	Size (10^6)		Average degree	O_{ss}	Description
	|V|	|E|			
audikw1	944	38354	81.2	5.48E+12	3D mechanics mesh, Par
cage15	5154	47022	18.2	4.06E+16	DNA electrophoresis,
quimonda07	8613	29143	6.7	6.92E+10	Circuit simulation, Quimor
23millions	23114	175686	7.6	1.29E+14	CEA/CESTA

Sandia National Laboratories

Experimental results (2)

Test case	Number of processes					
	2	4	8	16	32	64
audikw1						
O_{PTS}	5.73E+1	25.65E+1	25.54E+1	25.45E+1	25.45E+1	25.45E+1
O_{PM}	5.82E+1	26.37E+1	27.78E+1	28.88E+1	28.91E+1	21.07E+1
t_{PTS}	73.11	53.19	45.19	33.83	24.74	18.1

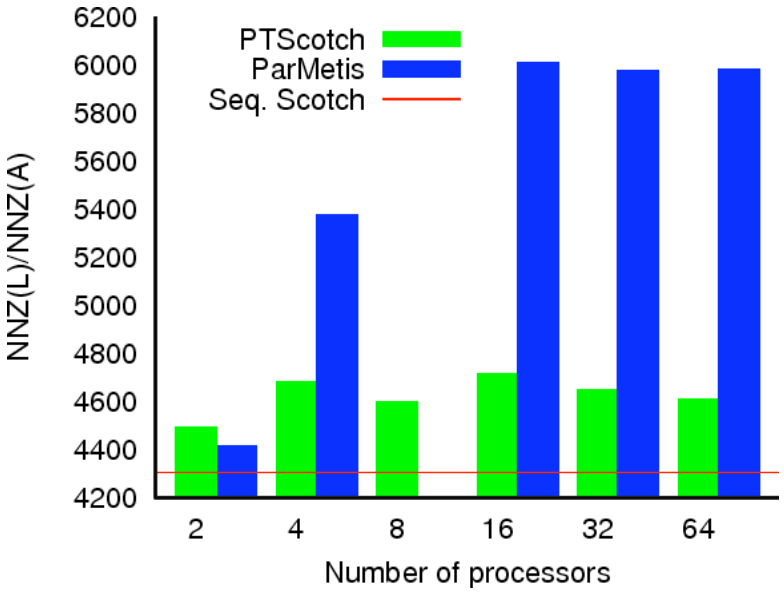


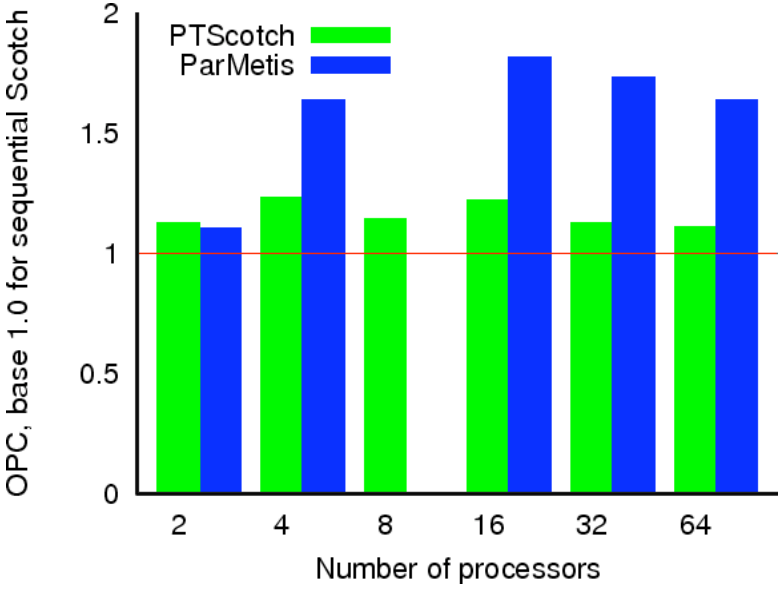


Sandia National Laboratories

Experimental results (3)

Test case	Number of processes					
	2	4	8	16	32	64
cage15						
O_{PTS}	4.58E+16	5.01E+16	4.64E+16	4.94E+16	4.58E+16	4.50E+1
O_{PM}	4.47E+16	6.64E+16	†	7.36E+16	7.03E+16	6.64E+1
t_{PTS}	540.46	427.38	371.70	340.78	351.38	380.6
t_{PM}	195.93	117.77	†	40.30	22.56	17.8





Experimental results (4)

- **ParMETIS crashes for all other graphs.**

Test case	Number of processes					
	2	4	8	16	32	64
quimonda07						
O_{PTS}	-	-	5.80E+10	6.38E+10	6.94E+10	7.70E+1
t_{PTS}	-	-	34.68	22.23	17.30	16.62
23millions						
O_{PTS}	1.45E+14	2.91E+14	3.99E+14	2.71E+14	1.94E+14	2.45E+1
t_{PTS}	671.60	416.45	295.38	211.68	147.35	103.7

Summary of Matrix Ordering

Sandia National Laboratories

- Zoltan provides access to efficient parallel ordering for sparse matrices (especially with Scotch).
- Zoltan provides a standard way to call parallel ordering.
- Zoltan will provide also its own ordering tool in the future, dealing with ordering for non-symmetric problems.

Sandia National Laboratories

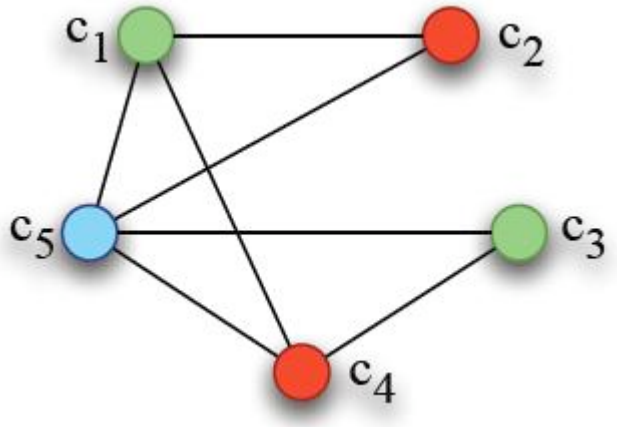
Zoltan Graph Coloring

- 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.

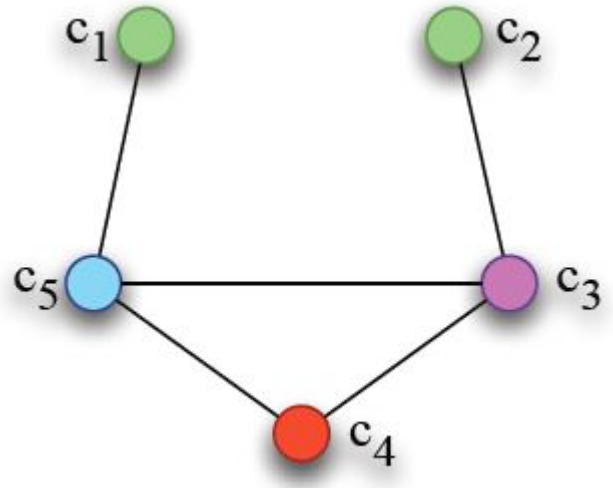
- **Applications**
  - **Iterative solution of sparse linear systems**
  - **Preconditioners**
  - **Sparse tiling**
  - **Eigenvalue computation**
  - **Parallel graph partitioning**



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**



A Parallel Coloring Framework

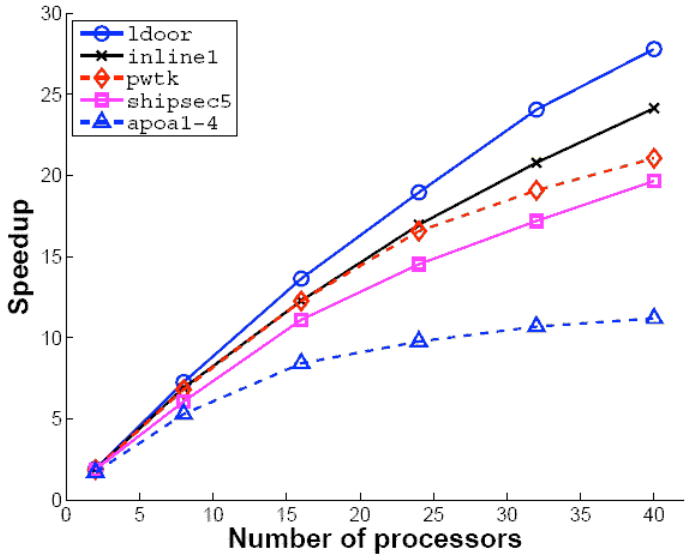
- Color vertices iteratively in rounds using a first fit strategy
- Each round is broken into supersteps
  - Color a certain number of vertices
  - Exchange recent color information
- Detect conflicts at the end of each round
- Repeat until all vertices receive consistent colors

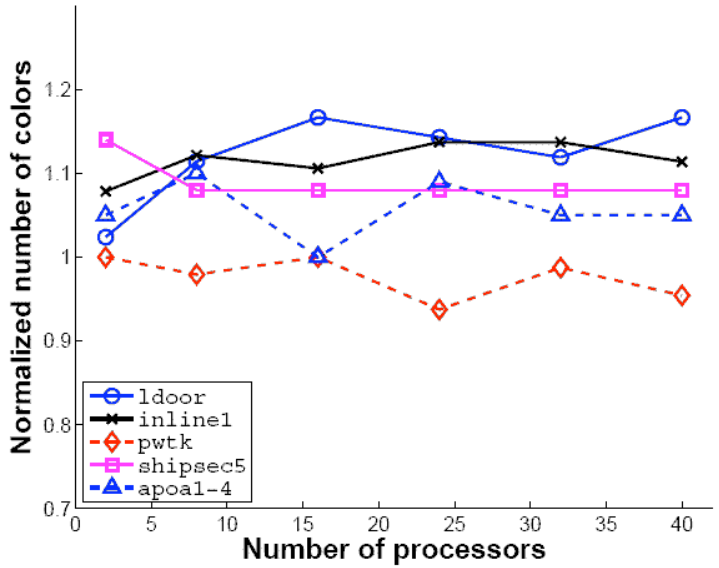
Coloring Interface in Zoltan

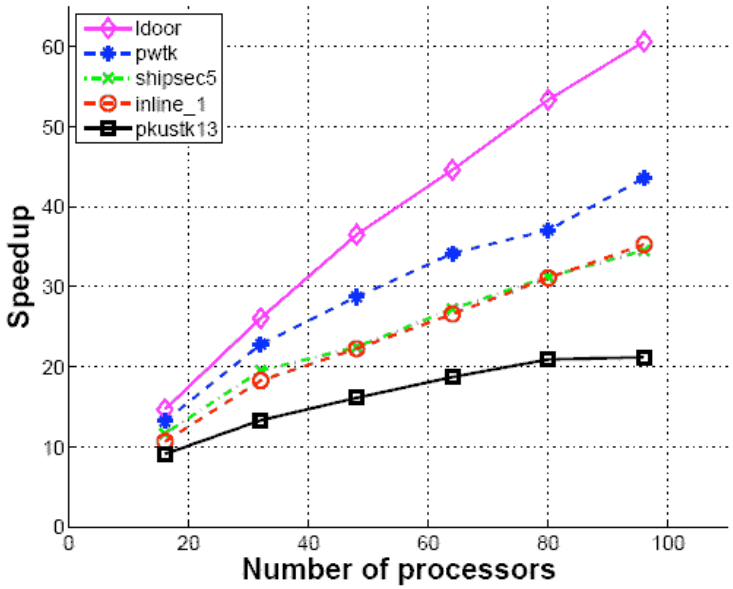
- Both distance-1 and distance-2 coloring routines can be invoked by Zoltan_Color function.
- The colors assigned to the objects are returned in an array.

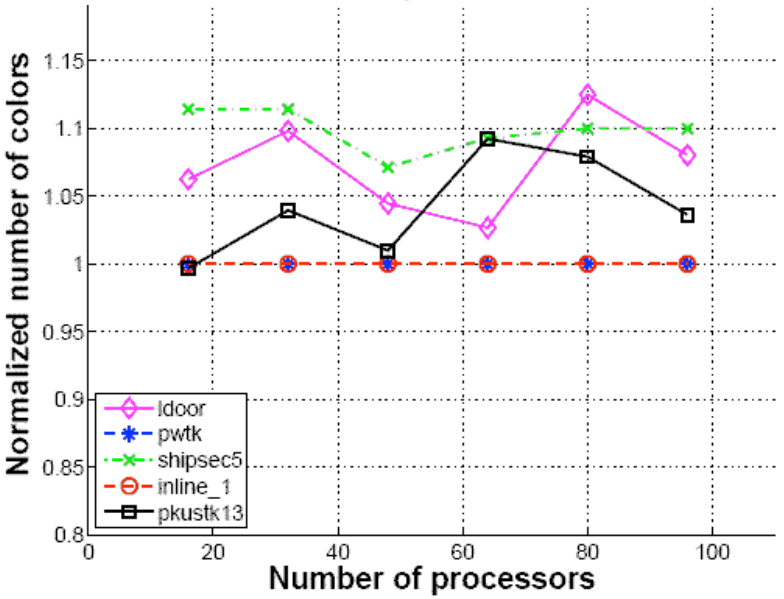
Sandia National Laboratories

Experimental Results









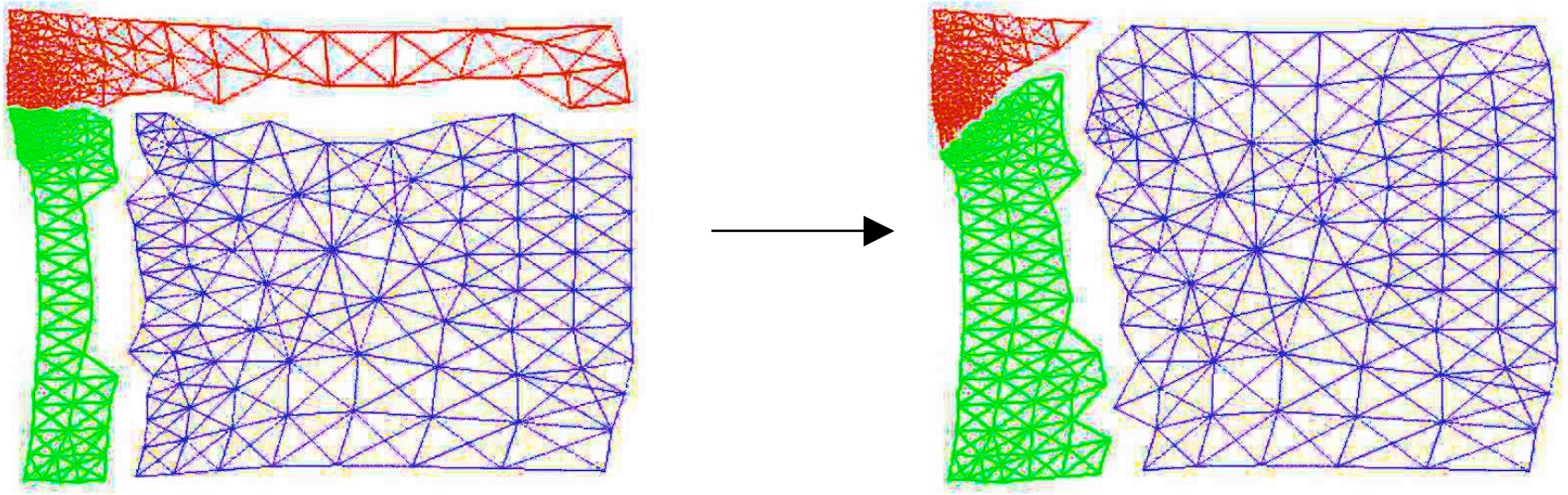
Other Zoltan Functionality

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

Sandia National Laboratories

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

Sandia National Laboratories

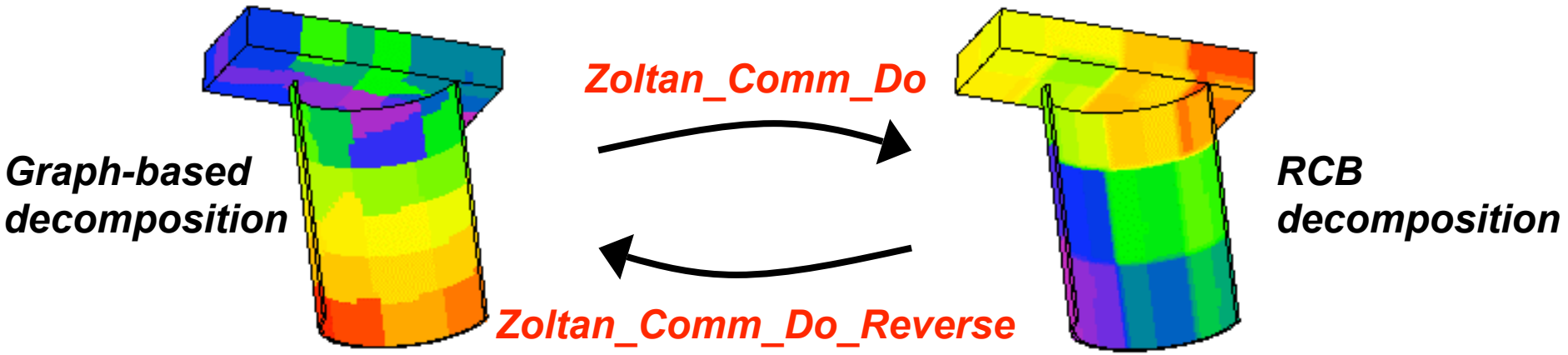
- **Required migration query functions:**
  - **ZOLTAN_OBJ_SIZE_MULTI_FN:**
    - Returns size of data (in bytes) for each object to be exported to a new processor.
  - **ZOLTAN_PACK_MULTI_FN:**
    - Remove data from application data structure on old processor;
    - Copy data to Zoltan communication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

Sandia National Laboratories

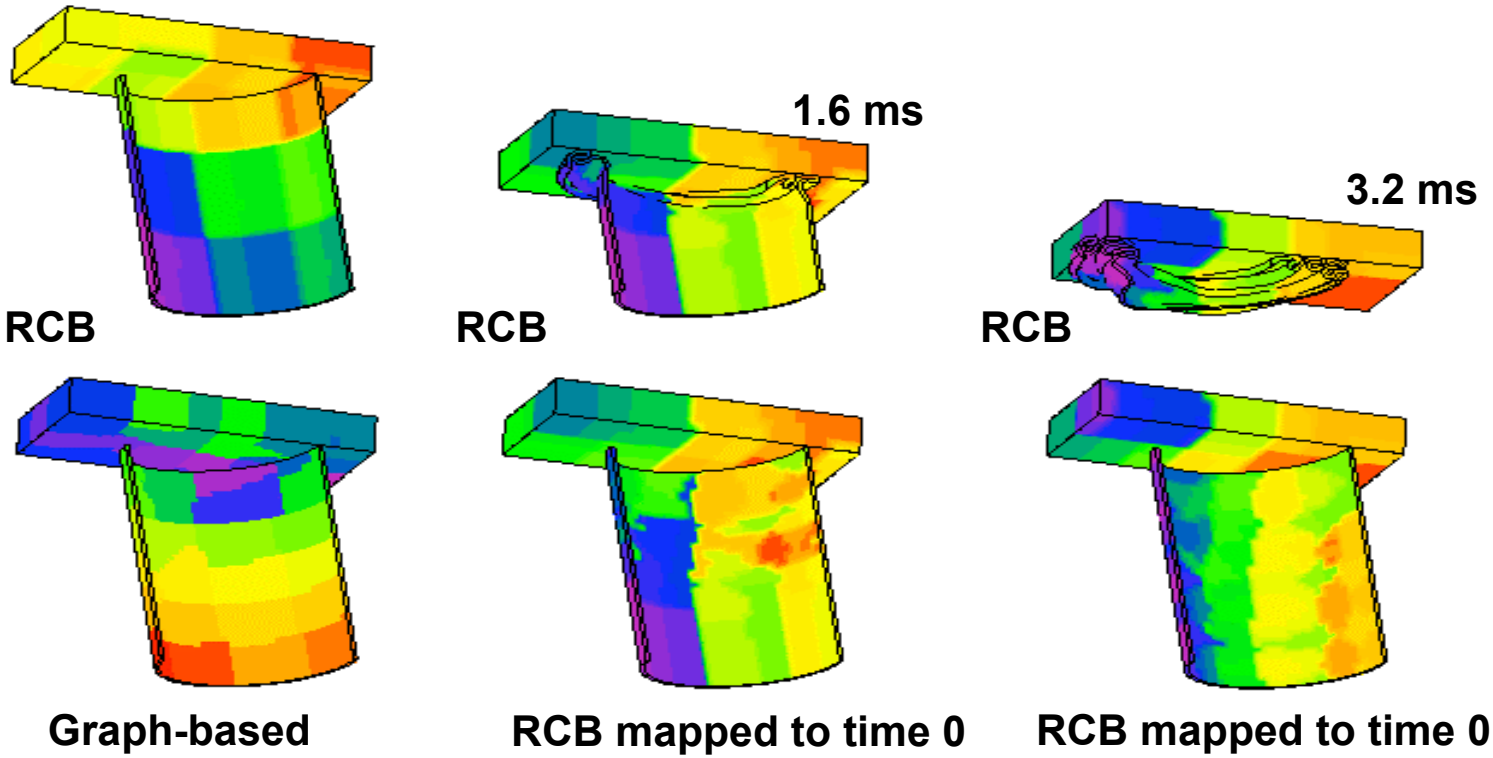
- **Simple primitives for efficient irregular communication.**
  - **Zoltan_Comm_Create: Generates communication plan.**
    - **Processors and amount of data to send and receive.**
  - **Zoltan_Comm_Do: Send data using plan.**
    - **Can reuse plan. (Same plan, different data.)**
  - **Zoltan_Comm_Do_Reverse: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



Example Application: Crash Simulations

Sandia National Laboratories

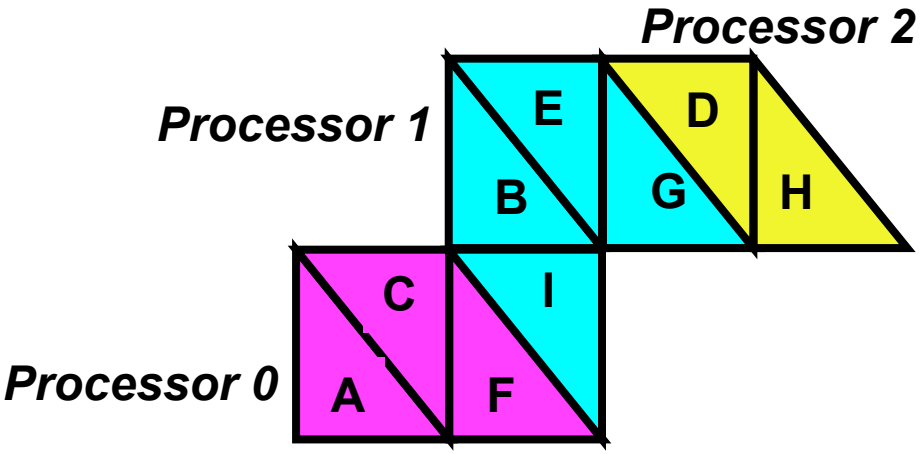
- **Multiphase simulation:**
  - **Graph-based decomposition of elements for finite element calculation.**
  - **Dynamic geometric decomposition of surfaces for contact detection.**
  - **Migration tools and Unstructured Communication package map between decompositions.**

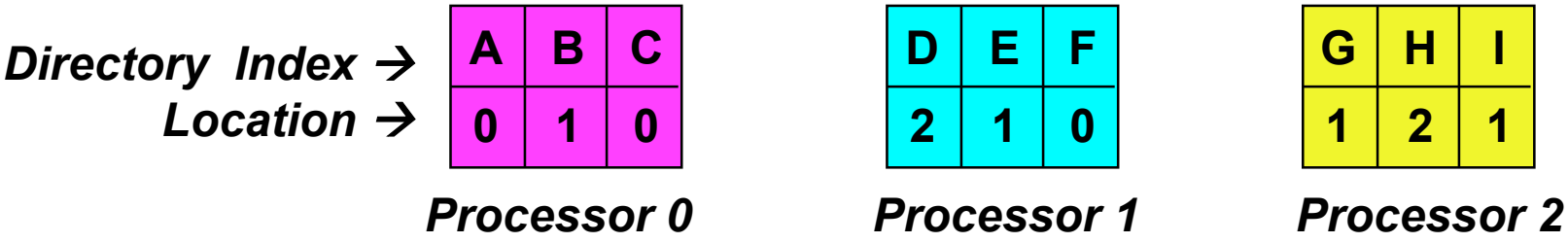


Zoltan Distributed Data Directory

Sandia National Laboratories

- **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.**





Alternate Interfaces to Zoltan

Sandia National Laboratories

- Isorropia package in Trilinos solver toolkit.
  - Epetra Matrix interface to Zoltan partitioning.
    - B = Isorropia::Epetra::create_balanced_copy(A, params);
  - Trilinos v9 will also include ordering and coloring interfaces in Isorropia.
  - SciDAC TOPS-2 CET.
- ITAPS iMesh interface to Zoltan.
  - New iMeshP parallel mesh interface to be incorporated in FY09.
  - SciDAC ITAPS CET.

Sandia National Laboratories

Future Work

- Two-dimensional matrix partitioning interfaces in Isorropia.
- Performance improvements for hypergraph partitioning.
- Multi-criteria hypergraph partitioning.
- Non-symmetric matrix ordering.

For More Information...

- **Zoltan Home Page**
  - http://www.cs.sandia.gov/Zoltan
  - User's and Developer's Guides
  - Download Zoltan software under GNU LGPL.

- **Email:**
  - {kddevin,ccheval,egboman}@sandia.gov
  - umit@bmi.osu.edu

Sandia National Laboratories

The End

Configuring and Building Zoltan

Sandia National Laboratories

- Create and enter the Zoltan directory:
  - gunzip zoltan_distrib_v3.0.tar.gz
  - tar xf zoltan_distrib_v3.0.tar
  - cd Zoltan
- Configure and make Zoltan library
  - Not autotooled; uses manual configuration file.
  - "make zoltan" attempts a generic build; library libzoltan.a is in directory Obj_generic.
  - To customize your build:
    - cd Utilities/Config; cp Config.linux Config.your_system
    - Edit Config.your_system
    - cd ../..
    - setenv ZOLTAN_ARCH your_system
    - make zoltan
    - Library libzoltan.a is in Obj_your_system

Config file

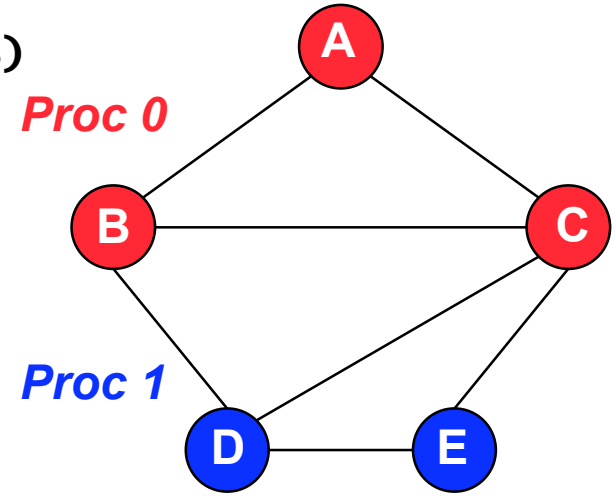

```
DEFS                    =
RANLIB                  = ranlib
AR                      = ar r

CC                      = mpicc -Wall
CPPC                    = mpic++
INCLUDE_PATH            =
DBG_FLAGS               = -g
OPT_FLAGS               = -O
CFLAGS                  = $(DBG_FLAGS)

F90                     = mpif90
LOCAL_F90               = f90
F90CFLAGS               = -DFMANGLE=UNDERSCORE -DNO_MPI2
FFLAGS                  =
SPPR_HEAD               = spprinc.most
F90_MODULE_PREFIX       = -I
FARG                    = farg_typical

MPI_LIB                 =
MPI_LIBPATH             =

PARMETIS_LIBPATH        = -L/Users/kddevin/code/ParMETIS3_1
PARMETIS_INCPATH        = -I/Users/kddevin/code/ParMETIS3_1
#PATOH_LIBPATH           = -L/Users/kddevin/code/PaToH
#PATOH_INCPATH           = -I/Users/kddevin/code/PaToH
```

Example Graph Callbacks

```
void ZOLTAN_NUM_EDGES_MULTI_FN(void *data,
  int num_gid_entries, int num_lid_entries,
  int num_obj, ZOLTAN_ID_PTR global_id, ZOLTAN_ID_PTR local_id,
  int *num_edges, int *ierr);
```

Proc 0 Input from Zoltan:
num_obj = 3
global_id = {A,C,B}
local_id = {0,1,2}

Output from Application on Proc 0:
num_edges = {2,4,3}
(i.e., degrees of vertices A, C, B)
ierr = ZOLTAN_OK



Example Graph Callbacks

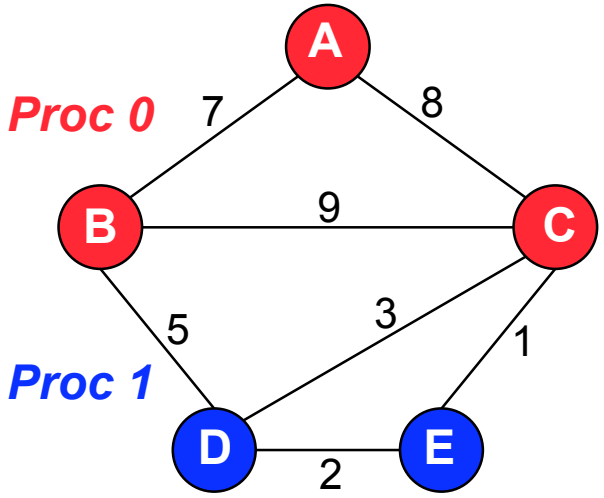
```
void ZOLTAN_EDGE_LIST_MULTI_FN(void *data,
  int num_gid_entries, int num_lid_entries,
  int num_obj, ZOLTAN_ID_PTR global_ids, ZOLTAN_ID_PTR local_ids,
  int *num_edges,
  ZOLTAN_ID_PTR nbor_global_id, int *nbor_procs,
  int wdim, float *nbor_ewgts,
  int *ierr);
```

Proc 0 Input from Zoltan:

```
num_obj = 3
global_ids = {A, C, B}
local_ids  = {0, 1, 2}
num_edges  = {2, 4, 3}
wdim = 0 or EDGE_WEIGHT_DIM parameter value
```

Output from Application on Proc 0:

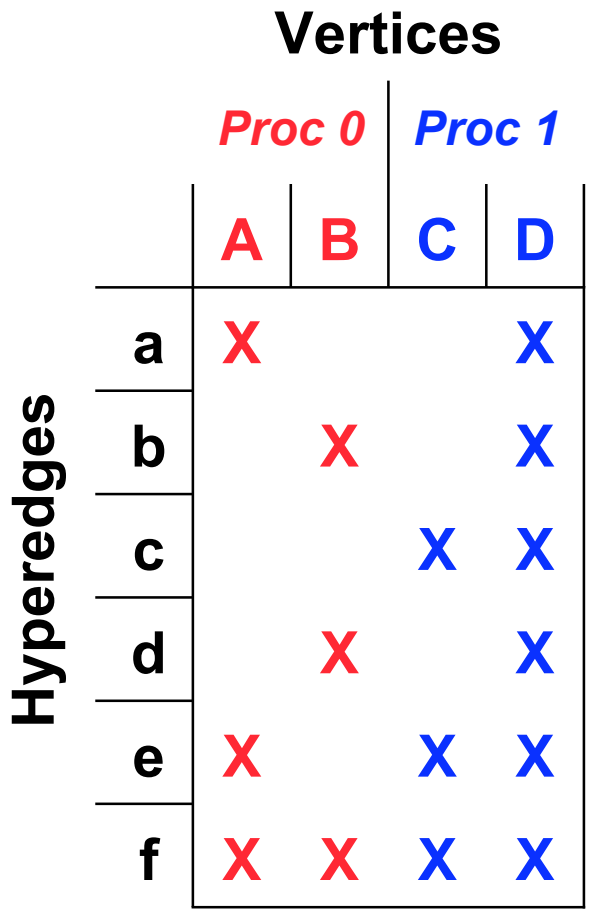
```
nbor_global_id = {B, C, A, B, E, D, A, C, D}
nbor_procs     = {0, 0, 0, 0, 1, 1, 0, 0, 1}
nbor_ewgts     = if wdim then
                   {7, 8, 8, 9, 1, 3, 7, 9, 5}
ierr = ZOLTAN_OK
```



Example Hypergraph Callbacks


```
void ZOLTAN_HG_SIZE_CS_FN(void *data, int *num_lists, int *num_pins,
  int *format, int *ierr);
```

```
Output from Application on Proc 0:
  num_lists = 2
  num_pins = 6
  format = ZOLTAN_COMPRESSED_VERTEX
           (owned non-zeros per vertex)
  ierr = ZOLTAN_OK

OR

Output from Application on Proc 0:
  num_lists = 5
  num_pins = 6
  format = ZOLTAN_COMPRESSED_EDGE
           (owned non-zeros per edge)
  ierr = ZOLTAN_OK
```



Example Hypergraph Callbacks

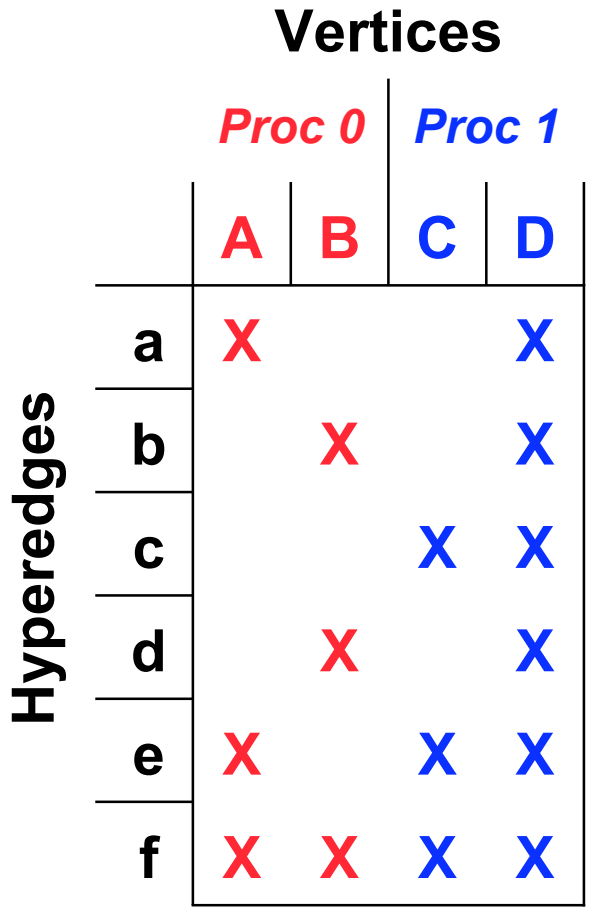

```
void ZOLTAN_HG_CS_FN(void *data, int num_gid_entries,
  int nvtxedge, int npins, int format,
  ZOLTAN_ID_PTR vtxedge_GID, int *vtxedge_ptr, ZOLTAN_ID_PTR pin_GID,
  int *ierr);
```

Proc 0 Input from Zoltan:

- nvtxedge = 2 or 5
- npins = 6
- format = ZOLTAN_COMPRESSED_VERTEX or ZOLTAN_COMPRESSED_EDGE

Output from Application on Proc 0:

- if (format = ZOLTAN_COMPRESSED_VERTEX)
  - vtxedge_GID = {A, B}
  - vtxedge_ptr = {0, 3}
  - pin_GID = {a, e, f, b, d, f}
- if (format = ZOLTAN_COMPRESSED_EDGE)
  - vtxedge_GID = {a, b, d, e, f}
  - vtxedge_ptr = {0, 1, 2, 3, 4}
  - pin_GID = {A, B, B, A, A, B}
- ierr = ZOLTAN_OK



Simple Example

- Zoltan/examples/C/zoltanSimple.c
- Application data structure:
  - int MyNumPts;
    - Number of points on processor.
  - int *Gids;
    - array of Global ID numbers of points on processor.
  - float *Pts;
    - Array of 3D coordinates of points on processor (in same order as Gids array).

Example zoltanSimple.c: Initialization

Sandia National Laboratories

```
/* Initialize MPI */
MPI_Init(&argc, &argv);
MPI_Comm_rank(MPI_COMM_WORLD, &me);
MPI_Comm_size(MPI_COMM_WORLD, &nprocs);

/*
** Initialize application data.  In this example,
** create a small test mesh and divide it across processors
*/

exSetDivisions(32);    /* rectilinear mesh is div X div X div */

MyNumPts = exInitializePoints(&Pts, &Gids, me, nprocs);

/*  Initialize Zoltan */
rc = Zoltan_Initialize(argc, argv, &ver);

if (rc != ZOLTAN_OK){
  printf("sorry...\n");
  free(Pts); free(Gids);
  exit(0);
}
```

Example zoltanSimple.c: Prepare for Partitioning

Sandia National Laboratories

```
/* Allocate and initialize memory for Zoltan structure */
zz = Zoltan_Create(MPI_COMM_WORLD);

/* Set general parameters */
Zoltan_Set_Param(zz, "DEBUG_LEVEL", "0");
Zoltan_Set_Param(zz, "LB_METHOD", "RCB");
Zoltan_Set_Param(zz, "NUM_GID_ENTRIES", "1");
Zoltan_Set_Param(zz, "NUM_LID_ENTRIES", "1");
Zoltan_Set_Param(zz, "RETURN_LISTS", "ALL");

/* Set RCB parameters */
Zoltan_Set_Param(zz, "KEEP_CUTS", "1");
Zoltan_Set_Param(zz, "RCB_OUTPUT_LEVEL", "0");
Zoltan_Set_Param(zz, "RCB_RECTILINEAR_BLOCKS", "1");

/* Register call-back query functions. */
Zoltan_Set_Num_Obj_Fn(zz, exGetNumberOfAssignedObjects, NULL);
Zoltan_Set_Obj_List_Fn(zz, exGetObjectList, NULL);
Zoltan_Set_Num_Geom_Fn(zz, exGetObjectSize, NULL);
Zoltan_Set_Geom_Multi_Fn(zz, exGetObject, NULL);
```

Example zoltanSimple.c: Partitioning

Sandia National Laboratories

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)

```
/* Perform partitioning */
rc = 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);
```

Example zoltanSimple.c: Use the Partition

Sandia National Laboratories

```
/* Process partitioning results;
** in this case, print information;
** in a "real" application, migrate data here.
*/
if (!rc){
  exPrintGlobalResult("Recursive Coordinate Bisection",
                      nprocs, me,
                      MyNumPts, numImport, numExport, changes);
}
else{
  free(Pts);
  free(Gids);
  Zoltan_Destroy(&zz);
  MPI_Finalize();
  exit(0);
}
```

Example zoltanSimple.c: Cleanup

Sandia National Laboratories

```
/* Free Zoltan memory allocated by Zoltan_LB_Partition. */
Zoltan_LB_Free_Part(&importGlobalGids, &importLocalGids,
                    &importProcs, &importToPart);
Zoltan_LB_Free_Part(&exportGlobalGids, &exportLocalGids,
                    &exportProcs, &exportToPart);

/* Free Zoltan memory allocated by Zoltan_Create. */
Zoltan_Destroy(&zz);

/* Free Application memory */
free(Pts); free(Gids);

/************************
** all done ************
************************/

MPI_Finalize();
```

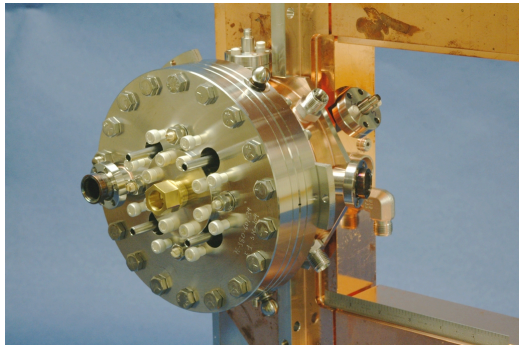
Sandia National Laboratories

Performance Results

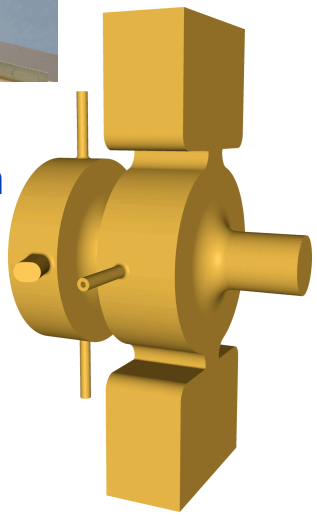
- Experiments on Sandia's Thunderbird cluster.
  - Dual 3.6 GHz Intel EM64T processors with 6 GB RAM.
  - Infiniband network.
- Compare RCB, HSFC, graph and hypergraph methods.
- Measure …
  - Amount of communication induced by the partition.
  - Partitioning time.

Sandia National Laboratories

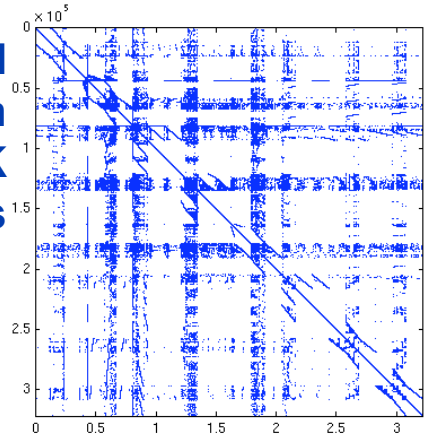
Test Data



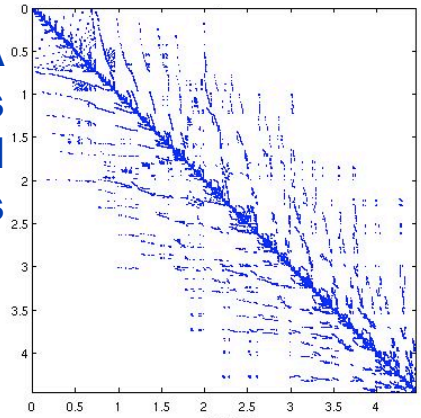
SLAC *LCLS
Radio Frequency Gun
6.0M x 6.0M
23.4M nonzeros



Xyce 680K ASIC Stripped
Circuit Simulation
680K x 680K
2.3M nonzeros



Cage15 DNA
Electrophoresis
5.1M x 5.1M
99M nonzeros



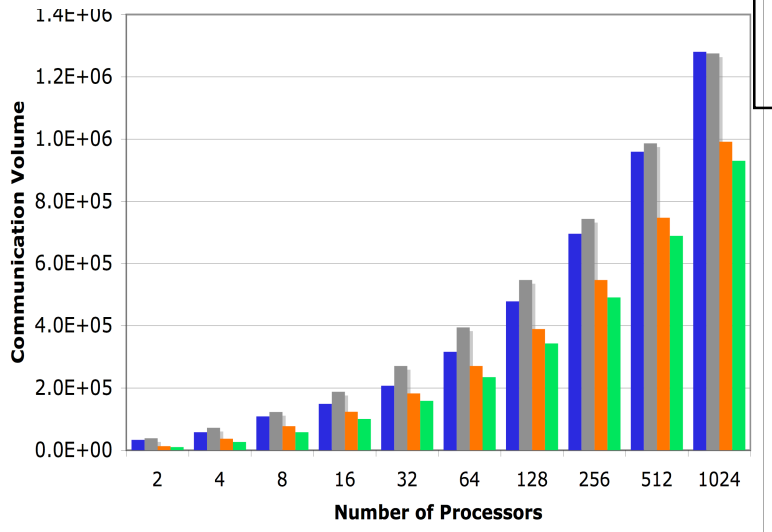
SLAC Linear Accelerator
2.9M x 2.9M
11.4M nonzeros

Communication Volume: Lower is Better

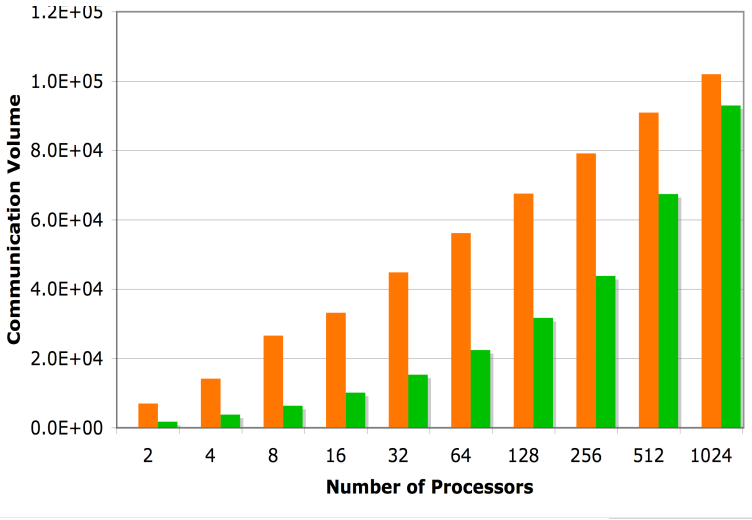
Sandia National Laboratories

Number of parts = number of processors.

SLAC 6.0M LCLS

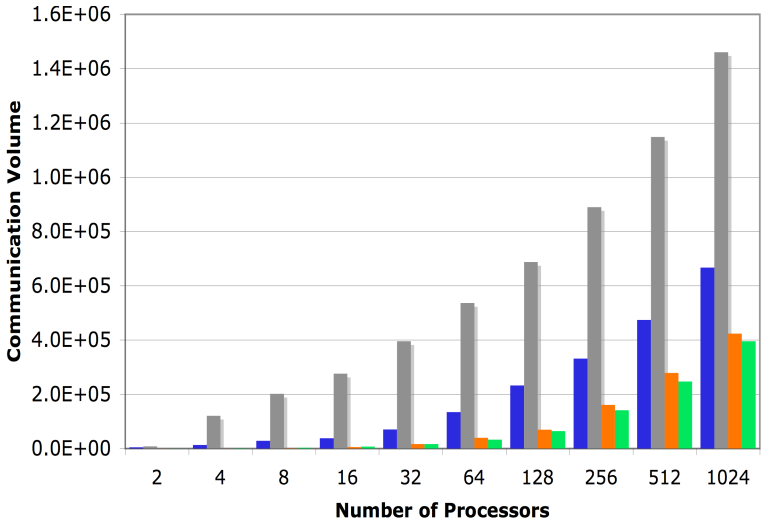


Xyce 680K circuit

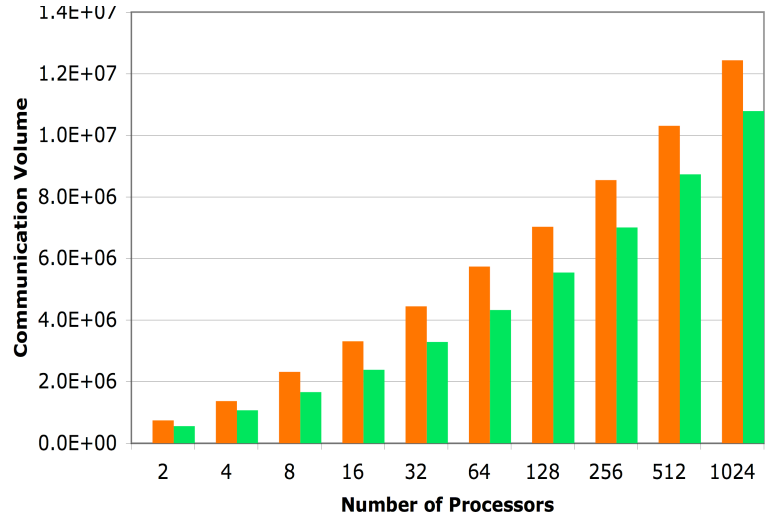


RCB
HSFC
Graph
Hypergraph

SLAC 2.9M Linear Accelerator



Cage15 5.1M electrophoresis



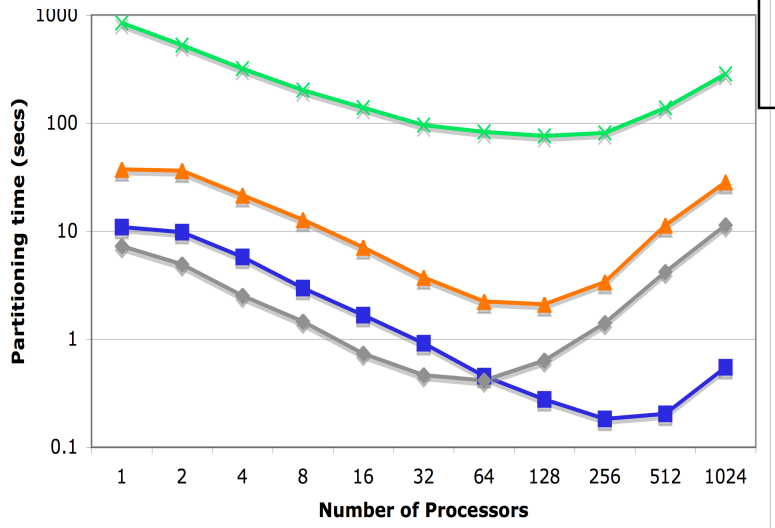
Partitioning Time: Lower is better

Sandia National Laboratories

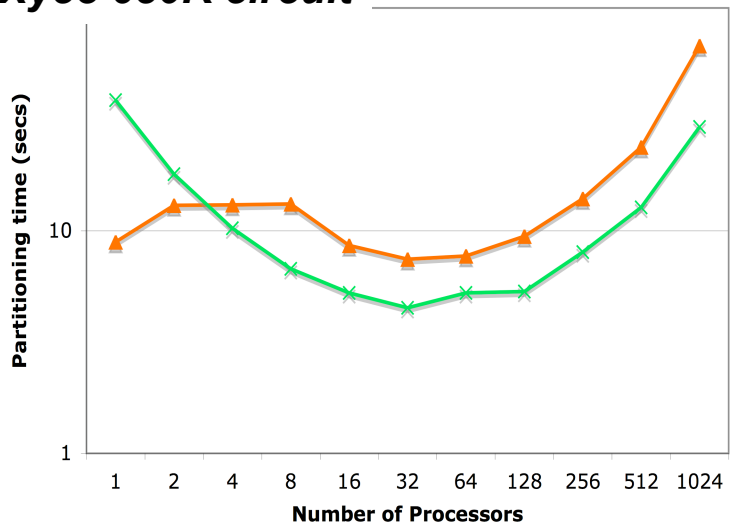
1024 parts. Varying number of processors.

RCB
HSFC
Graph
Hypergraph

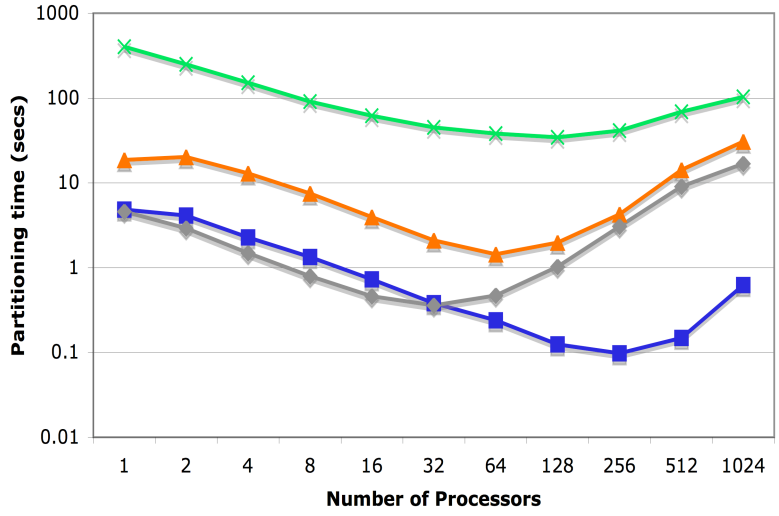
SLAC 6.0M LCLS



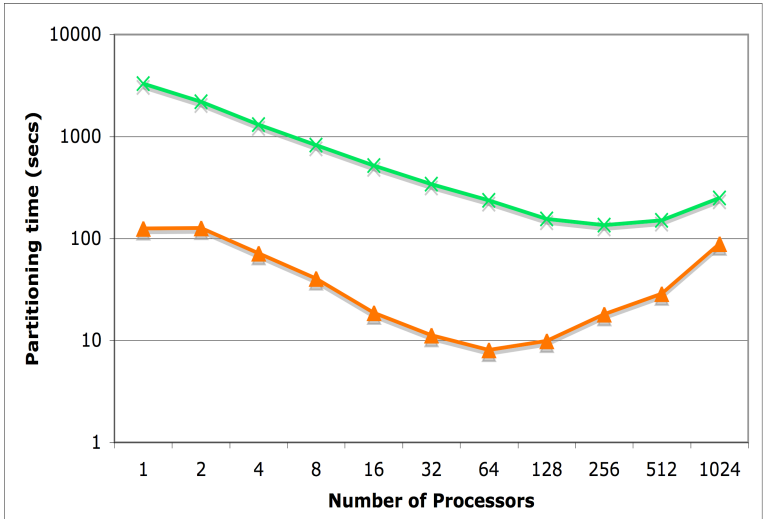
Xyce 680K circuit



SLAC 2.9M Linear Accelerator



Cage15 5.1M electrophoresis



Sandia National Laboratories

Repartitioning Experiments

- Experiments with 64 parts on 64 processors.
- Dynamically adjust weights in data to simulate, say, adaptive mesh refinement.
- Repartition.
- Measure repartitioning time and total communication volume:

  Data redistribution volume

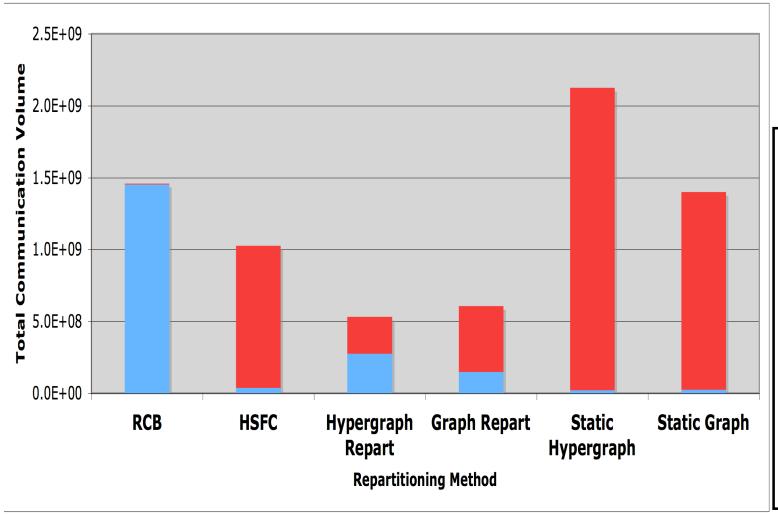
  \+ Application communication volume

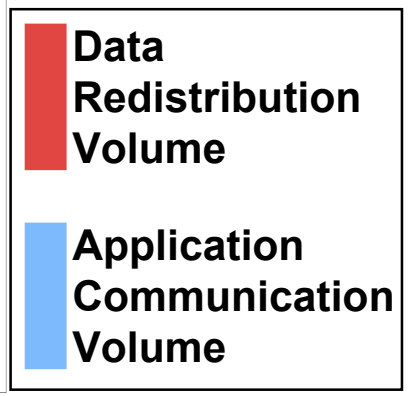
  Total communication volume

Repartitioning Results: Lower is Better

Sandia National Laboratories

SLAC 6.0M LCLS





Xyce 680K circuit