Parallel Hypergraph Partitioning for Irregular Problems

Karen Devine, Erik Boman, Robert Heaphy
Sandia National Laboratories, Albuquerque
kddevin@sandia.gov

Umit Çatalyürek
Ohio State University, Columbus

Rob Bisseling
Utrecht University, The Netherlands

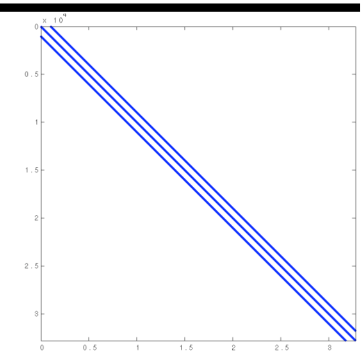
Sandia is a multiprogram laboratory operated by Sandia Corporation, a Lockheed Martin Company,
for the United States Department of Energy's National Nuclear Security Administration
under contract DE-AC04-94AL85000.

Graph Partitioning

- Work-horse of load-balancing community.
- Highly successful model for PDE problems.
- Model problem as a graph:
  - vertices = work associated with data (computation)
  - edges = relationships between data (communication)
- Goal: Evenly distribute vertex weight while minimizing weight of cut edges.
- Excellent software available.
  - Serial: Chaco (SNL), Jostle (U. Greenwich), METIS (U. Minn.), Party (U. Paderborn), Scotch (U. Bordeaux)
  - Parallel: ParMETIS (U. Minn.), PJostle (U. Greenwich)

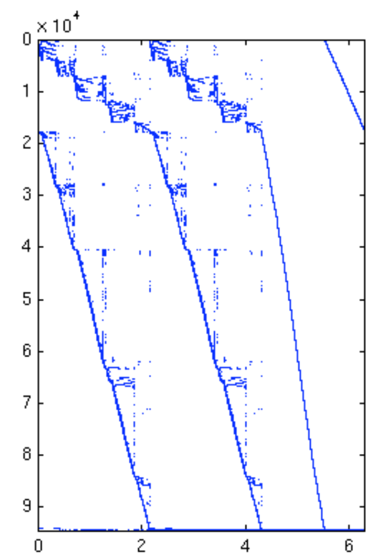
Limited Applicability of Graph Models

- Assume symmetric square problems.
  - Symmetric = undirected graph.
  - Square = inputs and outputs of operation are same size.



Hexahedral finite element matrix

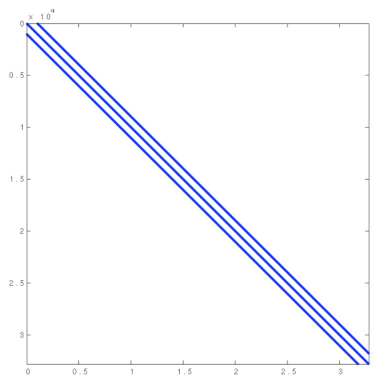
- Do not naturally support:
  - Non-symmetric systems.
    - Require directed or bipartite graph.
    - Partition $A+A^T$.
  - Rectangular systems.
    - Require decompositions for differently sized inputs and outputs.
    - Partition AA^T.



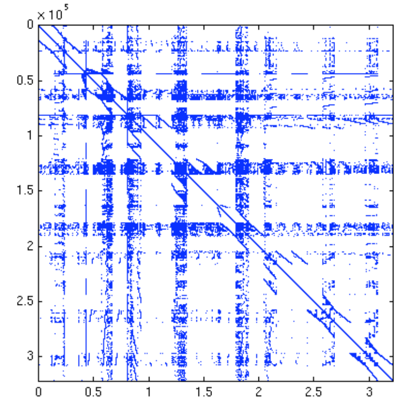
Linear programming matrix for sensor placement

Approximate Communication Metric in Graph Models

- Graph models assume
  - Weight of edge cuts = Communication volume.
- But edge cuts only *approximate* communication volume.
- Good enough for many PDE applications.
- Not good enough for irregular problems.

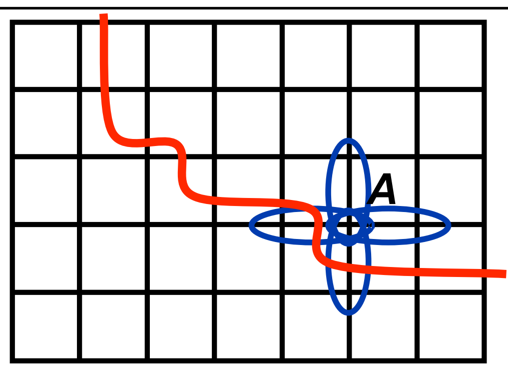
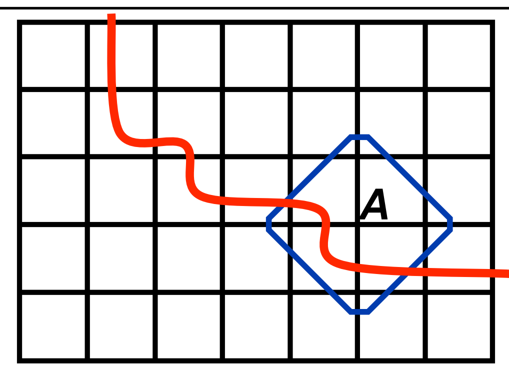


Hexahedral finite element matrix



Xyce ASIC matrix

Another Option: Hypergraph Models

Graph Partitioning Kernighan, Lin, Schweikert, Fiduccia, Mattheyes, Simon, Hendrickson, Leland, Kumar, Karypis, et al.	Hypergraph Partitioning Alpert, Kahng, Hauck, Borriello, Çatalyürek, Aykanat, Karypis, et al.
Vertices: computation.	Vertices: computation.
Edges: two vertices.	Hyperedges: two or more vertices.
Edge cuts approximate communication volume.	Hyperedge cuts accurately measure communication volume.
Assign equal vertex weight while minimizing edge cut weight.	Assign equal vertex weight while minimizing hyperedge cut weight.
	

Impact of Hypergraph Partitioning

Sandia National Laboratories

- Greater expressiveness ⇒ Greater applicability.
  - Structurally non-symmetric systems
    - circuits, biology
  - Rectangular systems
    - linear programming, least-squares methods
  - Non-homogeneous, highly connected topologies
    - circuits, nanotechnology, databases
- Accurate communication model ⇒ lower application communication costs.

- Several serial hypergraph partitioners available.
  - hMETIS (Karypis)
  - PaToH (Çatalyürek)
  - Mondriaan (Bisseling)

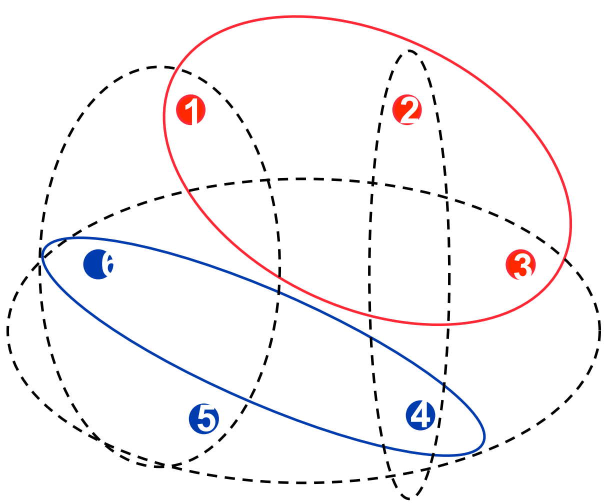
- Parallel partitioners needed for large, dynamic problems.
  - Zoltan PHG (Sandia)
  - Parkway (Trifunovic)

Matrix Representation

- **View hypergraph as matrix (Çatalyürek & Aykanat)**
  - **Vertices == columns**
  - **Edges == rows**
- **Communication volume associated with edge *e*:**

  CV_e = **(# processors in edge *e*) - 1**

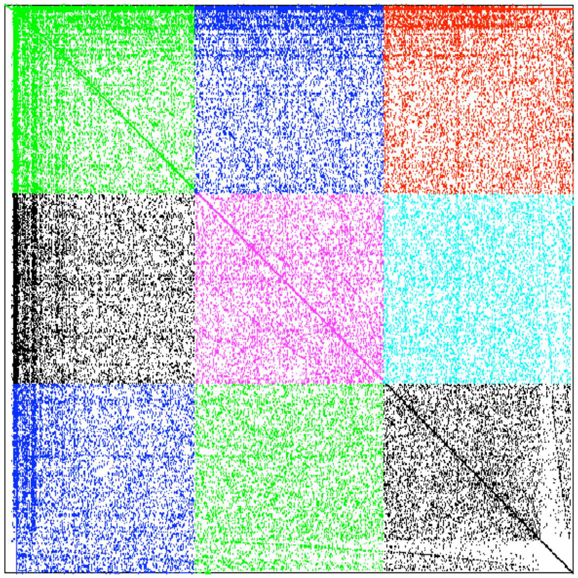
- **Total communication volume =** $\sum_e CV_e$



$$\begin{pmatrix} y \\ y \\ y \\ y \\ y \end{pmatrix} = \begin{pmatrix} * & * & * & & & \\ & * & & * & & \\ * & & & & * & * \\ & & * & * & * & * \\ & & & * & & * \end{pmatrix} \begin{pmatrix} x \\ x \\ x \\ x \\ x \\ x \end{pmatrix}$$

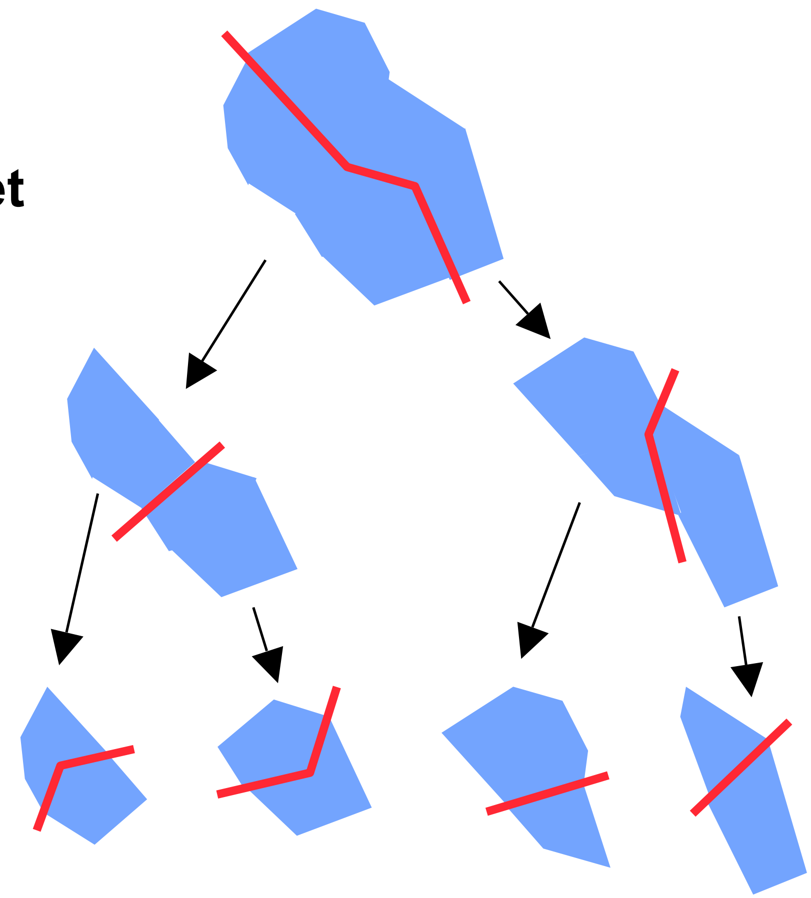
Data Layout

- **2D data layout within hypergraph partitioner.**
  - **Does not affect the layout returned to the application.**
  - **Vertex/hyperedge broadcasts to only *sqrt*(*P*) processors.**
  - **Maintain scalable memory usage.**
    - **No “ghosting” of off-processor neighbor info.**
    - **Differs from parallel graph partitioners and Parkway.**
  - **Design allows comparison of 1D and 2D distributions.**



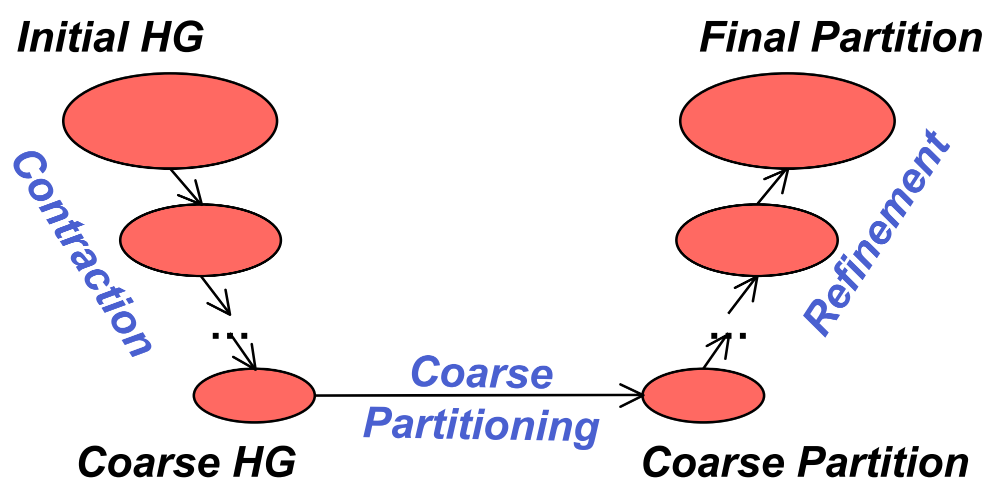
Recursive Bisection

- **Recursive bisection approach:**
  - **Partition data into two sets.**
  - **Recursively subdivide each set into two sets.**
  - **Only minor modifications needed to allow $P \neq 2^n$.**
- **Implementation:**
  - **Split *both the data and processors* into two sets.**
  - **Solve branches in parallel.**



Multilevel Scheme

- Multilevel hypergraph partitioning (Çatalyürek, Karypis)
  - Analogous to multilevel graph partitioning (Bui&Jones, Hendrickson&Leland, Karypis&Kumar).
  - Contraction: reduce HG to smaller representative HG.
  - Coarse partitioning: assign coarse vertices to partitions.
  - Refinement: improve balance and cuts at each level.



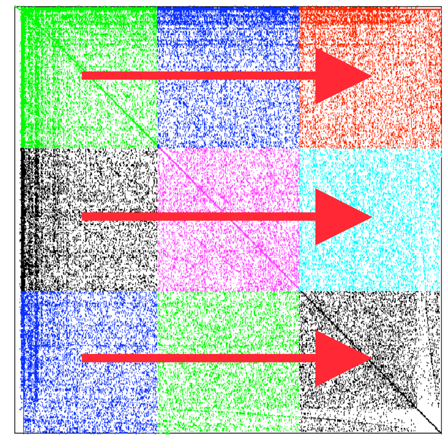
Multilevel Partitioning V-cycle

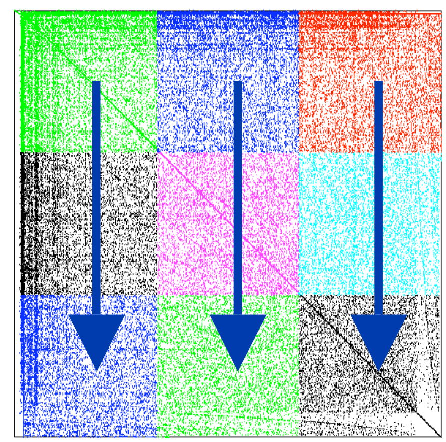
Contraction

- Merge pairs of "similar" vertices: matching.
- Greedy maximal weight matching heuristics.
- We use:
  - Heavy connectivity matching (Aykanat & Çatalyürek)
    - Inner-product matching (Bisseling)
    - First-Choice (Karypis)
  - Match columns (vertices) with greatest inner product $\Rightarrow$ greatest similarity in connectivity.

Parallel Matching in 2D Data Layout

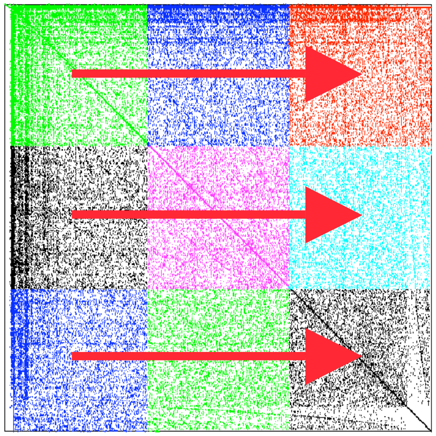
- On each processor:
  - Broadcast subset of vertices ("candidates") along processor row.
  - Compute (partial) inner products of received candidates with local vertices.
  - Accrue inner products in processor column.
  - Identify best local matches for received candidates.
  - Send best matches to candidates' owners.
  - Select best global match for each owned candidate.
  - Send "match accepted" messages to processors owning matched vertices.
- Repeat until all unmatched vertices have been sent as candidates.

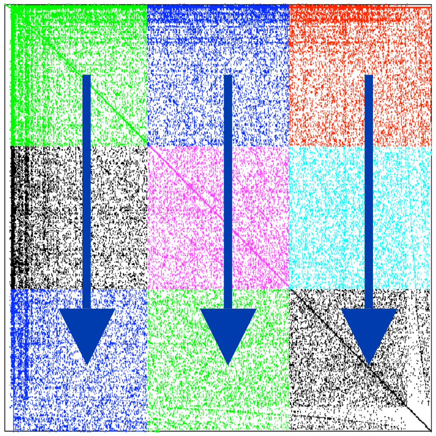




Coarse Partitioning

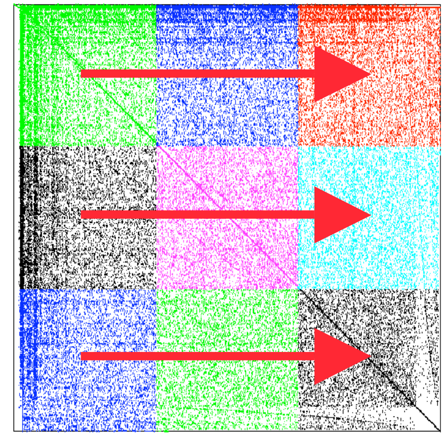
- **Gather coarsest hypergraph to each processor.**
  - **Gather edges to each processor in column.**
  - **Gather vertices to each processor in row.**
- **Compute different coarse partitions on each processor.**
- **Compute best over all processors.**

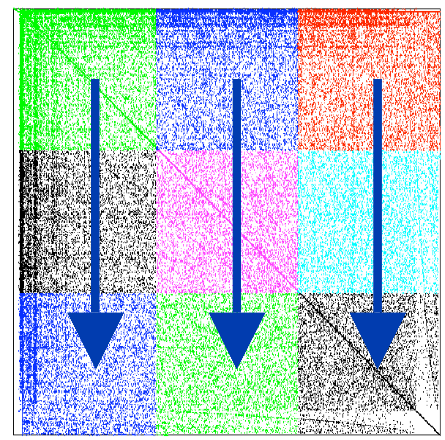




Refinement

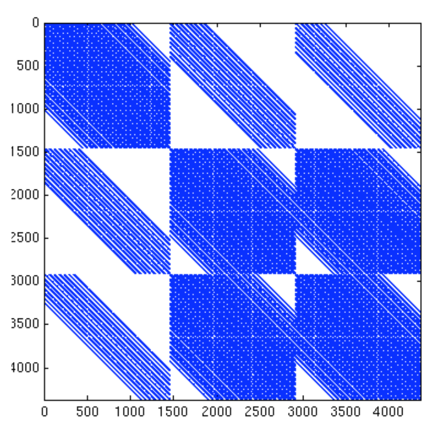
- For each level in V-cycle:
  - Project coarse partition to finer hypergraph.
  - Use local optimization (KL/FM) to improve balance and reduce cuts.
    - Compute "root" processor in each processor column: processor with most nonzeros.
    - Root processor computes moves for vertices in processor column.
    - All column processors provide cut information; receive move information.



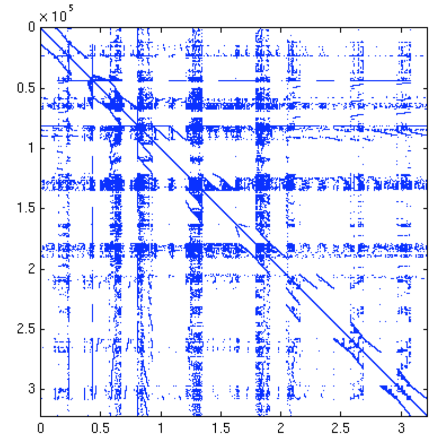


Hypergraph Partitioning Results

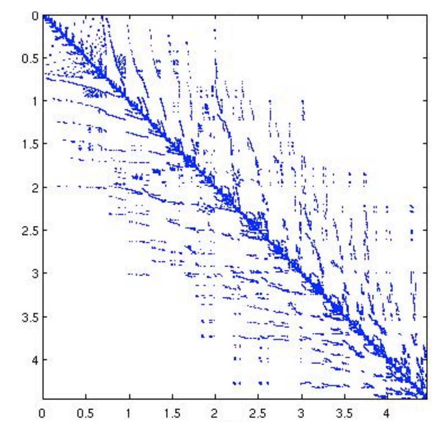
- Experiments on 3.06GHz Xenon Myrinet cluster.
- Compared Zoltan PHG to
  – Parkway hypergraph partitioner
  – ParMETIS graph partitioner
- Test problems:
  – Tramonto 2DLipidFMat: 4K x 4K; 5.5M nonzeros
  – Xyce ASIC Stripped: 680K x 680K; 2.3M nonzeros
  – Cage14 Electrophoresis: 1.5M x 1.5M; 27M nonzeros



Tramonto 2DLipidFMat



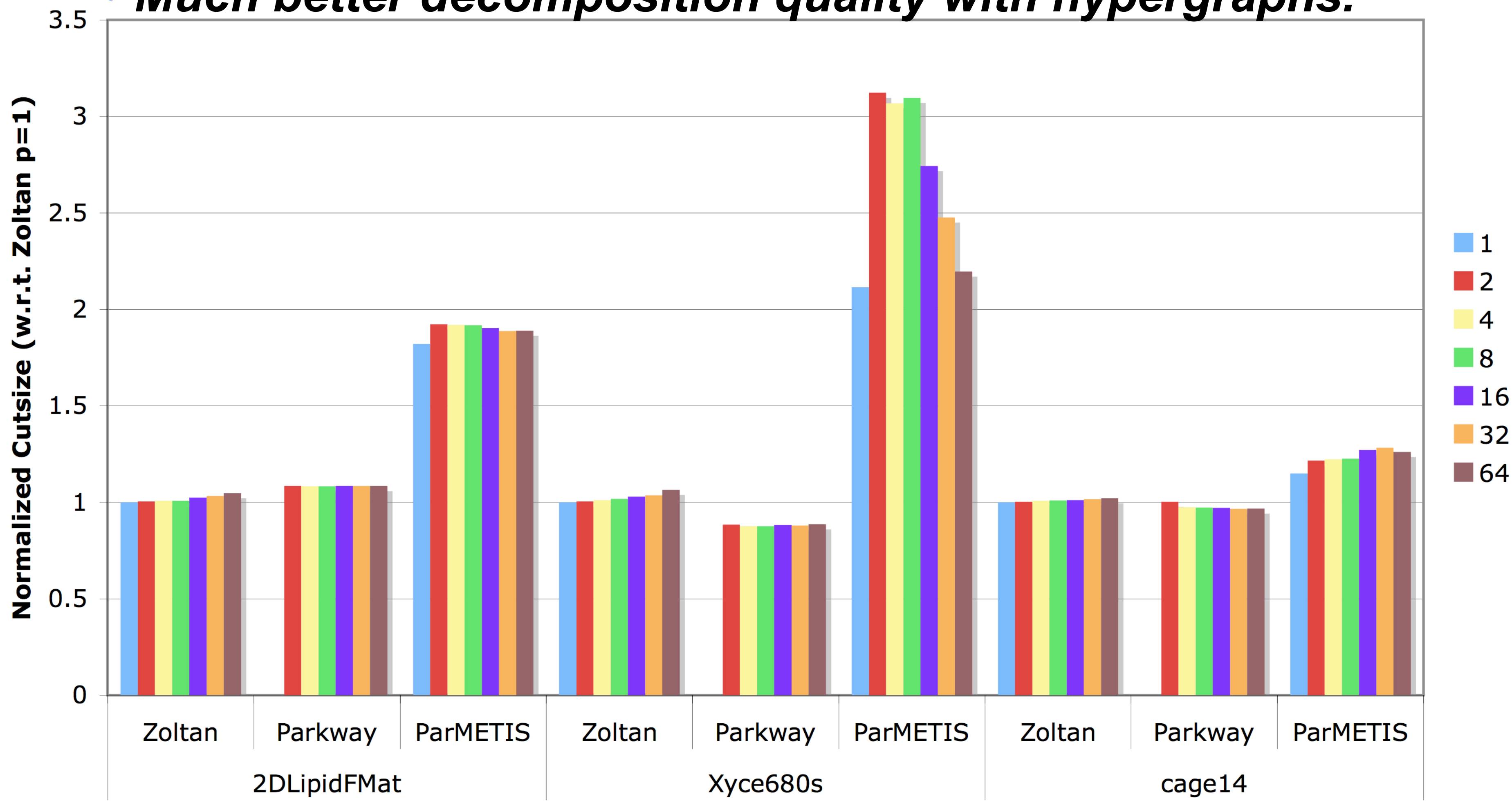
Xyce ASIC Stripped



Cage Electrophoresis

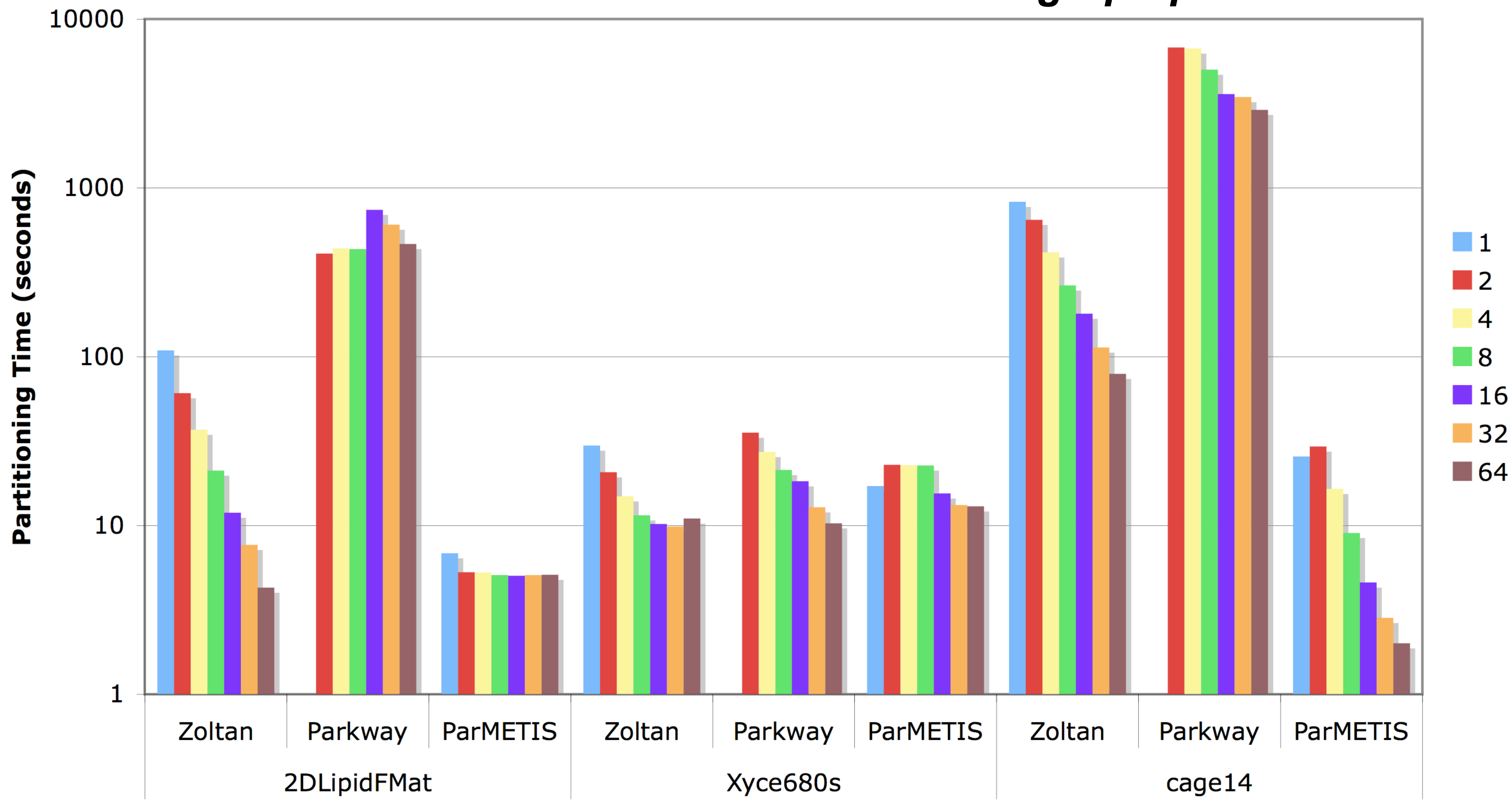
Hypergraph Partitioning: Decomposition Quality

- **64 partitions; p = 1, 2, 4, 8, 16, 32, 64**
- **Cut metric normalized w.r.t. Zoltan with p = 1.**
- ***Much better decomposition quality with hypergraphs.***



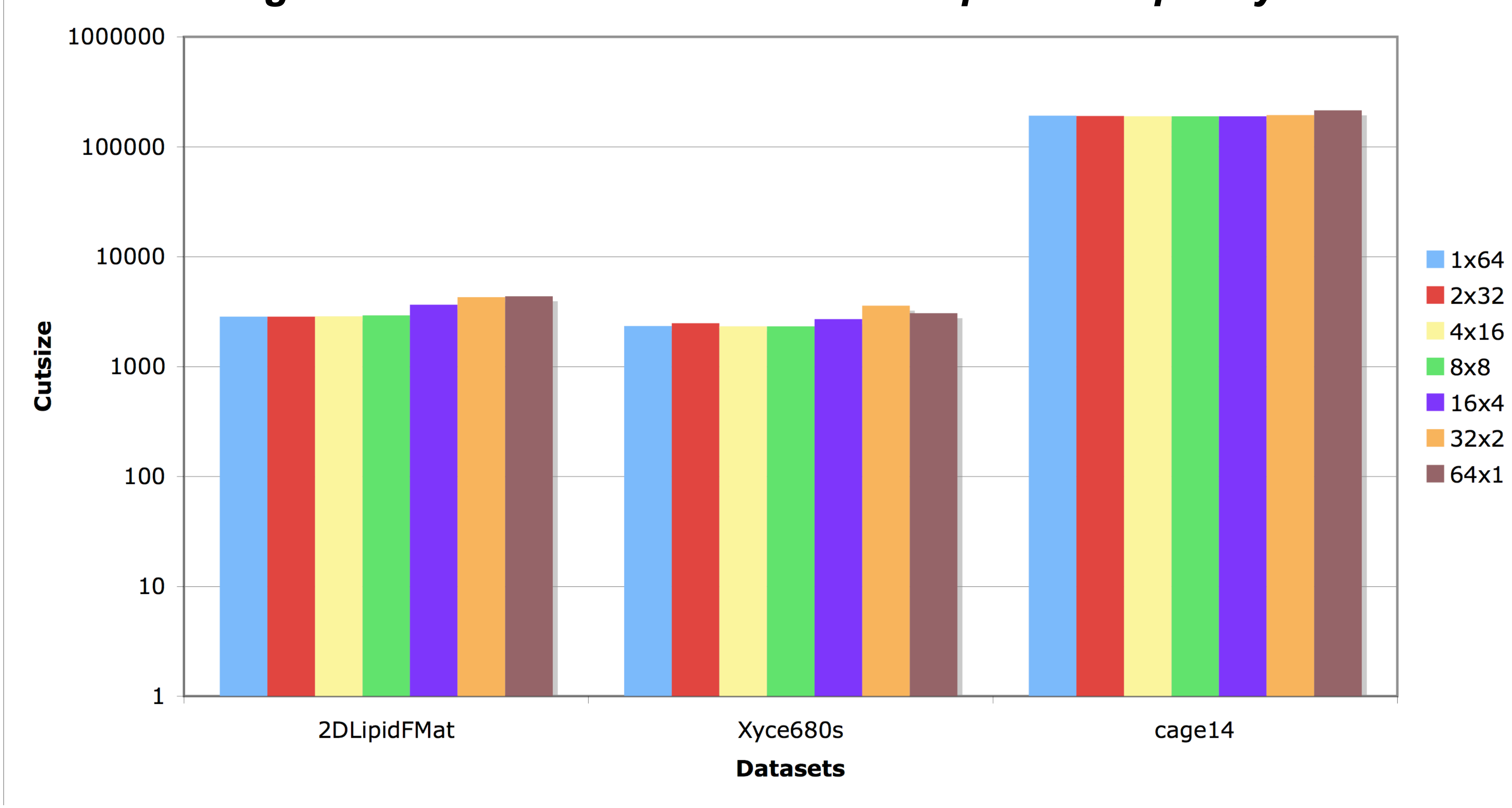
Hypergraph Partitioning Parallel Performance

- **64 partitions; p = 1, 2, 4, 8, 16, 32, 64**
- ***Execution time can be higher with hypergraphs, but not always.***
- ***Zoltan PHG scales as well as or better than graph partitioner.***



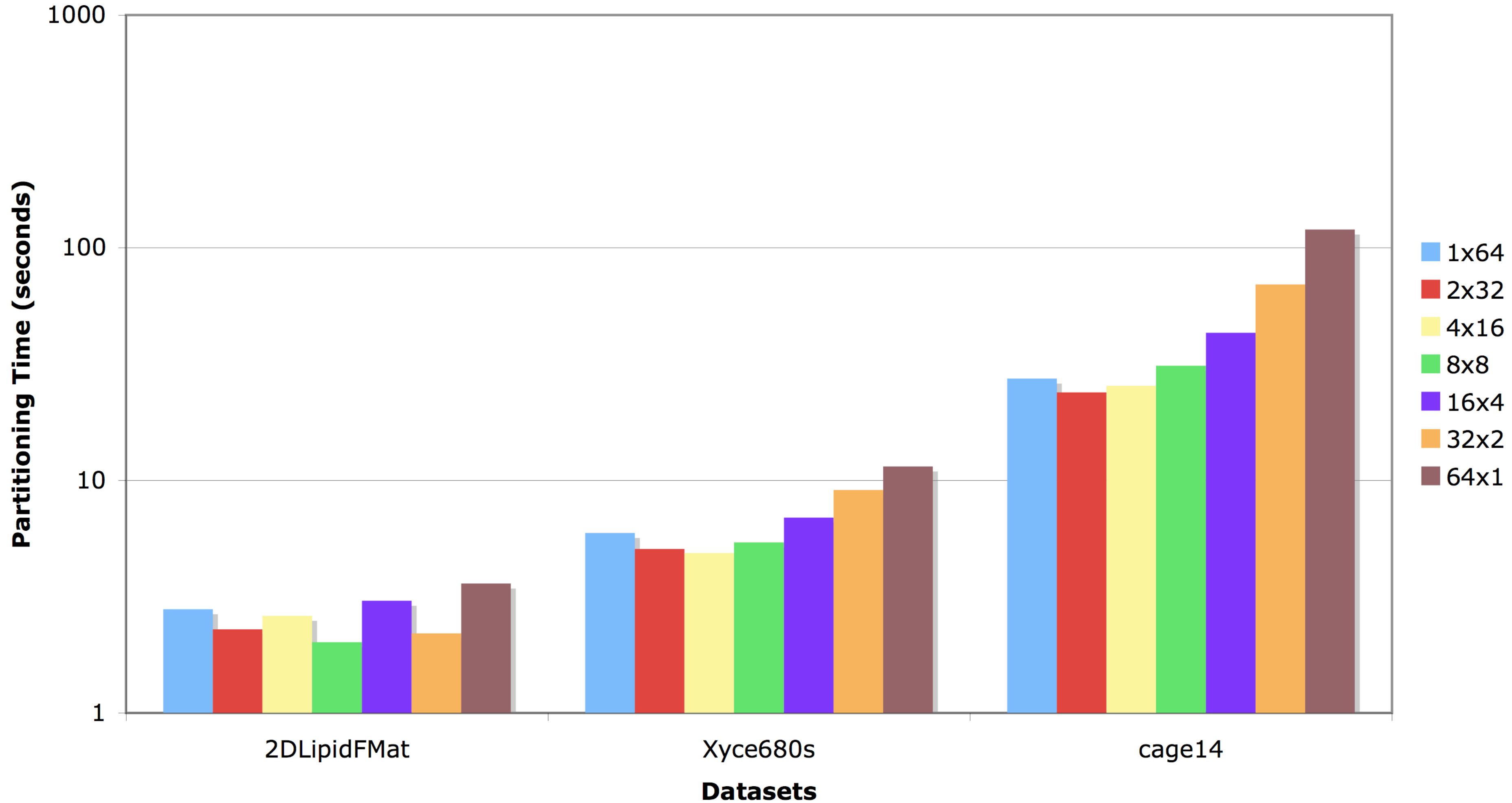
Zoltan 2D Distribution: Decomposition Quality

- **Processor configurations: 1x64, 2x32, 4x16, 8x8, 16x4, 32x2, 64x1.**
- ***Configuration has little affect on decomposition quality.***



Zoltan 2D Distribution: Parallel Performance

- **Processor configurations 1x64, 2x32, 4x16, 8x8, 16x4, 32x2, 64x1.**
- ***Configurations 2x32, 4x16, 8x8 are best.***



Summary and Future Work

- Hypergraph partitioning offers effective partitioning for irregular problems.
  - Supports wide range of applications.
  - Accurate communication metric results in lower application communication volume.
- Future work:
  - Incremental partitioning for dynamic applications.
    - Faster partitioning.
    - Minimize data migration.
  - Multicriteria partitioning.
    - Support multiphase simulations.
  - 2D Matrix partitioning.
    - 2D Cartesian
    - 2D Recursive
    - Fine-grained

For more information…

- Zoltan web site
  - http://www.cs.sandia.gov/Zoltan

- Download available March 2006: Zoltan 2.0.
  - Open-source, LGPL.

- Email:
  - kddevin@sandia.gov