



Parallel Hypergraph Partitioning for Scientific Computing

Erik Boman, Karen Devine, Robert Heaphy, Bruce Hendrickson
Sandia National Laboratories, Albuquerque

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 & hypergraph partitioning

- Applications in scientific computing:
 - Load balancing
 - minimize communication
 - Sparse matrix-vector product
 - Decomposition for LP
 - Matrix orderings
 - Parallel preconditioners
- Graph partitioning
 - Commonly used, but has deficiencies
 - Doesn't accurately represent communication volume
 - Not suitable for non-symmetric & rectangular matrices

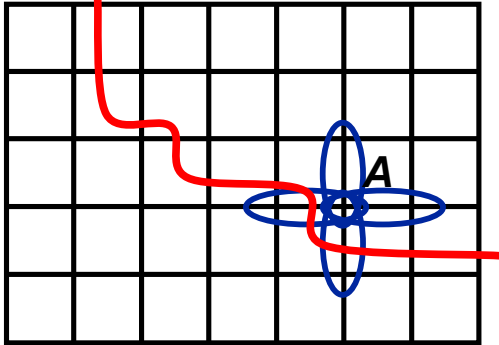
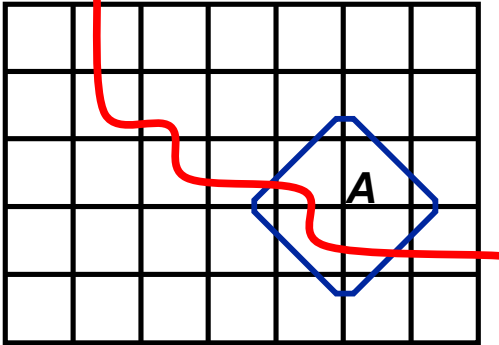


Hypergraph partitioning

- Hypergraph partitioning
 - More powerful than graph model
 - A hyperedge connects a set of vertices
 - Represents communication volume accurately
 - Aykanat & Catalyurek ('95-'99)
- Problem definition:
 - Given hypergraph $H=(V,E')$ and integer k
 - where E' is a set of hyperedges
 - Partition V into k disjoint subsets
 - Such that each subset has (approx.) same size and
 - Number of hyperedges cut between subsets is minimized (scaled by number of parts)
- NP-hard
 - But fast multilevel heuristics work well



Graph Partitioning vs. Hypergraph Partitioning

Graph Partitioning Kernighan, Lin, Schweikert, Fiduccia, Mattheyes, Pothen, Simon, Hendrickson, Leland, Kumar, Karypis, et al.	Hypergraph Partitioning Kernighan, Alpert, Kahng, Hauck, Borriello, Aykanat, Çatalyürek, 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.
	



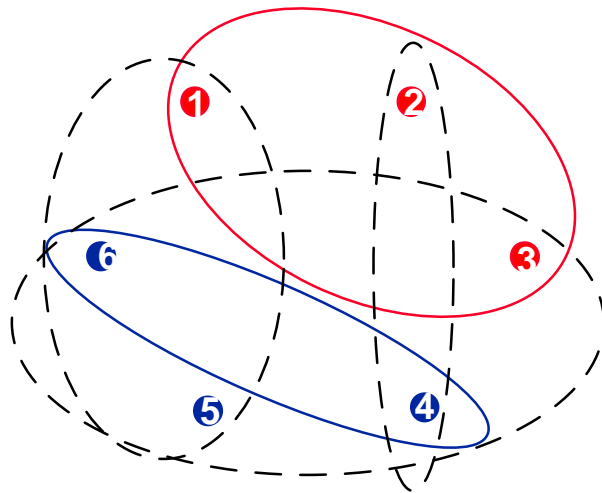
Hypergraph Partitioning

- Several serial hypergraph partitioners available.
 - hMETIS (Karypis)
 - PaToH (Çatalyürek)
 - Mondriaan (Bisseling)
- Parallel partitioners needed for large and dynamic problems.
 - Zoltan-PHG (Sandia)
 - ParKway (Trifunovic)
- Prediction:
 - Hypergraph model and partitioning tools will eventually replace graph partitioning in scientific computing
 - Except when partitioning time is important and quality matters less



Matrix Representation

- View hypergraph as matrix (Aykanat & Çatalyürek)
 - We use row-net model:
 - Vertices == columns
 - Edges == rows
- Ex: 1D partitioning of sparse matrix (along columns)

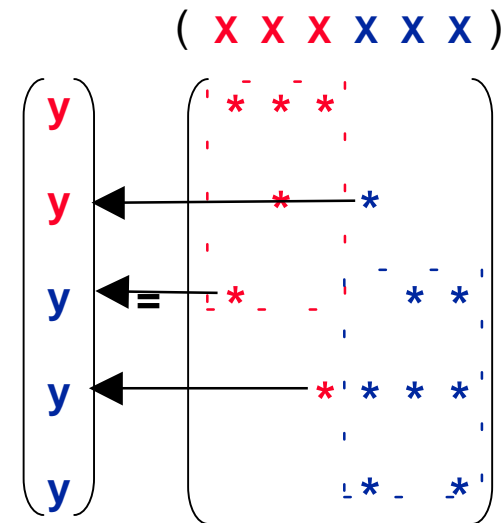


$$\begin{pmatrix} y \\ y \\ y \\ y \\ y \end{pmatrix} = \begin{pmatrix} * & * & * & & & \\ & * & & * & & \\ * & & & & * & * \\ & & * & * & * & * \\ & & & * & & * \end{pmatrix} \begin{pmatrix} x \\ x \\ x \\ x \\ x \\ x \end{pmatrix}$$

Sparse Matrix-Vector Product

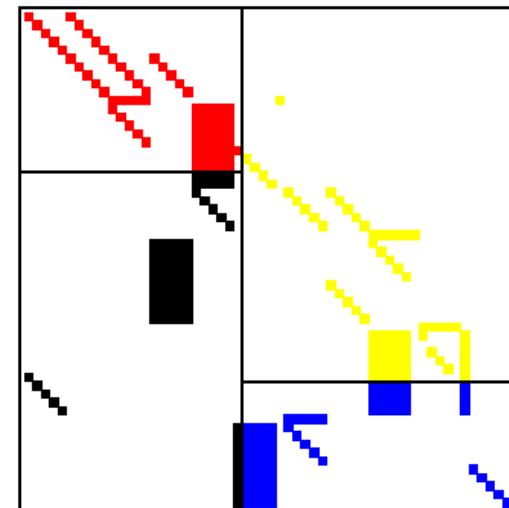
- Matrix-vector product
 - Important in scientific computing
 - Iterative methods
- Communication volume associated with edge e :

$$C_e = (\# \text{ processors in edge } e) - 1$$
- Total communication volume : $V = \sum_e C_e$



Sparse Matrix Partitioning

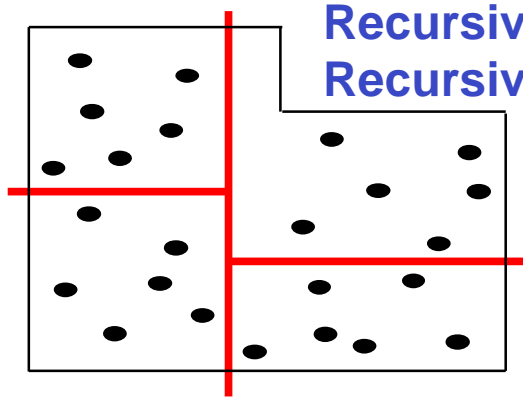
- 1D rows or columns:
 - hypergraph partitioning (Aykanat & Catalyurek)
- 2D Cartesian
 - Multiconstraint h.g.part. (Catalyurek & Aykanat)
- 2D recursive:
 - Mondriaan (Bisseling & Vastenhouw)
 - Non-Cartesian
 - Lower comm. volume
- Fine-grain model:
 - Hypergraph, each nonzero is a vertex (Catalyurek)
 - Ultimate flexibility



Courtesy: Rob
Bisseling

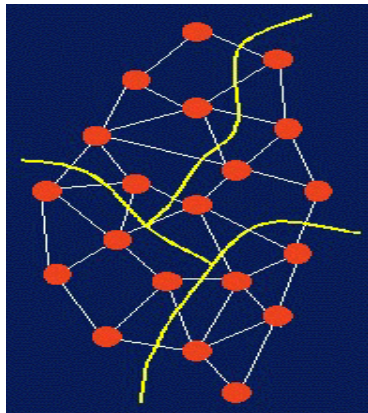
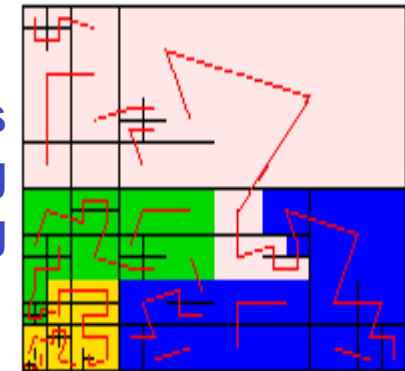


Zoltan Toolkit: Suite of Partitioning Algorithms



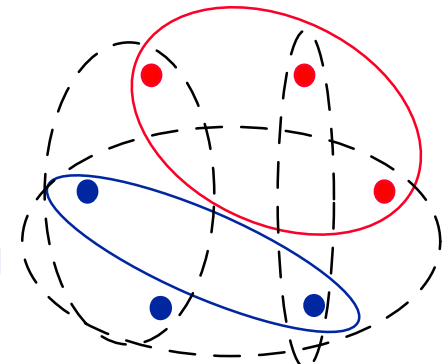
Recursive Coordinate Bisection
Recursive Inertial Bisection

Space Filling Curves
Refinement-tree Partitioning
Octree Partitioning



Graph Partitioning
ParMETIS , Jostle

Hypergraph Partitioning
NEW!





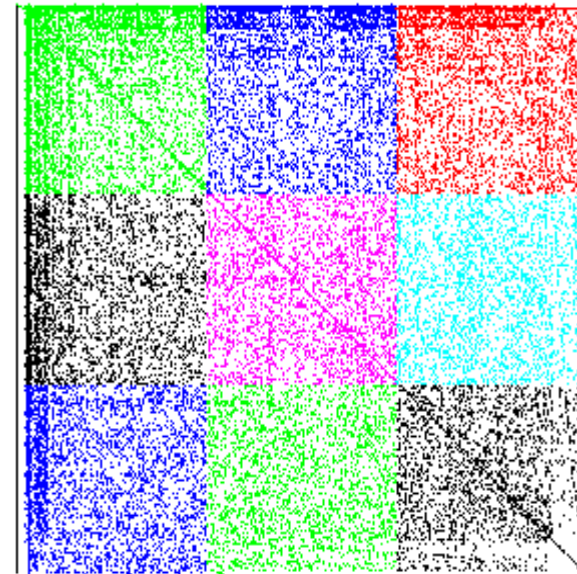
Zoltan Hypergraph Partitioner

- Parallel hypergraph partitioner
 - for large-scale problems
 - distributed memory (MPI)
- New package in the Zoltan toolkit
 - Available Fall 2005
 - Open source; LGPL



Data Layout

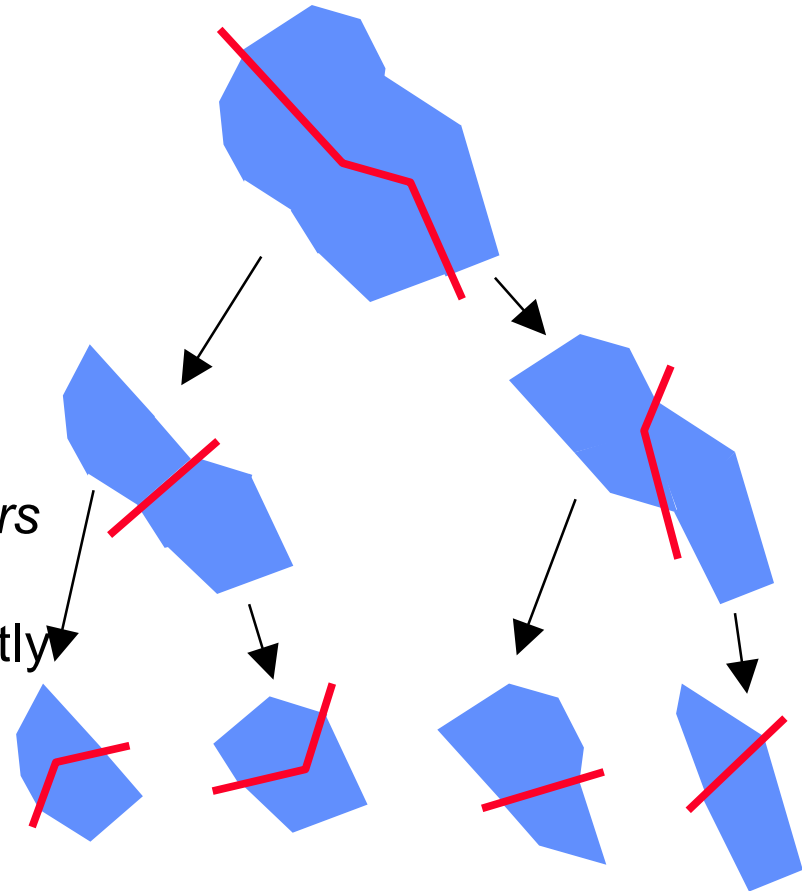
- **2D data layout** within hypergraph partitioner.
 - Does not affect the layout returned to the application.
 - Processors logically (not physically) organized as a 2D grid
 - Vertex/hyperedge communication limited to only \sqrt{p} processors (along rows/columns)
 - Maintain scalable memory usage.
 - No “ghosting” of off-processor neighbor info.
 - Differs from parallel graph partitioners and Parkway (1D).
 - Design allows comparison of 1D and 2D distributions.





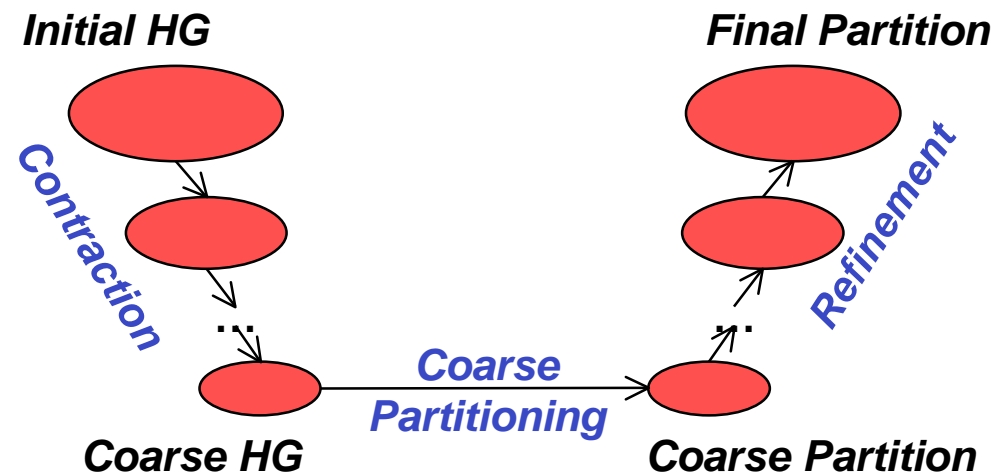
Recursive Bisection

- Recursive bisection approach:
 - Partition data into two sets.
 - Recursively subdivide each set into two sets.
 - We allow arbitrary k ($k \neq 2^n$).
- Parallelism:
 - Split *both the data and processors* into two sets; subproblems solved independently 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

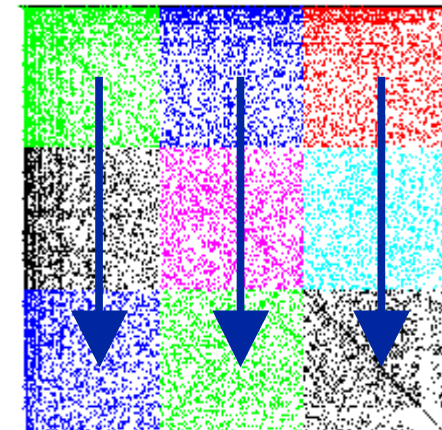
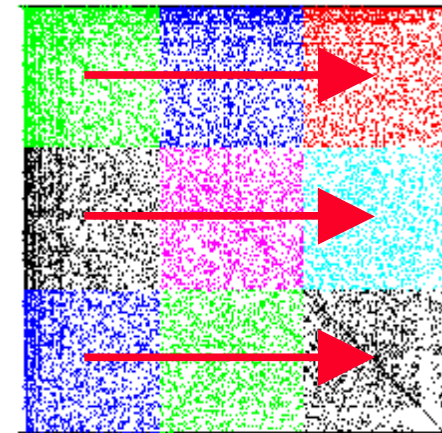


Contraction

- Merge pairs of “similar” vertices: matching
 - Currently no agglomeration of more than 2 vertices
- Greedy maximal weight matching heuristics
 - Matching is on a related graph (edges = similarities)
 - Maximum weight solution not necessary
- We use
 - Heavy connectivity matching (Aykanat & Çatalyürek)
 - Inner-product matching (Bisseling)
 - First-Choice (Karypis)
 - Match columns with greatest inner product \Rightarrow vertices with most shared hyperedges

Parallel Matching in 2D Data Layout

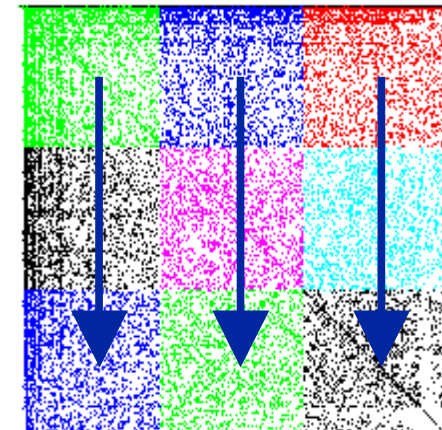
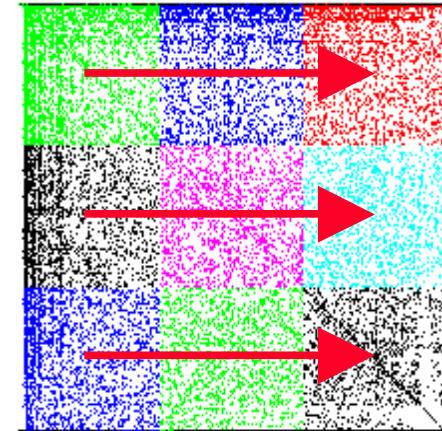
- On each processor:
 - Broadcast subset of vertices (“candidates”) along processor row.
 - Compute (partial) inner products of received candidates with local vertices.
 - Accrue inner products in processor column.
 - Identify best local matches for received candidates.
 - Send best matches to candidates’ 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

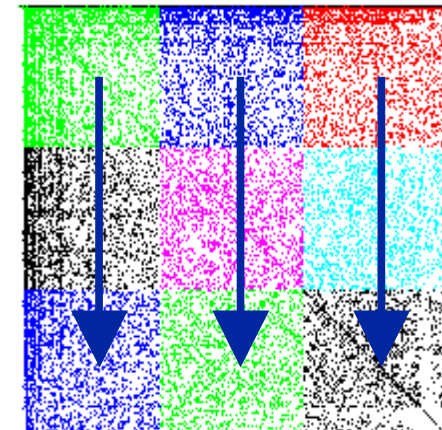
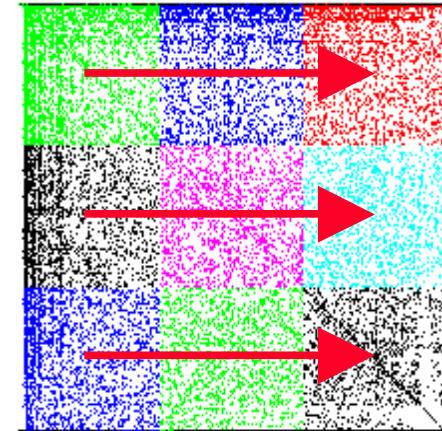
- Gather coarsest hypergraph to each processor.
 - Gather edges to each processor in column.
 - Gather vertices to each processor in row.
- Compute several different coarse partitions on each processor.
- Select best local partition.
- Compute best over all processors.
- Broadcast best partition to all.





Refinement

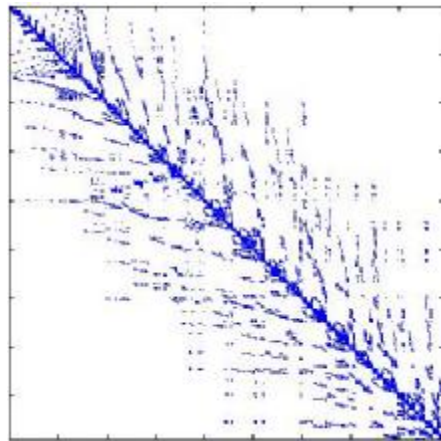
- For each level in V-cycle:
 - Project coarse partition to finer hypergraph.
 - Use local optimization (KL/FM) to improve balance and reduce cuts.
 - Compute “root” processor in each processor column: processor with most nonzeros.
 - Root processor computes moves for vertices in processor column.
 -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



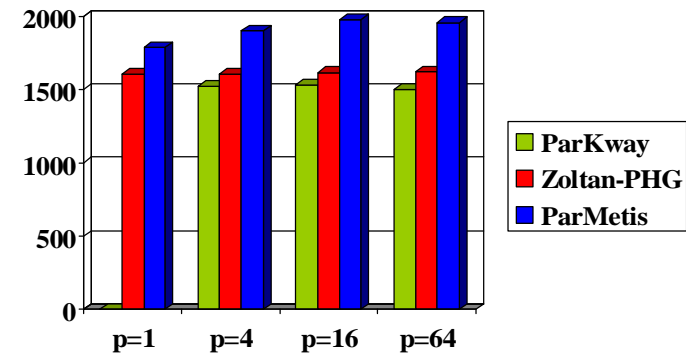


Results

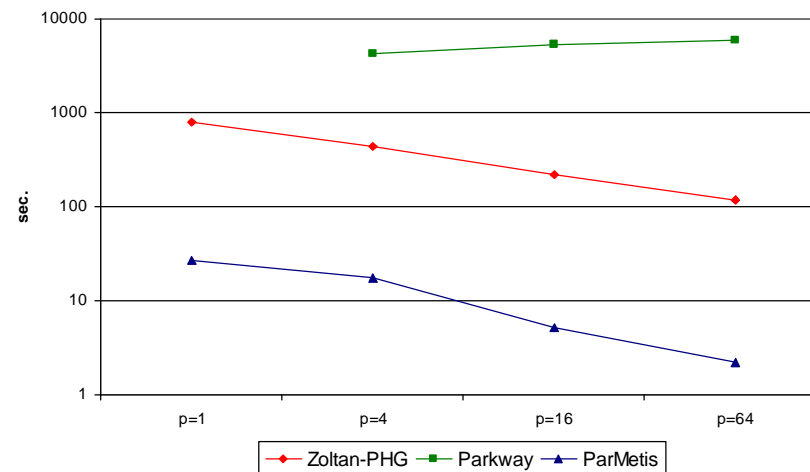
- Cage14: Cage model of DNA electrophoresis (van Heukelum)
 - 1.5M rows & cols; 27M nonzeros.
 - Symmetric structure
 - 64 partitions.
- Hypergraph partitioning reduced communication volume by 10-20% vs. graph partitioning.
- Zoltan much faster than ParkWay



Hypergraph cuts



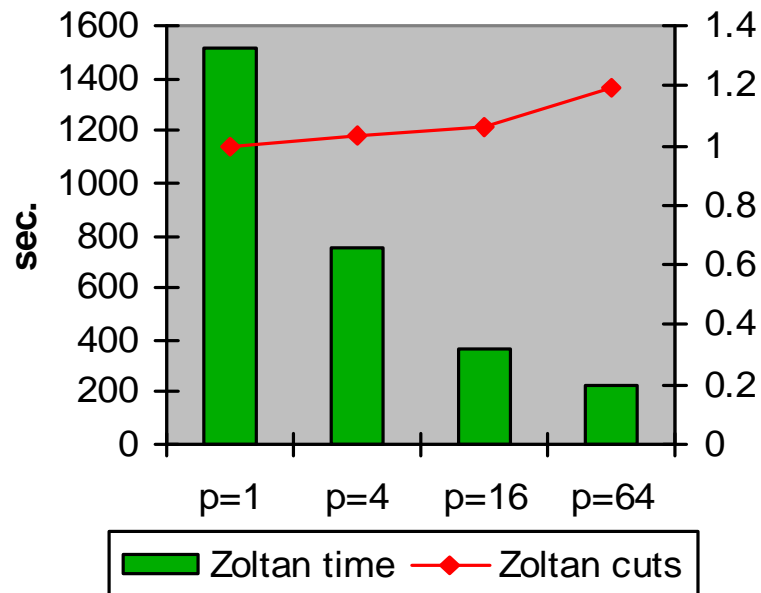
Time



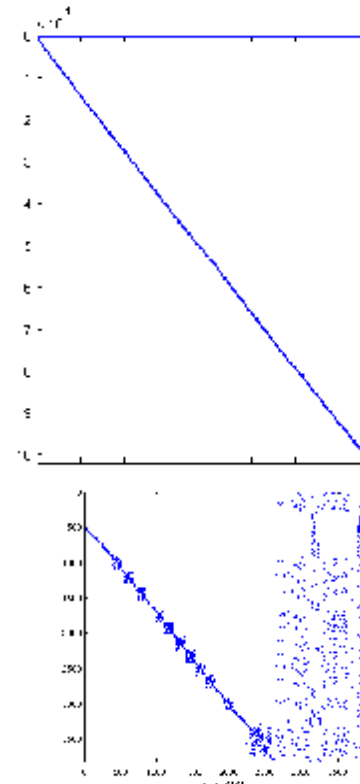


More Results

- Sensor placement – IP/LP model
 - 5M rows, 4M columns
 - 16M nonzeros



- ParKway ran out of memory
 - 1d with ghosting, not scalable





Future Work

- Increase speed while maintaining quality.
 - Heuristics for more local, less expensive matching
 - Better load balance within our code
 - K-way refinement?
- More evaluation of current design.
 - 1D vs. 2D data layouts
- Incremental partitioning for dynamic applications.
 - Minimize data migration.
- Multiconstraint partitioning
- Interface for 2D partitioning (sparse matrices)
- ***Watch for release in Zoltan later this year!***

