Recent Advances in Two-Dimensional Sparse Matrix Partitioning

SIAM PP10

2/26/2010

***Michael Wolf*, Erik Boman, Cédric Chevalier**

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.

Sparse Matrix Partitioning Motivation

- Sparse matrix-vector multiplication (SpMV) is common kernel in many numerical computations
  - Iterative methods for solving linear systems
  - PageRank computation
  - …
- Need to make parallel SpMV kernel as fast as possible

Parallel Sparse Matrix-Vector Multiplication

$$\begin{bmatrix} y_1 \\ y_2 \\ y_3 \\ y_4 \\ y_5 \\ y_6 \\ y_7 \\ y_8 \end{bmatrix} = \begin{bmatrix} 1 & 6 & 0 & 0 & 0 & 0 & 0 & 0 \\ 5 & 1 & 9 & 0 & 5 & 0 & 0 & 0 \\ 0 & 8 & 1 & 7 & 0 & 0 & 0 & 0 \\ 0 & 0 & 2 & 1 & 0 & 0 & 0 & 7 \\ 0 & 0 & 0 & 0 & 1 & 8 & 0 & 0 \\ 4 & 0 & 0 & 0 & 3 & 1 & 3 & 0 \\ 0 & 0 & 0 & 6 & 0 & 9 & 1 & 4 \\ 0 & 0 & 0 & 0 & 0 & 0 & 2 & 1 \end{bmatrix} \begin{bmatrix} 1 \\ 2 \\ 4 \\ 3 \\ 1 \\ 4 \\ 2 \\ 1 \end{bmatrix}$$

$$\mathbf{y} = \mathbf{A}\mathbf{x}$$

- Partition matrix nonzeros
- Partition vectors

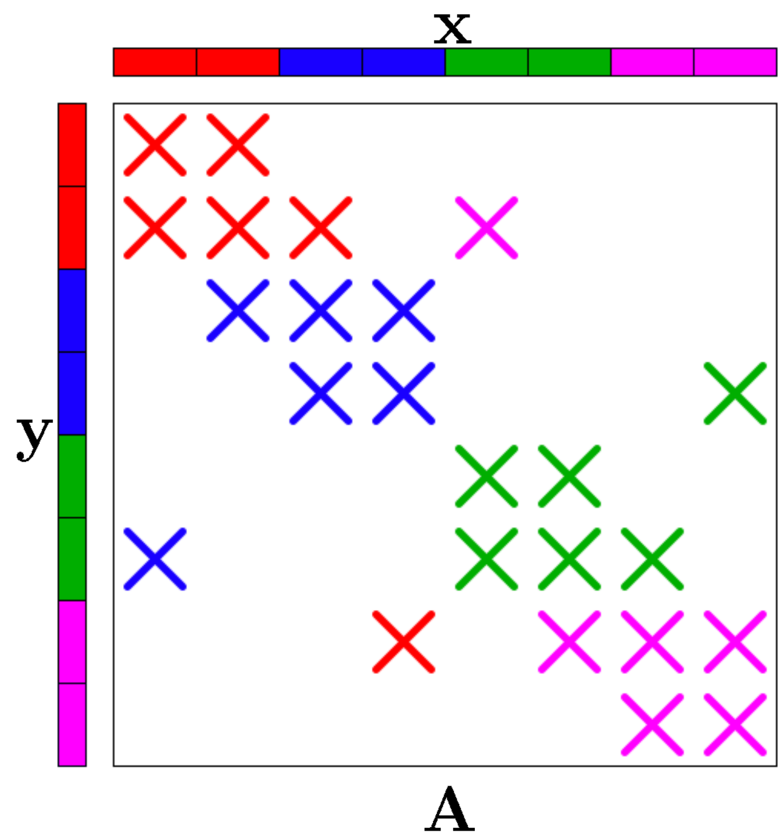
Objective

- Ideally we minimize total run-time
- Settle for easier objective
  - Work balanced
  - Minimize total communication volume
- Can partition matrices in different ways
  - 1D
  - 2D
- Can model problem in different ways
  - Graph
  - Bipartite graph
  - Hypergraph

Parallel Matrix-Vector Multiplication

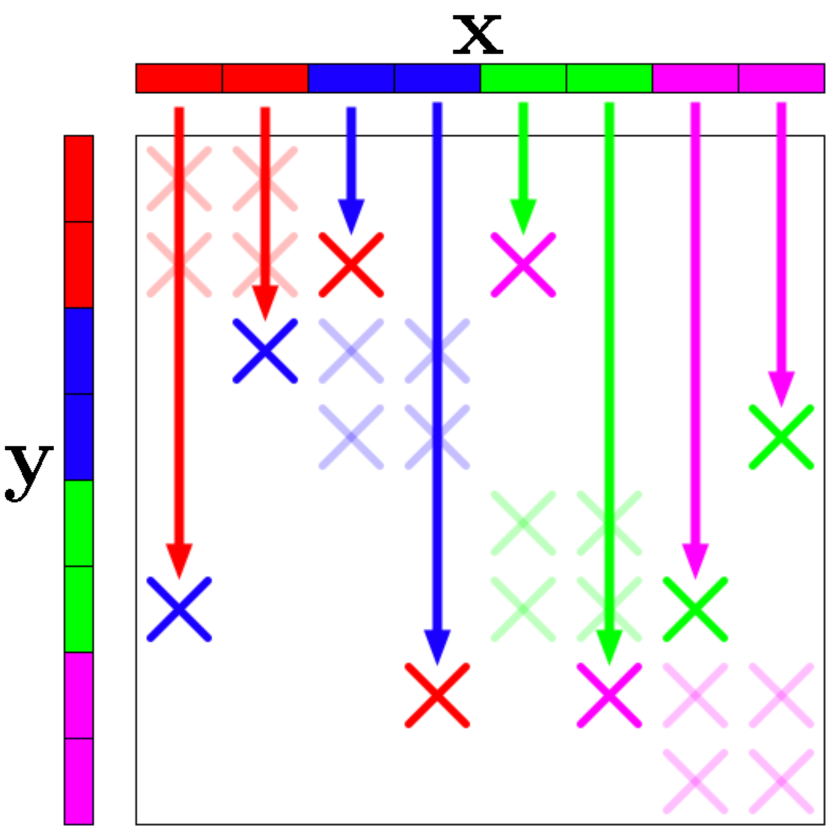
$$
\begin{bmatrix} y_1 \\ y_2 \\ y_3 \\ y_4 \\ y_5 \\ y_6 \\ y_7 \\ y_8 \end{bmatrix}
=
\begin{bmatrix}
1 & 6 & 0 & 0 & 0 & 0 & 0 & 0 \\
5 & 1 & 9 & 0 & 5 & 0 & 0 & 0 \\
0 & 8 & 1 & 7 & 0 & 0 & 0 & 0 \\
0 & 0 & 2 & 1 & 0 & 0 & 0 & 7 \\
0 & 0 & 0 & 0 & 1 & 8 & 0 & 0 \\
4 & 0 & 0 & 0 & 3 & 1 & 3 & 0 \\
0 & 0 & 0 & 6 & 0 & 9 & 1 & 4 \\
0 & 0 & 0 & 0 & 0 & 0 & 2 & 1
\end{bmatrix}
\begin{bmatrix} 1 \\ 2 \\ 4 \\ 3 \\ 1 \\ 4 \\ 2 \\ 1 \end{bmatrix}
$$

$$\mathbf{y} = \mathbf{A}\mathbf{x}$$

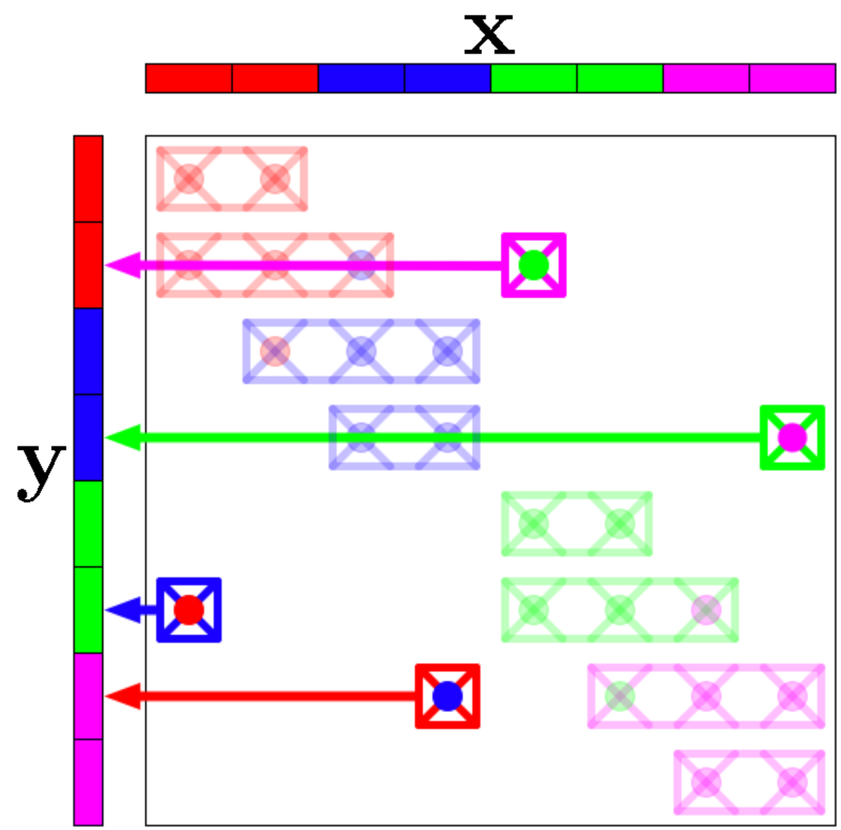


- Alternative way of visualizing partitioning

Parallel SpMV Communication

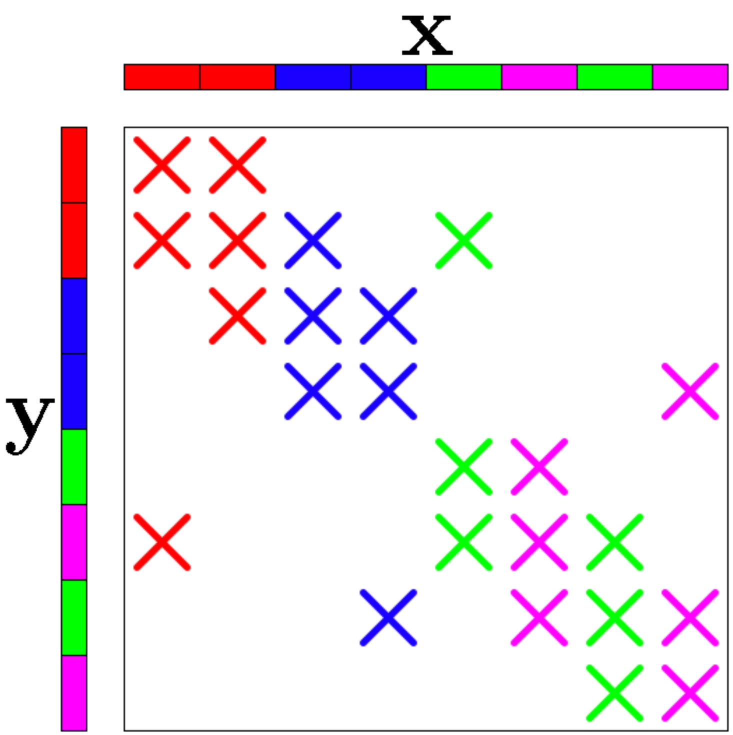


- x_j sent to remote processes that have nonzeros in column j

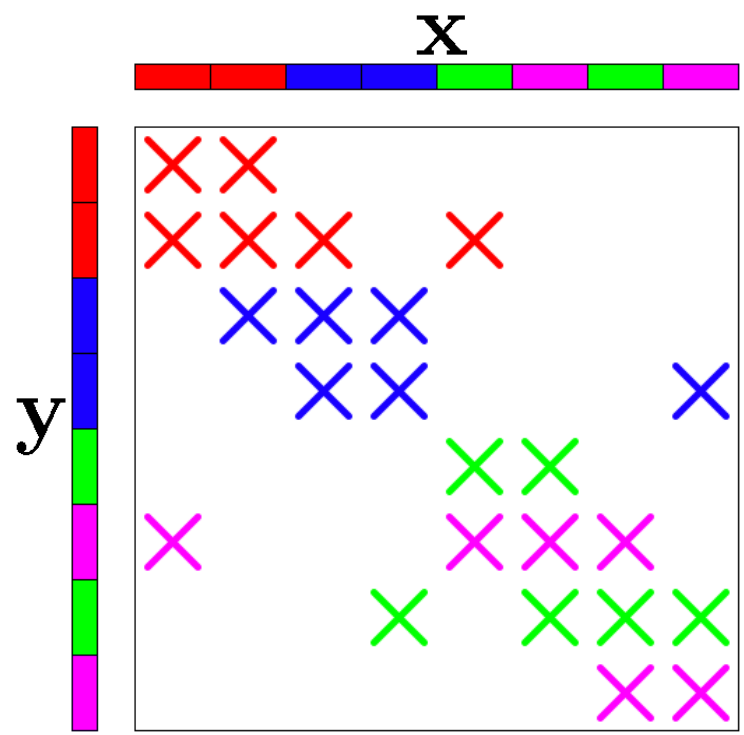


- Partial inner-products sent to process that owns vector element y_i

1D Partitioning



1D Column

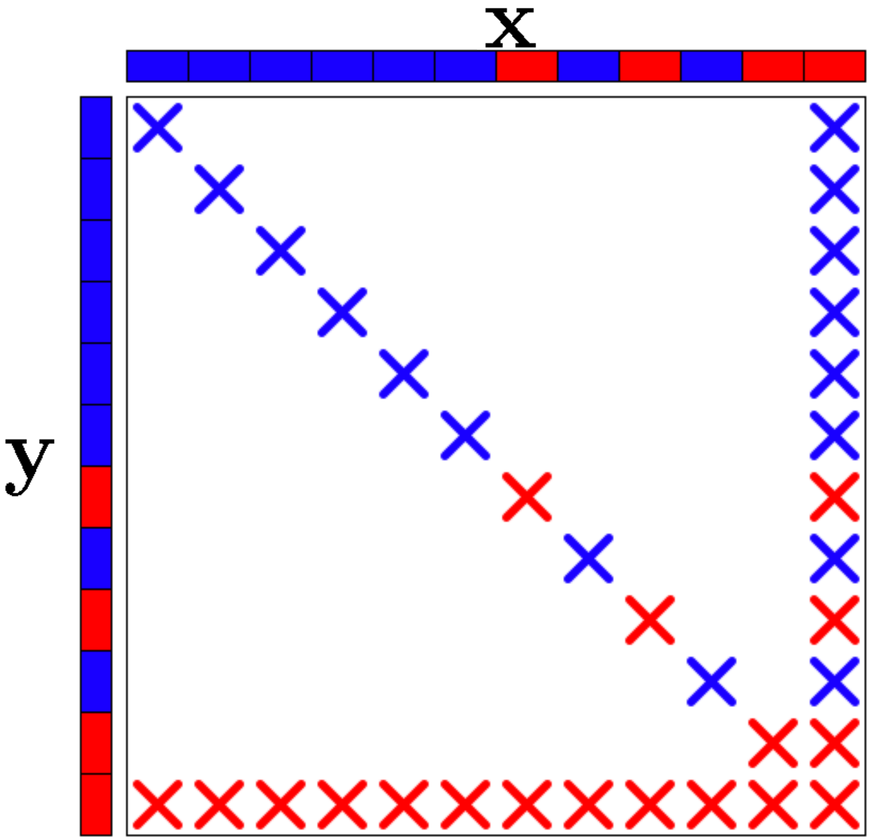


1D Row

- Each process assigned nonzeros for set of columns

- Each process assigned nonzeros for set of rows

When 1D Partitioning is Inadequate



"Arrowhead" matrix

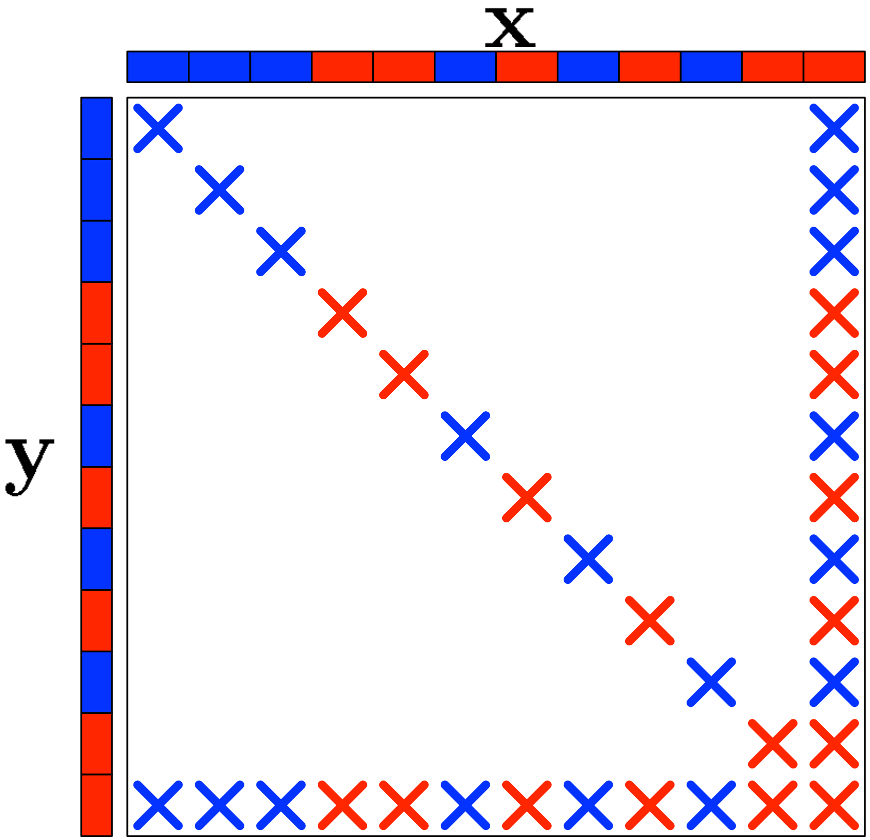
n=12

nnz=34 (18,16)

volume = 9

- For any 1D bisection of nxn arrowhead matrix:
  - nnz = 3n-2
  - Volume ≈ (3/4)n

When 1D Partitioning is Inadequate



"Arrowhead" matrix

n=12

nnz=34 (16,18)

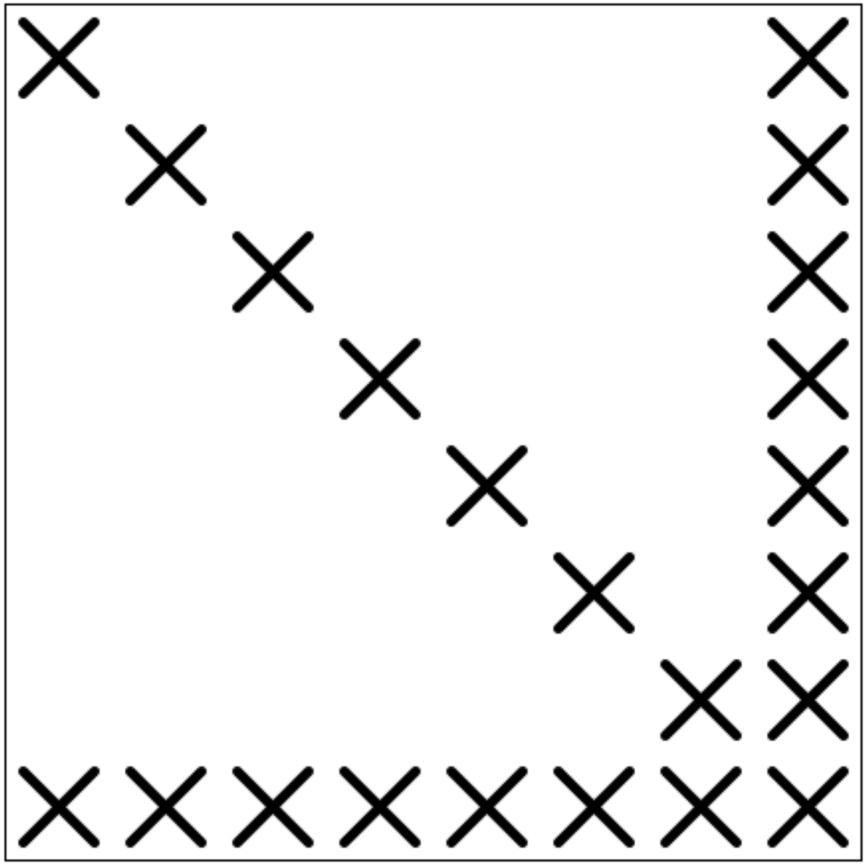
volume = 2

- 2D partitioning
- O(k) volume partitioning possible

2D Partitioning

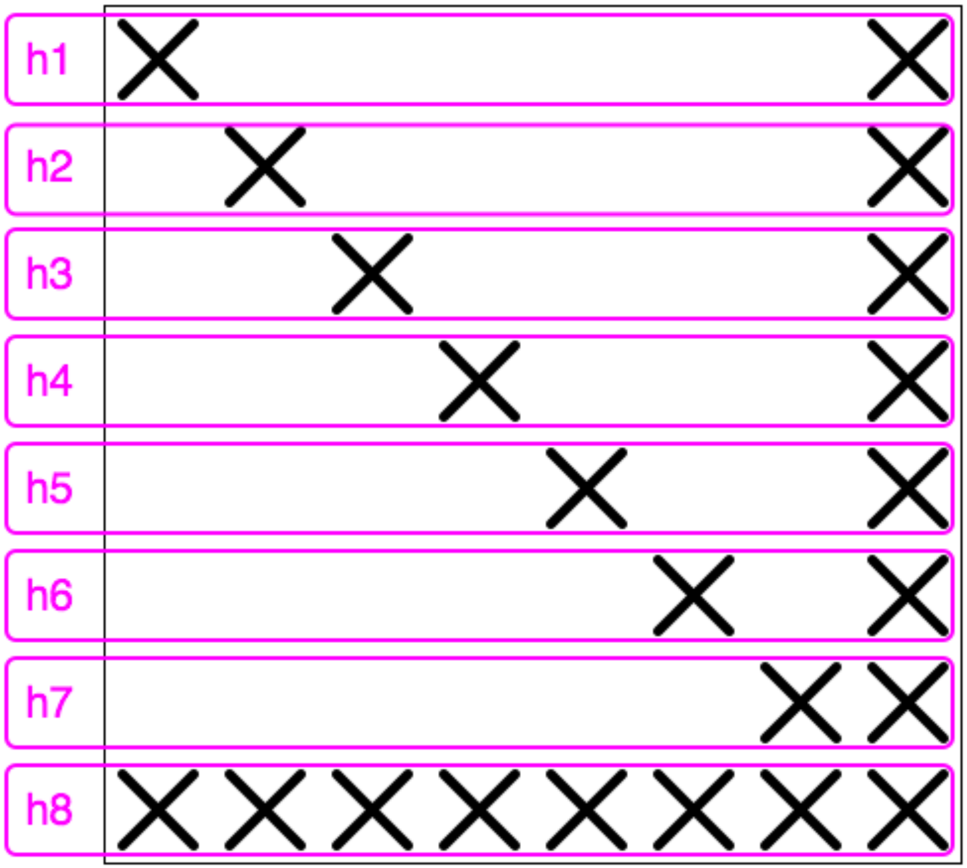
- More flexibility in partitioning
- No particular part for given row or column
- More general sets of nonzeros assigned parts
- Several methods of 2D partitioning
  - Fine-grain hypergraph
  - Coarse-grain hypergraph
  - Mondriaan
  - Nested dissection symmetric partitioning method

Fine-Grain (FG) Hypergraph Model



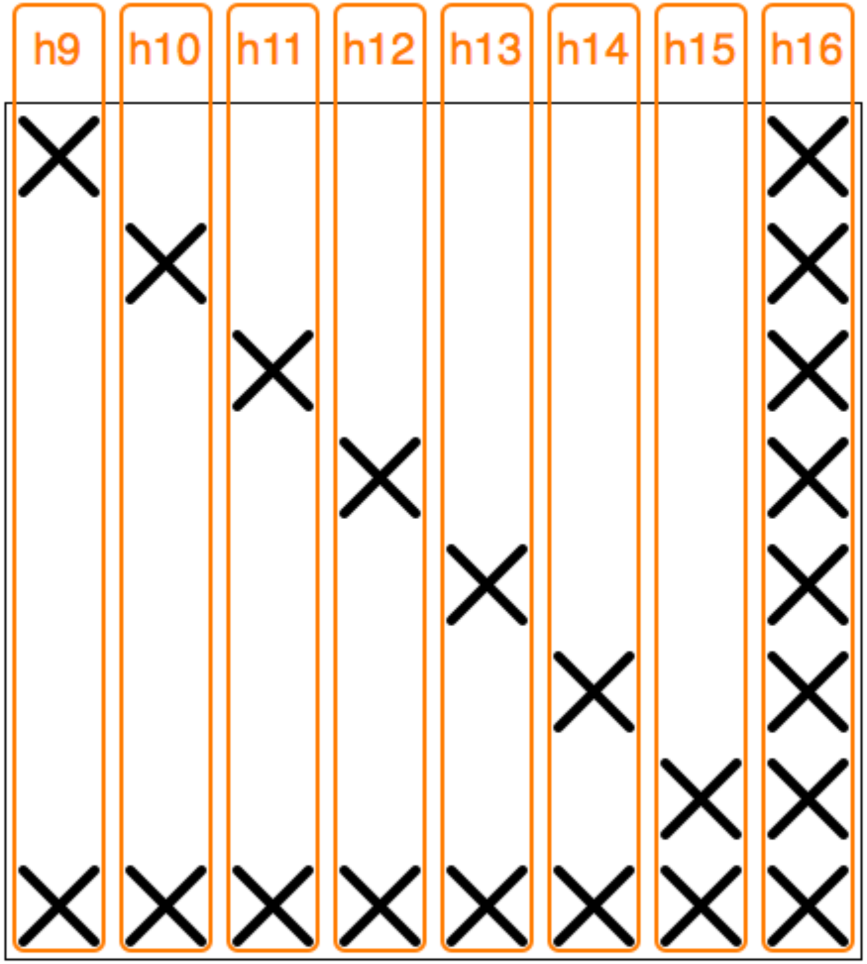
- Catalyurek and Aykanat (2001)
- Nonzeros represented by vertices in hypergraph

Fine-Grain Hypergraph Model



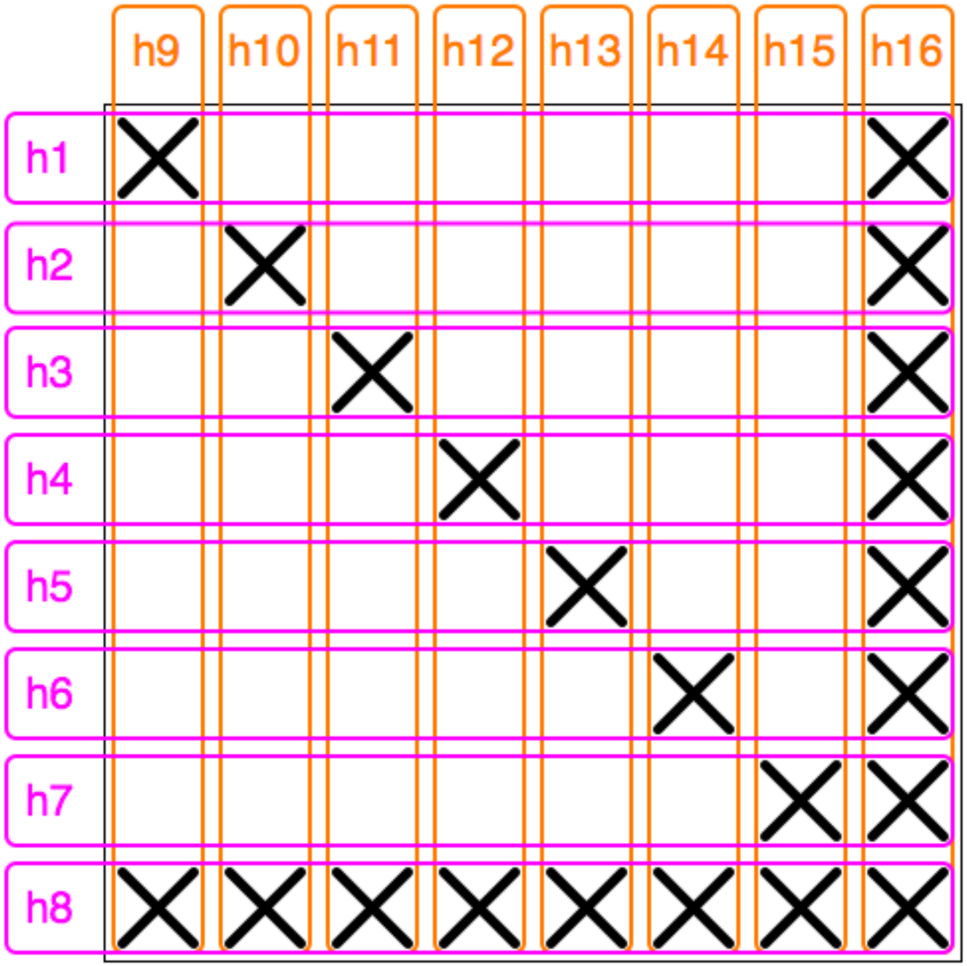
- Rows represented by hyperedges
- Hyperedge - set of one or more vertices

Fine-Grain Hypergraph Model



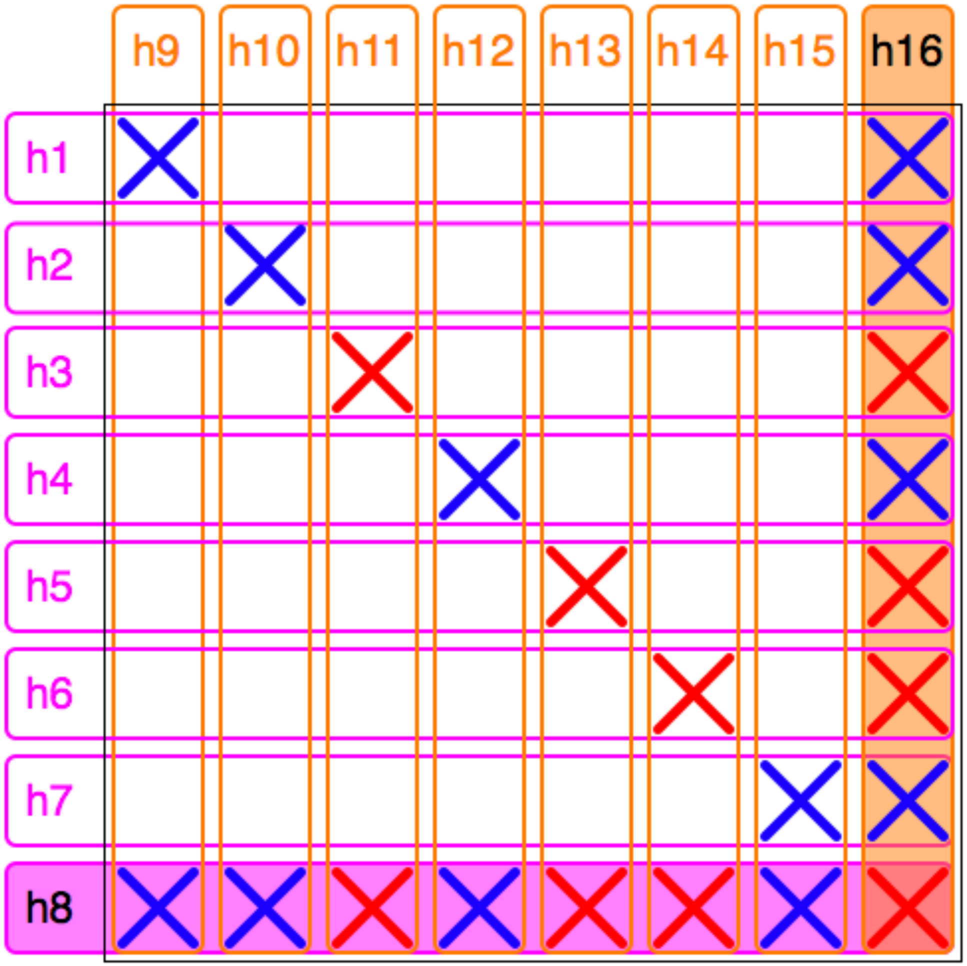
- Columns represented by hyperedges

Fine-Grain Hypergraph Model



- 2n hyperedges

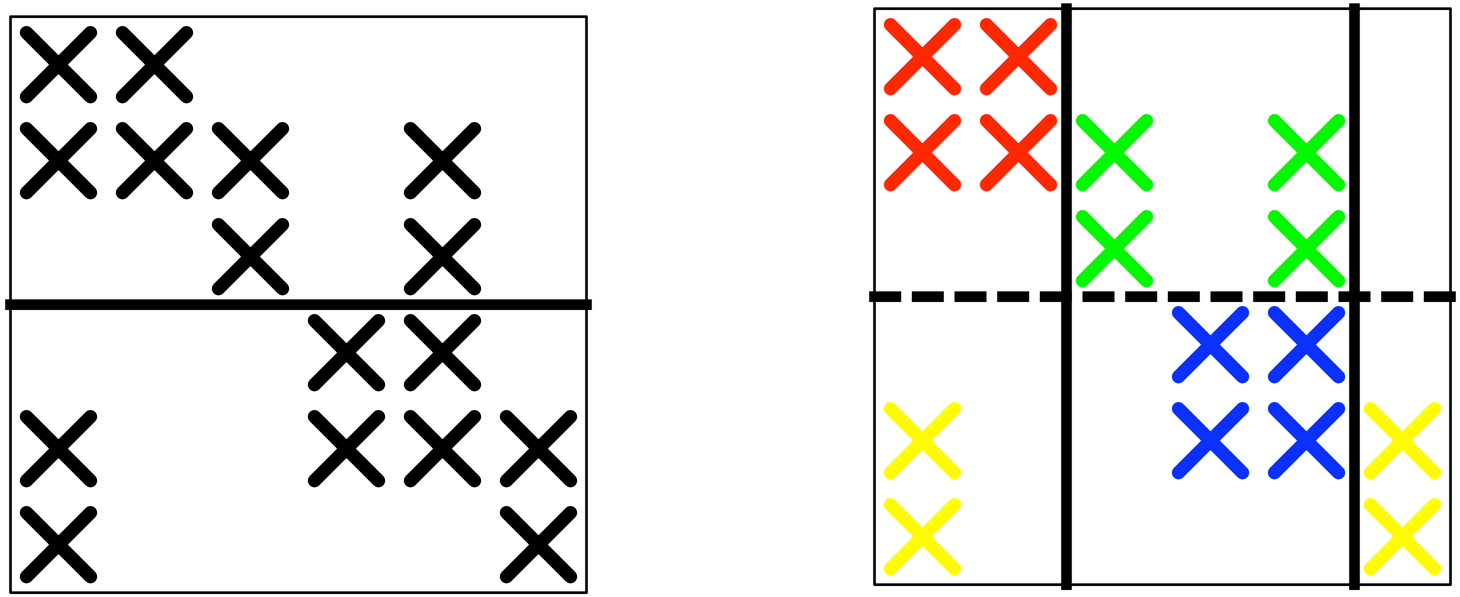
Fine-Grain Hypergraph Model



k=2, volume = cut = 2

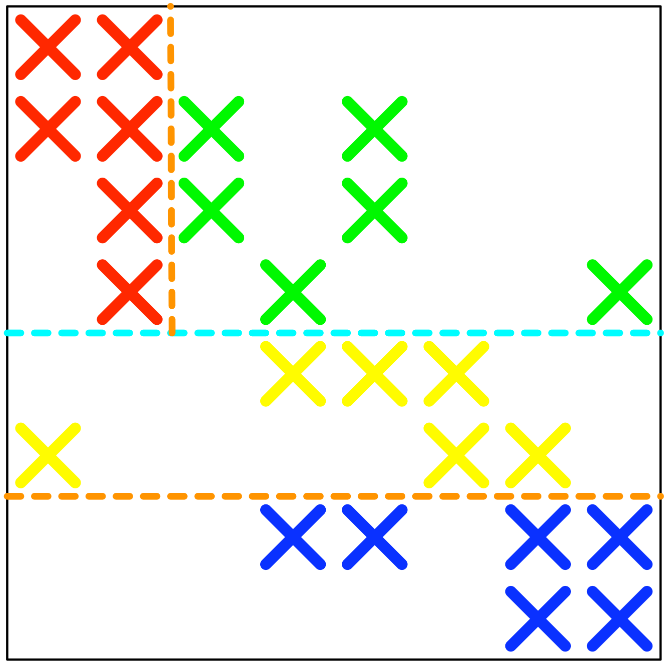
- Partition vertices into k equal sets
- For k=2
  - Volume = number of hyperedges cut
- Minimum volume partitioning when optimally solved
- Larger NP-hard problem than 1D

2D Coarse-Grain Partitioning



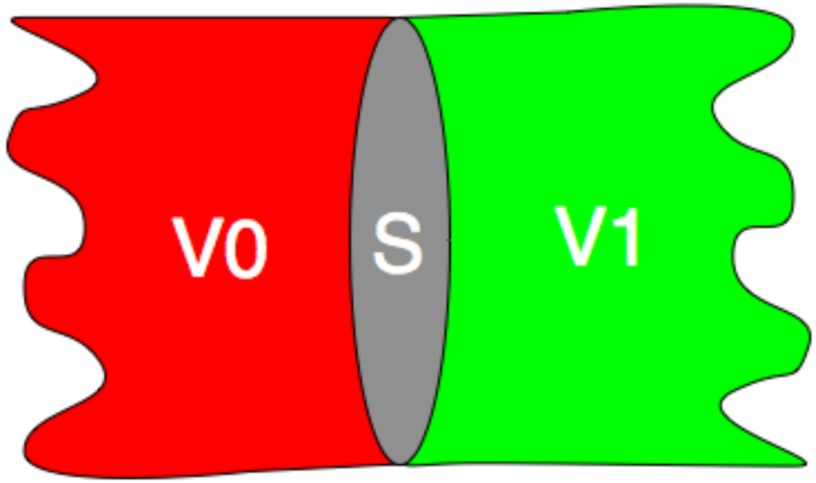
- Catalyurek and Aykanat (2001)
- Two stages:
  - 1D hypergraph partitioning
  - 1D multi-constraint hypergraph partitioning (ensures load balance)
- Bound on number of messages

Mondriaan Partitioning



- Vastenhouw and Bisseling (2005)
- Recursive bisection hypergraph partitioning
- Each level: 1D row or column partitioning

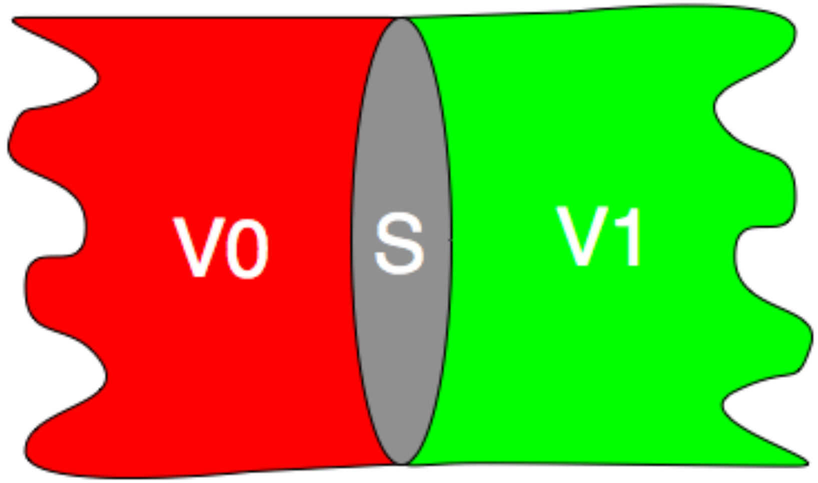
Nested Dissection Partitioning - Bisection



- Suppose A is structurally symmetric
- Let G(V,E) be graph of A
- Find small, balanced separator S
  - Yields vertex partitioning V = (**V0**,**V1**,S)
- Partition the edges such that
  - **E0** = {edges incident to a vertex in **V0**}
  - **E1** = {edges incident to a vertex in **V1**}

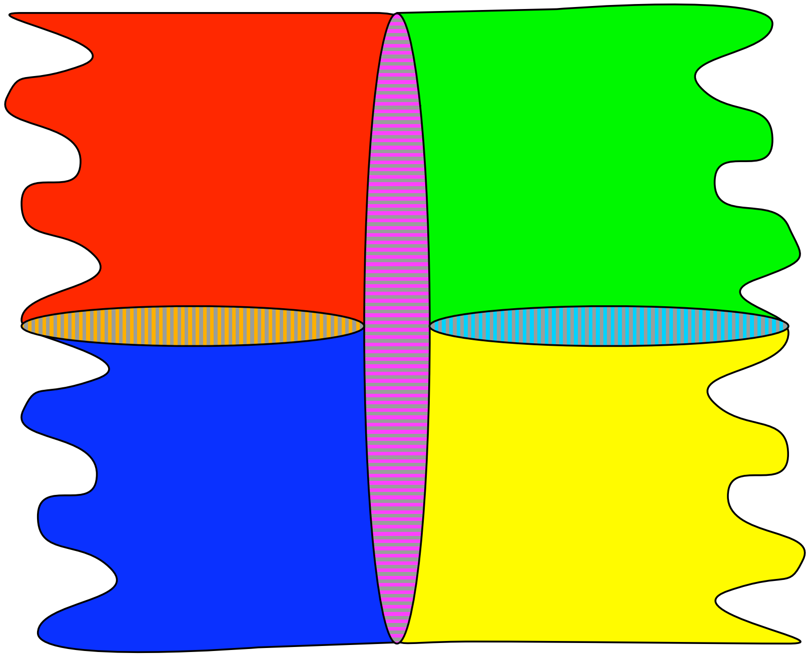
Sandia National Laboratories

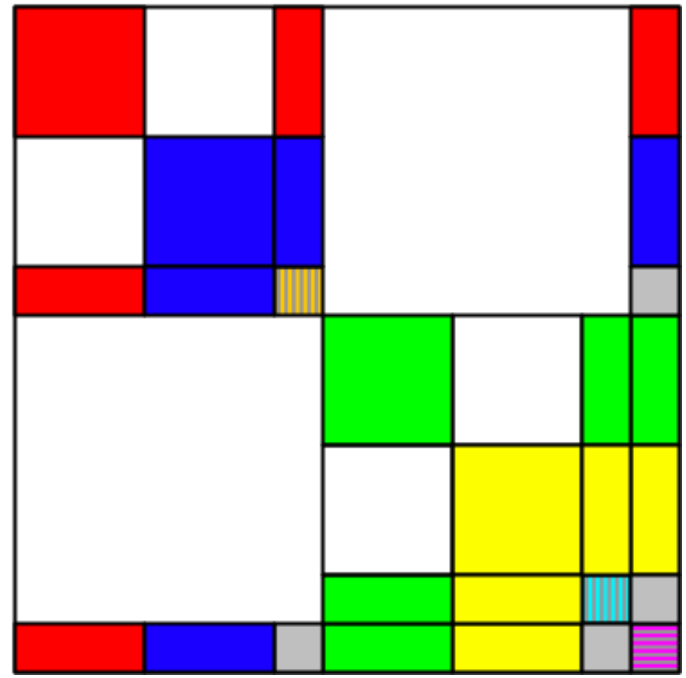
Nested Dissection Partitioning - Bisection



- Vertices in S and corresponding edges
  - Can be assigned to either part
  - Can use flexibility to maintain balance
- Communication Volume = 2*|S|
  - Regardless of S partitioning
  - |S| in each phase

Nested Dissection (ND) Partitioning Method





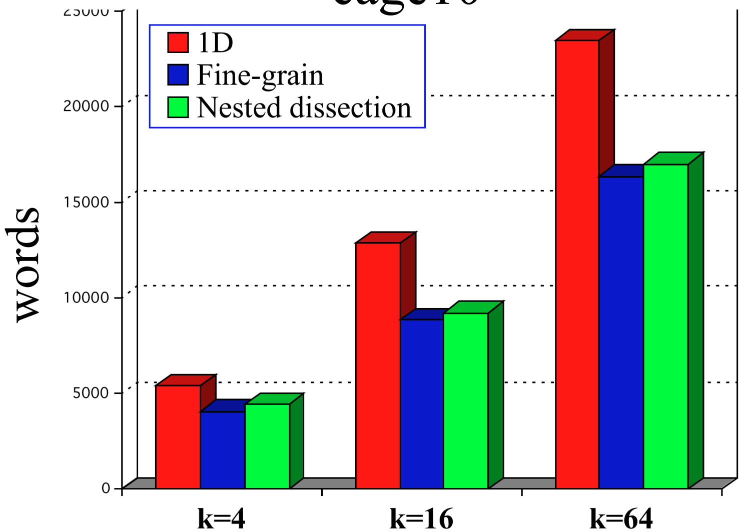
- Recursive bisection to partition into >2 parts
- Use nested dissection!

Numerical Experiments

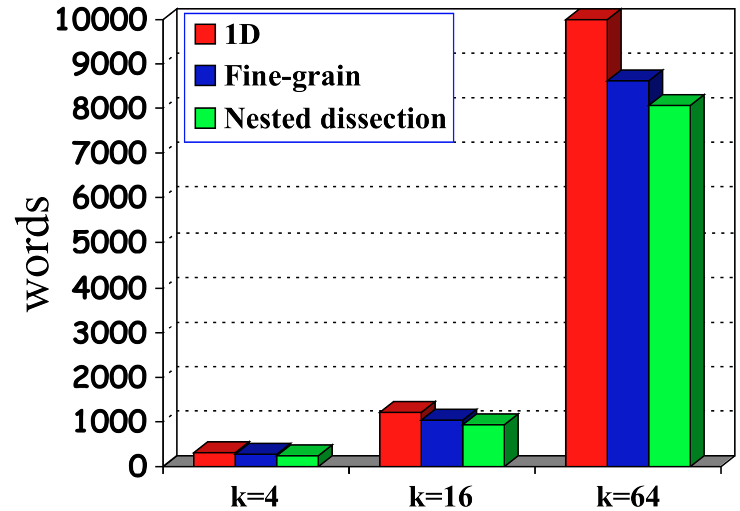
- Structurally symmetric matrices
- k = 4, 16, 64 parts using
  - 1D hypergraph partitioning
  - Fine-grain hypergraph partitioning (2D)
    - Good quality partitions but slow
  - Nested dissection partitioning (2D)
- Hypergraph partitioning for all methods
  - Zoltan (Sandia) with PaToH (Catalyurek, **serial**)
  - Allows "fair" comparison between methods
- Vertex separators derived from edge separators
  - MatchBox (Purdue: Pothen, et al.)

Communication Volume - Symmetric Matrices

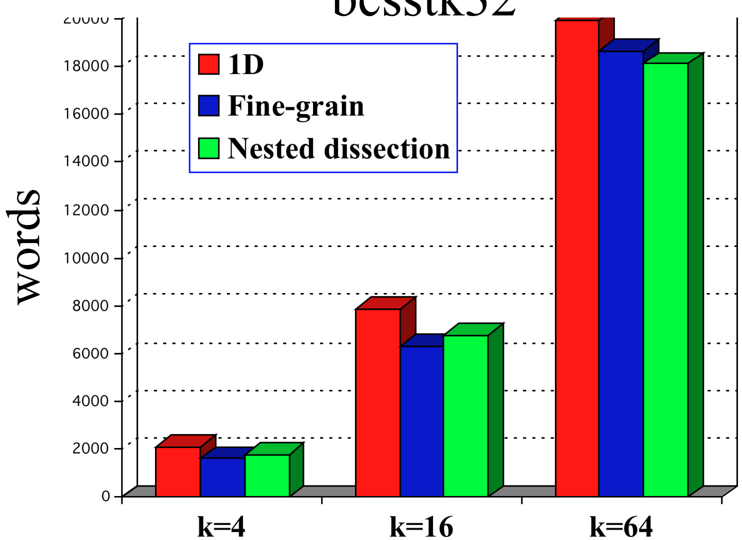
cage10



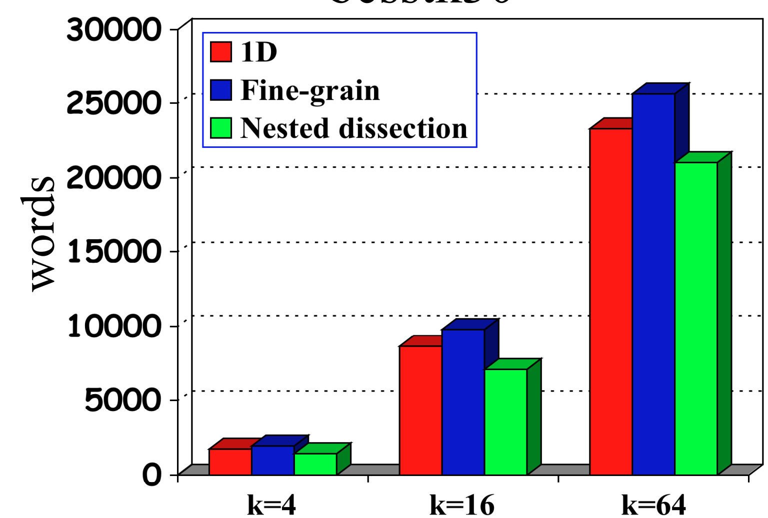
finan512



bcsstk32



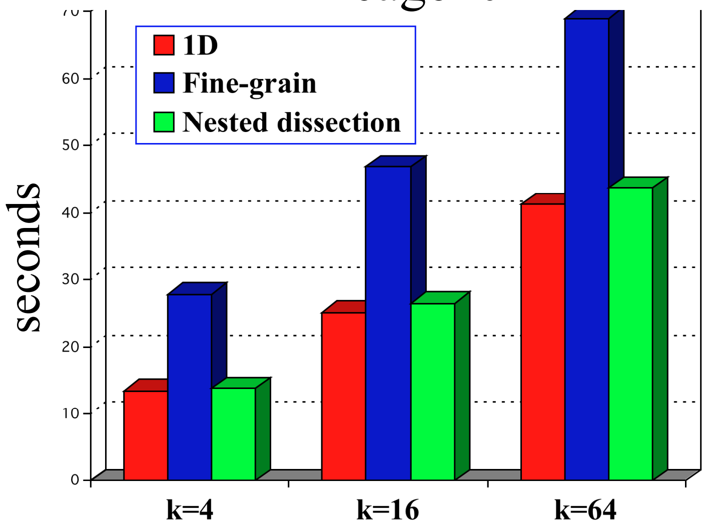
bcsstk30



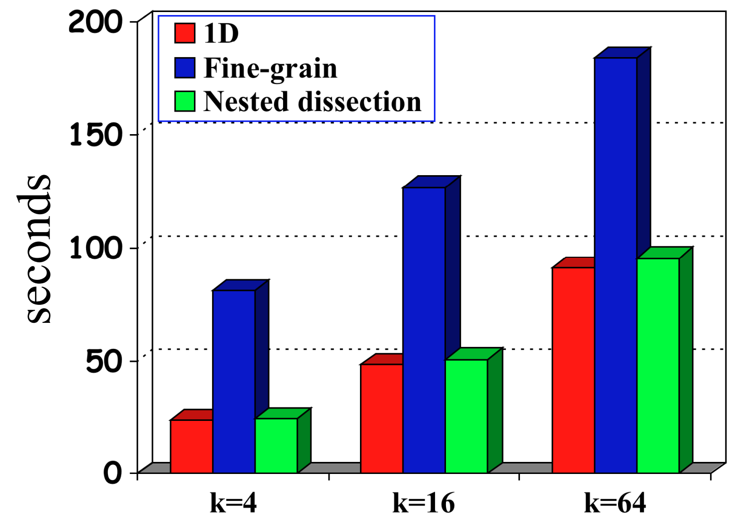
Test matrices from Rob Bisseling (Utrecht)

Runtimes of Partitioning Methods

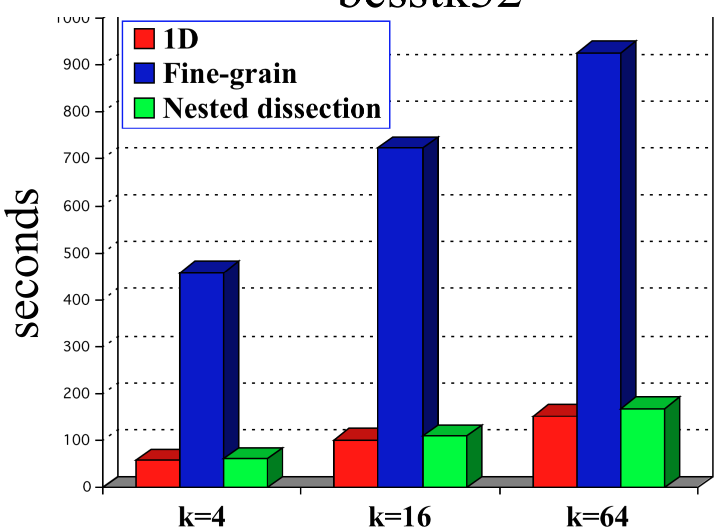
cage10



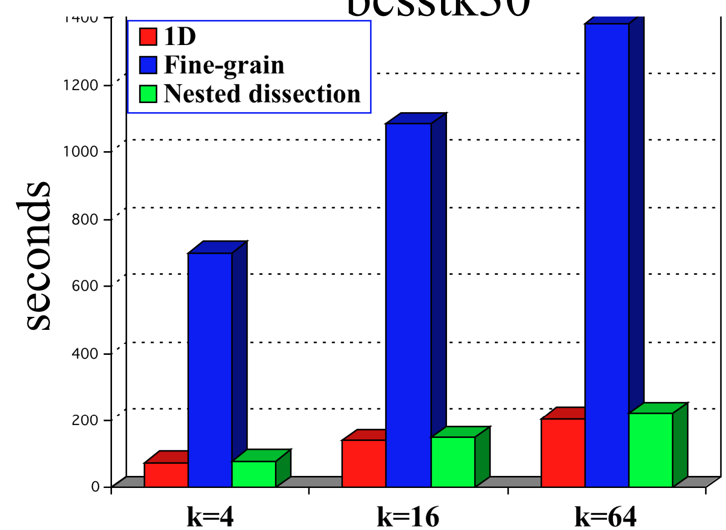
finan512



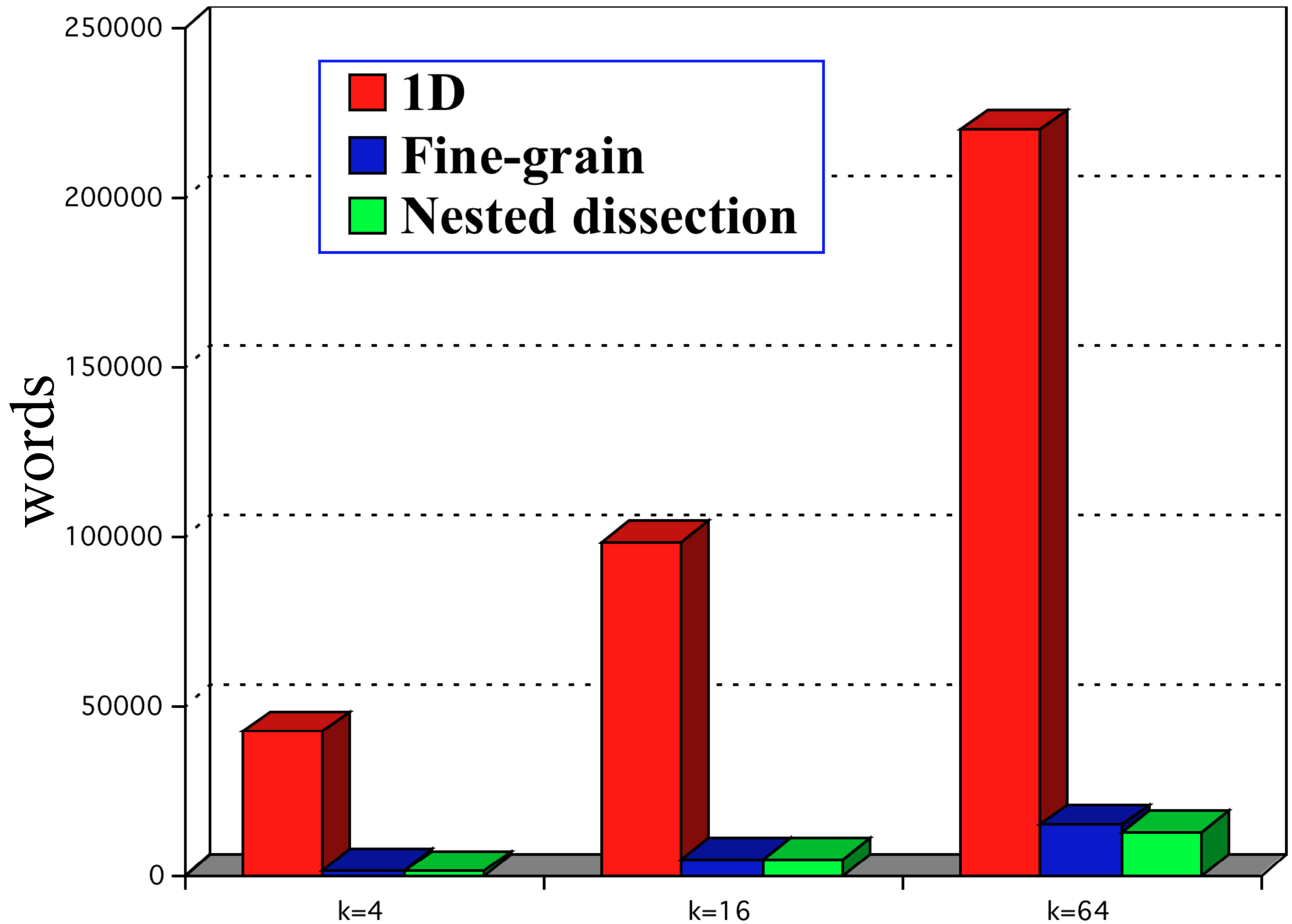
bcsstk32



bcsstk30

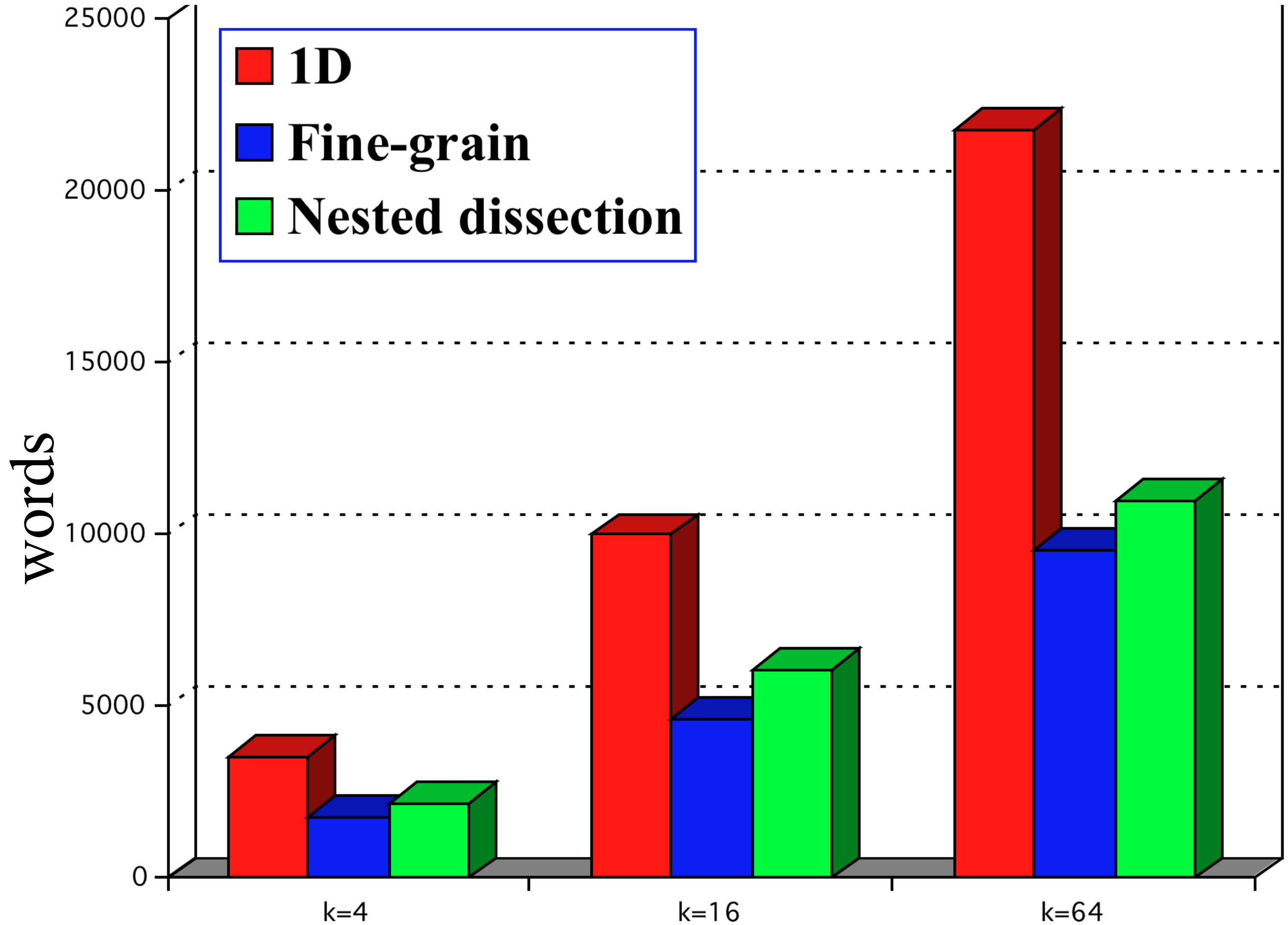


Communication Volume: 1D is Inadequate



- c-73: nonlinear optimization (Schenk)
  - UF sparse matrix collection
  - n=169,422   nnz=1,279,274

Communication Volume: 1D is Inadequate



- asic680ks: Xyce circuit simulation (Sandia)
  - n=682,712 nnz=2,329,176

Parallel Sparse Matrix Partitioning Software

- Developing HPC software for sparse matrix partitioning
  - 1D
  - 2D
- Idea is to implement sparse matrix partitioning algorithms in parallel
  - Efficient/fast
  - Simple to use
- Leverage existing software
  - Graph/hypergraph partitioners
  - Linear algebra packages
- Trilinos framework

Trilinos and Isorropia

- Trilinos
  - Framework for solving large-scale scientific problems
  - Focus on packages (independent pieces of software that are combined to solve these problems)
  - Epetra: parallel linear algebra package
- Isorropia
  - Trilinos package for combinatorial scientific computing
  - Partitioning, coloring, ordering algorithms applied to Epetra matrices
  - Utilizes many algorithms in Zoltan
  - "Zoltan for sparse matrices"

Isorropia: Sparse Matrix Partitioning Methods

- **Parallel** partitioning methods
- Currently exist
  - 1D linear/block, cyclic, random (New!)
  - 1D hypergraph
  - 1D graph
  - 2D fine-grain hypergraph (New!)
- Planned
  - 2D linear/block, cyclic, random
  - 2D RCB partitioning (of nonzeros)
  - 2D nested dissection
  - Vector partitioning (for 2D matrix partitioning)

Isorropia: Partitioning Example 1

```
using Isorropia::Epetra::Partitioner;

ParameterList params;
params.set("PARTITIONING_METHOD", "HYPERGRAPH");
params.set("BALANCE_OBJECTIVE","NONZEROS");
params.set("IMBALANCE_TOL","1.03");

// rowmatrix is an Epetra_RowMatrix
Partitioner partitioner(rowmatrix,params,false);
partitioner.partition();
```

- Simple partitioning of rowmatrix
  - 1D row hypergraph partitioning
  - Balancing number of nonzeros
  - Load imbalance tolerance of 1.03

Isorropia: Partitioning Example 2

```
using Isorropia::Epetra::Partitioner2D;

ParameterList params;
params.set("PARTITIONING_METHOD", "HGRAPH2D_FINEGRAIN");
params.set("IMBALANCE_TOL","1.03");

// rowmatrix is an Epetra_RowMatrix
Partitioner2D partitioner(rowmatrix, params, false);
partitioner.partition();
```

- 2D partitioning of rowmatrix
  - 2D fine-grain hypergraph partitioning
  - Balancing number of nonzeros (implicit)
  - Load imbalance tolerance of 1.03

Isorropia: Redistributing Matrix Data

```
partitioner->partition();

// Set up Redistributor based on partition
Isorropia::Epetra::Redistributor rd(partitioner);

// Redistribute data
newmatrix = rd.redistribute(*rowmatrix, true);
```

- After partitioning matrix
  - Build Redistributor from new partition
  - Redistribute data based on new partition
  - Obtain new matrix

Isorropia: Redistributing Matrix Data

```
using Isorropia::Epetra::createBalancedCopy;

ParameterList params;
params.set("IMBALANCE_TOL","1.03");
params.set("BALANCE_OBJECTIVE","NONZEROS");
params.set("PARTITIONING_METHOD", "HYPERGRAPH");

// crsmatrix and newmatrix are Epetra_CrsMatrix
newmatrix = createBalancedCopy(*crsmatrix, params);
```

- Shortcut
  - Combines partitioning/redistibution of data

Isorropia: Preliminary results

- Isorropia and Epetra can be used to study matrix partitioning
  - Easy to experiment with different matrix partitionings
  - Can see impact of partitionings on different Epetra parallel linear algebra kernels
- Numerical experiments
  - Runtime of SpMV for different matrix partitionings
  - 3 different methods: 1D linear, 1D hypergraph, 2D fine-grain
  - **Parallel** implementations of partitioning methods
  - Test problems: bcsstk30, bcsstk32, c-73, asic680ks

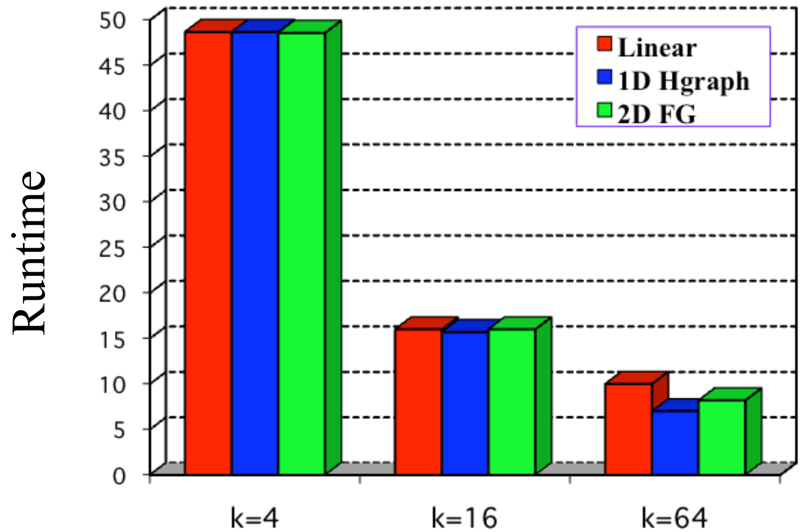
Isorropia: Preliminary results

NERSC Franklin

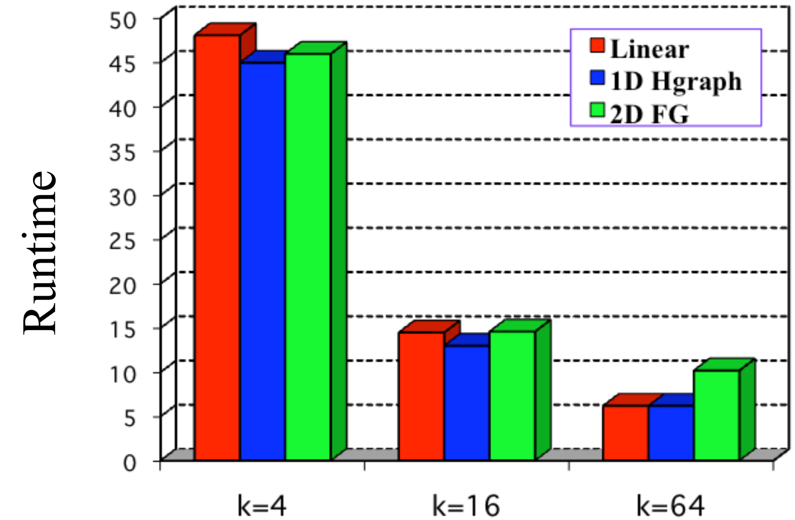
- Platforms
  - NERSC Franklin (Cray XT4, Opteron 2.3 GHz quad core)
  - SNL Odin cluster (dual 2.2GHz Opteron, Myrinet)

Isorropia: SpMV Timings (Franklin)

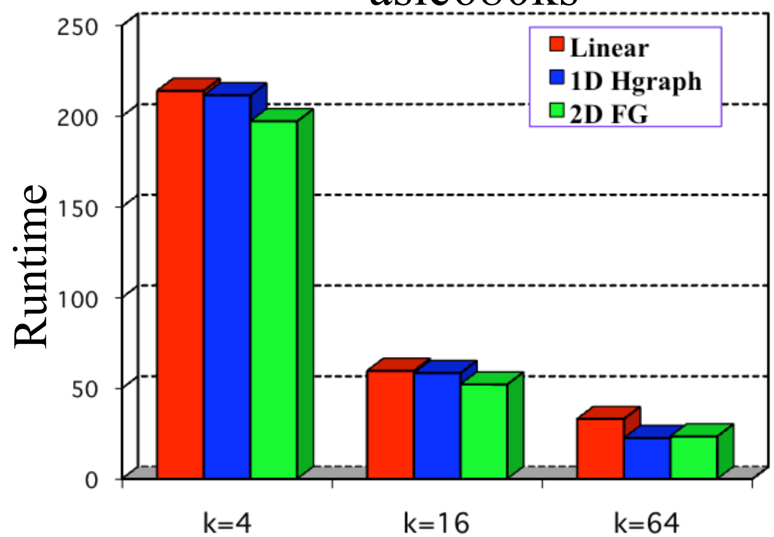
bcsstk32



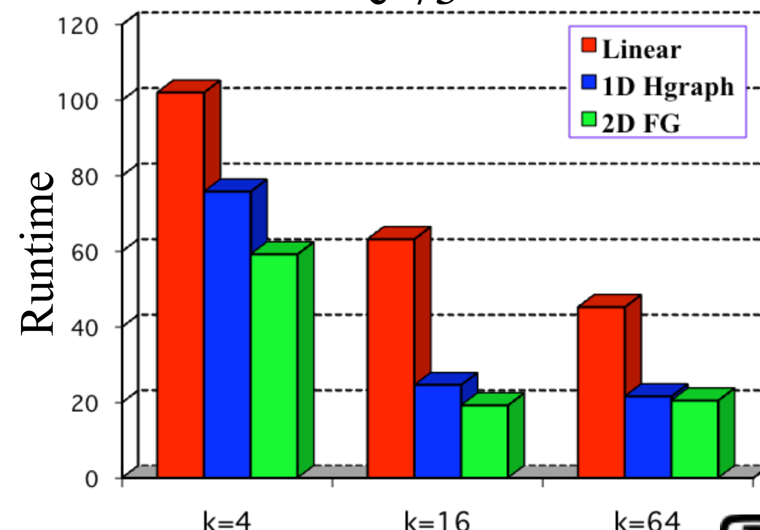
bcsstk30



asic680ks

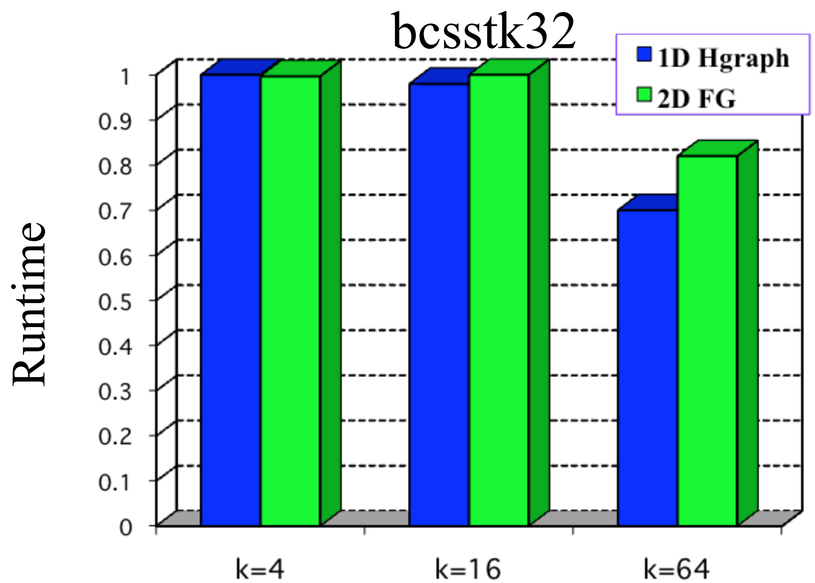


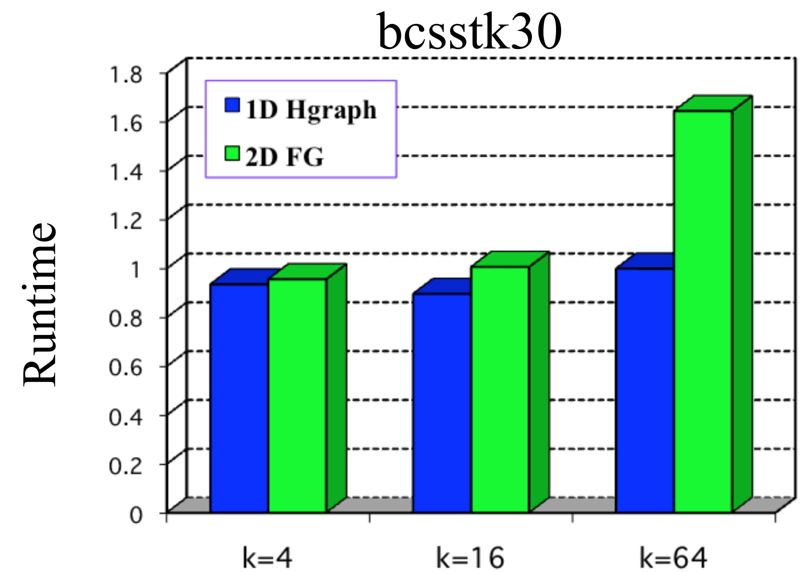
c-73

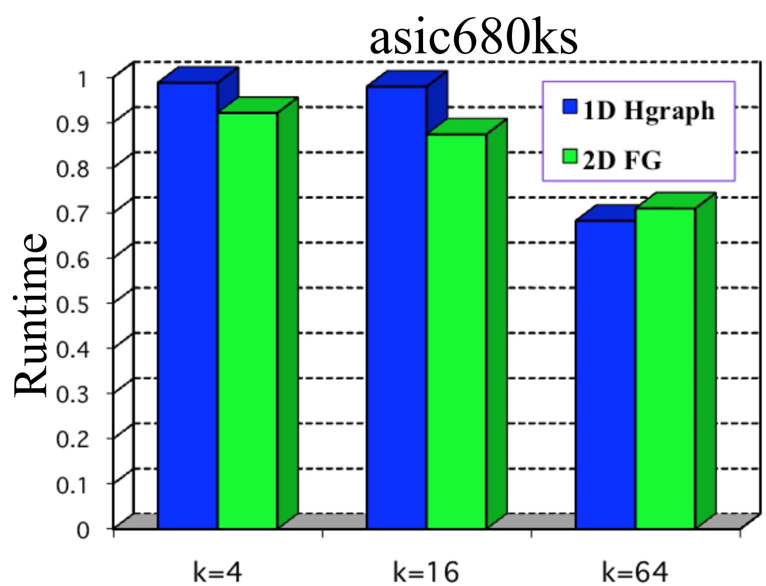


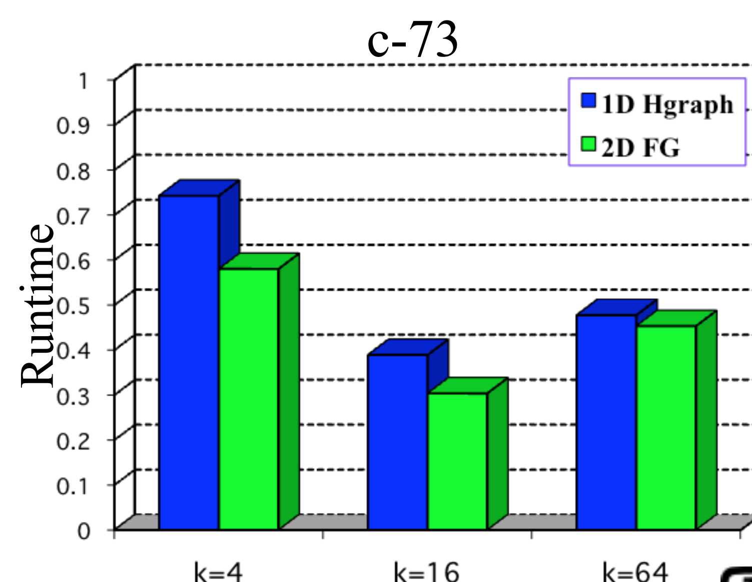
Sandia National Laboratories

SpMV Timings (Franklin, normalized)



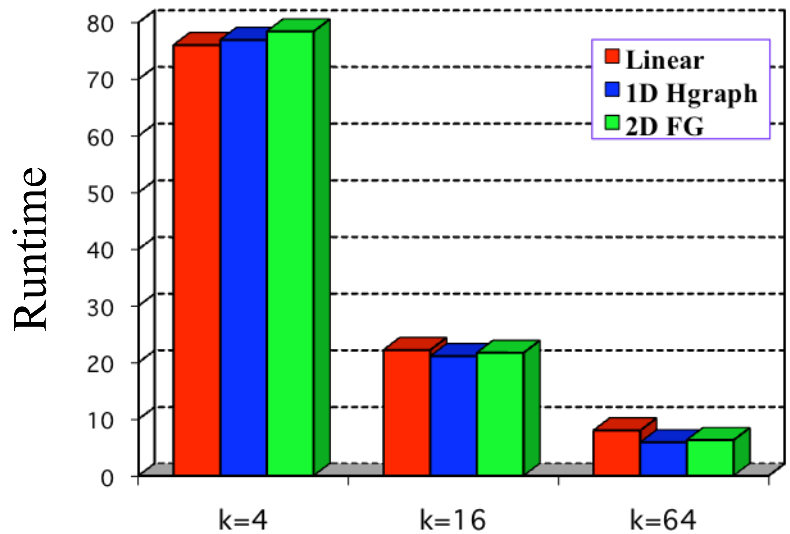




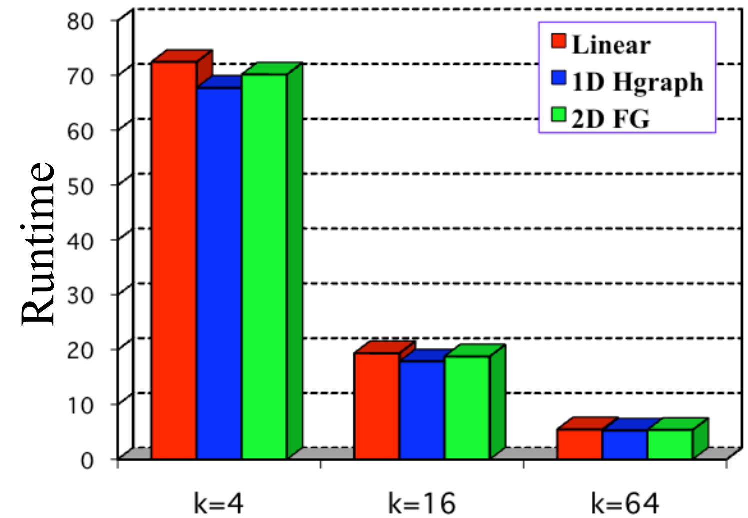


Isorropia: SpMV Timings (Odin)

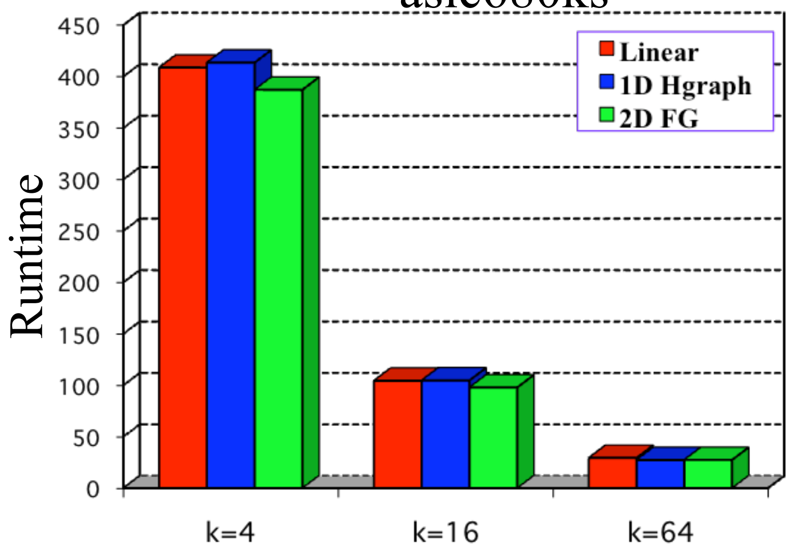
bcsstk32



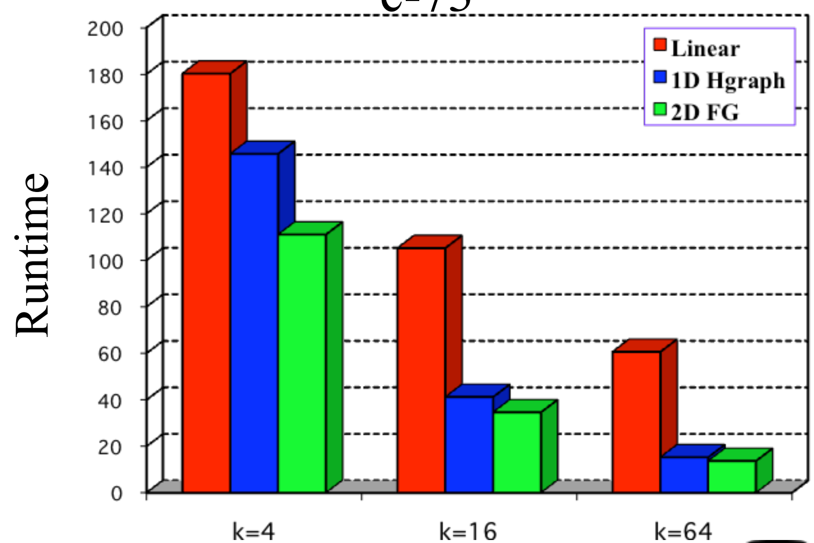
bcsstk30



asic680ks



c-73



Summary

- Motivation for and overview of 2D partitioning
- New 2D matrix partitioning algorithm
- ND matrix partitioning algorithm
  - ND used in new context
  - Good trade off between communication volume and partitioning time
    - Communication volume (comparable to fine-grain)
    - Partitioning time (comparable to 1D)
- Presented simple framework for sparse matrix partitioning for Trilinos/Epetra applications
  - First production code that supports parallel 2D sparse matrix partitioning

Summary of Isorropia Work

- Mixed results for SpMV runtimes
  - Decrease not proportional to decrease in communication volume
  - Results for bcsstk30 and bcsstk32 not significantly better than linear
    - 2D FG worse than 1D hypergraph
  - Improvement over linear for asic680k and c-73
    - 2D FG significantly better than 1D hypergraph for some k
- 2D partitioning can be effective for some matrices
- Improvements needed to make 2D methods viable
  - Room for improvement (e.g., PHG for FG)
    - 2D fine-grain partitioning in next Trilinos release

Selection of Related Papers/Info

2D Partitioning:

U. Catalyurek and C. Aykanat, "A fine-grain hypergraph model for 2d decomposition of sparse matrices," In *Proc. IPDPS 8th Int'l Workshop on Solving Irregularly Structured Problems in Parallel* (Irregular 2001), April 2001.

U. Catalyurek, C. Aykanat, and B. Ucar. On two-dimensional sparse matrix partitioning: Models, methods, and a recipe. To appear in *SIAM Journal on Scientific Computing*.

B. Vastenhouw and R. H. Bisseling. A two-dimensional data distribution method for parallel sparse matrix-vector multiplication. *SIAM Review*, 47(1):67–95, 2005.

Nested Dissection Partitioning:

E.G. Boman and M.M. Wolf, "A Nested Dissection Approach to Sparse Matrix Partitioning for Parallel Computations," SANDIA Technical Report 2008-5482J. (Submitted for publication)

M. Wolf, E. Boman, and C. Chevalier, "Improved Parallel Data Partitioning by Nested Dissection with Applications to Information Retrieval," SANDIA Technical Report 2008-7908J.

Trilinos/Isorropia:

http://trilinos.sandia.gov

http://trilinos.sandia.gov/packages/isorropia/