Exceptional service in the national interest

Sandia National Laboratories

Scalable Matrix Computations on Large Scale-Free Graphs using 2D Graph Partitioning

Erik Boman, Karen Devine, Sivasankaran Rajamanickam
Sandia National Laboratories

Sparse Days, CERFACS, June 17, 2013

U.S. DEPARTMENT OF ENERGY
NNSA National Nuclear Security Administration

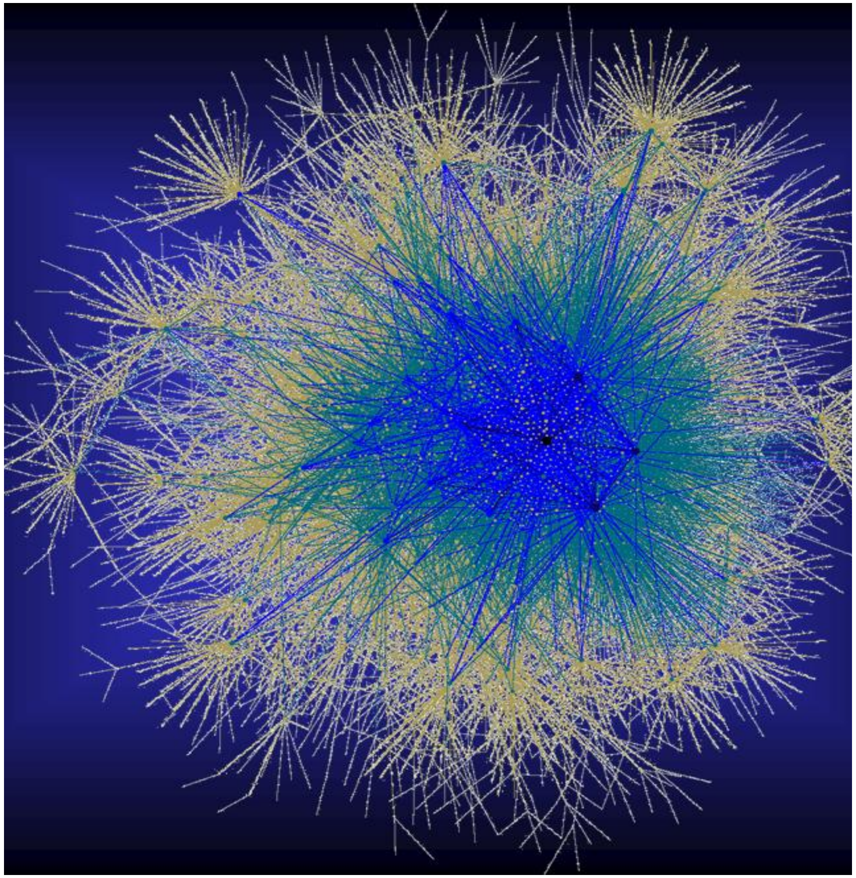
Sandia National Laboratories is a multi-program laboratory managed and operated by Sandia Corporation, a wholly owned subsidiary of Lockheed Martin Corporation, for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-AC04-94AL85000.

Overview

- We are interested in matrix computations to analyze large graphs on distributed-memory supercomputers
  - In particular, eigensolvers
  - Our focus is on SpMV, a kernel in iterative methods
- We present results of various data distribution strategies for distributed-memory computing on scale-free graphs.
  - 1D vs 2D matrix layout
  - Use of graph and hypergraph partitioners
- We present a new method combining (hyper)graph partitions with 2D distributions, and show its benefit for scale-free graphs.

Background

- Large graphs are pervasive
  - WWW, social networks
- Often scale-free
  - Power-law degree distr.
  - Small diameter
- Very different from PDE discretizations
  - Need to adapt scientific computing methods and tools?



BGP graph (credit: Ross Richardson, Fan Chung)
http://math.ucsd.edu/~fan/graphs/gallery

Matrix Computations: SpMV is key

- Linear algebra is a useful analysis tool for graphs
  - Eigen-analysis using extreme eigenpairs
  - SpMV is core kernel in iterative methods
- Sparse matvec (SpMV) is bottleneck for scale-free graphs on large distributed-memory computers
  - High-degree vertices cause lots of communication
  - Some processors need to communicate with almost all other!

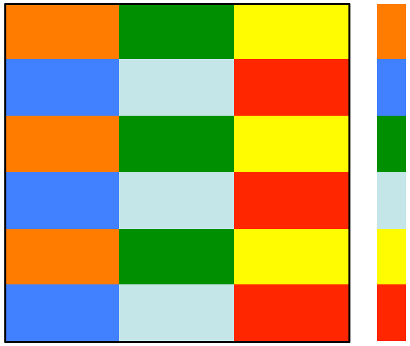
Partitioning

- Graph partitioning generally reduces communication for SpMV
  - Hypergraph model exactly models communication volume (Catalyurek & Aykanat, 2000)
- Graph partitioners are widely regarded as ineffective on scale-free graphs
  - Software tools (e.g., Metis, Scotch, Zoltan) were designed for meshes and PDE discretizations
    - Not optimized for scale-free graphs
  - Focus on communication volume
    - We wish to reduce both #messages and communication volume
- Partitioning strategy depends on type of distribution
  - 1D (row-based) distribution is most common

1D and 2D Matrix Distributions

- 1D matrix distribution:
  - Entire rows (or columns) of matrix assigned to a processor
  - Same mapping used for vectors
  - Default distribution in Trilinos
- 2D matrix distribution:
  - Block-based Cartesian layout
  - Long used in parallel dense solvers (ScaLapack)
  - Also works for sparse matrices (Hendrickson et al. ‘95, Bisseling ‘04)
  - Yoo et al. (SC’11) demonstrated benefit over 1D layouts for eigensolves on scale-free graphs

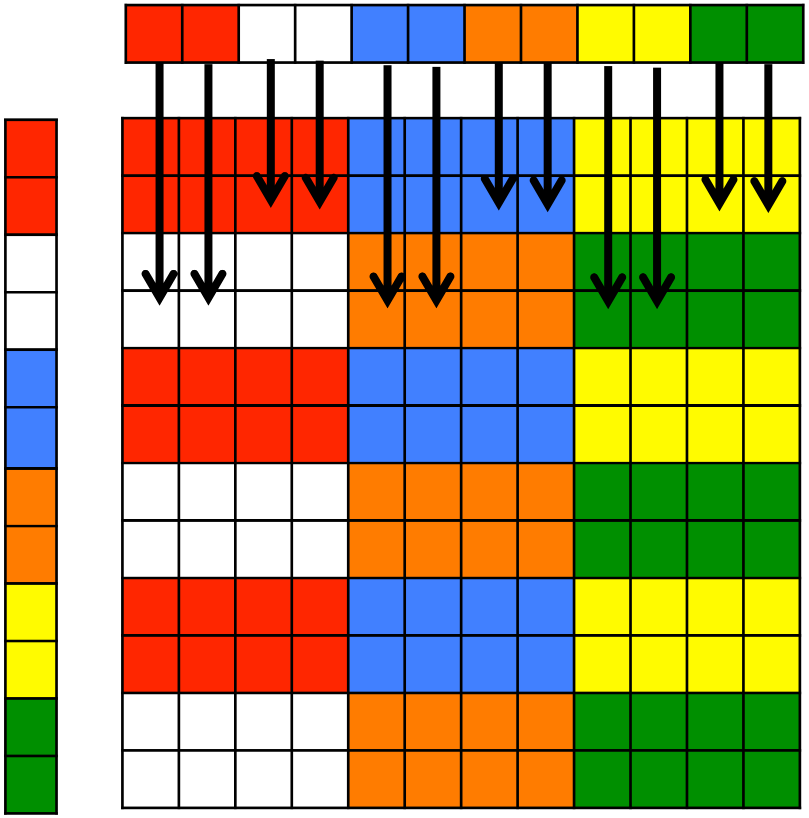
1D row-wise matrix distribution; 6 processes



2D matrix distribution; 6 processes

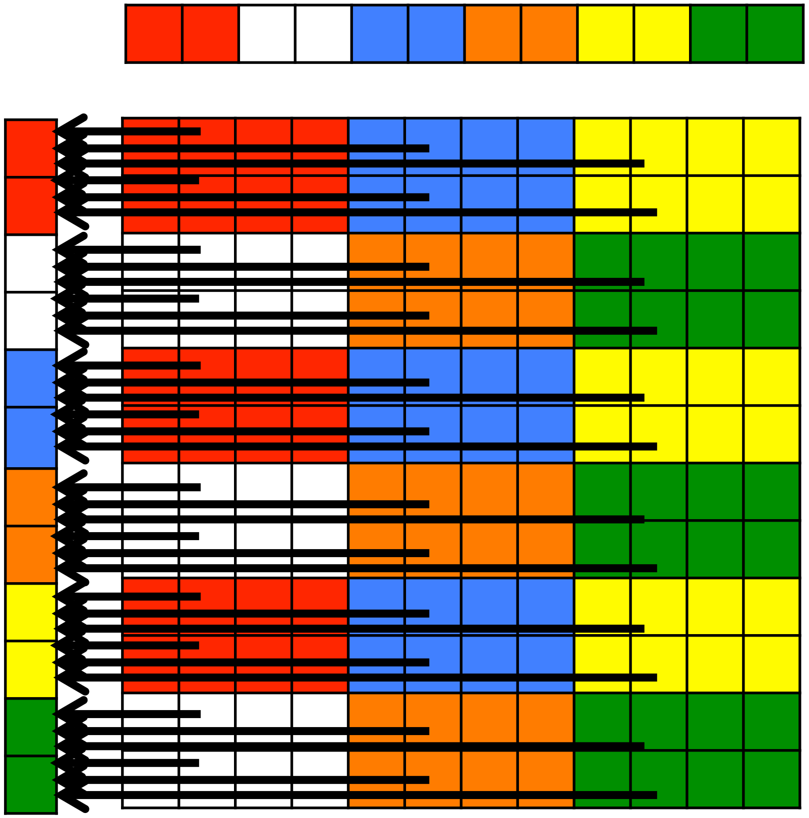
Benefit of 2D Matrix Distribution

- During matrix-vector multiplication, communication occurs only along rows or columns of processors.
  - Expand (vertical):
    Vector entries x_j sent to column processors to compute local product $y^p = A^p x$
  - Fold (horizontal):
    Local products y^p summed along row processors; $y = \Sigma y^p$
- In 1D, fold is not needed, but expand may be all-to-all.



Benefit of 2D Matrix Distribution

- During matrix-vector multiplication, communication occurs only along rows or columns of processors.
  - Expand (vertical): Vector entries x_j sent to column processors to compute local product $y^p = A^p x$
  - Fold (horizontal): Local products y^p summed along row processors; $y = \Sigma y^p$
- In 1D, fold is not needed, but expand may be all-to-all.



Trilinos Computational Science Toolkit

- Heroux et al., Sandia

- Trilinos Capabilities:
  - Scalable Linear & Eigen Solvers
  - Discretizations, Meshes & Load Balancing
  - Nonlinear, Transient & Optimization Solvers
  - Software Engineering Technologies & Integration

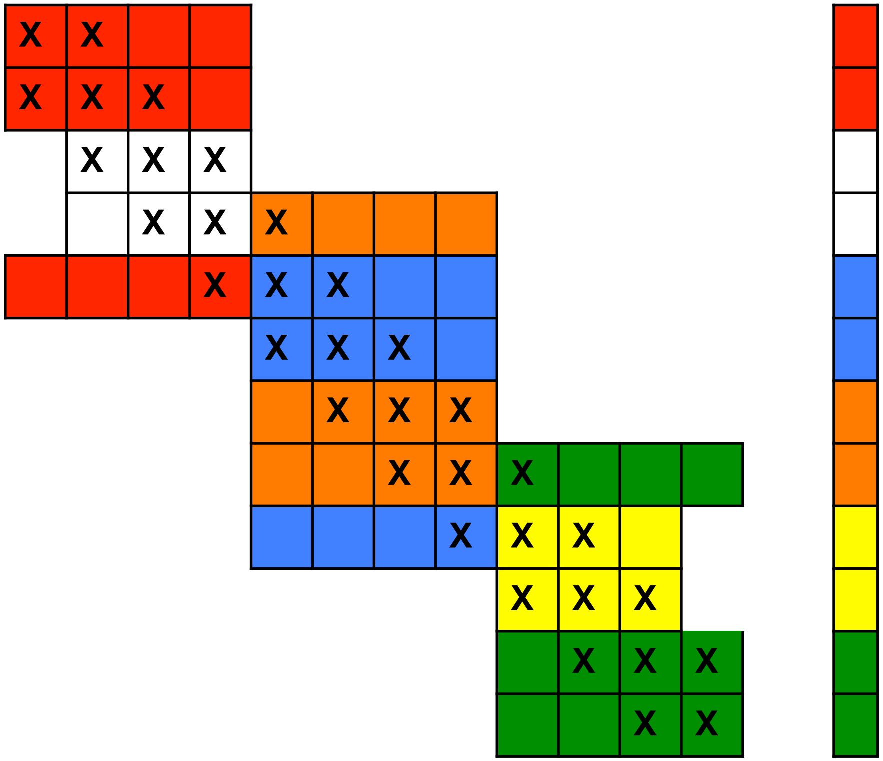
- Trilinos features:
  - Block-based data structures and algorithms
    - Block-based linear and eigen solvers use "multivector" data structures.
  - Toolkit/package-based design
    - Packages can be combined, but not all of Trilinos is needed to get work done.

- In this project, we use Trilinos'…
  - Distributed Matrix/Vector classes *Epetra* and *Epetra64*
  - Eigensolver package *Anasazi*
  - Linear solver package *Belos*
  - Preconditioning package *Ifpack*
  - Utilities package *Teuchos* (e.g., communicators, parameters, ref-counted pointers)

Trilinos Maps

- Maps describe the distribution of global IDs for rows/columns/vector entries to processors.
- Four maps needed in most general case:
  - Row map for matrix
  - Column map for matrix
  - Range map for vector
  - Domain map for vector
- Part of *Epetra* package

Rank 3 (Blue)
Row Map = {4, 5, 8}
Column Map = {4, 5, 6, 7}
Range/Domain Map = {4, 5}



1D vs 2D Strong Scaling Experiments

- Compare times for matrix-vector multiplication with 1D and 2D distributions
- Hera cluster at LLNL (AMD quad-core, quad-socket Opteron processors operating at 2.2/2.3 GHz)
- Matrices from the University of Florida matrix collection
- Symmetrized and largest connected component extracted

Name	Description	Number of Rows	Number of Nonzeros
Hollywood-2009	Hollywood movie actor network (Boldi, Rosa, Santini, Vigna)	1.1M	113M
Wikipedia-20070206	Links between wikipedia pages (Gleich)	3.5M	85M
Ljournal-2008	LiveJournal social network (Boldi, Rosa, Santini, Vigna)	5.6M	99M
Wb-edu	Links between *.edu webpages (Gleich)	8.9M	88M
Cit-Patents	Citation network among US patents (Hall, Jaffe, Trajtenberg)	3.8M	33M

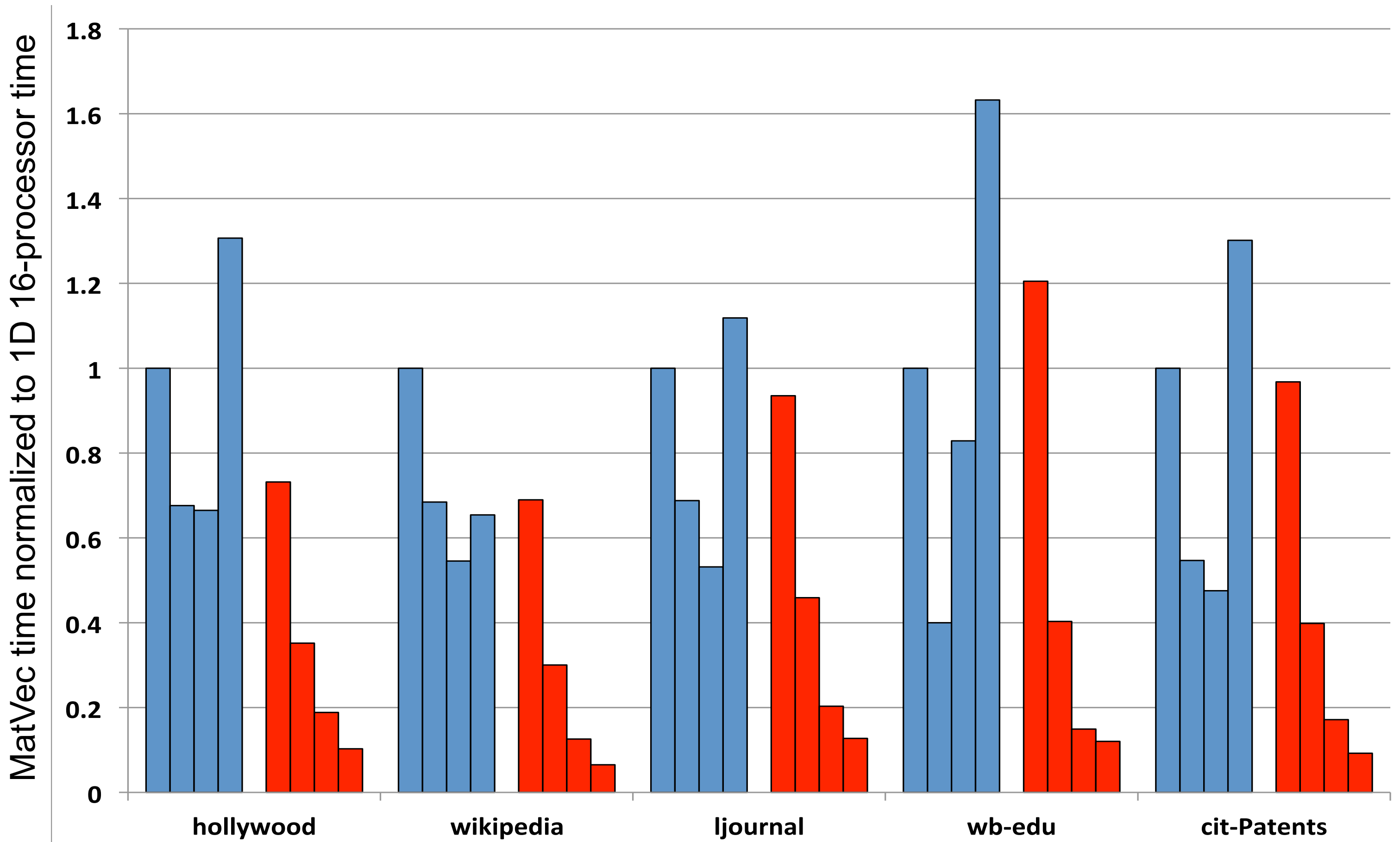
1D vs 2D Strong Scaling experiments

For each matrix:

Blue = Trilinos 1D Matrix Distribution on 16, 64, 256, 1024 processors (left to right)

Red = Trilinos 2D Matrix Distribution on 16, 64, 256, 1024 processors (left to right)

Times are normalized to the 1D 16-processor runtime for each matrix.



Randomization

- On input, randomly permute matrix rows/columns
  - Eliminates any inherent structure in input file (e.g., high degree nodes first)
  - Gives better balance in number of nonzeros per processor for 1D and 2D
  - But can drastically increase communication volume

liveJournal matrix (4M rows; 73M nonzeros) on 1024 processes				
Method	Imbalance in nonzeros (Max/Avg per proc)	Max # Messages per SpMV	Comm. Vol. per SpMV (doubles)	100 SpMV time (secs)
1D-Block	12.8	1023	34.5M	2.14
1D-Random	1.3	1023	55.3M	1.52
2D-Block	11.4	62	43.4M	0.95
2D-Random	1.0	62	64.2M	0.43

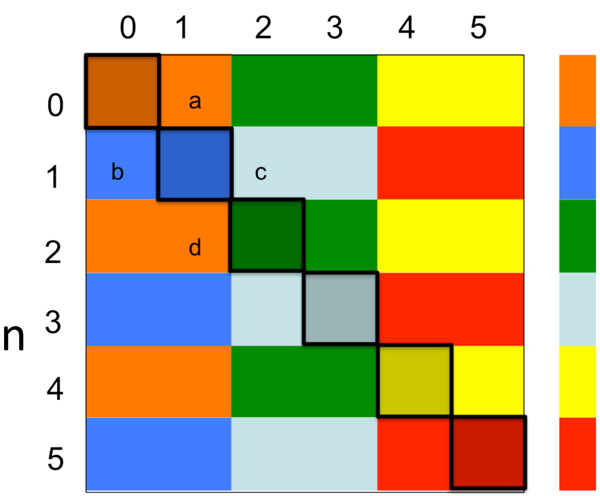
Advanced 2D Partitioning Methods

The Cartesian 2D block distributions are simple to compute but ignore the structure of the graph. Can we do better?

- Coarse-grain hypergraph (Catalyurek & Aykanat '01)
  - Cartesian product, but expensive to compute
  - Requires multiconstraint hypergraph partitioning
- Fine-grain hypergraph (Catalurek & Aykanat '01)
  - Assign each nonzero separately, not Cartesian
  - Much larger hypergraph, impractical for big problems
- Mondriaan (Vastenhouw & Bisseling '05)
  - Recursive hypergraph partitioning
  - Only serial software available

New idea: Graph Partitioning + 2D

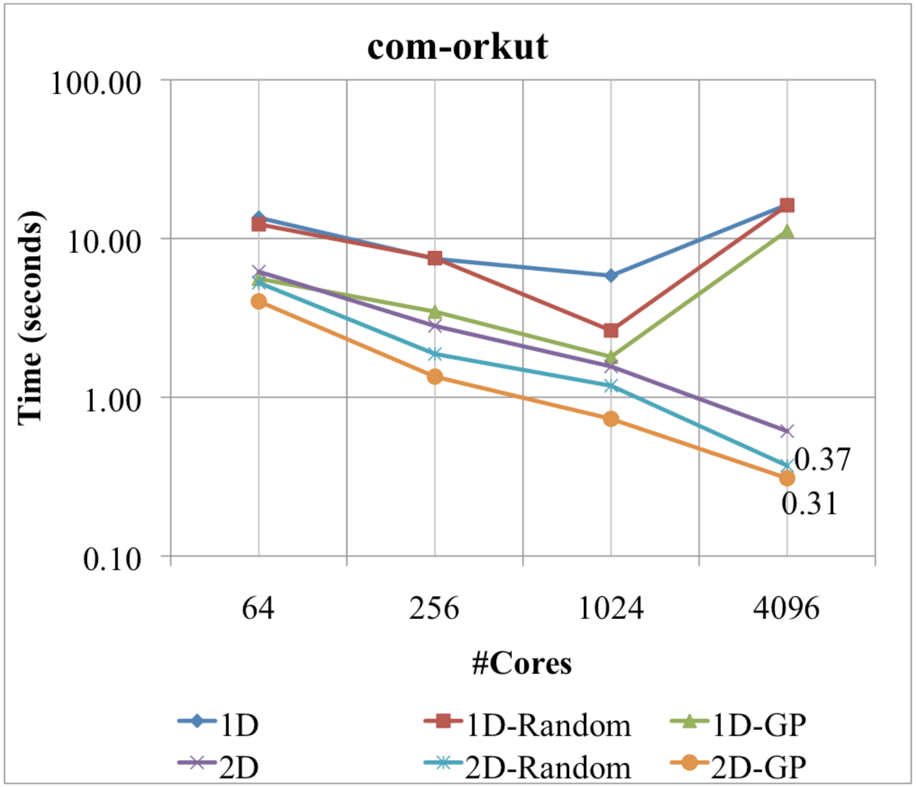
- Cartesian 2D block distributions limit #messages but ignore structure of the graph.
- (Hyper)Graph partitioning (e.g., Zoltan, ParMETIS, Scotch) balances work (nonzeros per process) while attempting to minimize total communication volume.
  - Thought to be ineffective on scale-free graphs
- Our idea: Apply (hyper)graph partitioning and 2D distribution together
  - Compute vertex-based partition of graph using ParMETIS or Zoltan
  - Apply 2D distribution to a logical permutation based on the (hyper)graph partition
- Advantages:
  - Balance the number of nonzeros per process
  - Exploit structure in the graph to reduce communication volume
  - Reduce the number of messages via 2D distribution



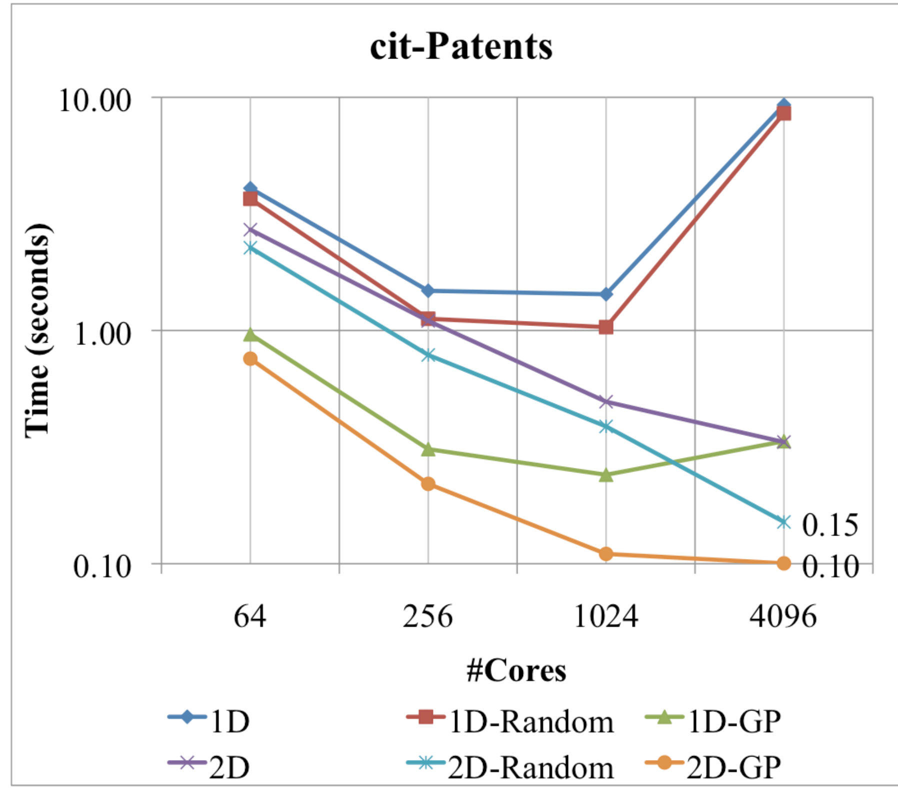
2D-GP: Graph partitioning with 2D Distribution

liveJournal matrix (4M rows; 73M nonzeros) on 1024 processes				
Method	Imbalance in nonzeros (Max/Avg per proc)	Max # Messages per SpMV	Comm. Vol. per SpMV (doubles)	100 SpMV time (secs)
1D-Block	12.8	1023	34.5M	2.14
1D-Random	1.3	1023	55.3M	1.52
1D-GP	1.2	1011	18.9M	0.53
2D-Block	11.4	62	43.4M	0.95
2D-Random	1.0	62	64.2M	0.43
2D-GP	1.4	62	22.4M	0.22

Strong scaling



Orkut social network
3.1M rows; 237M nonzeros
Max nonzeros/row = 33K

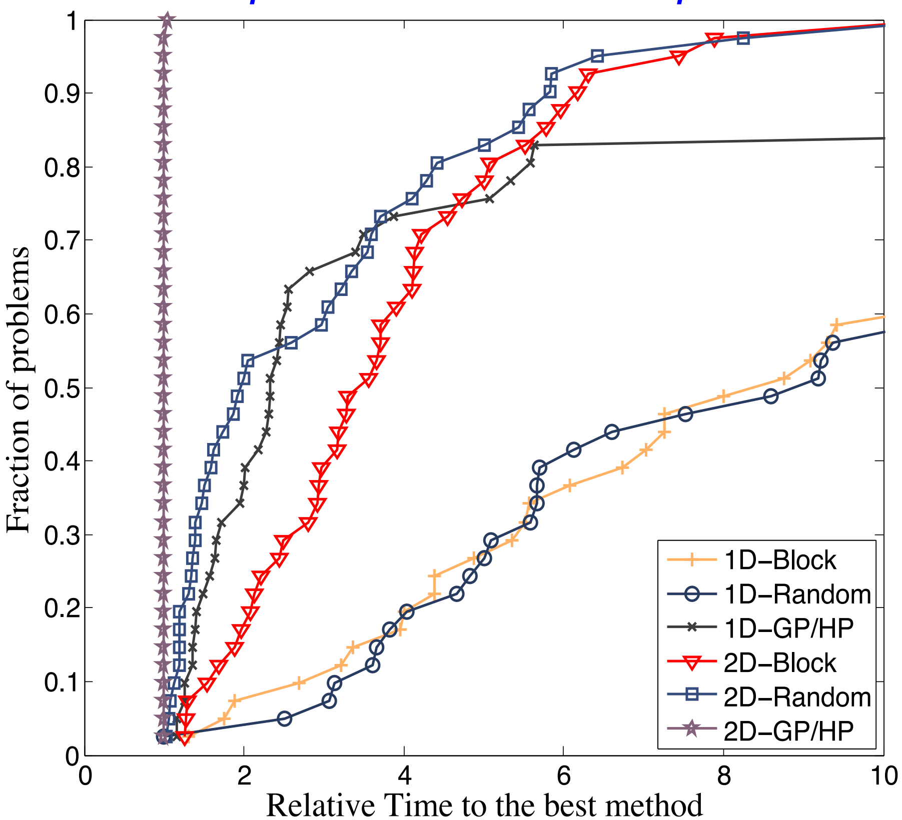


Patent citations network
3.8M rows; 37M nonzeros
Max nonzeros/row = 1K

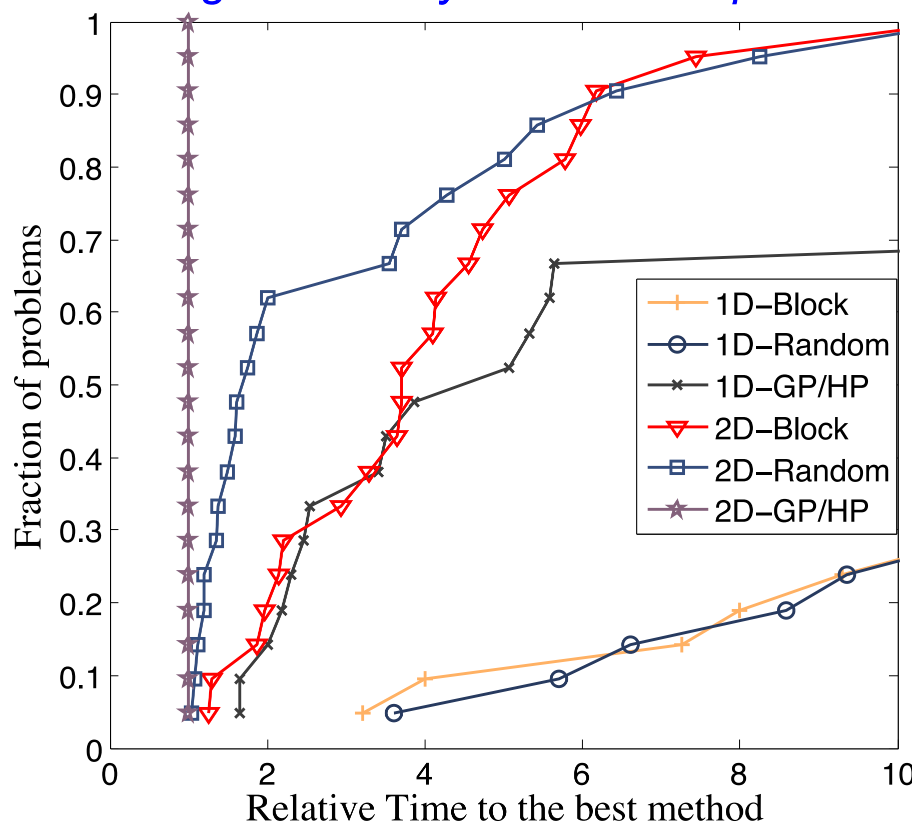
Performance comparisons

- 10 matrices: 1.1M - 67.5M rows; 36M-1.6B nonzeros
- 2D-GP/HP best in all but one experiment
- Benefit even greater for large numbers of processes

All experiments: 64-4096 procs

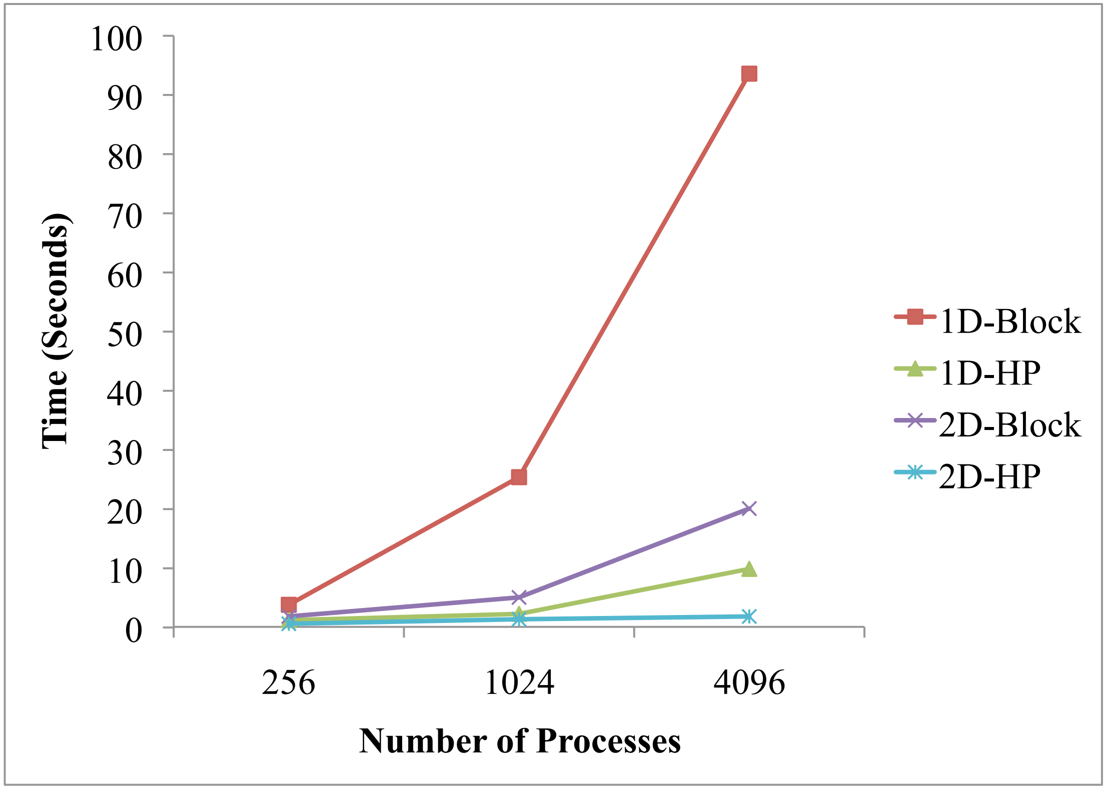


Large runs only: 1024-4096 procs



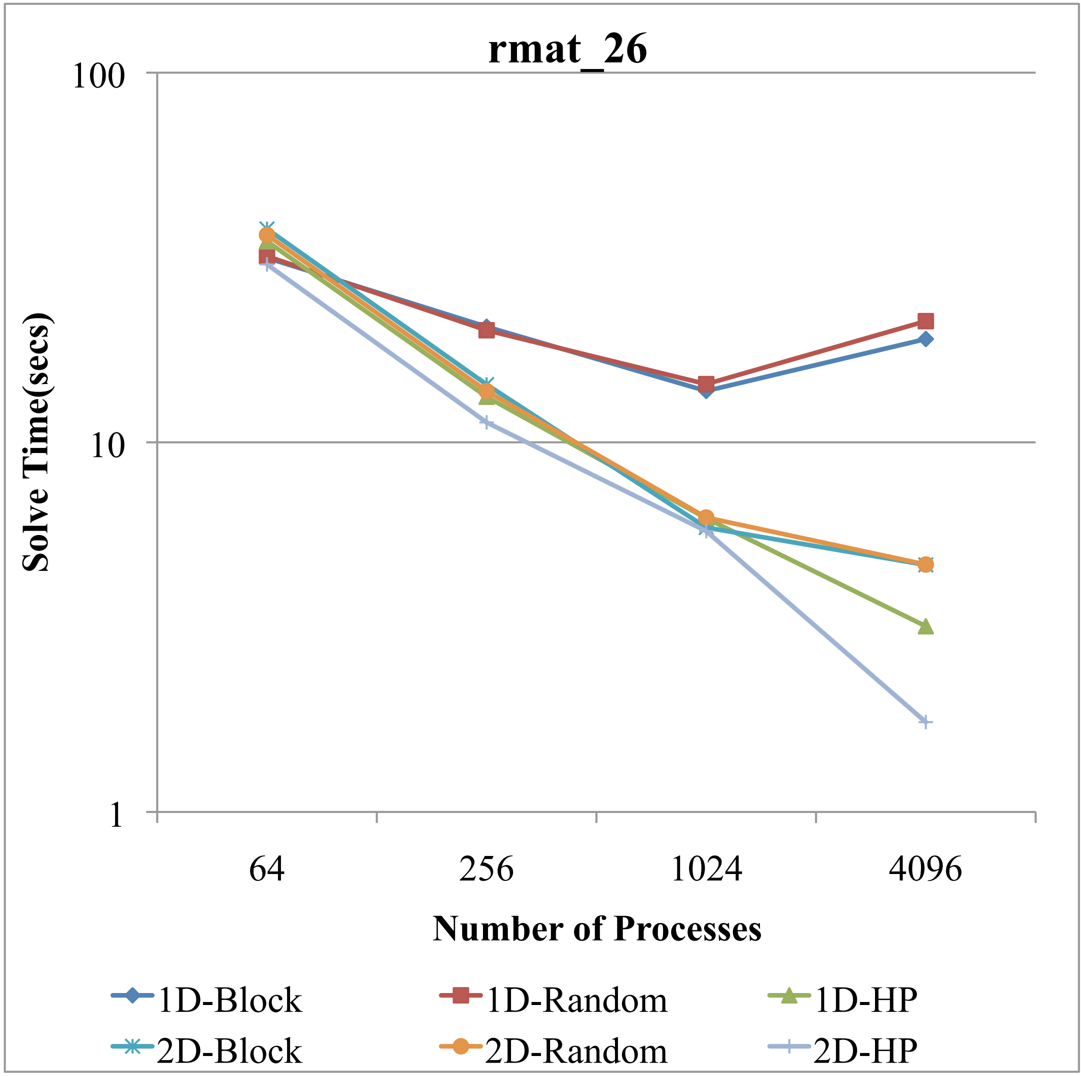
Weak Scaling

- R-MAT matrices (Chakrabarti et al., 2004) with Graph-500 parameters (a=0.57; b=c=0.19; d=0.05)
  - rmat_22 on 256 procs
    - 4.2M vertices
    - 38M edges
  - rmat_24 on 1024 procs
    - 16.8M vertices
    - 151M edges
  - rmat_26 on 4096 procs
    - 67.1M vertices
    - 604M edges
- 2D-HP maintains best weak scaling.



Eigensolver Experiments

- Anasazi Toolkit in Trilinos
  - Baker, Hetmaniuk, Lehoucq, Thornquist; ACM TOMS 2009
  - Block-based eigensolvers: Solve AX = XΛ or AX = BXΛ
- Experiment:
  - Find 10 largest eigenvalues of Laplacian using Block Krylov-Schur (BKS) solver
  - rmat_26 matrix: 67.1M rows; 604M nonzeros
  - HP = Hypergraph partitioning in Zoltan



Conclusions

- 2D distribution has clear benefit for scale-free graphs, especially at high process counts.
  - Reduces max number of messages per process
- Randomization can be effective to restore load balance.
  - But can increase communication volume
- (Hyper)graph partitioning can maintain load balance while keeping communication volume low.
  - More effective for scale-free graphs than thought
- Combining 2D distribution with (hyper)graph partitioning gives best results.
  - Low number of messages, low communication volume, low imbalance
  - Allows reuse of existing partitioning software

Extra Slides

Distributions for Anasazi

- Matrix-vector multiplication an important kernel
  - 55-87% of solve time for hollywood-2009 matrix with block 2D distribution on 64-4096 processes
- Other operations contribute to solve time
  - Remaining time primarily in orthogonalization
  - Balance with respect to vector entries, not matrix entries
- Benefit in balancing BOTH matrix nonzeros and vector entries
  - Randomization can achieve this balance, but increases communication volume drastically.
  - Multiconstraint graph partitioning can be used to achieve balance while keeping communication volume low.
    - Two weights per vertex: [1, number of nonzeros per row]
    - Find one partition that balances both weights.

Example: Eigensolve with multiconstraint graph partitioning

Find 10 largest eigenvalues of hollywood-2009 matrix (1.1M rows; 114M nz) using Anasazi's BKS (0.001 tolerance) on 1024 processes					
Method	Nonzero Imbalance (max/avg)	Vector imbalance (max/avg)	Total Comm Volume for one SpMV (doubles)	SpMV time (secs)	Total Solve time (secs)
2D-Block	26.0	1.0	15.7M	0.93	1.15
2D-Random	1.1	1.0	35.6M	0.44	0.62
2D-GP	1.6	30.3	17.2M	0.33	0.96
2D-GP-MC Multiconstraint	1.6	1.1	17.5M	0.27	0.44

Scaling in Anasazi

- Use Anasazi's Block Krylov Schur method to find ten largest eigenvalues of the normalized Laplacian matrix (tol=0.001)