SAND2015-2983C

Exceptional service in the national interest

Sandia National Laboratories

Distributing Linear Systems for Parallel Computation

Karen Devine
with Erik Boman and Sivasankaran Rajamanickam
Sandia National Laboratories

AWM Research Symposium, April 2015

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.

Leveraging investments in computing

Sandia National Laboratories

- Decades of research in parallel, high-performance computing for scientific applications in the national laboratories
  - Driven by stockpile stewardship mission: ensure reliability and safety of nuclear weapons without nuclear testing
- Large investments in infrastructure and people
  - Hardware design, purchases, operations
  - Operating systems, file systems, runtime systems
  - Libraries of linear/nonlinear/eigensolvers, optimization algorithms, uncertainty quantification methods, partitioners
- Relatively straightforward transfer to other PDE-based applications
  - Climate simulation, reactor design and safety, carbon sequestration, nanotechnology, etc.
- New exciting research areas: e.g., "big data" analysis, cybersecurity
- Can we leverage our scientific computing investment to address broader range of application areas?

Case study:
Partitioning for Parallel Computing

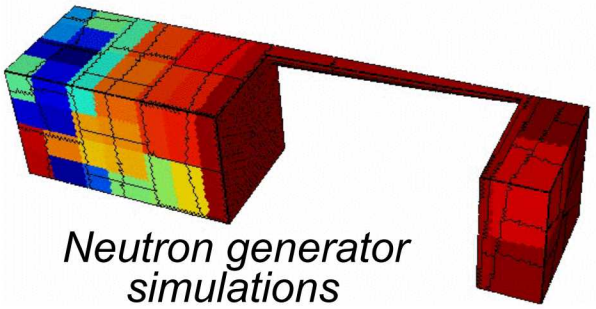
Sandia National Laboratories

- First step in parallel computing: distributing work among the processors
- Partitioning: Divide work so that total execution time is minimized
  - Constraint: Processors have equal amounts of work
    - Processors are not waiting for other processors to finish computation
  - Objective: Interprocessor communication (data movement) is minimized
- Note: this definition differs from that often used in graph-analysis
  - Here, think "load balancing," not "clustering"
- Use matrix distribution for matrix-vector multiplication as model problem
  - Key kernel of many scientific applications (e.g., finite element analysis)
  - Important in graph analysis (e.g., spectral analysis using extreme eigenpairs)
  - Distribute matrix and vector to minimize matrix-vector multiplication time

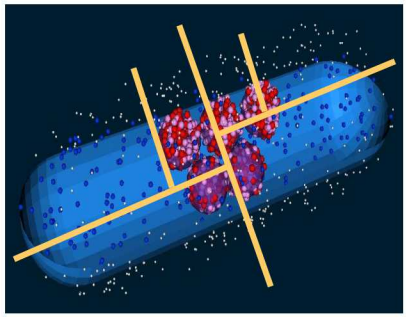
Partitioning software

Sandia National Laboratories

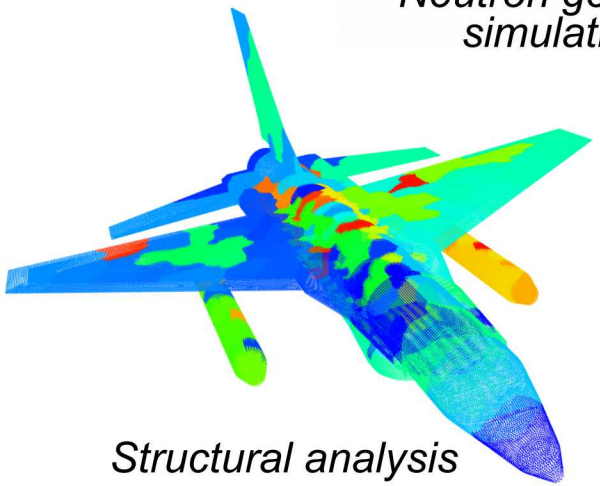
- Many high quality parallel partitioning tools developed for physics-based applications
  - Zoltan toolkit of geometric, graph & hypergraph partitioners (Sandia)
  - ParMETIS (U. Minnesota) & PT-Scotch (U. Bordeaux) graph partitioners



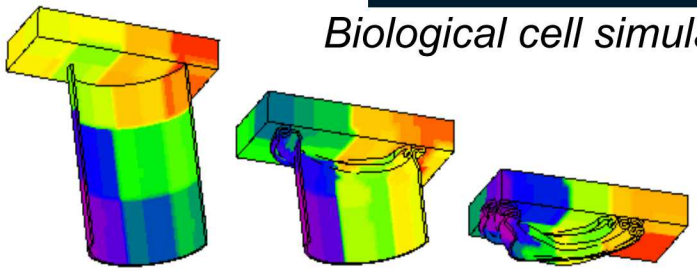
Neutron generator simulations



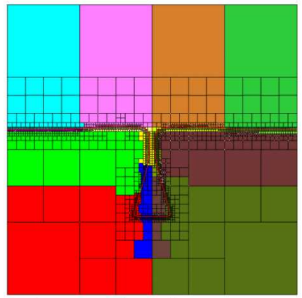
Biological cell simulations



Structural analysis



Contact detection in crash simulations

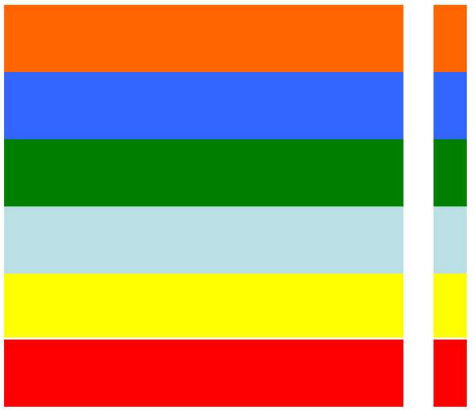


Chemical vapor deposition

Typical matrix partition

Sandia National Laboratories

- 1D Distribution:
  - Entire row (or column) of matrix assigned to a single processor
  - Vector uses same distribution
  - During SpMV, processor receives (via communication) vector entries needed to match non-zeros in owned rows.



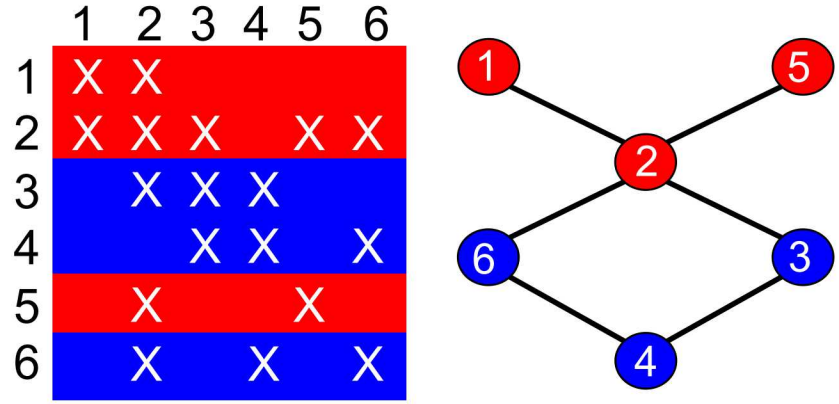
1D row-wise matrix distribution; 6 processes

- 1D-Block distribution of *N* x *N* matrix onto *p* processors:
  - First *N/p* rows given to processor 0
  - Next *N/p* rows given to processor 1
  - And so on…
  - Default in many parallel linear algebra libraries (e.g., Trilinos)

Graph partitioning: 1D-GP

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

- Explicitly attempts to minimize communication costs induced by partition
- Represent matrix *A* as a graph:
  - One vertex *j* per row a_j
  - Edge (*i, j*) exists iff $a_{ij} \neq 0$
  - Vertex weights = # nonzeros in row

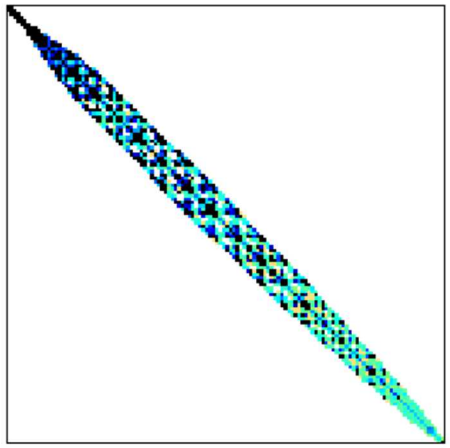


- Goal: Assign equal vertex weight to parts while minimizing weight of edges between parts (i.e., cut by part boundary)
- Highly effective for mesh-based PDE problems
  - Mostly local connectivity (e.g., local support for basis functions)
  - Regular structure (e.g., dual graph of mesh)

Example: Finite element matrix

Sandia National Laboratories

- Structural problem discretizing a gas reservoir with tetrahedral finite elements
- Platform: SNL Redsky cluster
  - 2.93 GHz dual socket/quad core Nehalem X5570 procs
  - 3D torus InfiniBand network
- Graph partition gives 25% reduction in SpMV time relative to 1D-Block
  - Improves load balance
  - Reduces communication volume



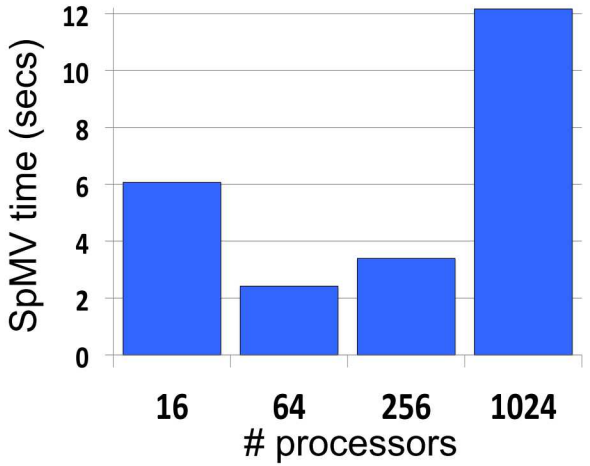
Serena matrix
Janna & Ferronato
U.Florida Sparse Matrix Collection

Serena: 1.4M rows; 65M nonzeros; Max 249 nz/row; Avg 46 nz/row 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	1.2	55	4.4M	0.20
1D-Random	1.0	1023	62.1M	13.62
1D-GP	1.1	98	1.1M	0.15

CounterExample: Social network matrix

Sandia National Laboratories

- Social networks, web graphs, etc., have very different structure from PDE discretizations
  - Power-law degree distributions; scale-free properties
- Graph partitioning can reduce SpMV time
  - Reduces imbalance and communication volume
- But large number of messages hurts scaling
  - Nearly all-to-all communication



Strong scaling of 1D-GP for com-liveJournal matrix
Yang & Leskovec
Stanford SNAP collection

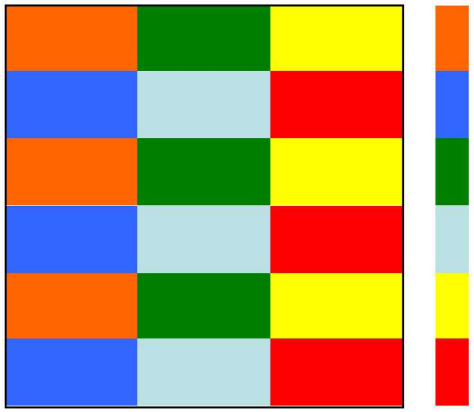
com-liveJournal: 4M rows; 73M nonzeros; Max 15K nz/row; Avg 18 nz/row 1024 processes				
Method	Imbalance in nonzeros (Max/Avg per proc)	Max # Messages per SpMV	Comm. Vol. per SpMV (doubles)	100 SpMV time (secs)
1D-Block	12.8	1023	34.5M	14.72
1D-Random	1.3	1023	66.3M	14.00
1D-GP	1.2	1011	18.9M	12.17

Goal: Reduce number of messages

Sandia National Laboratories

- 1D distribution:
  - Entire rows (or columns) of matrix assigned to a processor
- 2D distribution:
  - Cartesian methods: Each process owns intersection of some rows & columns
  - Processes are *logically* arranged in a 2D grid
  - Limits max #messages per process to *O(sqrt(#processors))*
  - Long used in parallel dense solvers (ScaLapack)
  - Beneficial also for sparse matrices (Fox et al. '88, Lewis & van de Geijn '93, Hendrickson et al. '95)
  - Yoo et al. (SC'11) demonstrated benefit over 1D layouts for eigensolves on scale-free graphs

1D row-wise matrix distribution; 6 processes

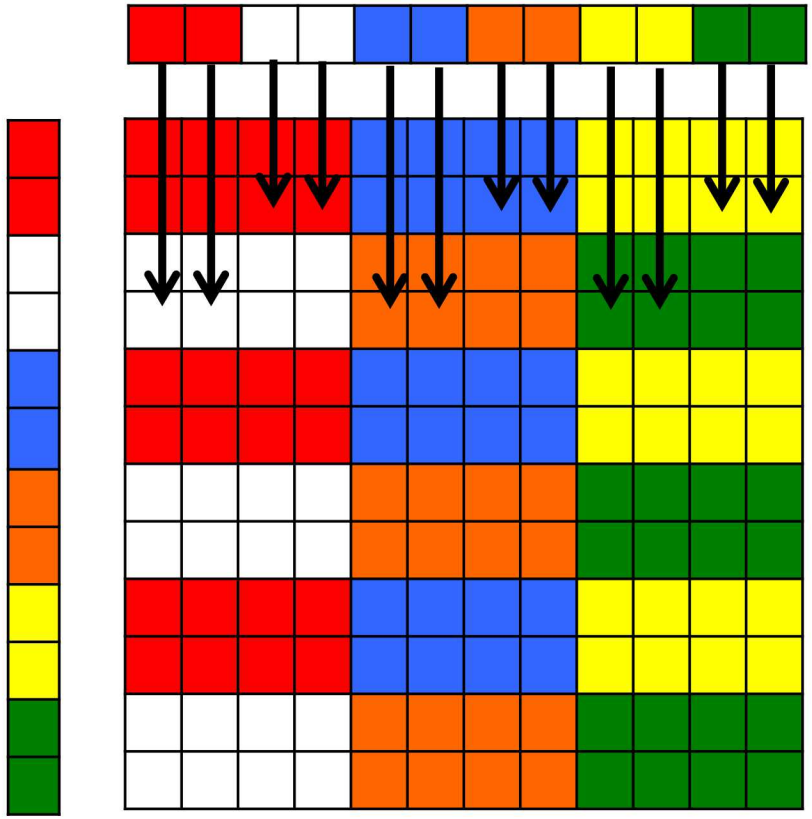


2D matrix distribution; 6 processes

Benefit of 2D Matrix Distribution

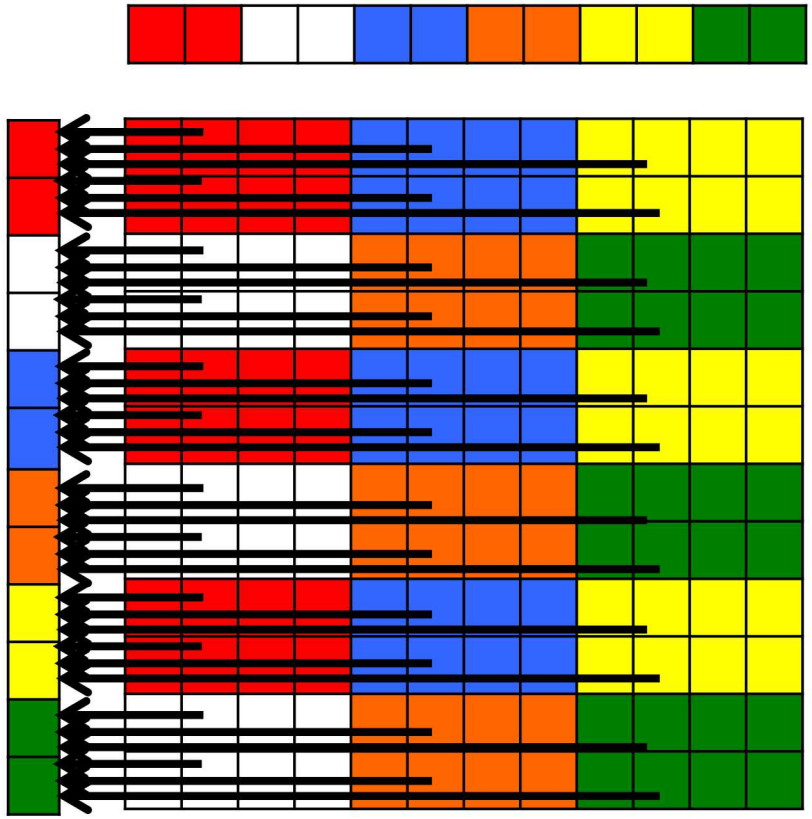
Sandia National Laboratories

- During matrix-vector multiplication (y=Ax), 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; $y = \sum y^p$
- 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; $y = \sum y^p$
- In 1D, fold is not needed, but expand may be all-to-all.



2D Partitioning of Social Network

Sandia National Laboratories

- Drastic reduction in max number of messages and SpMV time
  - Even with expand & fold, max number of messages is smaller
- Communication volume high with 2D partitions
  - Ignoring the non-zero structure of the matrix.
  - Can we exploit it as we did with 1D-GP?

liveJournal: 4M rows; 73M nonzeros; Max 15K nz/row; Avg 18 nz/row 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	14.72
1D-Random	1.3	1023	66.3M	14.00
1D-GP	1.2	1011	18.9M	12.17
2D-Block	11.4	62	43.4M	1.31
2D-Random	1.0	62	64.2M	0.97

New Method: 2D + Graph Partitioning

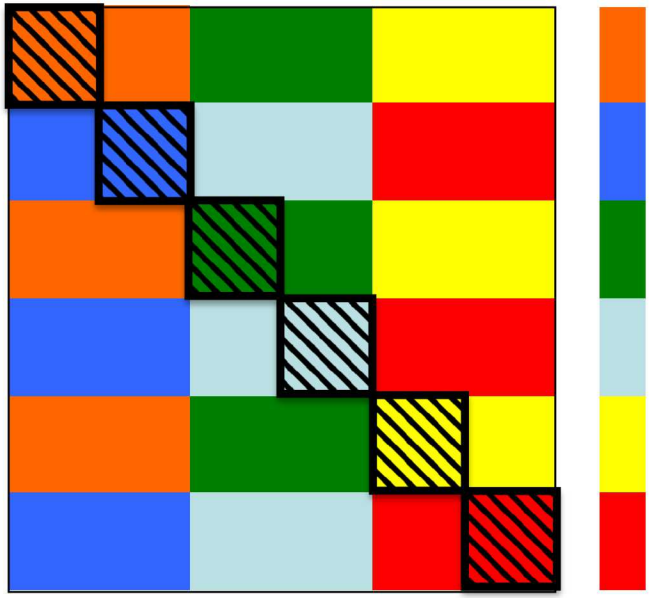
Sandia National Laboratories

- Existing research into direct 2D partitioning of nonzeros (treat nonzeros as graph/hypergraph vertices)
  - Catalyurek & Aykanat; Vastenhouw & Bisseling
  - Much larger problem → very expensive
  - Only serial software available
- Our idea: Apply parallel graph partitioning and 2D distribution together
  - Compute 1D-GP row (vertex) partition of matrix (graph)
  - Apply 2D distribution to the resulting permuted matrix (graph)
- Advantages:
  - Balance the number of nonzeros per process,
  - Exploit structure in the graph to reduce communication volume, AND
  - Reduce the number of messages via 2D distribution
- Don't optimize a single objective but try do fairly well in all

2D Graph Partitioning (2D-GP)

- Partition rows (vertices) of original matrix (graph) into p parts
  - Using standard graph partitioner
- Implicitly, let $A_{perm} = PAP^T$
  - Where P is permutation from partitioning above
- Assign A_{perm} to processes using Cartesian block 2D layout

Due to partitioning, diagonal blocks of A_{perm} will be denser:



Results 1D vs 2D (Block, Random, GP)

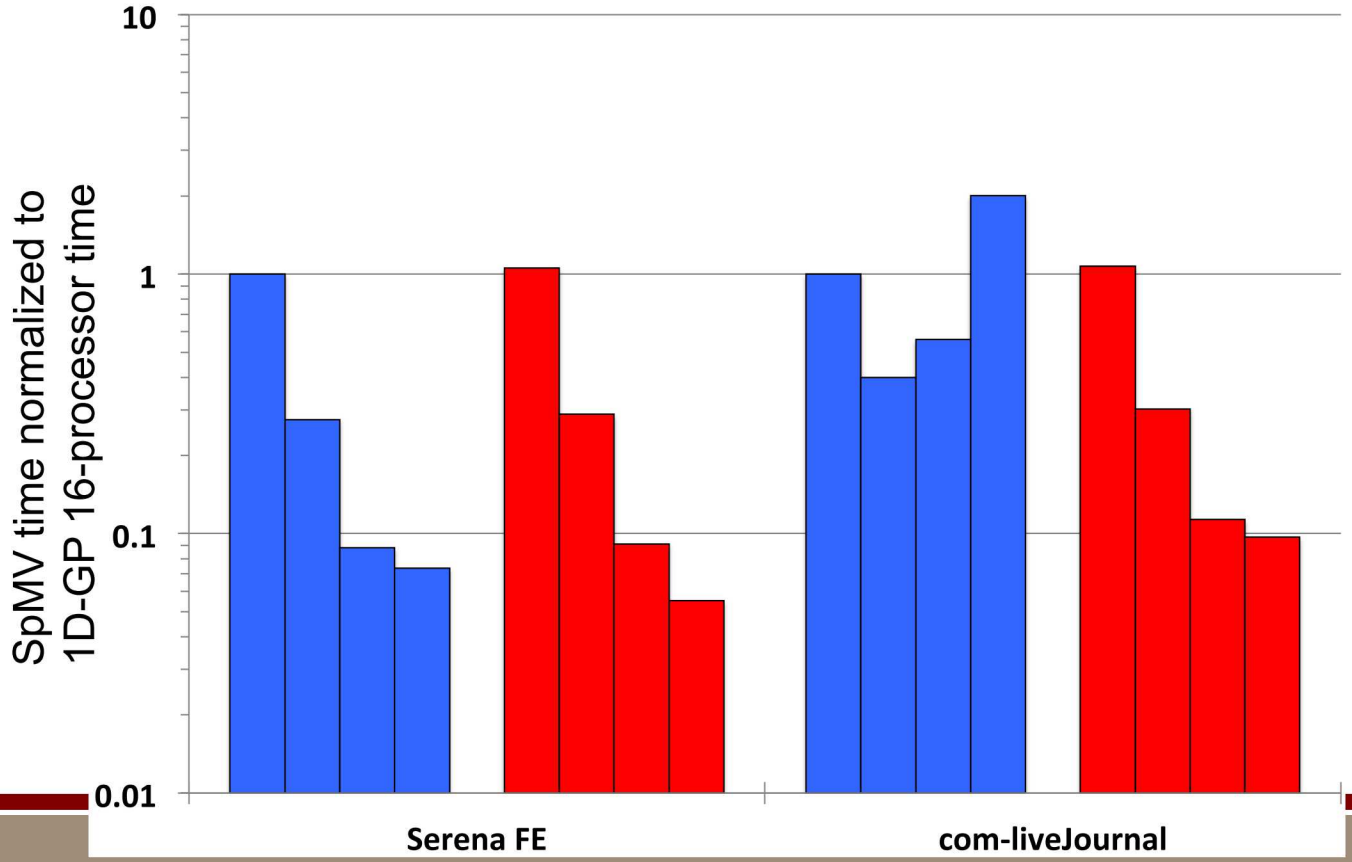
Sandia National Laboratories

- With 2D-GP,
  - Low number of messages as with 2D-Block, 2D-Random
  - Reduced communication volume due to using structure of matrix
  - Reduced SpMV execution time

liveJournal: 4M rows; 73M nonzeros; Max 15K nz/row; Avg 18 nz/row 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	14.72
1D-Random	1.3	1023	66.3M	14.00
1D-GP	1.2	1011	18.9M	12.17
2D-Block	11.4	62	43.4M	1.31
2D-Random	1.0	62	64.2M	0.97
2D-GP	1.4	62	22.4M	0.59

Strong scaling: 1D-GP vs 2D-GP

- Performance for fixed problem as increase number of processors
- For each matrix:
  - **Blue = 1D-GP on 16, 64, 256, 1024 processors (left to right)**
  - **Red = 2D-GP on 16, 64, 256, 1024 processors (left to right)**
  - Times are normalized to the 1D-GP 16-processor runtime

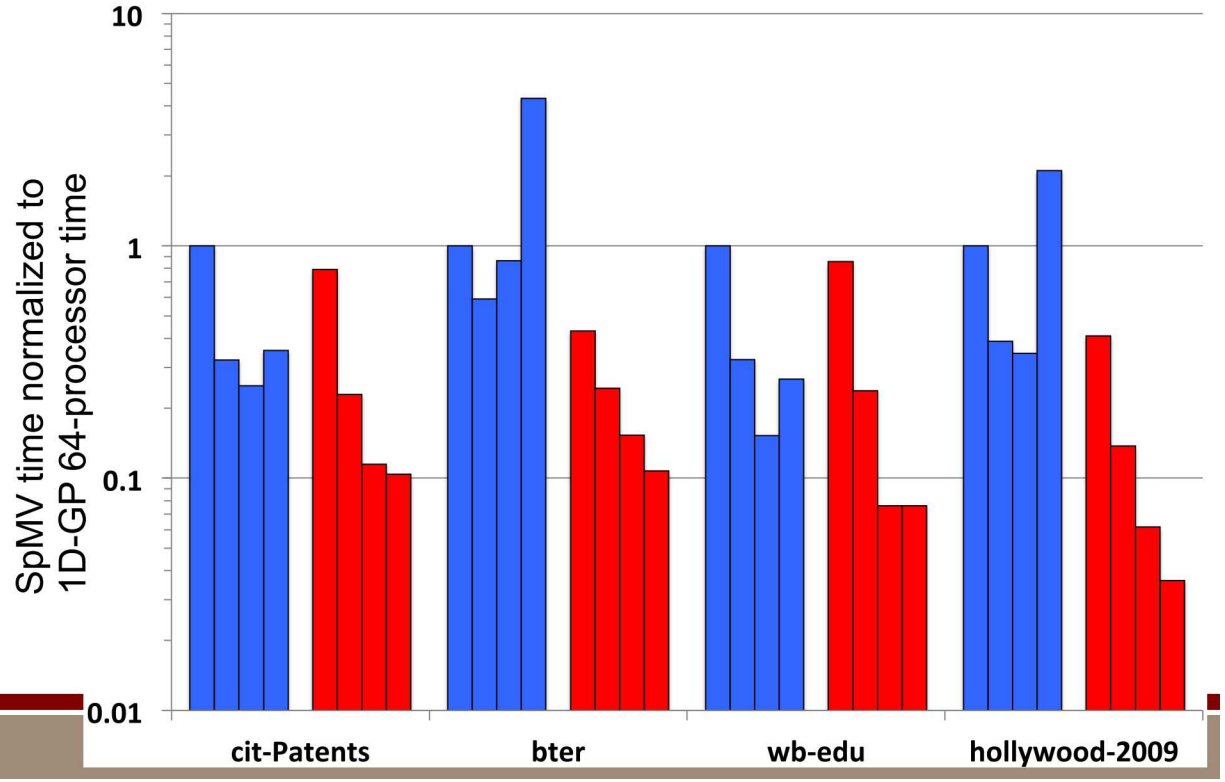


More 1D vs 2D experiments

Platform: cab cluster at LLNL (2.6GHz Intel Xeon E5 16-core nodes, Infiniband)

Name	Description	# Rows	# Nonzeros
cit-patents (UFL)	Citation network of US patents (Hall, Jaffe, Trajtenberg)	3.8M	33M
bter (generated)	Block Two-Level Erdös-Rényi (Seshadhri, Kolda, Pinar)	3.9M	63M
wb-edu (UFL)	Links between *.edu webpages (Gleich)	8.9M	88M
hollywood-2009 (UFL)	Hollywood movie actor network (Boldi, Rosa, Santini, Vigna)	1.1M	113M

- For each matrix:
  - Blue = 1D-GP on 64, 256, 1024, 4096 processors (left to right)
  - Red = 2D-GP on 64, 256, 1024, 4096 processors (left to right)
  - Times are normalized to the 1D-GP 64-processor runtime



Conclusions

Sandia National Laboratories

- Parallel distribution strategies depend on structure of data
  - Sparsity, regularity, dimensions of matrix are important
  - Demonstrated with Finite Element vs Social Network matrices
- Tools developed for PDEs can be applied *cleverly* in other application domains
  - Exploited partitioners and linear algebra libraries (developed for scientific computing) in network analysis scenario
  - Partitioners: Zoltan (SNL) and ParMETIS (U. Minnesota)
  - Matrix/Vector classes: Trilinos (SNL) using Map class to describe 1D and 2D distributions
- Challenging and exciting opportunities for cross-utilization of hardware, systems, algorithms and software

For more information…

Sandia National Laboratories

- "Scalable Matrix Computations on Large Scale-Free Graphs Using 2D Graph Partitioning."
  Erik Boman, Karen Devine, and Sivasankaran Rajamanickam
  Proc. of the International Conference on High Performance Computing, Networking, Storage and Analysis (SC13)

- Trilinos home page: http://trilinos.org
- Zoltan home page: http://www.cs.sandia.gov/Zoltan

- Email: kddevin@sandia.gov