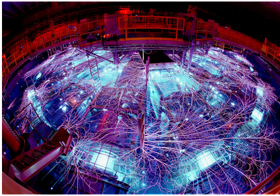
*Exceptional service in the national interest*

Sandia National Laboratories

SAND2016-6136C



# Neighbor Discovery for Algebraic Multigrid and Matrix Migration

Chris Siefert
Sandia National Laboratories

6/22/16

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. SAND NO. 2016-00000

# Outline

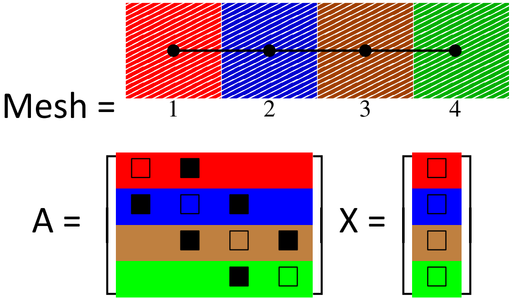
- What is neighbor discovery?
- Efficient neighbor discovery for matrix migration
- Conclusions

# Parallel Sparse Matrices

- Congratulations! You can store a parallel sparse matrix w/ MPI! What's next?
- You probably want to be able to *multiply* this matrix by a vector.
- What sort of communication structures do we need (presuming row-wise storage)?
  - The *domain* distribution of the vector.
  - The *column* distribution of the matrix.
  - The list of (data,destination) pairs each rank *sends*.
  - The list of (data,source) pairs each rank *receives*.

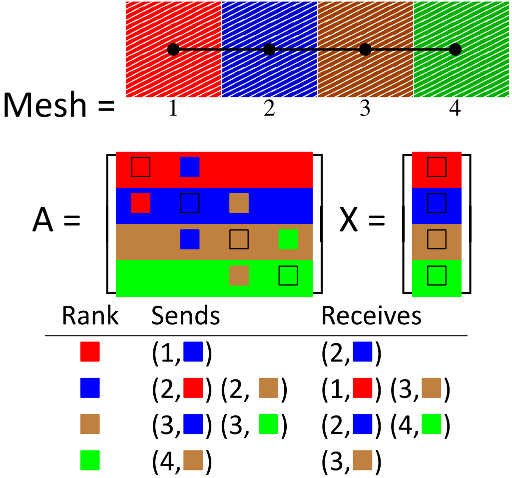
# Finding Neighbors in General

- So, supposing we had this:




- How do we fill out our send and receive lists?

# Data Distribution #1



# General Algorithms

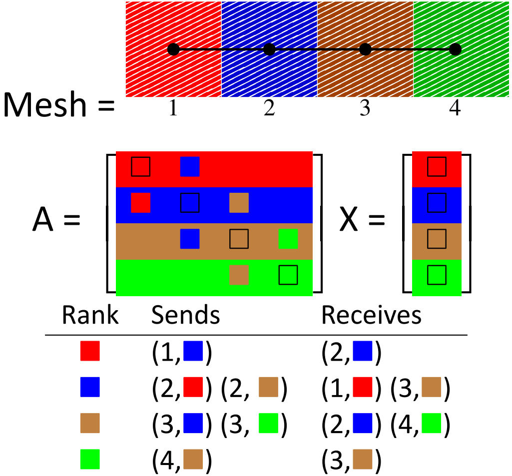
- Idea: Use assumed partition [1] or rendezvous scheme.
  - Create assumed partition w/ easy to calculate range.
  - Each owning proc talks to assumed owner.
  - Each proc asks assumed owner who owns needed unknowns.
  - Requires $O(\log(p))$ distributed termination detection [2].
- Message: You need to exploit structure (of some kind) to get $O(1)$ storage and communication.
- BUT, once you have a hammer, everything looks like a nail.

[1] Barker, Falgout and Yang, 2006.
[2] Pinar and Hendrickson, 2001.

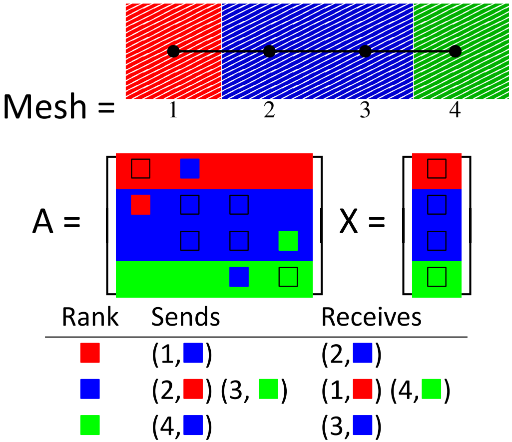
# Outline

- What is neighbor discovery?
- Efficient neighbor discovery for matrix migration
- Conclusions

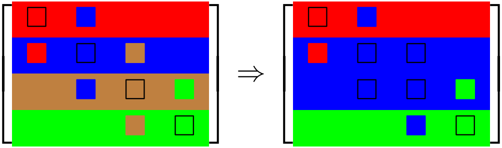
# Data Distribution #1



# Data Distribution #2

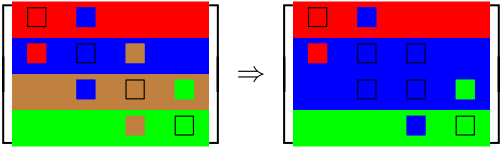


# So what do we do?



- What happens?
  - Rank ■ gets row 3 from Rank ■.
  - Rank ■ now has a new neighbor, Rank ■ (and the reverse).
- How do we solve this? Try Rank ■'s perspective.
  - Rank ■ passes row 3 to Rank ■.
  - Rank ■ already knows that Rank ■ is a neighbor.
  - Basic Idea: What if Rank ■ tells ■ that?

# Algorithm (■ edition)



- Forward round: If I pass a row to ■, I tell ■ who that row's neighbors are (e.g. ■).
- Reverse round: If I passed a row to ■ telling him that ■ is now his neighbor, I must tell ■ that ■ is now her neighbor.
- Idea: Use the send/recv structure in *both* directions.

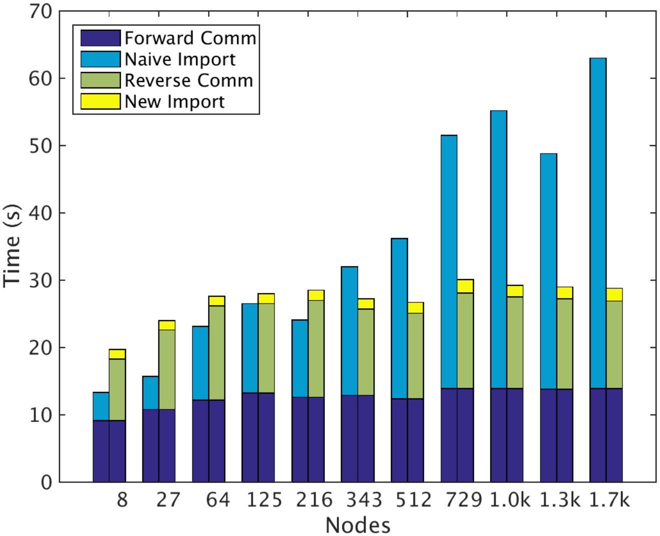
# Algorithm (in more detail)

- Forward round
  - ∀ send row id *i*, ∀ nonzeros in row *i* , pass a (value, global column id, owning rank) triplet.
  - Combine recv'd global column ids with existing global column ids to generate a column distribution map.
- Reverse round
  - ∀ recv'd row id *i*, pass a list of ranks to whom an entry in global column *i* was sent during the forward round.
- Send/recv list generation
  - Combine forward round "owning ranks" with other existing recvs to get recv list.
  - Combine reverse round "communicated ranks" with other existing sends to get send list.
- Note: "New" stuff is in blue.

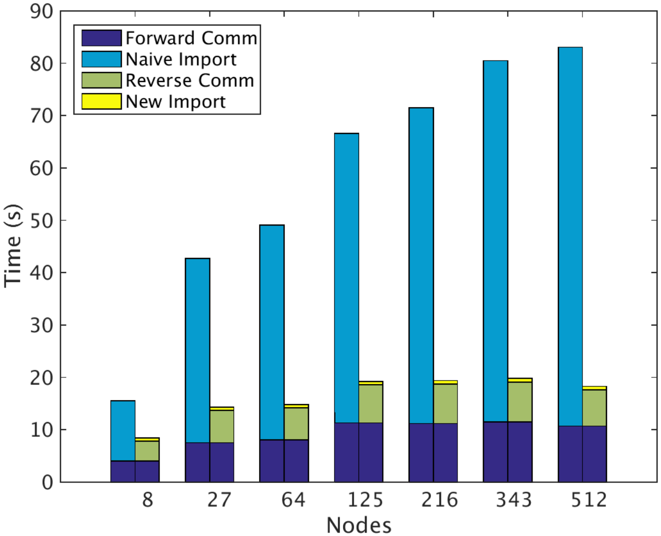
# Computational Example

- Example: 3D Laplacian (*A*) and prolongator (*P*) from Trilinos/MueLu.
- Matrix migration: Off-processor portions of *P* needed to compute $C = A\,P$.
- Compare: Communication costs
  - Building communication structures *ex nihilo*.
  - Building them via the aforementioned algorithm.
  - Trilinos/Epetra code used in both cases.
- Two machines
  - SNL's Redsky.
  - NERSC's Edison.
- Note: Pack/unpack costs will be neglected to focus on comm.

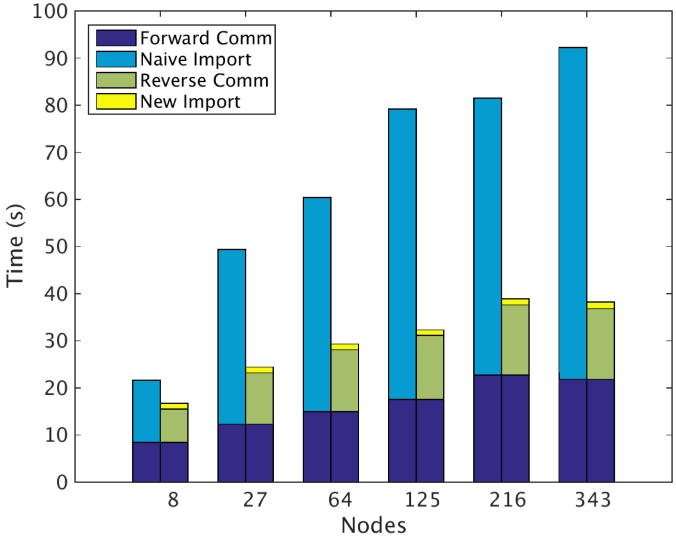
# Edison: 15k Unknowns / Core



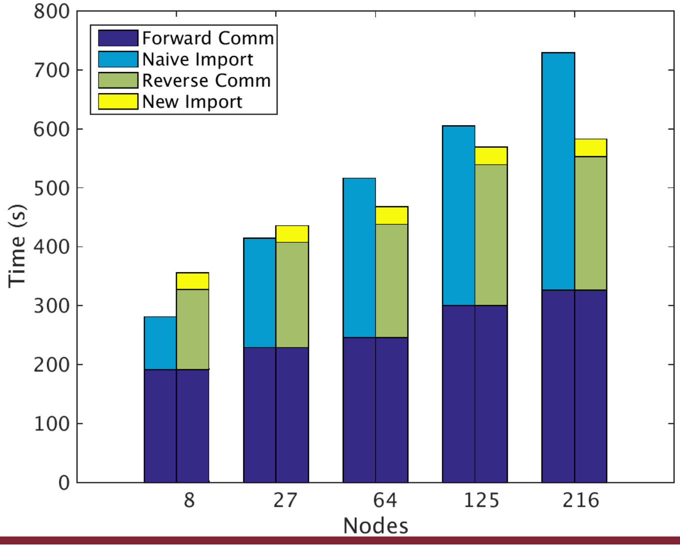
# Redsky: 5k Unknowns / Core



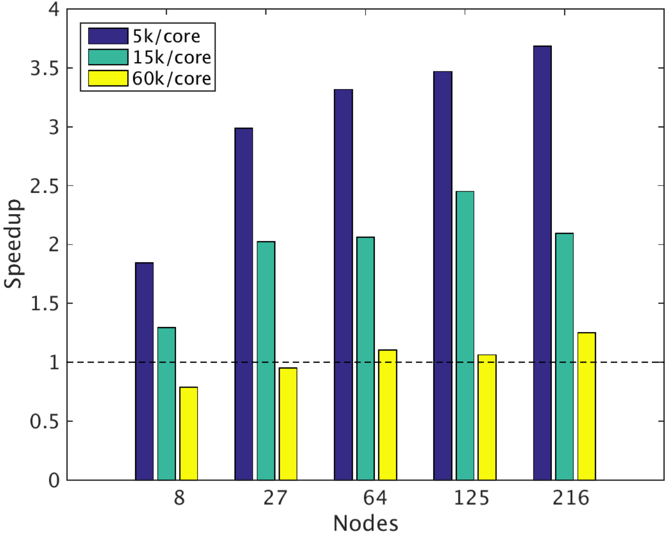
# Redsky: 15k Unknowns / Core



# Redsky: 60k Unknowns / Core



# Redsky: Speedup



# Outline

- What is neighbor discovery?
- Efficient neighbor discovery for matrix migration
- Conclusions

# Conclusions

- There is enough structure in matrix migration to get $O(1)$ cost neighbor discovery.
  - This kernel is especially useful in AMG's triple product.
  - A screwdriver usually does a better job than a hammer…
  - But with the right machine and enough data per core, maybe a hammer is good enough.
- Future directions
  - Other applications: Repartitioning, off-processor FEM assembly.
  - Complete deployment in Trilinos/Tpetra utility routines.
  - Further optimization of communication.