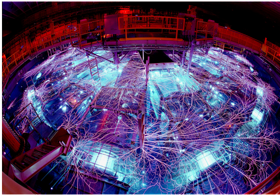
*Exceptional service in the national interest*

Sandia National Laboratories

SAND2018-1752C

# Low Communication Neighbor Discovery for Matrix Migration


Chris Siefert (Sandia National Laboratories) and Chris Luchini (Sci Tac LLC)


2/15/18

U.S. DEPARTMENT OF ENERGY

NNSA National Nuclear Security Administration


Sandia National Laboratories is a multimission laboratory managed and operated by National Technology and Engineering Solutions of Sandia LLC, a wholly owned subsidiary of Honeywell International Inc. for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-NA0003525. SAND NO. 2016-00000

# Outline

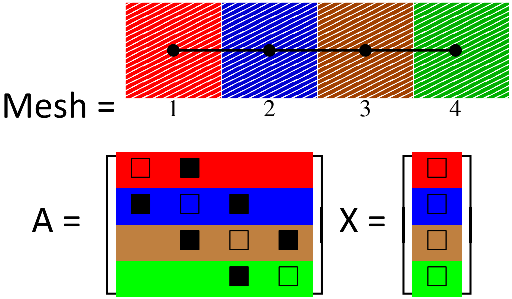
- What is neighbor discovery?
- Efficient neighbor discovery for matrix migration
- Conclusions

# 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*.

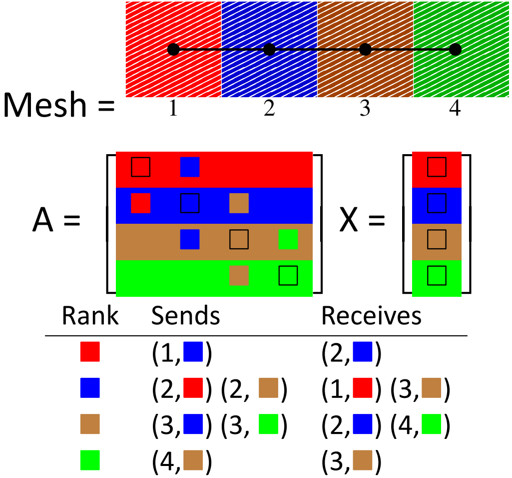
# Finding Neighbors in General

- So, supposing we had this:




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

# Data Distribution #1



# General Algorithms

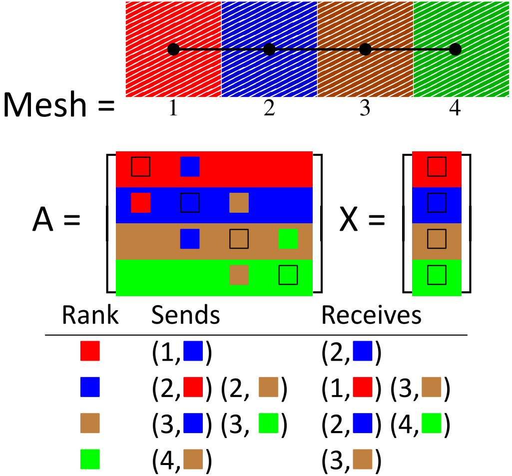
- Idea: Use assumed partition [1] or rendezvous scheme.
  - Create assumed partition w/ easy to calculate range.
  - Each owning proc talks to assumed owner.
  - Each proc asks assumed owner who owns needed unknowns.
  - Requires $O(\log(p))$ distributed termination detection [2].
- Message: You need 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.

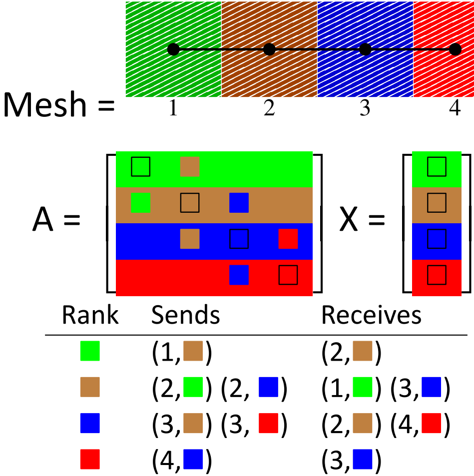
# Outline

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

# Data Distribution #1

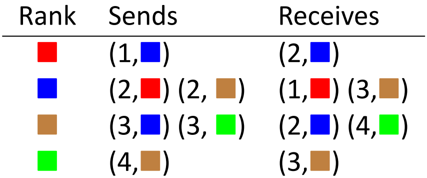


# Data Distribution #2 (Reversed)

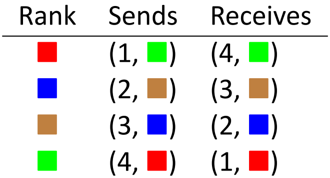


# Migration from #1 to #2

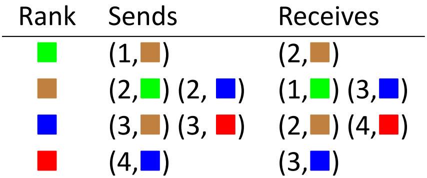
- From this: Xfer 1 =



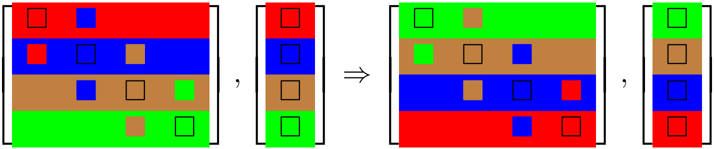

- Via this: Mat / X Migration =




- To this: Xfer 2 =

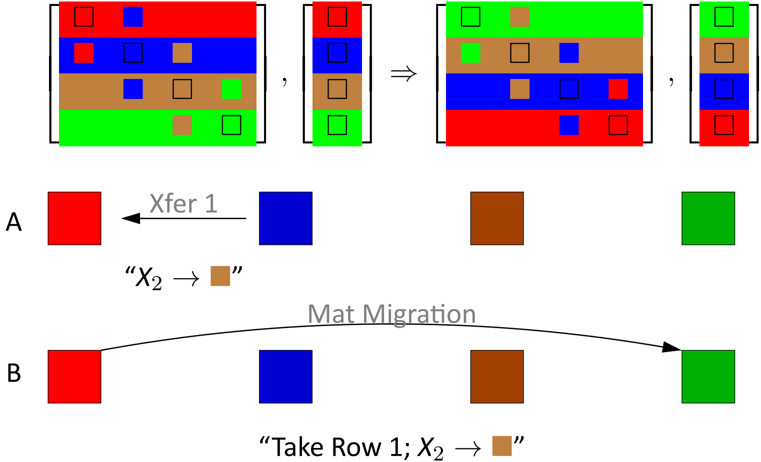


# What happens: Focus on Row 1



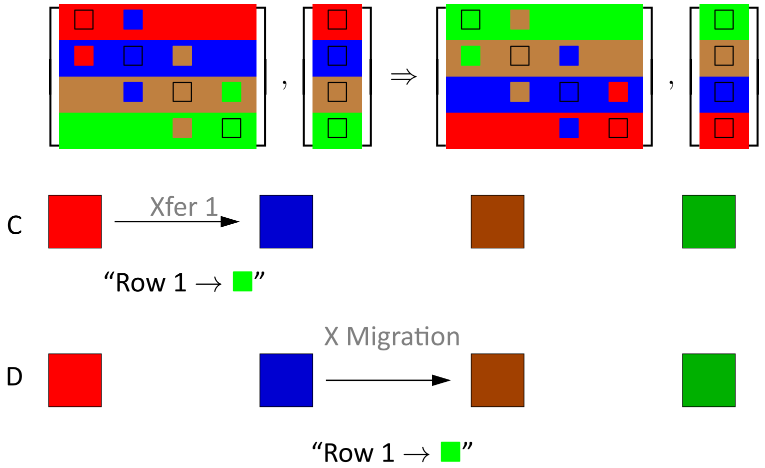
- Row 1 goes from to .
- Row 1 recv'd $X_2$ from , but now that is owned by . How does know this?
- Moreover, how does know it needs to send $X_2$ to ?
- What should we do?

# Forward Algorithm by Picture (Row 1)



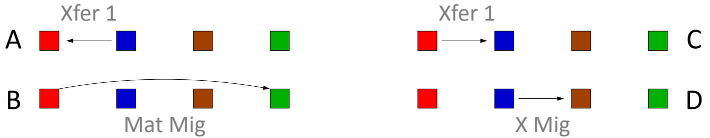

- This solves the recv problem, but what about sends?

# Reverse Algorithm by Picture (Row 1)



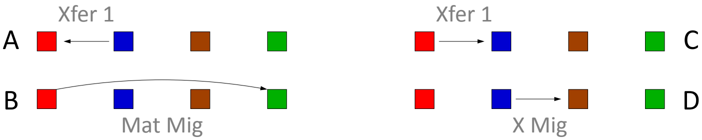

- Now we know our recv's.

# Algorithm (in more detail)




- Forward round
  - A: ∀ row in (xfer 1) send id ∩ (x migration) send id, pass a (global domain id, owning domain rank) pair.
  - B: ∀ nonzeros in (mat migration) send row ids, pass a (value, global column id, owning domain rank) triplet.
- Reverse round
  - C: ∀ (xfer 1) recv'd id *i*, pass a list of ranks to whom an entry in global column *i* was sent during B.
  - D: ∀ (xfer 1) recv'd id *i* from C that is no longer owned, pass that message along (x migration).

# Algorithm Optimizations

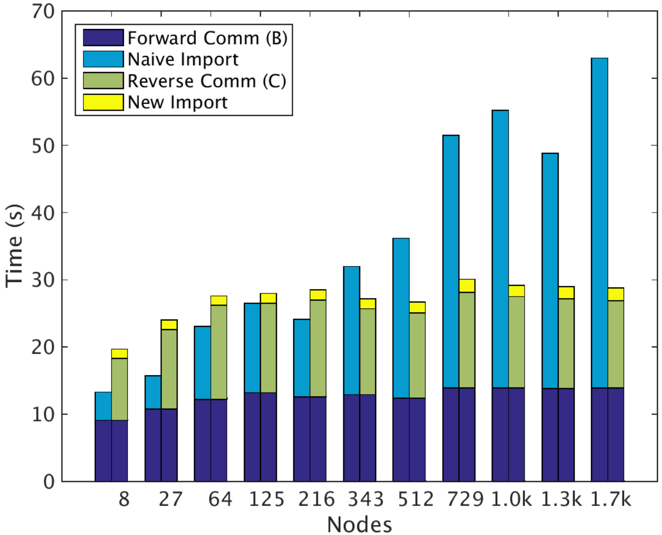



- This can be implemented as a 4 round algorithm.
- However, the only data dependencies are B on A and D on C $\rightarrow$ this can be implemented in as little as two rounds.
- If rows in the *X* vector aren't moving, we can skip steps A & D (this is the case for matrix-matrix multiplication and transpose). And we can combine B & C if we want to.

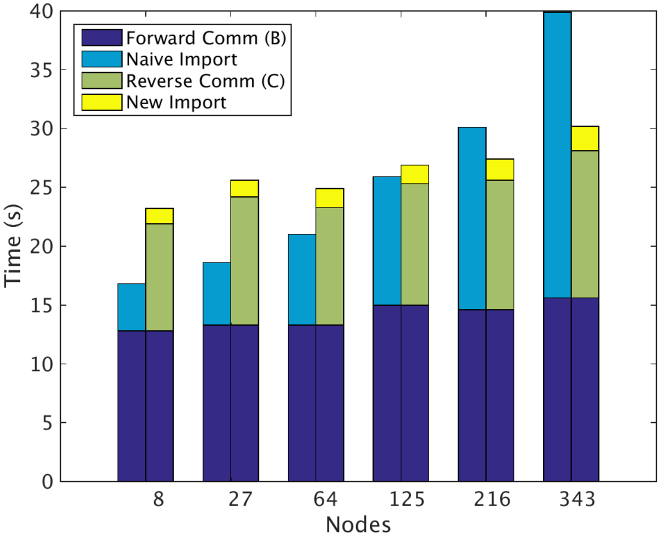
# Computational Example

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

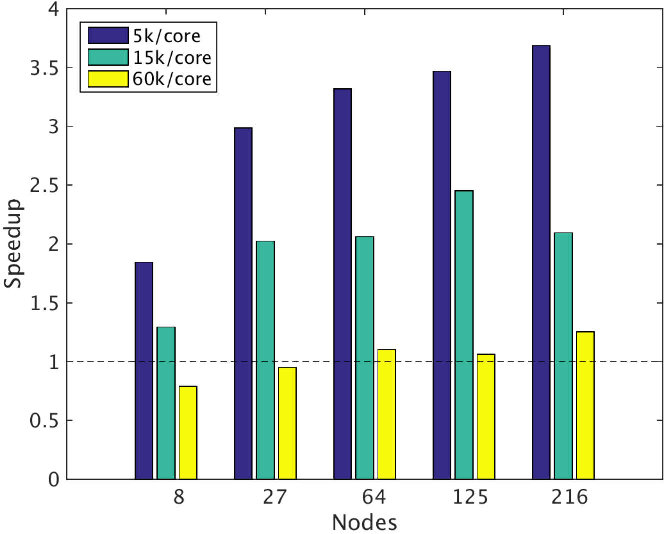
# Edison: 15k Unknowns / Core



# Serrano: 20k 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.
  - Four rounds in general, but can be reduced to 1 in the matrix multiplication & transpose case.
  - 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
  - Complete deployment in Trilinos/Tpetra.
  - Implement the 1-round combination of B & C.
  - Demonstrate relevance on Xeon Phi / Cuda architectures.