OUTLINE

Motivation and background

Selected Inversion Algorithm

`PSelInv`: a parallel implementation of Selected Inversion

Initial performance evaluation

Communication load analysis (Symmetric case)

Performance evaluation

Conclusion

Motivations:

- Sparse matrices arise in many applications:
  - Optimization problems
  - Discretized PDEs
  - Electronic structure theory
  - ...
- Some sparse direct methods require:
  - Sparse factorizations
  - Computing some inverse elements

Motivations:

- Sparse matrices arise in many applications:
  - Optimization problems
  - Discretized PDEs
  - Electronic structure theory
  - ...
- Some sparse direct methods require:
  - Sparse factorizations
  - **Computing some inverse elements**

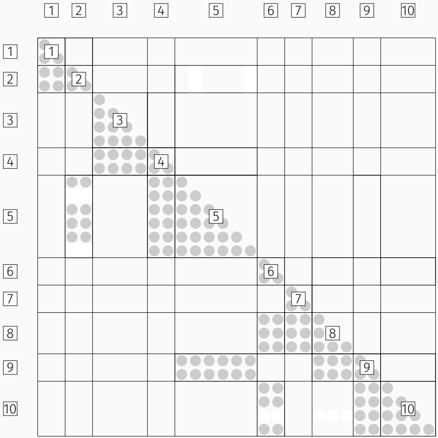
Motivations:

- Sparse matrices arise in many applications:
  - Optimization problems
  - Discretized PDEs
  - Electronic structure theory
  - ...
- Some sparse direct methods require:
  - Sparse factorizations
  - **Computing some inverse elements**

Challenges for current and future platforms:

- Lower amount of memory per core
- Higher relative communication costs
- Large performance variations

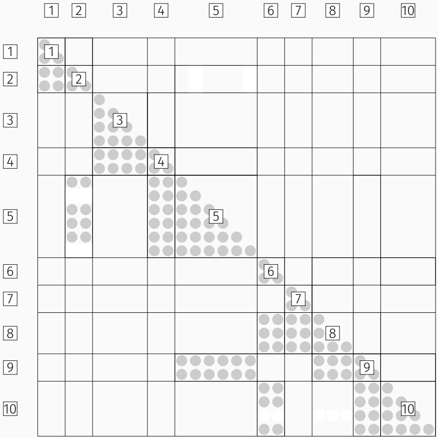
SPARSE MATRICES AND ELIMINATION TREE

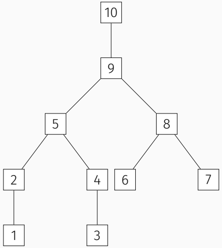


- Fill in, $\Omega(A) \subseteq \Omega(L)$

$$A = LL^T$$

$\Omega(A)$ is the sparsity pattern of A

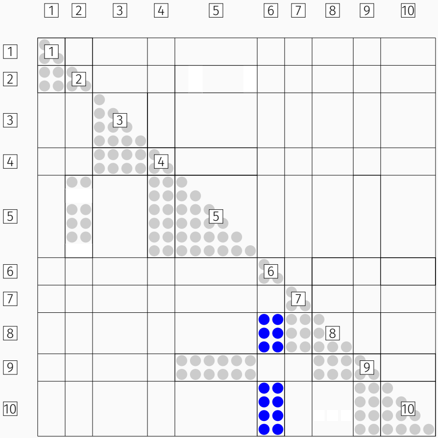


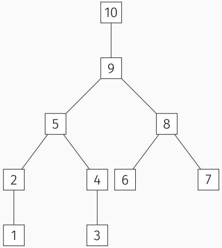


- Elim. tree represents column dependences
- Fill in, $\Omega(A) \subseteq \Omega(L)$

$$A = LL^T$$

$\Omega(A)$ is the sparsity pattern of A





- Elim. tree represents column dependences
- Fill in, $\Omega(A) \subseteq \Omega(L)$
- Supernode, same structure below diagonal block

$$A = LL^T$$

$\Omega(A)$ is the sparsity pattern of A

Motivation:

- Many applications require **some** elements of an inverse matrix:
  - Electronic structure theory
  - Confidence interval estimation
  - Poisson-Boltzmann
  - Quantum transport theory

Ex: Kohn-Sham density functional theory

$$\Gamma = f(H - \mu I) \approx \sum_{i=1}^{Q} \frac{\omega_i}{H - z_i I}$$

$$\rho(\Gamma) \approx \sum_{i=1}^{Q} diag\left(\frac{\omega_i}{H - z_i I}\right)$$

Objective:

- Compute **selected entries** of A^{-1}

$$\{(A^{-1})_{ij} \mid A_{ij} \neq 0 \;\; or \;\; A_{ji} \neq 0, 1 \leq i,j \leq N\}$$

- "Naive" solution: sequence of solve on e_j
  - $A = LU, A^{-1} = [x_1, x_2 \ldots x_N] \implies$ Solve $LUx_j = e_j$
- Better algorithms:

  [Takahashi et al. 1973], [Erismann and Tinney 1975]

- "Naive" solution: sequence of solve on e_j
  - $A = LU, A^{-1} = [x_1, x_2 \dots x_N] \implies$ Solve $LUx_j = e_j$
- Better algorithms:
  [Takahashi et al. 1973], [Erismann and Tinney 1975]
- Let $A = LU$ be the LU factorization of A
- Selected Inversion: entries of A^{-1} corresponding to $\Omega(A)$

- "Naive" solution: sequence of solve on e_j
  - $A = LU, A^{-1} = [x_1, x_2 \dots x_N] \implies$ Solve $LUx_j = e_j$
- Better algorithms:
  [Takahashi et al. 1973], [Erismann and Tinney 1975]
- Let $A = LU$ be the LU factorization of A
- Selected Inversion: entries of A^{-1} corresponding to $\Omega(A)$
- If A sparse, entries of A^{-1} in $\Omega(A)$ require entries in $\Omega(L+U)$

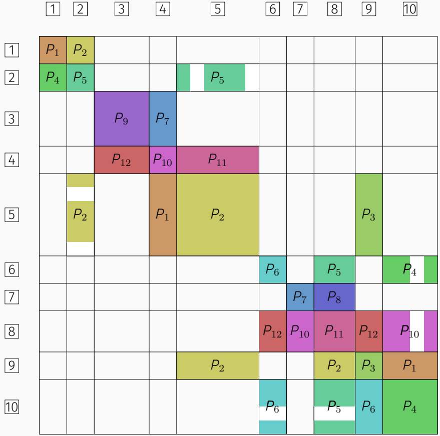
- "Naive" solution: sequence of solve on e_j
  - $A = LU, A^{-1} = [x_1, x_2 \dots x_N] \implies$ Solve $LUx_j = e_j$
- Better algorithms:
  [Takahashi et al. 1973], [Erismann and Tinney 1975]

- Let $A = LU$ be the LU factorization of A
- Selected Inversion: entries of A^{-1} corresponding to $\Omega(A)$
- If A sparse, entries of A^{-1} in $\Omega(A)$ require entries in $\Omega(L+U)$
- All entries of A^{-1} in $\Omega(L+U)$ are actually computed

for *Supernode* $\mathcal{K} = \mathcal{N}$ *down to* 1 **do**

  Compute selected elements of $A^{-}1$ within $\mathcal{K}$

end



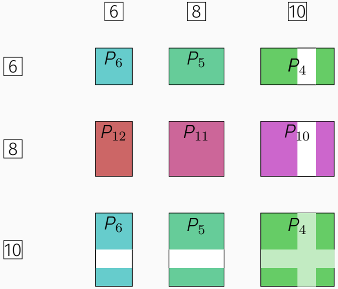
P_1	P_2	P_3
P_4	P_5	P_6
P_7	P_8	P_9
P_{10}	P_{11}	P_{12}

- 2D Block Cyclic layout
- 4-by-3 processor grid
- No explicit load balancing
- Works well in practice [Gupta]

PARALLEL PROCESSING OF A SUPERNODE

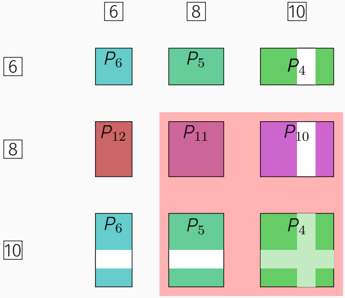
$$A^{-1} = \begin{pmatrix} D^{-1} + US_{-1}L & -US^{-1} \\ -S^{-1}L & S^{-1} \end{pmatrix}$$

- Supernode depends on S^{-1}



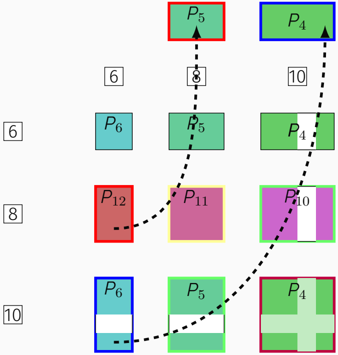
$$A^{-1} = \begin{pmatrix} D^{-1} + US_{-1}L & -US^{-1} \\ -S^{-1}L & S^{-1} \end{pmatrix}$$

- Supernode depends on S^{-1}
- Supernode processed in parallel
  - $S^{-1} \leftrightarrow$ Ancestors in the elimination tree
  - Ancestors compute contributions
  - Contributions reduced



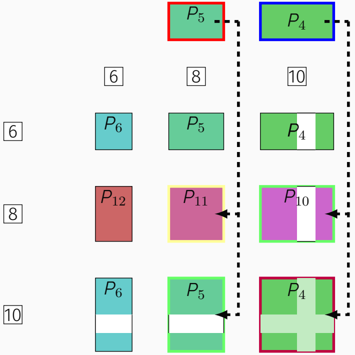
PARALLEL PROCESSING OF A SUPERNODE

$$A^{-1} = \begin{pmatrix} D^{-1} + US_{-1}L & -US^{-1} \\ -S^{-1}L & S^{-1} \end{pmatrix}$$



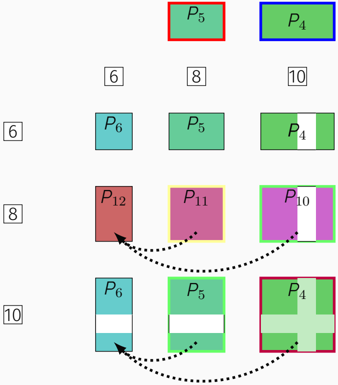
- Supernode depends on S^{-1}
- Supernode processed in parallel
  - $S^{-1} \leftrightarrow$ Ancestors in the elimination tree
  - Ancestors compute contributions
  - Contributions reduced
- Complex communication pattern
  - Sending to cross-diagonal processors $\leftrightarrow$ simpler pattern

$$A^{-1} = \begin{pmatrix} D^{-1} + US_{-1}L & -US^{-1} \\ -S^{-1}L & S^{-1} \end{pmatrix}$$



- Supernode depends on S^{-1}
- Supernode processed in parallel
  - $S^{-1} \leftrightarrow$ Ancestors in the elimination tree
  - Ancestors compute contributions
  - Contributions reduced
- Complex communication pattern
  - Sending to cross-diagonal processors $\leftrightarrow$ simpler pattern
  - Communication only within rows / columns of processors

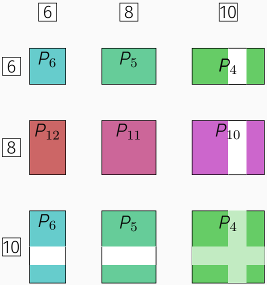
$$A^{-1} = \begin{pmatrix} D^{-1} + US_{-1}L & -US^{-1} \\ -S^{-1}L & S^{-1} \end{pmatrix}$$



- Supernode depends on S^{-1}
- Supernode processed in parallel
  - $S^{-1} \leftrightarrow$ Ancestors in the elimination tree
  - Ancestors compute contributions
  - Contributions reduced
- Complex communication pattern
  - Sending to cross-diagonal processors $\leftrightarrow$ simpler pattern
  - Communication only within rows / columns of processors

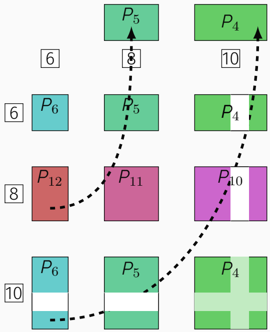
PARALLEL PROCESSING OF A SUPERNODE

$$A^{-1} = \begin{pmatrix} D^{-1} + US_{-1}L & -US^{-1} \\ -S^{-1}L & S^{-1} \end{pmatrix}$$



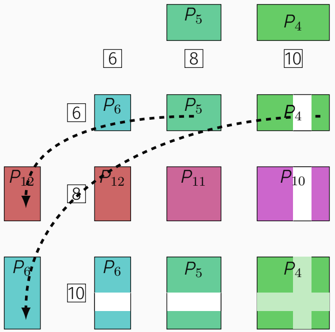
- D: diagonal block
- L: lower triangular block
- U: upper triangular block

$$A^{-1} = \begin{pmatrix} D^{-1} + US_{-1}L & -US^{-1} \\ -S^{-1}L & S^{-1} \end{pmatrix}$$



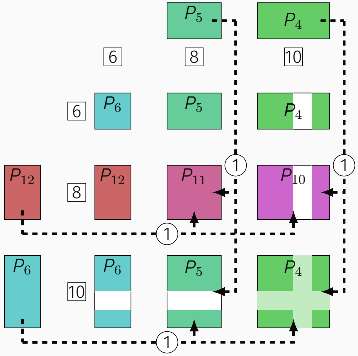
- D: diagonal block
- L: lower triangular block
- U: upper triangular block
- Alloc. L and U on cross-diagonal processors

$$A^{-1} = \begin{pmatrix} D^{-1} + US_{-1}L & -US^{-1} \\ -S^{-1}L & S^{-1} \end{pmatrix}$$



- D: diagonal block
- L: lower triangular block
- U: upper triangular block
- Alloc. L and U on cross-diagonal processors

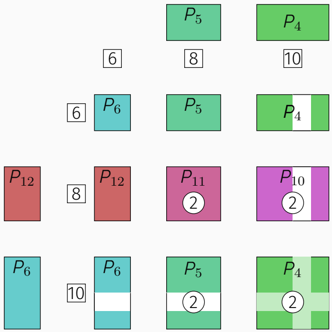
$$A^{-1} = \begin{pmatrix} D^{-1} + US_{-1}L & -US^{-1} \\ -S^{-1}L & S^{-1} \end{pmatrix}$$



- D: diagonal block
- L: lower triangular block
- U: upper triangular block

- Alloc. L and U on cross-diagonal processors
- Bcast. L along cols., U along rows

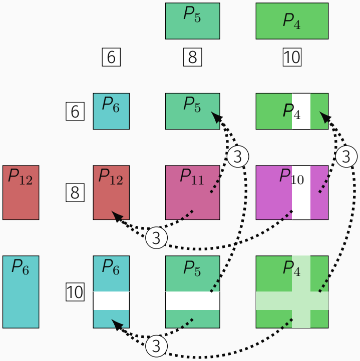
$$A^{-1} = \begin{pmatrix} D^{-1} + US_{-1}L & -US^{-1} \\ -S^{-1}L & S^{-1} \end{pmatrix}$$



- D: diagonal block
- L: lower triangular block
- U: upper triangular block

- Alloc. L and U on cross-diagonal processors
- Bcast. L along cols., U along rows

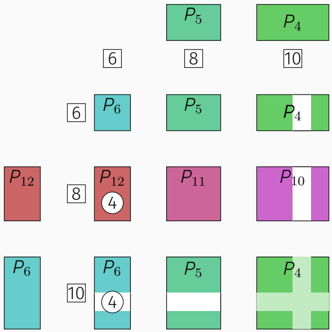
$$A^{-1} = \begin{pmatrix} D^{-1} + US_{-1}L & -US^{-1} \\ -S^{-1}L & S^{-1} \end{pmatrix}$$



- D: diagonal block
- L: lower triangular block
- U: upper triangular block

- Alloc. L and U on cross-diagonal processors
- Bcast. L along cols., U along rows
- Reduce contrib. to L along rows, U along cols.

$$A^{-1} = \begin{pmatrix} D^{-1} + US_{-1}L & -US^{-1} \\ -S^{-1}L & S^{-1} \end{pmatrix}$$

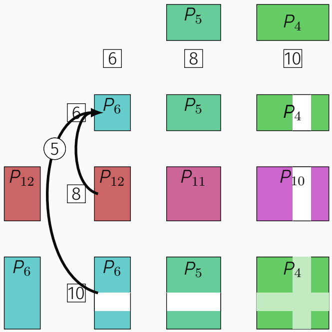


- D: diagonal block
- L: lower triangular block
- U: upper triangular block

- Alloc. L and U on cross-diagonal processors
- Bcast. L along cols., U along rows
- Reduce contrib. to L along rows, U along cols.

PARALLEL PROCESSING OF A SUPERNODE

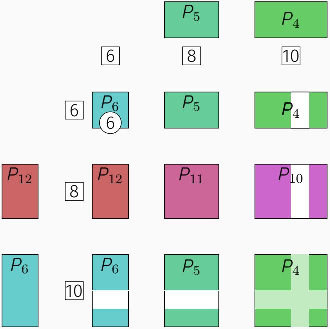
$$A^{-1} = \begin{pmatrix} D^{-1} + US_{-1}L & -US^{-1} \\ -S^{-1}L & S^{-1} \end{pmatrix}$$



- D: diagonal block
- L: lower triangular block
- U: upper triangular block

- Alloc. L and U on cross-diagonal processors
- Bcast. L along cols., U along rows
- Reduce contrib. to L along rows, U along cols.
- Reduce contrib. to D within column

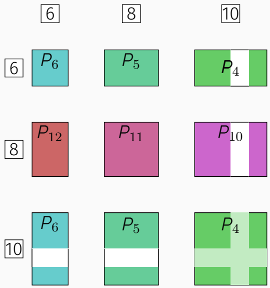
$$A^{-1} = \begin{pmatrix} D^{-1} + US_{-1}L & -US^{-1} \\ -S^{-1}L & S^{-1} \end{pmatrix}$$



- D: diagonal block
- L: lower triangular block
- U: upper triangular block

- Alloc. L and U on cross-diagonal processors
- Bcast. L along cols., U along rows
- Reduce contrib. to L along rows, U along cols.
- Reduce contrib. to D within column

$$A^{-1} = \begin{pmatrix} D^{-1} + US_{-1}L & -US^{-1} \\ -S^{-1}L & S^{-1} \end{pmatrix}$$



- D: diagonal block
- L: lower triangular block
- U: upper triangular block

- Alloc. L and U on cross-diagonal processors
- Bcast. L along cols., U along rows
- Reduce contrib. to L along rows, U along cols.
- Reduce contrib. to D within column
- Delete temporary L and U

CONCURRENCY BETWEEN SUPERNODES

for *Supernode* $\mathcal{K} = \mathcal{N}$ *down to* 1 **do**

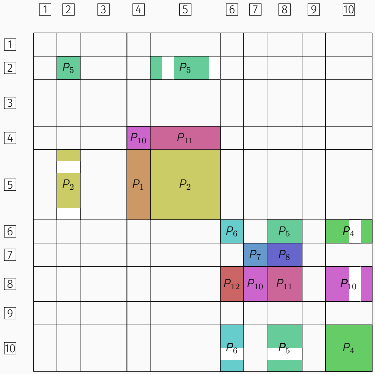
  $\mathcal{R}_\mathcal{K} \leftarrow$ non-zero rows in supernode $\mathcal{K}$

  $Y = S^{-1}_{\mathcal{R}_\mathcal{K},\mathcal{R}_\mathcal{K}} \ell_{\mathcal{R}_\mathcal{K},\mathcal{K}}$

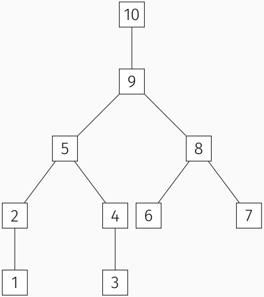
  $A_{\mathcal{K},\mathcal{K}} \leftarrow d^{-1} + Y^T \ell_{\mathcal{R}_\mathcal{K},\mathcal{K}}$

  $A_{\mathcal{R}_\mathcal{K},\mathcal{K}} \leftarrow -Y$

end



- Top-Down elimination tree traversal
- Exploit elimination tree to increase concurrency



for *Supernode* $\mathcal{K} = \mathcal{N}$ *down to* 1 **do**

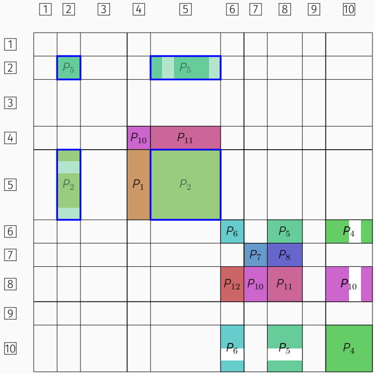
$\quad \mathcal{R}_\mathcal{K} \leftarrow$ non-zero rows in supernode $\mathcal{K}$

$\quad Y = S^{-1}_{\mathcal{R}_\mathcal{K},\mathcal{R}_\mathcal{K}} \ell_{\mathcal{R}_\mathcal{K},\mathcal{K}}$

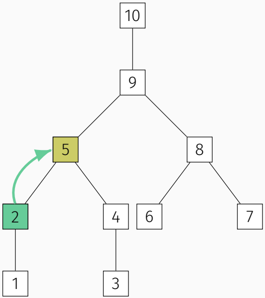
$\quad A_{\mathcal{K},\mathcal{K}} \leftarrow d^{-1} + Y^T \ell_{\mathcal{R}_\mathcal{K},\mathcal{K}}$

$\quad A_{\mathcal{R}_\mathcal{K},\mathcal{K}} \leftarrow -Y$

end



- Top-Down elimination tree traversal
- Exploit elimination tree to increase concurrency



for *Supernode* $\mathcal{K} = \mathcal{N}$ *down to* 1 **do**

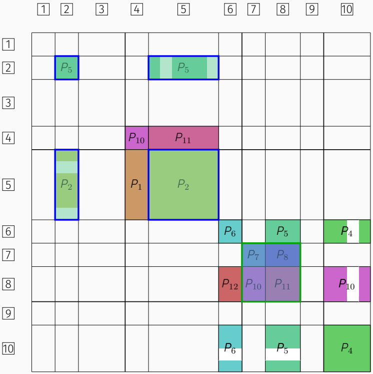
$\quad \mathcal{R}_\mathcal{K} \leftarrow$ non-zero rows in supernode $\mathcal{K}$

$\quad Y = S^{-1}_{\mathcal{R}_\mathcal{K},\mathcal{R}_\mathcal{K}} \ell_{\mathcal{R}_\mathcal{K},\mathcal{K}}$

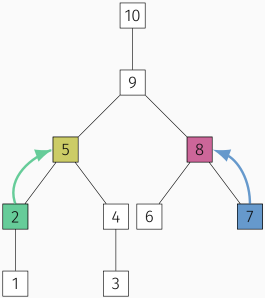
$\quad A_{\mathcal{K},\mathcal{K}} \leftarrow d^{-1} + Y^T \ell_{\mathcal{R}_\mathcal{K},\mathcal{K}}$

$\quad A_{\mathcal{R}_\mathcal{K},\mathcal{K}} \leftarrow -Y$

end



- Top-Down elimination tree traversal
- Exploit elimination tree to increase concurrency



for *Supernode* $\mathcal{K} = \mathcal{N}$ *down to* 1 **do**

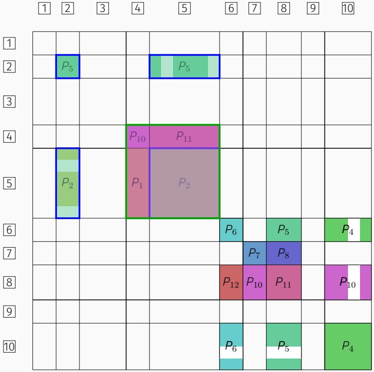
$\mathcal{R}_{\mathcal{K}} \leftarrow$ non-zero rows in supernode $\mathcal{K}$

$Y = S^{-1}_{\mathcal{R}_{\mathcal{K}},\mathcal{R}_{\mathcal{K}}} \ell_{\mathcal{R}_{\mathcal{K}},\mathcal{K}}$

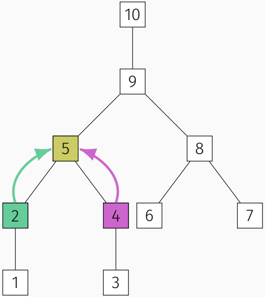
$A_{\mathcal{K},\mathcal{K}} \leftarrow d^{-1} + Y^T \ell_{\mathcal{R}_{\mathcal{K}},\mathcal{K}}$

$A_{\mathcal{R}_{\mathcal{K}},\mathcal{K}} \leftarrow -Y$

end



- Top-Down elimination tree traversal
- Exploit elimination tree to increase concurrency



CONCURRENCY BETWEEN SUPERNODES

for *Supernode* $\mathcal{K} = \mathcal{N}$ *down to* 1 **do**

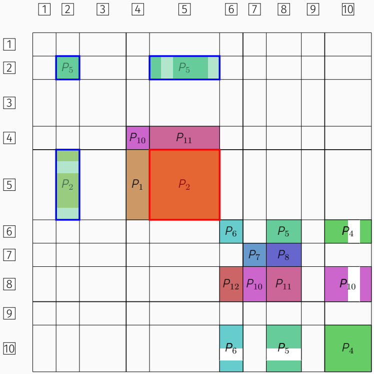
$\mathcal{R}_{\mathcal{K}} \leftarrow$ non-zero rows in supernode $\mathcal{K}$

$Y = S^{-1}_{\mathcal{R}_{\mathcal{K}},\mathcal{R}_{\mathcal{K}}} \ell_{\mathcal{R}_{\mathcal{K}},\mathcal{K}}$

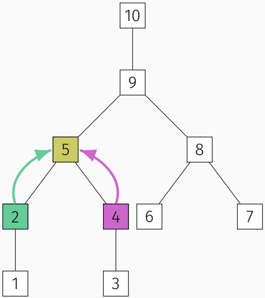
$A_{\mathcal{K},\mathcal{K}} \leftarrow d^{-1} + Y^T \ell_{\mathcal{R}_{\mathcal{K}},\mathcal{K}}$

$A_{\mathcal{R}_{\mathcal{K}},\mathcal{K}} \leftarrow -Y$

end



- Top-Down elimination tree traversal
- Exploit elimination tree to increase concurrency



- Serializations from common ancestors

for *Supernode* $\mathcal{K} = \mathcal{N}$ *down to* 1 **do**

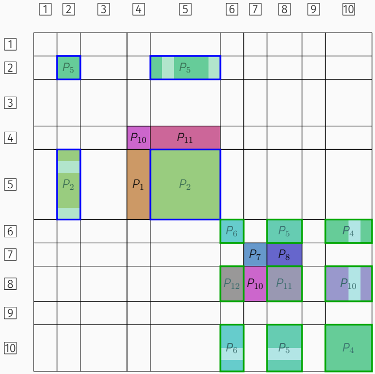
$\mathcal{R}_\mathcal{K} \leftarrow$ non-zero rows in supernode $\mathcal{K}$

$Y = S^{-1}_{\mathcal{R}_\mathcal{K},\mathcal{R}_\mathcal{K}} \ell_{\mathcal{R}_\mathcal{K},\mathcal{K}}$

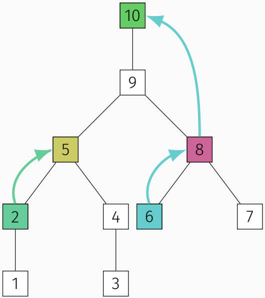
$A_{\mathcal{K},\mathcal{K}} \leftarrow d^{-1} + Y^T \ell_{\mathcal{R}_\mathcal{K},\mathcal{K}}$

$A_{\mathcal{R}_\mathcal{K},\mathcal{K}} \leftarrow -Y$

end



- Top-Down elimination tree traversal
- Exploit elimination tree to increase concurrency



- Serializations from common ancestors

for *Supernode* $\mathcal{K} = \mathcal{N}$ *down to* 1 **do**

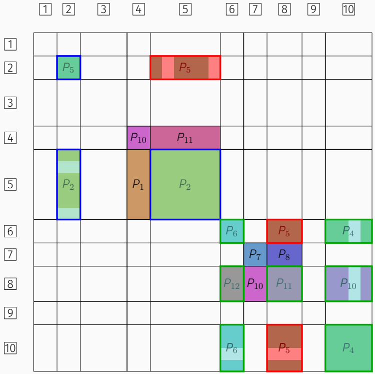
  $\mathcal{R}_\mathcal{K} \leftarrow$ non-zero rows in supernode $\mathcal{K}$

  $Y = S^{-1}_{\mathcal{R}_\mathcal{K},\mathcal{R}_\mathcal{K}} \ell_{\mathcal{R}_\mathcal{K},\mathcal{K}}$

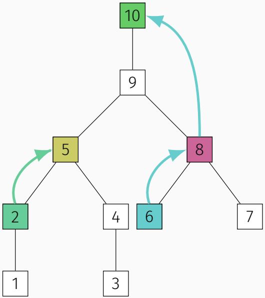
  $A_{\mathcal{K},\mathcal{K}} \leftarrow d^{-1} + Y^T \ell_{\mathcal{R}_\mathcal{K},\mathcal{K}}$

  $A_{\mathcal{R}_\mathcal{K},\mathcal{K}} \leftarrow -Y$

end



- Top-Down elimination tree traversal
- Exploit elimination tree to increase concurrency



- Serializations from common ancestors
- Serializations from layout

for *Supernode* $\mathcal{K} = \mathcal{N}$ *down to* 1 **do**

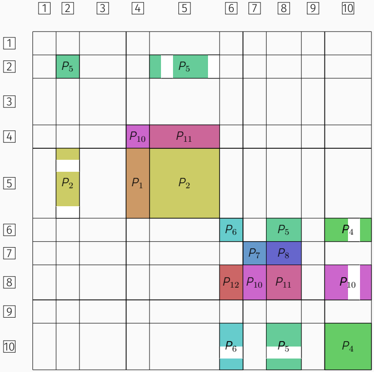
$\mathcal{R}_\mathcal{K} \leftarrow$ non-zero rows in supernode $\mathcal{K}$

$Y = S^{-1}_{\mathcal{R}_\mathcal{K},\mathcal{R}_\mathcal{K}} \ell_{\mathcal{R}_\mathcal{K},\mathcal{K}}$

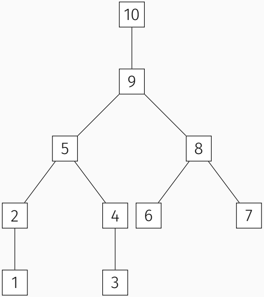
$A_{\mathcal{K},\mathcal{K}} \leftarrow d^{-1} + Y^T \ell_{\mathcal{R}_\mathcal{K},\mathcal{K}}$

$A_{\mathcal{R}_\mathcal{K},\mathcal{K}} \leftarrow -Y$

end



- Top-Down elimination tree traversal
- Exploit elimination tree to increase concurrency



- Serializations from common ancestors
- Serializations from layout

for *Supernode* $\mathcal{K} = \mathcal{N}$ *down to* 1 **do**

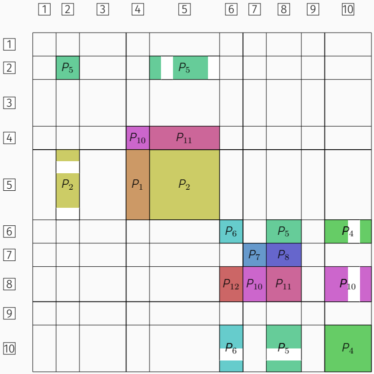
$\mathcal{R}_{\mathcal{K}} \leftarrow$ non-zero rows in supernode $\mathcal{K}$

$Y = S^{-1}_{\mathcal{R}_{\mathcal{K}},\mathcal{R}_{\mathcal{K}}} \ell_{\mathcal{R}_{\mathcal{K}},\mathcal{K}}$

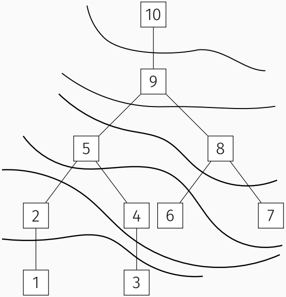
$A_{\mathcal{K},\mathcal{K}} \leftarrow d^{-1} + Y^T \ell_{\mathcal{R}_{\mathcal{K}},\mathcal{K}}$

$A_{\mathcal{R}_{\mathcal{K}},\mathcal{K}} \leftarrow -Y$

end



- Top-Down elimination tree traversal
- Exploit elimination tree to increase concurrency



- Serializations from common ancestors
- Serializations from layout
- How to schedule supernodes ?

CONCURRENCY BETWEEN SUPERNODES

for *Supernode* $\mathcal{K} = \mathcal{N}$ *down to* 1 **do**

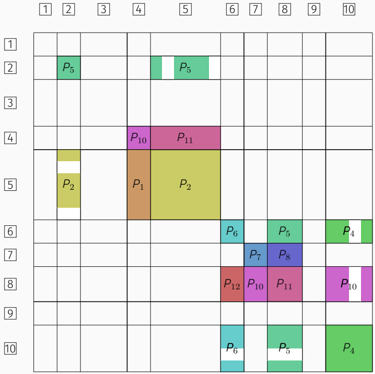
  $\mathcal{R}_{\mathcal{K}} \leftarrow$ non-zero rows in supernode $\mathcal{K}$

  $Y = S^{-1}_{\mathcal{R}_{\mathcal{K}},\mathcal{R}_{\mathcal{K}}} \ell_{\mathcal{R}_{\mathcal{K}},\mathcal{K}}$

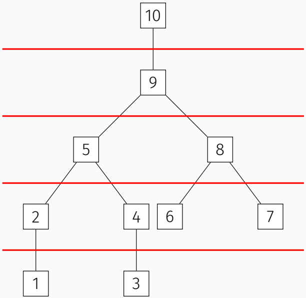
  $A_{\mathcal{K},\mathcal{K}} \leftarrow d^{-1} + Y^T \ell_{\mathcal{R}_{\mathcal{K}},\mathcal{K}}$

  $A_{\mathcal{R}_{\mathcal{K}},\mathcal{K}} \leftarrow -Y$

end



- Top-Down elimination tree traversal
- Exploit elimination tree to increase concurrency



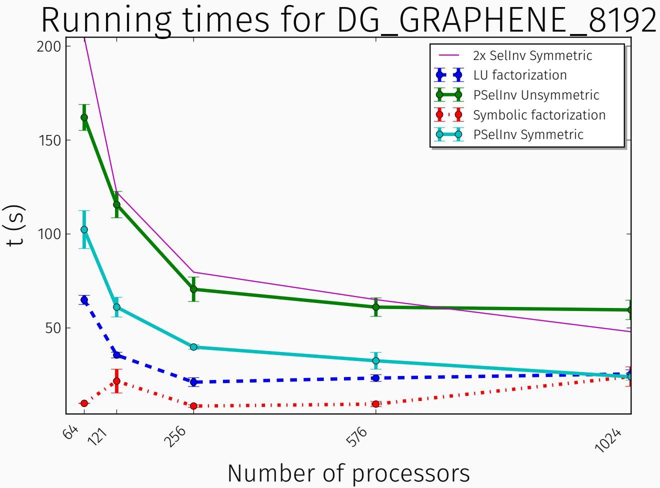
- Serializations from common ancestors
- Serializations from layout
- How to schedule supernodes ?
- Level-based heuristic as first step

- Experiments on NERSC Edison
  - Intel Ivy Bridge 2.4 GHz processors
  - 24 cores per node (two sockets)
  - 2.6 GB of memory per core
- `ParMETIS` used for matrix ordering
- `SuperLU_DIST` used for factorization
- `PSelInv`:
  - “Flat” MPI implementation
  - Only asynchronous P2P send/recv

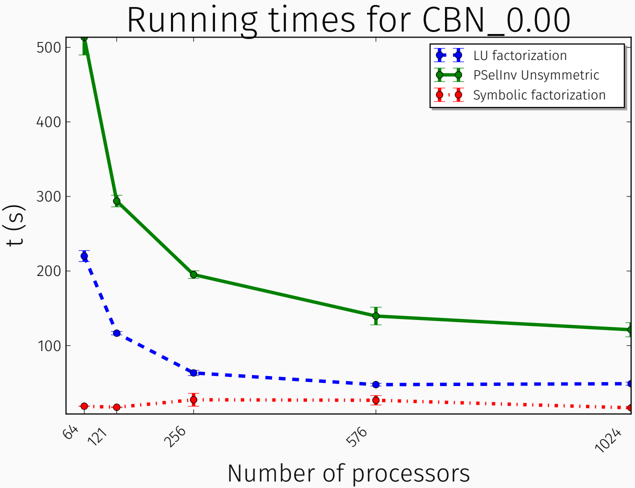
- Experiments on NERSC Edison
  - Intel Ivy Bridge 2.4 GHz processors
  - 24 cores per node (two sockets)
  - 2.6 GB of memory per core
- `ParMETIS` used for matrix ordering
- `SuperLU_DIST` used for factorization
- `PSelInv`:
  - "Flat" MPI implementation
  - Only asynchronous P2P send/recv

Focus on pipelining computations

STRONG SCALING: DG_GRAPHENE_8192

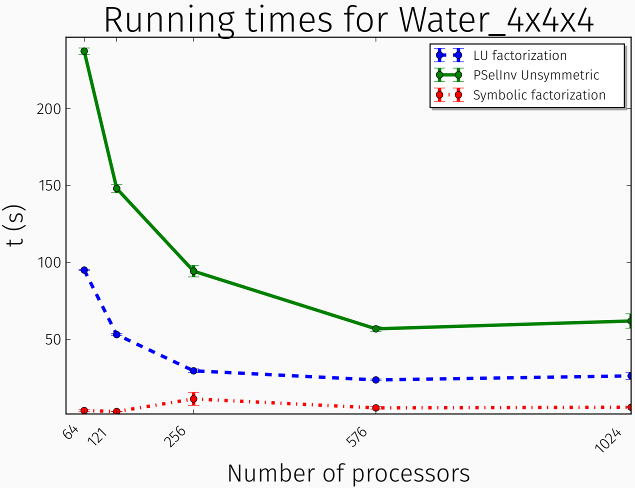


n=327,680 , nnz=238,668,800 , nnz(L+U)=1.9×10^9



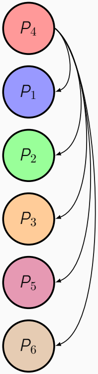
n=166,010 , nnz=251,669,372 , nnz(L+U)=2.8×10^9

STRONG SCALING: WATER_4X4X4

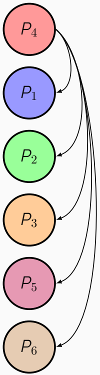


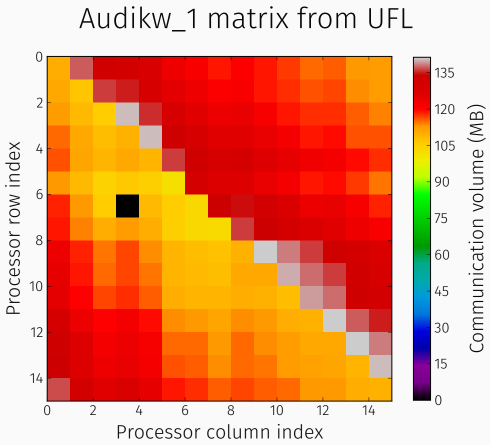
n=94,208 , nnz=32,706,432 , nnz(L+U)=1.3×10^9

Audikw_1 matrix from UFL



p steps

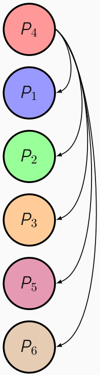


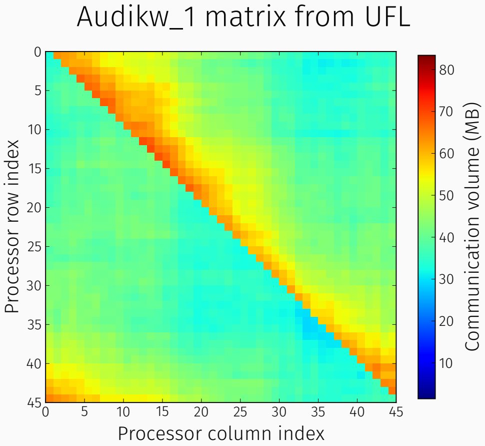


p steps

Col-Bcast using Flat-tree on 256 processors

Avg. volume: 120.06 MB, Std. dev.: 10.2%, Med.: 119.04 MB



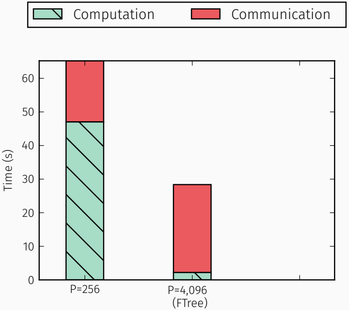


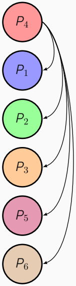
p steps

Col-Bcast using Flat-tree on 2116 processors

Avg. volume: 43 MB, Std. dev.: 19.2%, Med.: 40.8 MB

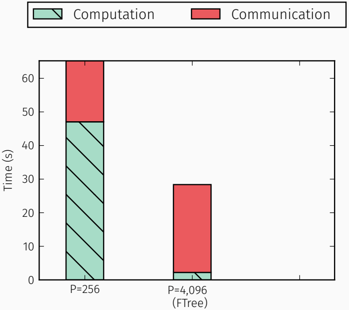
- "Flat-tree" communication pattern not efficient
- Restricted broadcast/reduce implemented by point-to-point cause imbalance

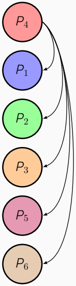




ENHANCING GROUP COMMUNICATIONS

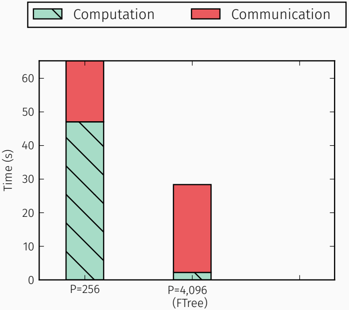
- “Flat-tree” communication pattern not efficient
- Restricted broadcast/reduce implemented by point-to-point cause imbalance

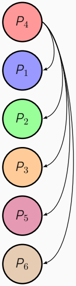




- Tree-based communication patterns
  - Asynchronous, non-collective

- “Flat-tree” communication pattern not efficient
- Restricted broadcast/reduce implemented by point-to-point cause imbalance

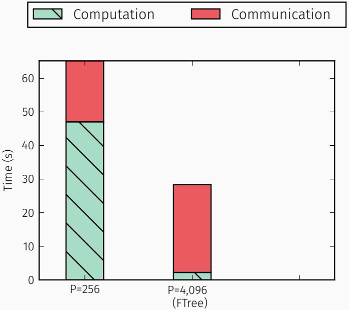


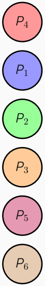


- Tree-based communication patterns
  - Asynchronous, non-collective
  - Structure is pre-allocated
  - Message sizes pre-computed

ENHANCING GROUP COMMUNICATIONS

- "Flat-tree" communication pattern not efficient
- Restricted broadcast/reduce implemented by point-to-point cause imbalance

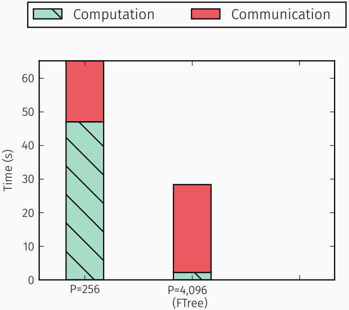


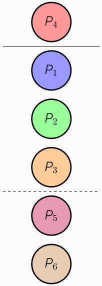


- Tree-based communication patterns
  - Asynchronous, non-collective
  - Structure is pre-allocated
  - Message sizes pre-computed
- Binary tree

ENHANCING GROUP COMMUNICATIONS

- “Flat-tree” communication pattern not efficient
- Restricted broadcast/reduce implemented by point-to-point cause imbalance

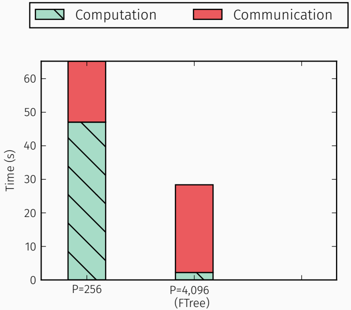


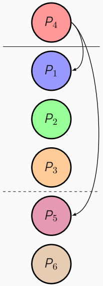


- Tree-based communication patterns
  - Asynchronous, non-collective
  - Structure is pre-allocated
  - Message sizes pre-computed
- Binary tree

ENHANCING GROUP COMMUNICATIONS

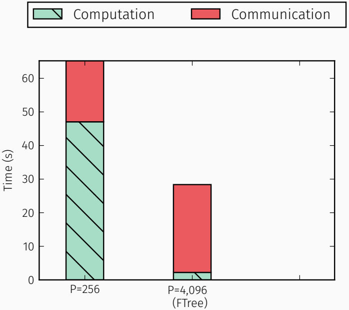
- “Flat-tree” communication pattern not efficient
- Restricted broadcast/reduce implemented by point-to-point cause imbalance

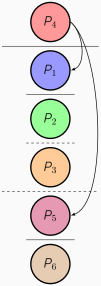




- Tree-based communication patterns
  - Asynchronous, non-collective
  - Structure is pre-allocated
  - Message sizes pre-computed
- Binary tree

- "Flat-tree" communication pattern not efficient
- Restricted broadcast/reduce implemented by point-to-point cause imbalance

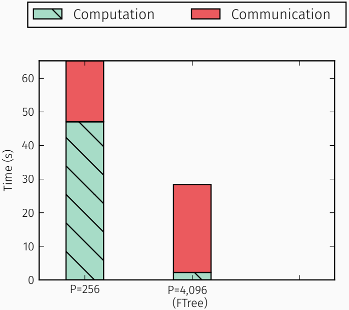


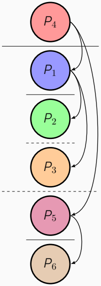


- Tree-based communication patterns
  - Asynchronous, non-collective
  - Structure is pre-allocated
  - Message sizes pre-computed
- Binary tree

ENHANCING GROUP COMMUNICATIONS

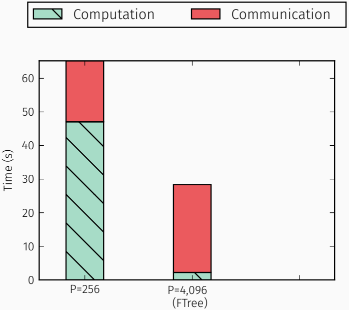
- "Flat-tree" communication pattern not efficient
- Restricted broadcast/reduce implemented by point-to-point cause imbalance

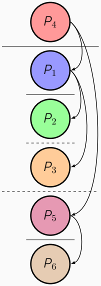




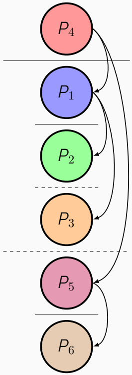
- Tree-based communication patterns
  - Asynchronous, non-collective
  - Structure is pre-allocated
  - Message sizes pre-computed
- Binary tree

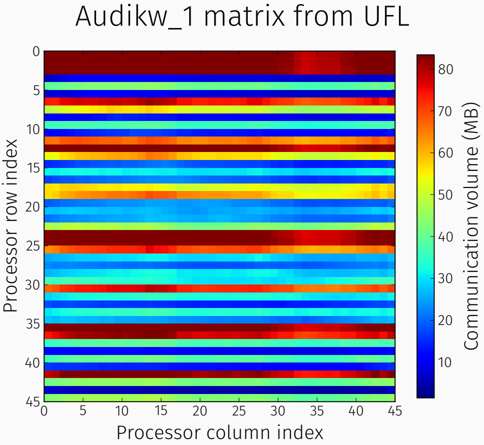
- "Flat-tree" communication pattern not efficient
- Restricted broadcast/reduce implemented by point-to-point cause imbalance





- Tree-based communication patterns
  - Asynchronous, non-collective
  - Structure is pre-allocated
  - Message sizes pre-computed
- Binary tree
- $\log(p)$ cost

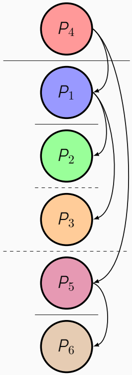


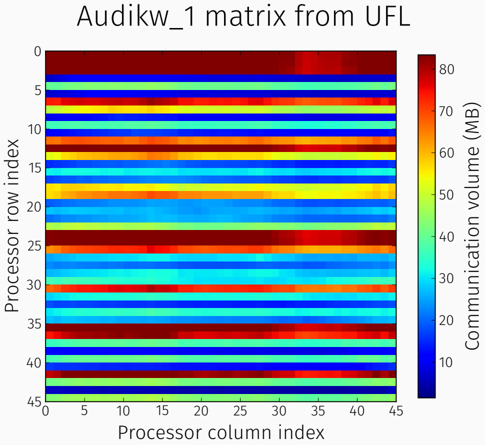


$\log(p)$ steps

Col-Bcast using Binary-tree on 2116 processors

Avg. volume: 42.96 MB, Std. dev.: 63%, Med.: 36.9 MB



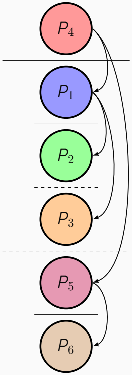


log(p) steps

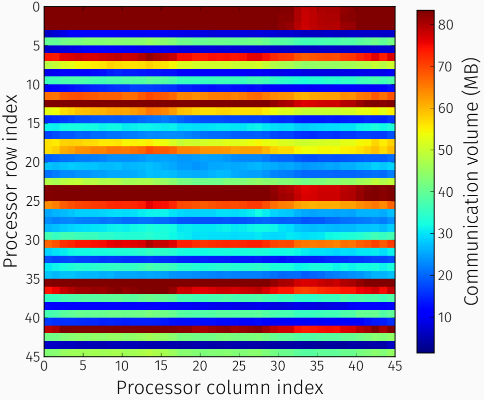
Col-Bcast using Binary-tree on 2116 processors

Avg. volume: 42.96 MB, Std. dev.: 63%, Med.: 36.9 MB

"Striped" pattern ⇔ imbalance



Audikw_1 matrix from UFL



log(p) steps

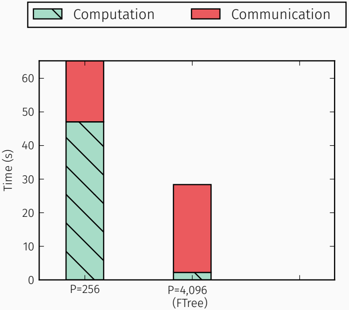
Col-Bcast using Binary-tree on 2116 processors

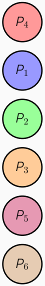
Avg. volume: 42.96 MB, Std. dev.: 63%, Med.: 36.9 MB

"Striped" pattern $\Leftrightarrow$ imbalance

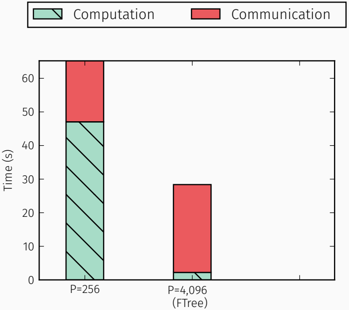
Same nodes are picked to forward data

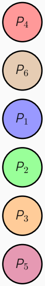
- "Flat-tree" communication pattern not efficient
- Restricted broadcast/reduce implemented by point-to-point cause imbalance





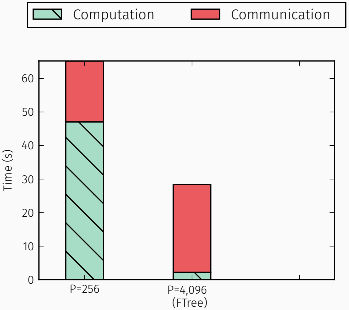
- “Flat-tree” communication pattern not efficient
- Restricted broadcast/reduce implemented by point-to-point cause imbalance

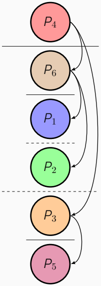




- Random circular shift

- "Flat-tree" communication pattern not efficient
- Restricted broadcast/reduce implemented by point-to-point cause imbalance

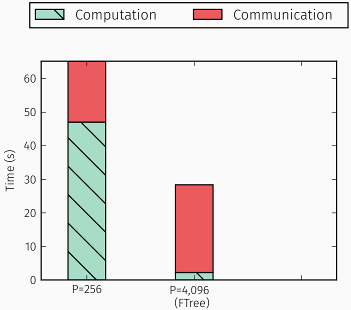


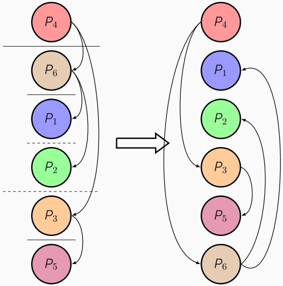


- Random circular shift
- Binary tree

ENHANCING GROUP COMMUNICATIONS

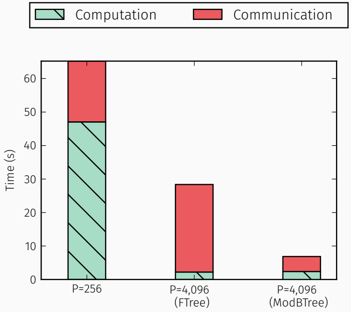
- "Flat-tree" communication pattern not efficient
- Restricted broadcast/reduce implemented by point-to-point cause imbalance

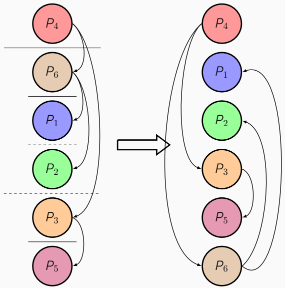




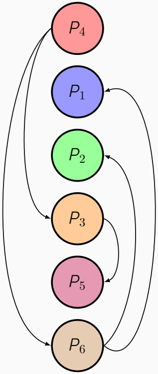
- Random circular shift
- Binary tree

- “Flat-tree” communication pattern not efficient
- Restricted broadcast/reduce implemented by point-to-point cause imbalance



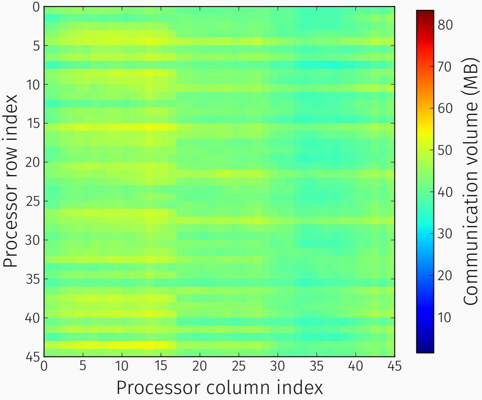


- Random circular shift
- Binary tree



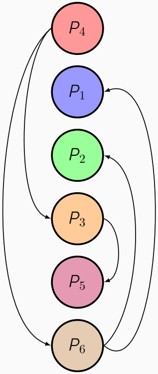
log(p) steps

Audikw_1 matrix from UFL



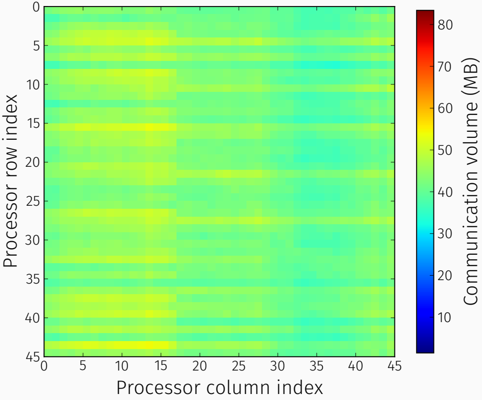
Col-Bcast using Shifted Binary-tree on 2116 processors

Avg. volume: 42.96 MB, Std. dev.: 7.7%, Med.: 42.6 MB



log(p) steps

Audikw_1 matrix from UFL

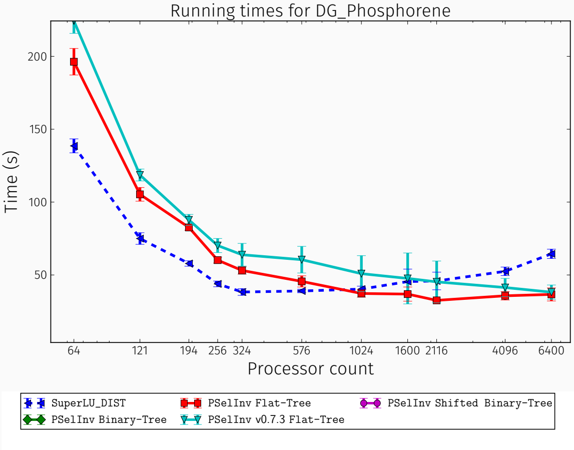


Col-Bcast using Shifted Binary-tree
on 2116 processors

Avg. volume: 42.96 MB, Std. dev.: 7.7%, Med.: 42.6 MB

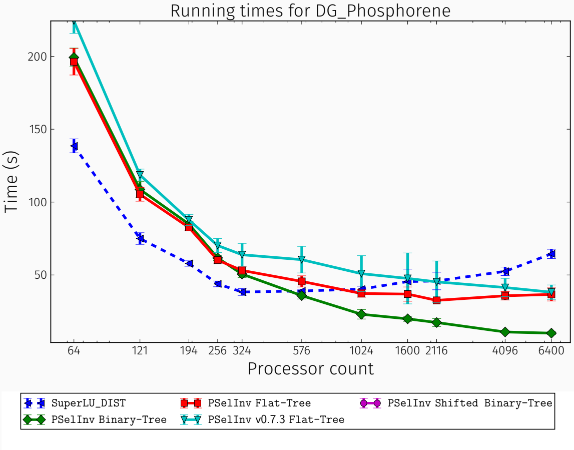
Volume of communication is now balanced

STRONG SCALING: DG_PHOSPHORENE (DGDFT)



n=512,000 , nnz=550,400,000 , nnz(L+U)=3.7×10^9

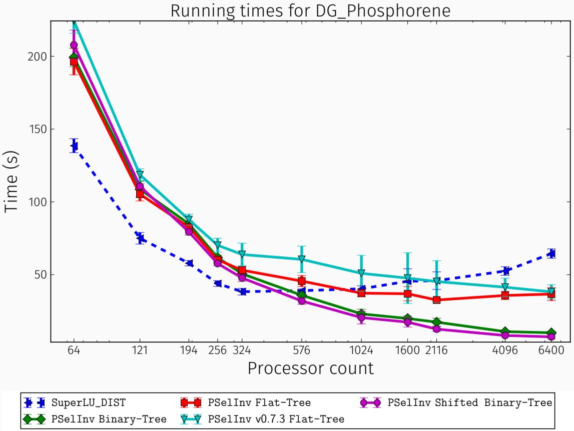
STRONG SCALING: DG_PHOSPHORENE (DGDFT)



n=512,000 , nnz=550,400,000 , nnz(L+U)=3.7×10^9

Binary-tree allows a better pipeline

STRONG SCALING: DG_PHOSPHORENE (DGDFT)



n=512,000 , nnz=550,400,000 , nnz(L+U)=3.7×10^9

Binary-tree allows a better pipeline

Shifted Binary-tree balance load, preserving pipeline

Conclusions:

- Parallel Selected Inversion for Unsymmetric matrices
- Trace computations may need transpose
- Pipelining and asynchronous task execution model are critical for performance
- Communication load has a severe impact on performance
- Faster than more general inversion algorithms

Conclusions:

- Parallel Selected Inversion for Unsymmetric matrices
- Trace computations may need transpose
- Pipelining and asynchronous task execution model are critical for performance
- Communication load has a severe impact on performance
- Faster than more general inversion algorithms
- `PSelInv` available in the PEXSI library

  `http://www.pexsi.org/`
- PEXSI used in many packages and SciDAC projects:

  SIESTA, CP2K, DGDFT (LBL), ELSI (NSF)

Future work:

- Concurrent supernodes scheduling
- Finer granularity tasks
- Remove need for duplicating data
- Tree-based communications for unsymmetric case
- Hybrid MPI/ OpenMP
- Dynamic task scheduling