Blocking Strategy Optimization for Sparse Direct Linear Solvers on Heterogeneous Architectures

June 12th, 2015 - SOLHAR Meeting

Mathieu Faverge, Grégoire Pichon, Pierre Ramet, Jean Roman

Outline

1

Sparse Direct Solvers

Context

Problem: solve $Ax = b$

- Cholesky: factorize $A = LL^T$ (symmetric pattern $(A + A^T)$ for LU)
- Solve $Ly = b$
- Solve $L^T x = y$

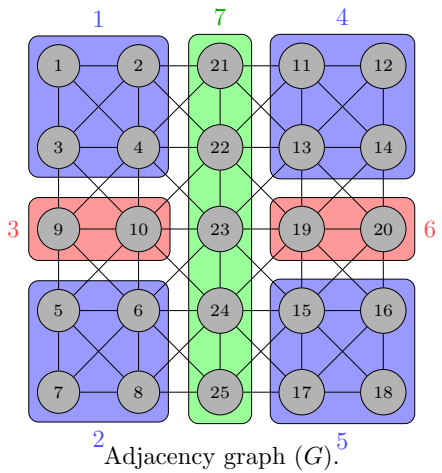
Sparse Direct Solvers: PaStiX approach

1. Order unknowns to minimize the fill-in
2. Compute a symbolic factorization to build L structure
3. Factorize the matrix in place on L structure
4. Solve the system with forward and backward triangular solves

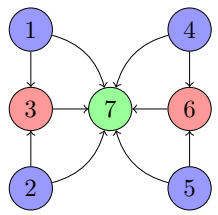
Symbolic Factorization

General approach

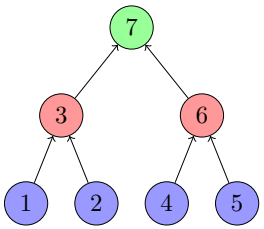
1. Use the nested dissection process to partition a sparse matrix
2. Use the minimum degree solution when leaves are small enough
3. Order a supernode thanks to the Reverse Cuthill-McKee algorithm



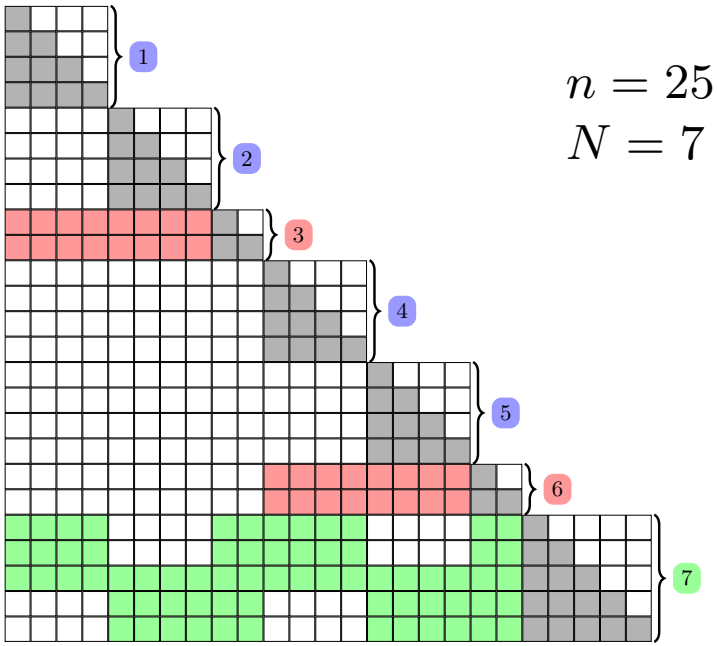
Adjacency graph (G).



Quotient graph (G^*/P) $= (G/P)^*$



Elimination tree (T).



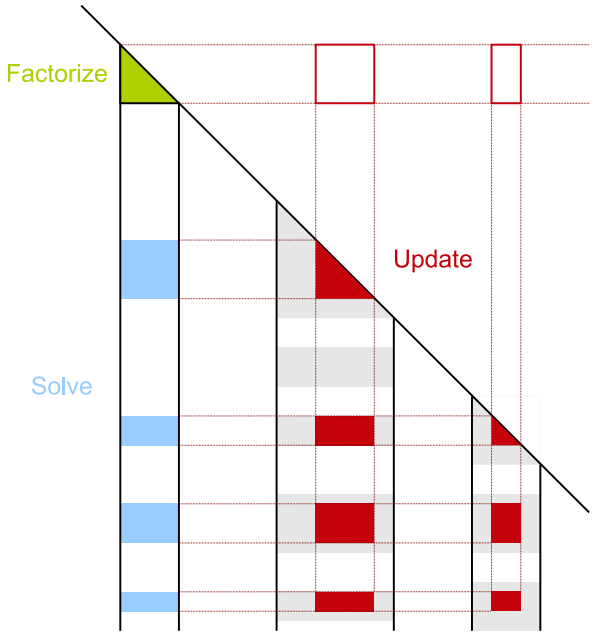
Factorized matrix (L).

$$n = 25$$
$$N = 7$$

Numerical Factorization

Algorithm to eliminate the block column k

1. Factorize the diagonal block
2. Solve off-diagonal blocks in the current column (TRSM)
3. Update the underlying matrix with the column's contribution (GEMM)



Update

- Compacted matrix-matrix product
- Update divided into the number of off-diagonal blocks receiving contributions

Motivations

Clustering techniques

- Operations on data blocks are more efficient
- Preprocessing stages on the matrix structure before numerical operations
- Those steps can be used for several systems presenting the same initial structure, or for several right-hand-sides

Objectives

- Increase BLAS efficiency by reducing the number of off-diagonal blocks
- Reduce RUNTIME overhead, with larger tasks

The number of non-zeros, as well as the number of operations, is kept the same

2

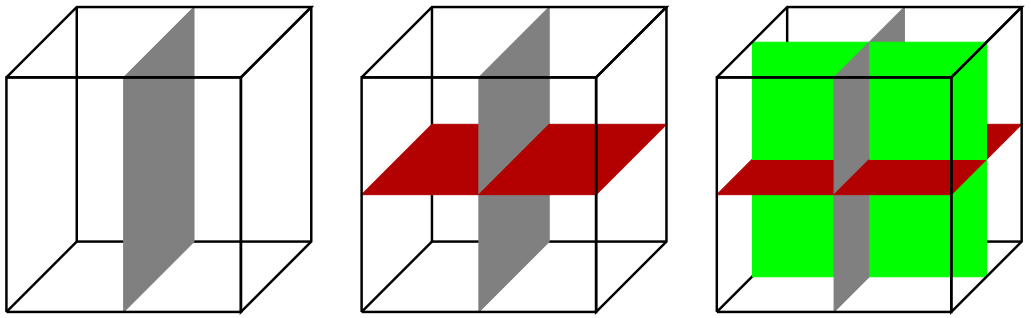
Supernode Ordering Problem

Nested Dissection (1)

Considering a graph $G = (V, E, \sigma_p)$
V: vertexes, E: edges, σ_p: unknowns permutation

Algorithm to compute σ_p

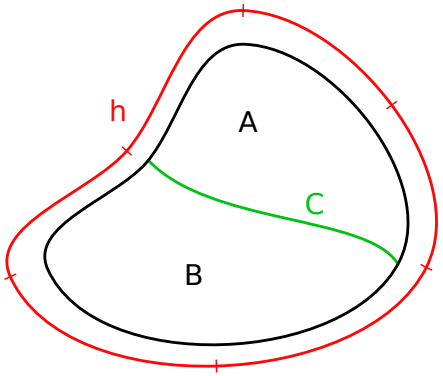
1. Partition $V = A \cup B \cup C$
2. Order C with larger numbers
3. Apply the process recursively on A and B



Three-levels of nested dissection on a regular cube

Nested Dissection (2)

Considering a subgraph appearing the nested dissection process, with $|V| = p$



p^σ-separation theorem

- $0 < \alpha < 1, \beta > 0, \frac{1}{2} \leq \sigma < 1$
- A and B do not interact
- $|A| \leq \alpha p$, $|B| \leq \alpha p$
- $|C| \leq \beta p^\sigma$

Characterisation theorem

Fill-in element in position (i, j) if it exists a path from i to j that only goes through vertexes with a lower number than i and j.

Related Work: Reverse Cuthill-McKee (RCM) algorithm

Off-diagonal blocks

The number of off-diagonal blocks contributing to a supernode corresponds to the number of sequences ordered consecutively on the supernode halo

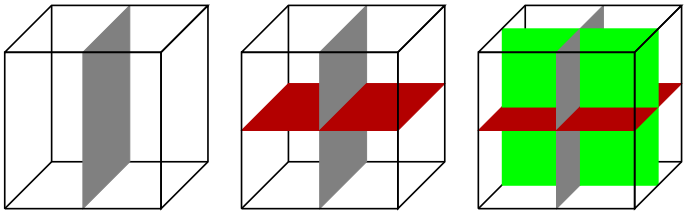
General idea - Breadth-First Search

1. Choose a peripheral vertex x, ordered as first vertex
2. Order vertexes interacting with x (neighbourhood at distance d)
3. Iterate starting with those vertexes (neighbourhood at distance $d+1$)

Drawbacks

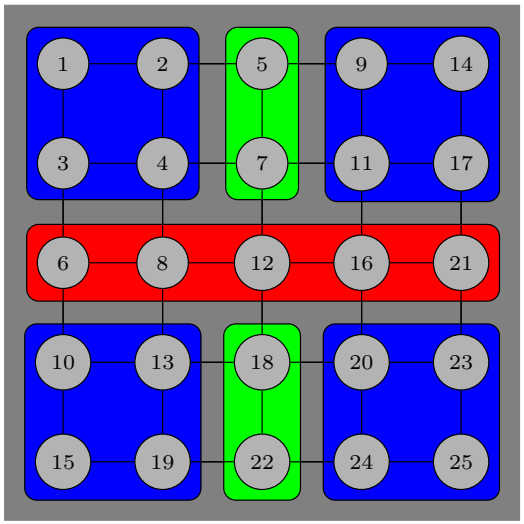
- Work on A structure instead of L structure
- Do not consider contributing supernodes, but only intra-node interactions
- Order supernodes during the nested dissection process while it could be realized after

Ordering Last Supernode

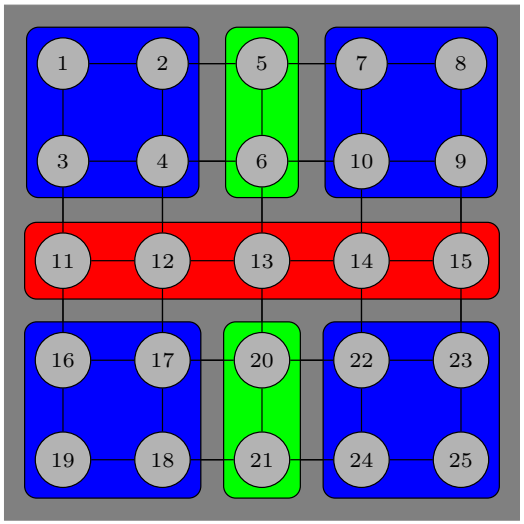


Example

- $n = 5 \times 5 \times 5$
- $N = 1 + 2 + 4 + 8 = 15$
- First separator of size 5×5



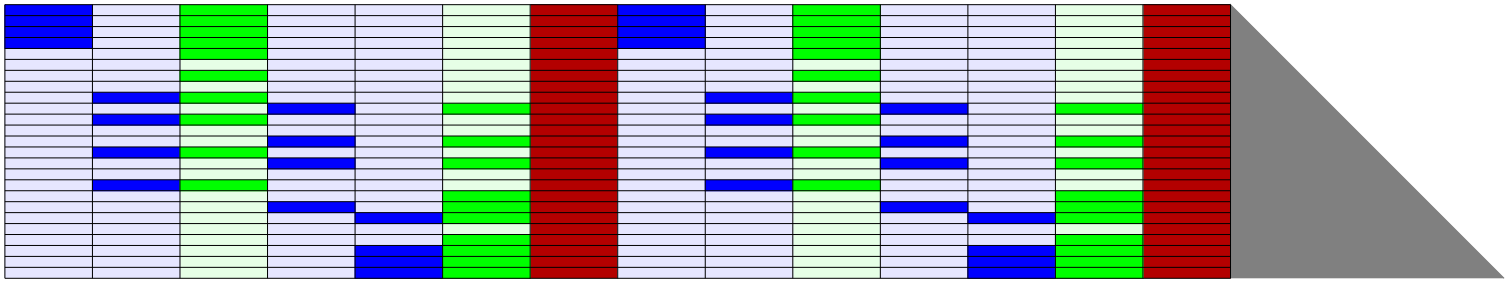
RCM



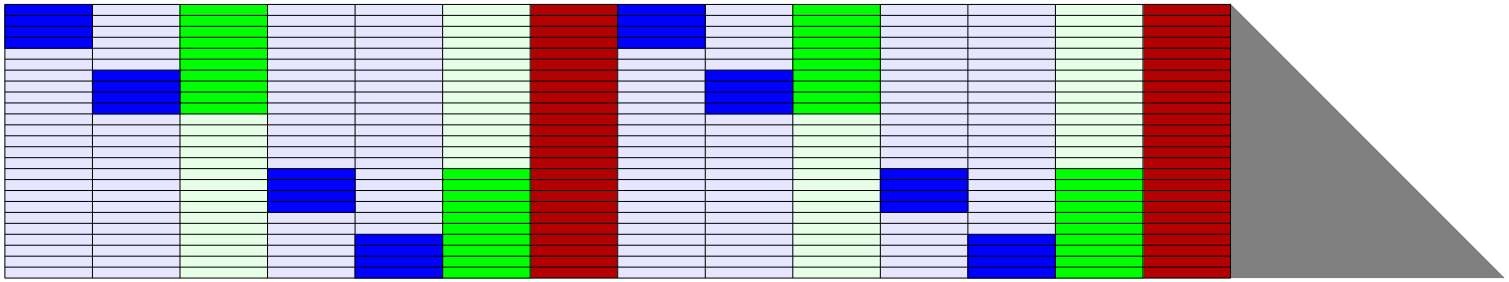
Optimal

Projection of contributing supernodes and ordering of the first separator

Resulting Symbolic Structures

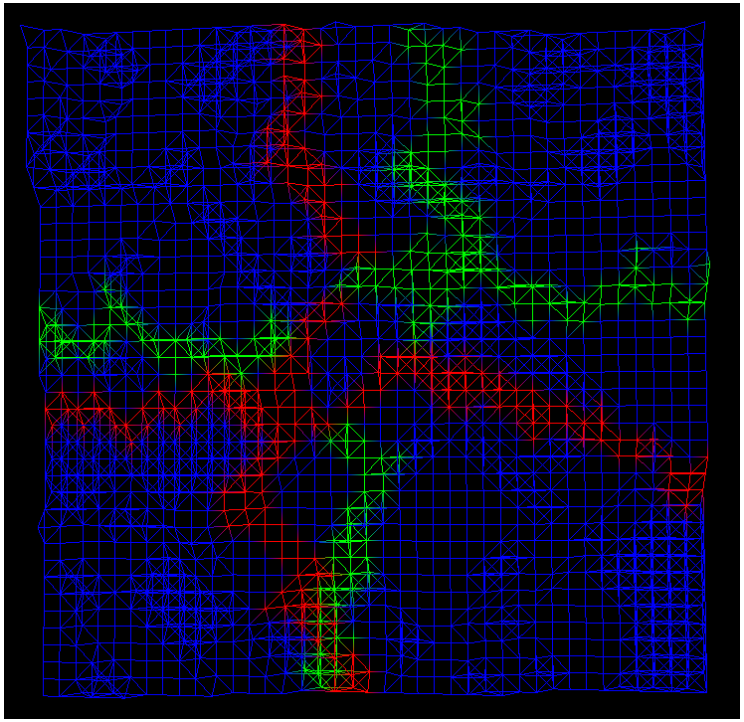


Symbolic Factorization: RCM



Symbolic Factorization: Optimal

Practical Ordering with Scotch



Last supernode of a 3D Laplacian (size 40)

Subparts A and B are partitioned differently. Thus, supernodes projection is less regular than in our previous example

3

Problem Modelization & Proposed Solution

Formalization of the Problem (1)

Notations

- We consider the ℓ^{th} diagonal block C_ℓ
- Contributing supernodes are included in $(C_k)_{k \in [1, \ell-1]}$

Supernode contributions to C_ℓ

$$row_{ik}^{\ell} = \begin{cases} 1 \text{ if vertex } i \text{ from } C_\ell \text{ is connected to } C_k \\ 0 \text{ otherwise} \end{cases}, k \in [1, \ell-1], i \in [1, |C_\ell|]$$

Set of contributions for line i:
$B_i^\ell = (row_{ik}^\ell)_{k \in [1, \ell-1]}$

Formalization of the Problem (2)

Metrics

- Weight of line i

$$w_i^\ell = \sum_{k=1}^{\ell-1} row_{ik}^\ell$$

- Distance between lines i and j

$$d_{i,j}^\ell = d(B_i^\ell, B_j^\ell) = \sum_{k=1}^{\ell-1} row_{ik}^\ell \oplus row_{jk}^\ell$$

with $\oplus$ the exclusive or operation.
measure the number of blocks created by line j and ended at line i

Formalization of the Problem (3)

Number of off-diagonal blocks

$$odb^{\ell} = \frac{1}{2}(w_1^{\ell} + \sum_{i=1}^{|C_{\ell}|-1} d_{i,i+1}^{\ell} + w_{|C_{\ell}|}^{\ell})$$

Optimal solution

- Minimize odb^{ℓ}
- Shortest Hamiltonian Path problem: find the shortest path visiting once each line, with a constraint on first and last line
- Complete symmetric graph

Proposition

Traveller Salesman Problem

- Find a cycle minimizing

$$\sum_{i=1}^{|C_\ell|} d^\ell_{i,(i+1)[|C_\ell|]}$$

- Add a fictive vertex S_0, without any contribution to build a cycle instead of a path

Algorithm

1. Build the set B^ℓ_i for each line i of C^ℓ
2. Compute the distance matrix
3. Insert lines to minimize the cycle length
4. Split the cycle at fictive vertex to get the path

Build Sets of Contributing Supernodes

Sparse properties

- We rely on the sparse properties of L
- Store for each line contributing supernodes only: $row_{ik}^{\ell} = 1$
- Direct accesses thanks to the symbolic structure
- Linear in the size of the symbolic structure
- Set of contributing supernodes with increasing supernodes number

for each line i in the supernode C_ℓ **do**
  for each contributing node $C_{k \in [1, \ell-1]}$ **do**
    Append k to the set of contributing supernodes to line i
  end for
end for

Compute the Distance Matrix

Computing a distance

- Compare sets B_i^ℓ and B_j^ℓ
- Advance progressively in each set, and exploit the fact that contributing supernodes are stored in a increasing fashion
- Complexity in $\Theta(|B_i^\ell| + |B_j^\ell|)$

for each line i in the supernode C_ℓ **do**
    for each line j in the supernode C_ℓ **do**
        Compute the distance between lines i and j
    end for
end for

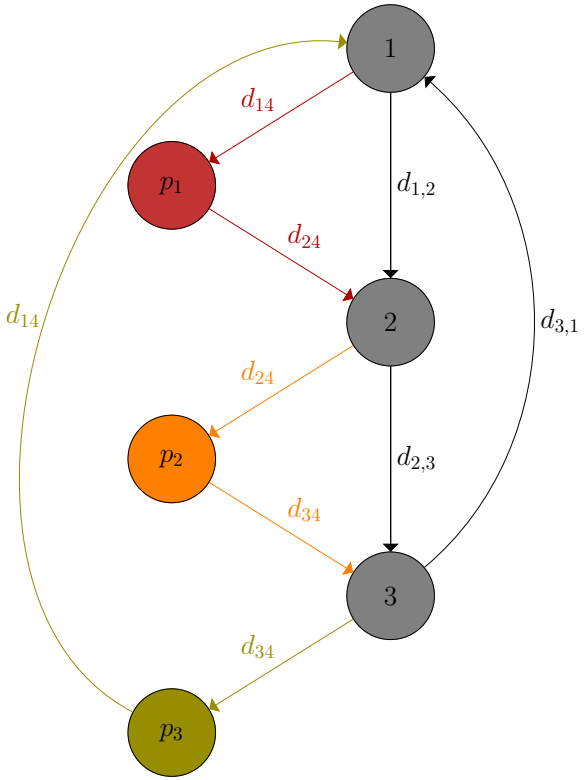
Build the New Lines Permutation (1)

Inserting a line

- Start with fictive vertex S_0
- Search for the position which will minimize the cycle
- The optimal solution is a NP-hard problem
- Heuristic: relative position between lines 1 to $i-1$ cannot evolve when inserting line i

$Cycle^{\ell} = \{S_0, 1\}$
for $i \in [2, |C_\ell|]$ **do**
    Insert row i in $Cycle^{\ell}$ to minimize the cycle length
end for

Build the New Lines Permutation (2)

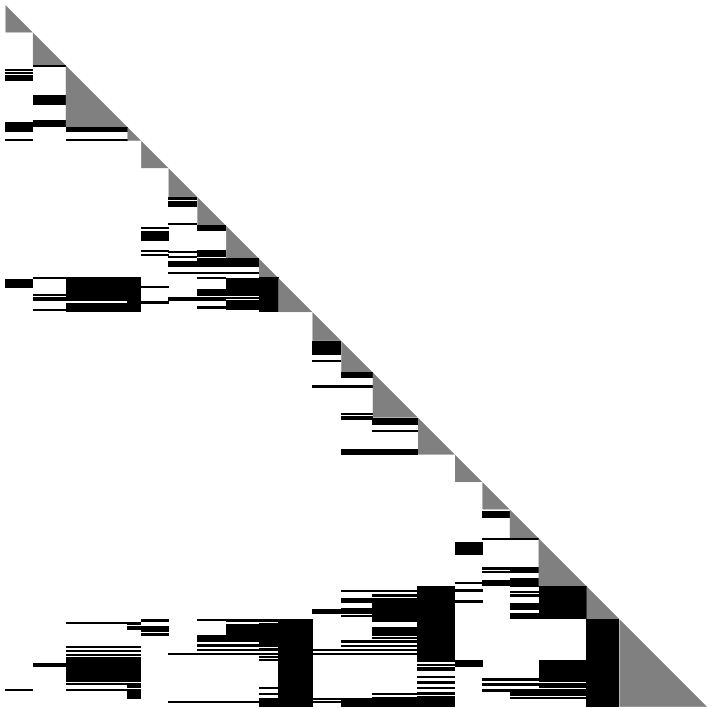


Possible positions to insert a fourth row

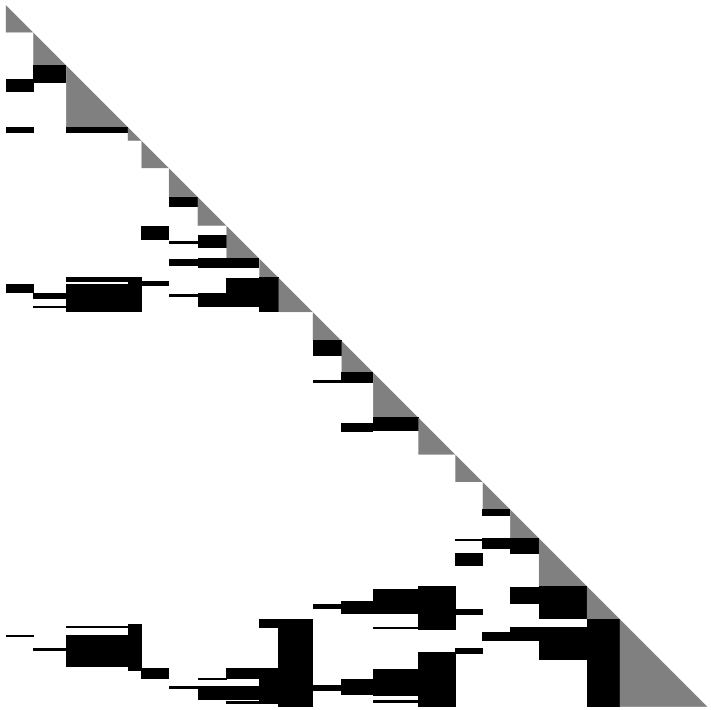
Inserting line i

- Find the position k to minimize $d^{\ell}_{i,k} + d^{\ell}_{i,k+1} - d^{\ell}_{k,k+1}$
- For identical position, minimize the cut *i.e.* $min(d^{\ell}_{i,k}, d^{\ell}_{i,k+1})$ between positions k leading to the same cycle length

Resulting Solution - Example



Without reordering



With reordering

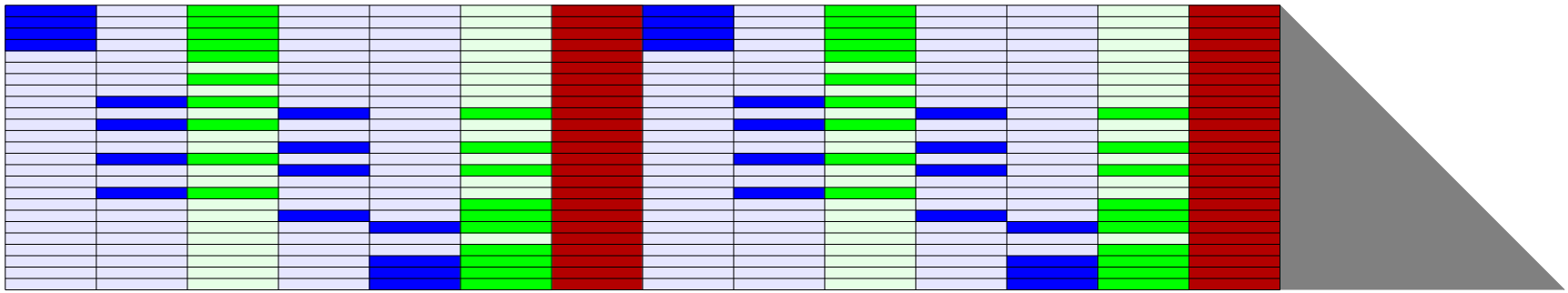
Reordering on a $8 \times 8 \times 8$ laplacian

Works whatever is the initial seed

Complexity (1)

Context

- For graphs respecting a n^σ-separation theorem
- Building the sets of contributing supernodes: $\Theta(n)$
- Inserting lines in C_l: $\Theta(|C_l|^2)$
- We study the complexity of the distance matrix computation



Considering supernode C_ℓ, each contributing line is used $(|C_\ell| - 1)$ times to compute distance with each other line

Complexity (2)

$$\mathcal{C} = \sum_{\ell=1}^{N} (\sum_{i=1}^{|C_\ell|} (row_{ik}^{\ell})_{k \in [1,\ell-1]} \times (|C_\ell| - 1))$$

Theorem (for meshes with balanced outer/inner contributions)

The number of off-diagonal blocks in the symbolic structure is in $\Theta(n)$. The demonstration considered off-diagonal lines instead of off-diagonal blocks, so the number of off-diagonal lines is in $\Theta(n)$ too.

$$\forall k \in [1, N], |C_k| \leq \alpha \times |C_N| = \Theta(n^\sigma)$$

$$\mathcal{C} \leq \Theta(n^\sigma) \times \underbrace{\sum_{\ell=1}^{N} (\sum_{i=1}^{|C_\ell|} (row_{ik}^{\ell})_{k \in [1,\ell-1]})}_{\Theta(n)} = \Theta(n^{\sigma+1})$$

Complexity (3)

Results

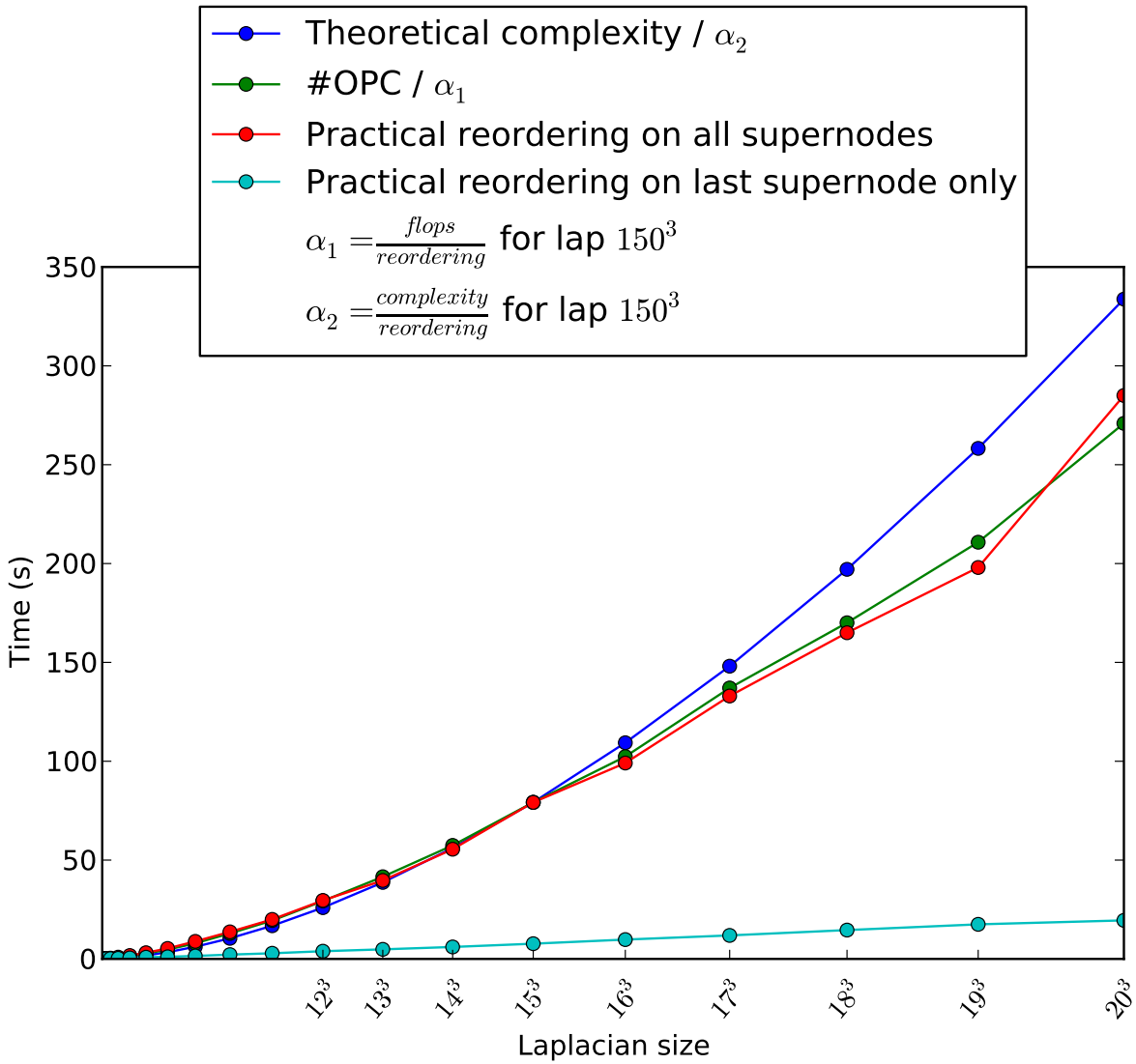
- Numerical factorization in $\Theta(n^{3\sigma})$
- Reordering in $\Theta(n^{\sigma+1})$

Type	σ	Reordering	Factorization
2D	$\frac{1}{2}$	$\Theta(n\sqrt{n})$	$\Theta(n\sqrt{n})$
3D	$\frac{2}{3}$	$\Theta(n^{\frac{5}{3}})$	$\Theta(n^2)$

Complexity for regular meshes

Asymptotically faster than the numerical factorization for $\sigma > \frac{1}{2}$
Remind that RCM is well working in 2D case

Complexity (4)



4

Benchmarks

Experimental Conditions

Set of matrices

- Large matrices, around 1 million unknowns (real-life meshes)
- Extracted from different applications
- Different average off-diagonal block size

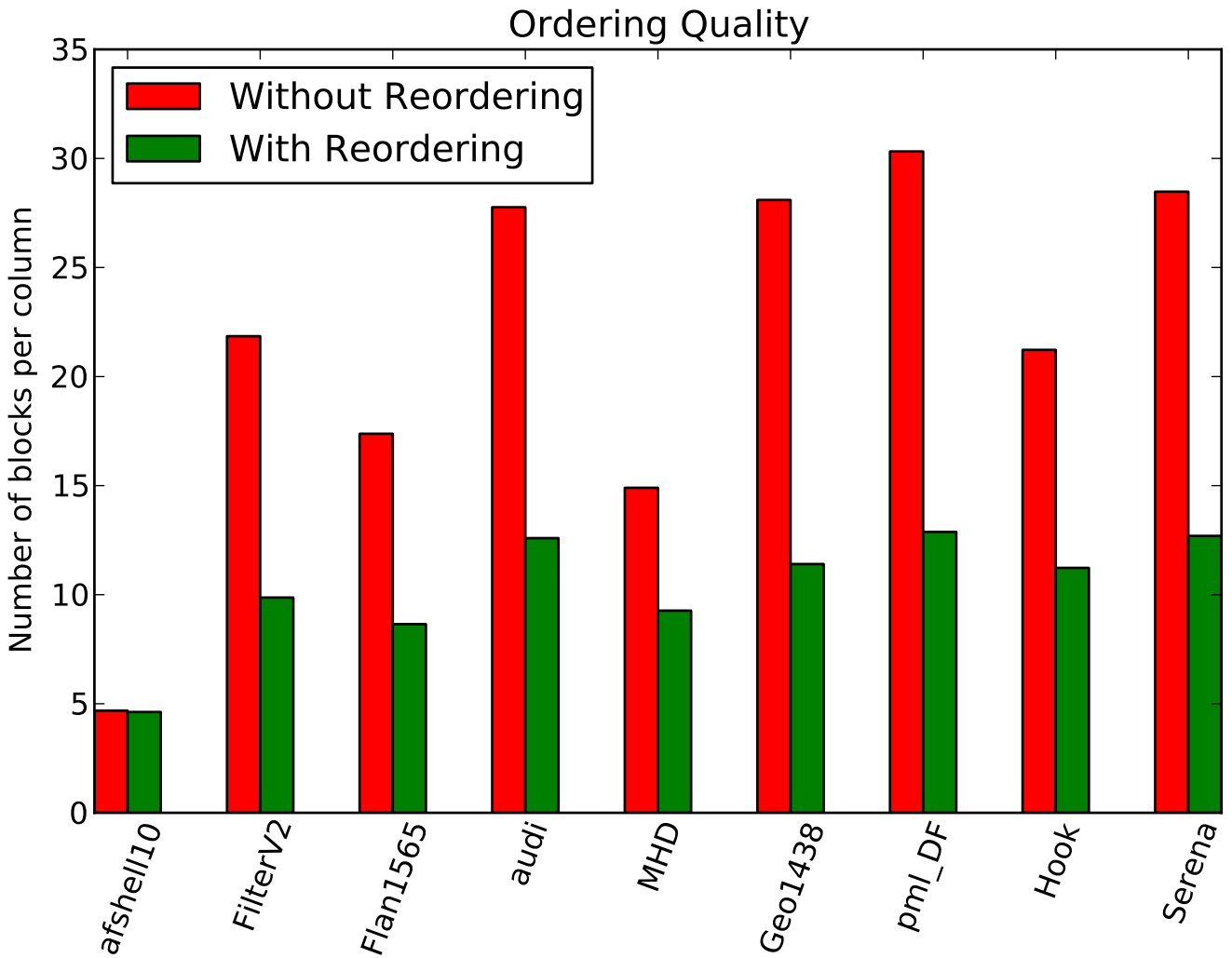
Machine - Curie TGCC

- Two quadcore INTEL `Westmere` running at 2.66 GHz
- Two NVIDIA `M2090 T20A`
- MKL `14.0.3.174`
- Cuda `5.5.22`

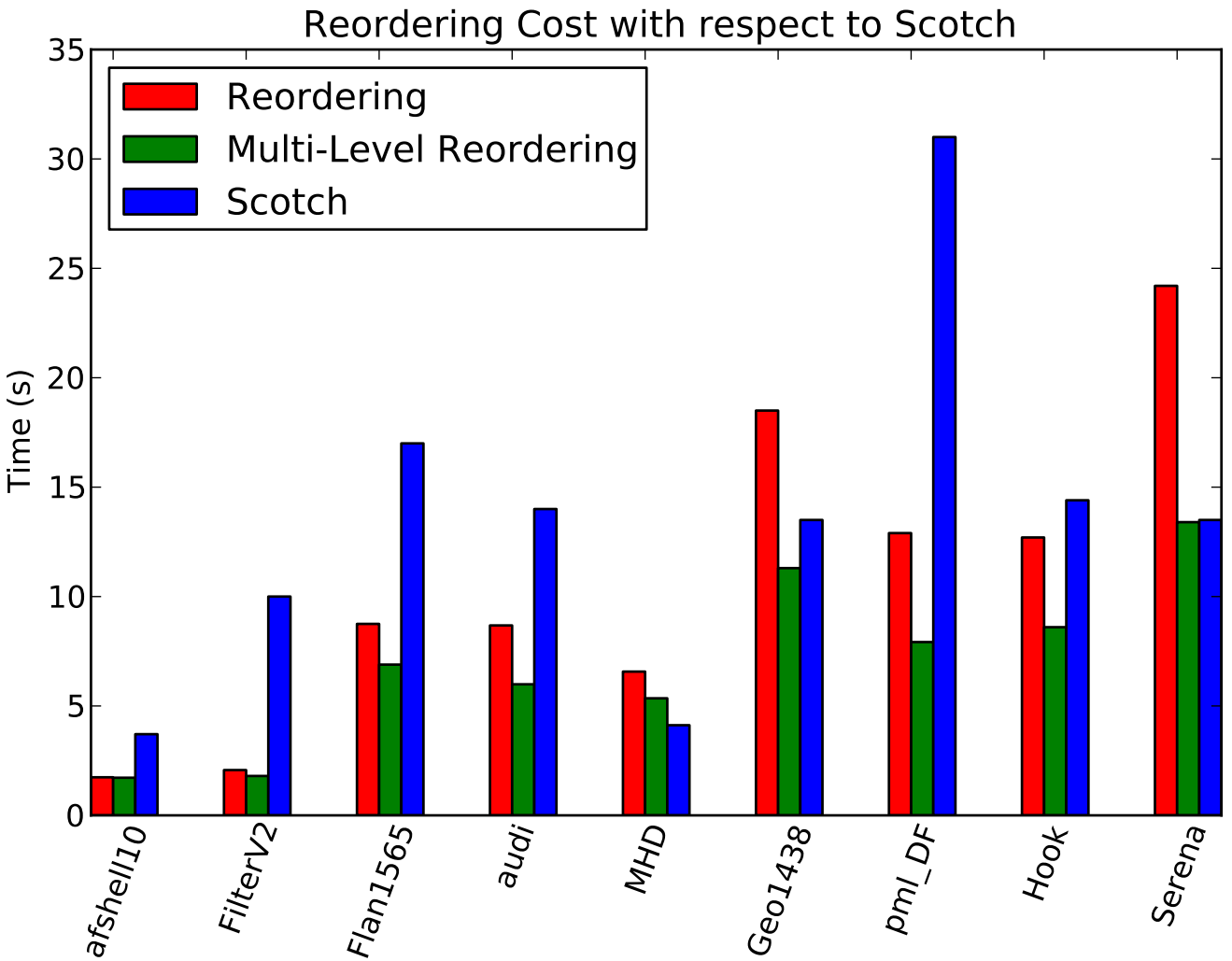
PaStiX

- Use StarPU implementation
- 6 threads + 2 GPUs
- Large minimum block size for GPUs efficiency

Tests on Large Matrices - Number of Off-Diagonal Blocks

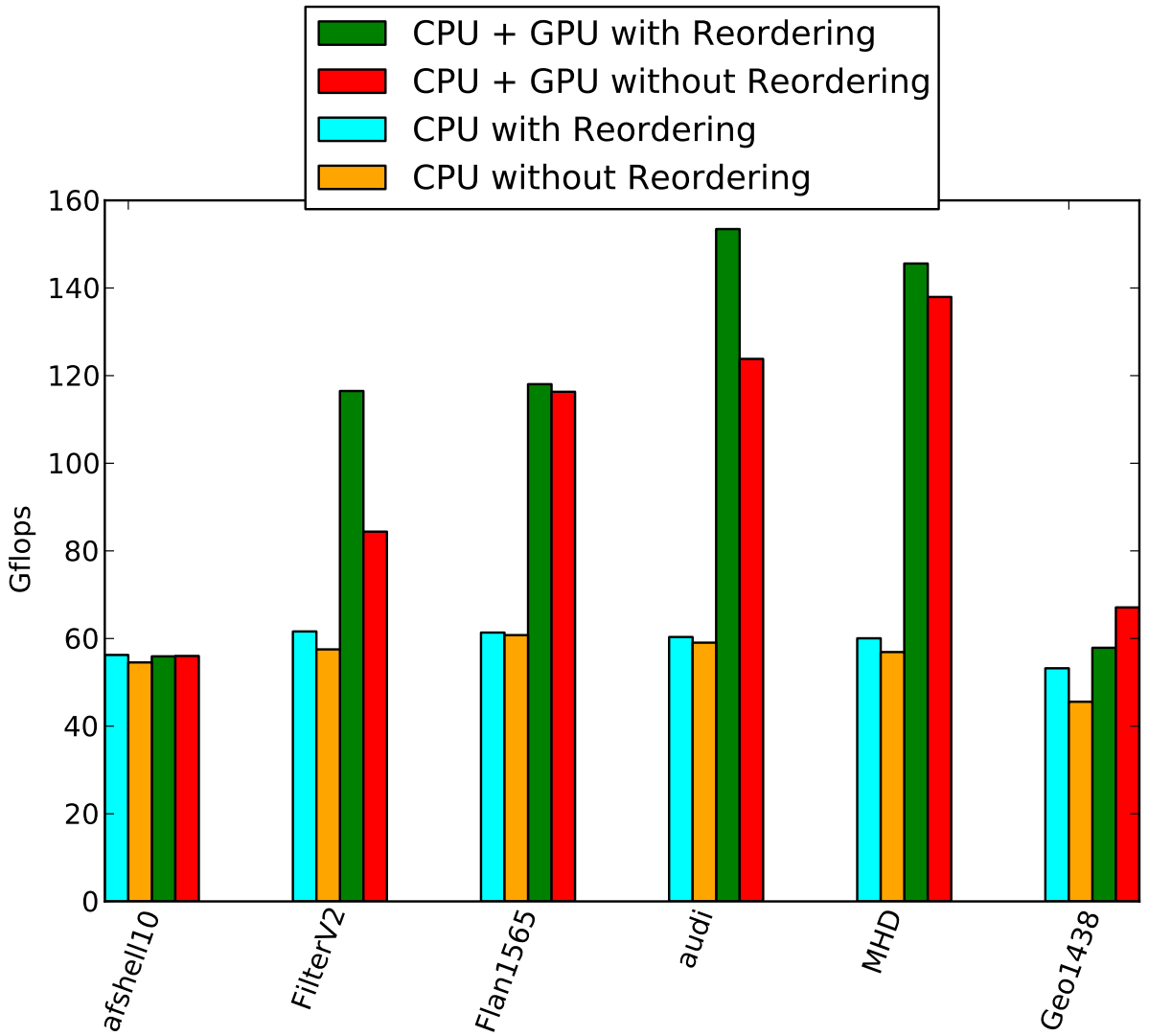


Tests on Large Matrices - Reordering Cost (sequential)



Performance on Curie (using 8 CPUs and 2 GPUs)

For the factorization only



Conclusion

Results

- Number of off-diagonal blocks reduced by a factor between 2 and 3
- Performance gain of the factorization up to 20% in an heterogeneous context
- Theoretical and practical reordering complexity small with respect to the numerical factorization for 3D graphs
- Works whatever is the initial seed

Perspectives

- Study such a strategy for a multifrontal solver (MUMPS)
- Implement the algorithm in a parallel context

Thanks !