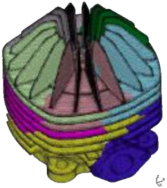
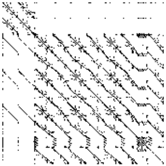
# Block Low-Rank multifrontal solvers: complexity, performance, and scalability

Théo Mary
Université de Toulouse

Ph.D. defense, 24 November 2017

# Context

 →  → 

### Linear system $Ax = b$

Often a keystone in scientific computing applications (discretization of PDEs, step of an optimization method, ...)

### Matrix sparsity

A sparse matrix is "any matrix with enough zeros that it pays to take advantage of them" (Wilkinson)

### Large-scale systems

Increasingly large numbers of cores available, need to efficiently make use of them

# Iterative vs direct methods

## Iterative methods

Build sequence $x_k$ converging towards $x$

☺ Computational cost: $\mathcal{O}(n)$ operations/iteration and memory

☹ Convergence is application-dependent

## Direct methods

Factorize $A = LU$ and solve $LUx = b$

☺ Numerically reliable

☹ Computational cost: $\mathcal{O}\left(n^2\right)$ operations, $\mathcal{O}\left(n^{4/3}\right)$ memory
Practical example on a $1000^3$ 27-point Helmholtz problem:
15 ExaFlops and 209 TeraBytes for factors!

# Iterative vs direct methods

## Iterative methods

Build sequence $x_k$ converging towards $x$

☺ Computational cost: $\mathcal{O}(n)$ operations/iteration and memory

☹ Convergence is application-dependent

## Direct methods

Factorize $A = LU$ and solve $LUx = b$

☺ Numerically reliable

☹ Computational cost: $\mathcal{O}(n^2)$ operations, $\mathcal{O}(n^{4/3})$ memory
Practical example on a $1000^3$ 27-point Helmholtz problem:
15 ExaFlops and 209 TeraBytes for factors!

**Objective of the thesis:**
**reduce the cost of sparse direct solvers ...**
**...while maintaining their numerical reliability**

# Main contributions

Principle: build approximated factorization $A_\varepsilon = L_\varepsilon U_\varepsilon$ at given accuracy $\varepsilon$

Contribution: design of novel algorithms with two fundamental properties

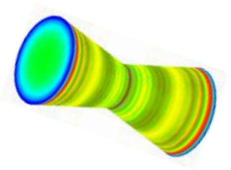
## 1st contribution: asymptotic complexity reduction

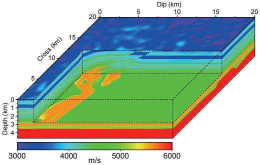
Theoretical proof and experimental validation that:

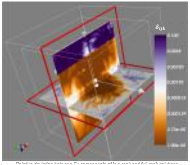
- Memory: $\mathcal{O}\left(n^{4/3}\right) \to \mathcal{O}\left(n \log n\right)$
- Operations: $\mathcal{O}\left(n^{2}\right) \to \mathcal{O}\left(n^{5/3}\right) \to \mathcal{O}\left(n^{4/3}\right)$

## 2nd contribution: efficient and scalable algorithms

Designed algorithms to efficiently translate the theoretical complexity reduction into actual performance and memory gains for large-scale computers and applications





Structural mechanics
Matrix of order 8M
Required accuracy: $10^{-9}$

Seismic imaging
Matrix of order 17M
Required accuracy: $10^{-3}$

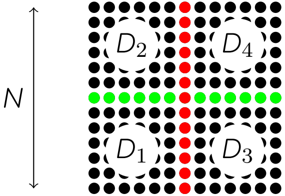
Electromagnetism
Matrix of order 21M
Required accuracy: $10^{-7}$

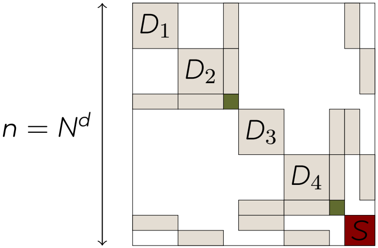
Results on 900 cores:

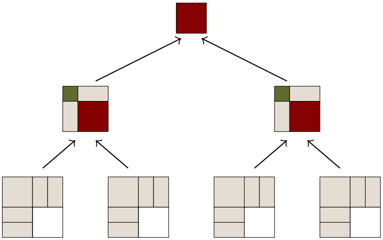
| | factorization time (s) | | | memory/proc (GB) | | |
|---|---|---|---|---|---|---|
| application | MUMPS | BLR | ratio | MUMPS | BLR | gain |
| structural | 289.3 | 104.9 | 2.5 | 7.9 | 5.9 | 25% |
| seismic | 617.0 | 123.4 | 4.9 | 13.3 | 10.4 | 22% |
| electromag. | 1307.4 | 233.8 | 5.3 | 20.6 | 14.4 | 30% |

# Introduction

# Multifrontal Factorization with Nested Dissection



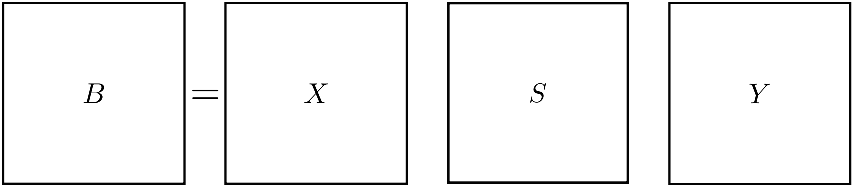
**3D problem complexity**

$\rightarrow$ Flops: $\mathcal{O}\left(n^2\right)$, mem: $\mathcal{O}\left(n^{4/3}\right)$

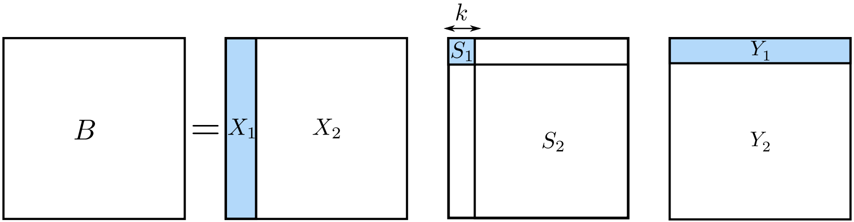
- George. *Nested dissection of a regular finite element mesh*, SIAM J. Numer. Anal., 1973.

Take a dense matrix $B$ of size $b \times b$ and compute its SVD $B = XSY$:

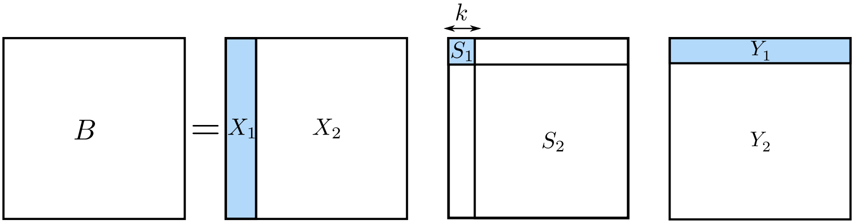


Take a dense matrix $B$ of size $b \times b$ and compute its SVD $B = XSY$:




$B = X_1 S_1 Y_1 + X_2 S_2 Y_2$ with $S_1(k,k) = \sigma_k > \varepsilon,\ S_2(1,1) = \sigma_{k+1} \leq \varepsilon$

Take a dense matrix $B$ of size $b \times b$ and compute its SVD $B = XSY$:

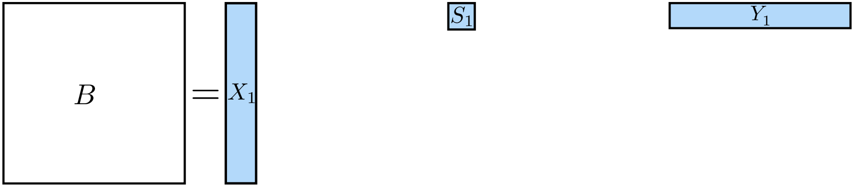



$B = X_1 S_1 Y_1 + X_2 S_2 Y_2$ with $S_1(k,k) = \sigma_k > \varepsilon, \; S_2(1,1) = \sigma_{k+1} \leq \varepsilon$

If $\tilde{B} = X_1 S_1 Y_1$ then $\|B - \tilde{B}\|_2 = \|X_2 S_2 Y_2\|_2 = \sigma_{k+1} \leq \varepsilon$

# Low-rank matrices

Take a dense matrix $B$ of size $b \times b$ and compute its SVD $B = XSY$:



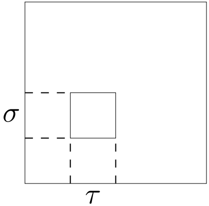

$B = X_1 S_1 Y_1 + X_2 S_2 Y_2$ with $S_1(k,k) = \sigma_k > \varepsilon,\ S_2(1,1) = \sigma_{k+1} \leq \varepsilon$

If $\tilde{B} = X_1 S_1 Y_1$ then $\|B - \tilde{B}\|_2 = \|X_2 S_2 Y_2\|_2 = \sigma_{k+1} \leq \varepsilon$

If the singular values of $B$ decay very fast (e.g. exponentially) then $k \ll b$ even for very small $\varepsilon$ (e.g. $10^{-14}$) $\Rightarrow$ memory and CPU consumption can be reduced considerably with a controlled loss of accuracy ($\leq \varepsilon$) if $\tilde{B}$ is used instead of $B$

# Low-rank sub-blocks

Frontal matrices are not low-rank but in some applications they exhibit low-rank blocks

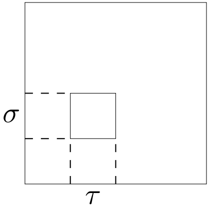



A block $B$ represents the interaction between two subdomains $\sigma$ and $\tau$.
If they have a small diameter and are far away their interaction is weak $\Rightarrow$ rank is low.

The block-admissibility condition formalizes this intuition:

$$\sigma \times \tau \text{ is admissible } \Leftrightarrow \max\left(\operatorname{diam}(\sigma), \operatorname{diam}(\tau)\right) \leq \eta \operatorname{dist}(\sigma, \tau)$$

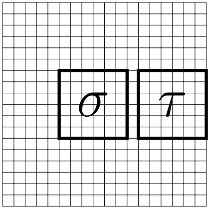
Frontal matrices are not low-rank but in some applications they exhibit low-rank blocks

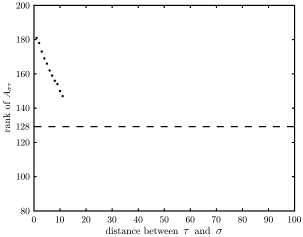



A block $B$ represents the interaction between two subdomains $\sigma$ and $\tau$.
If they have a small diameter and are far away their interaction is weak $\Rightarrow$ rank is low.

The block-admissibility condition formalizes this intuition:

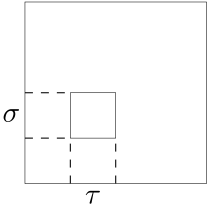
$$\sigma \times \tau \text{ is admissible } \Leftrightarrow \max\left(\operatorname{diam}(\sigma), \operatorname{diam}(\tau)\right) \leq \eta \operatorname{dist}(\sigma, \tau)$$



# Low-rank sub-blocks

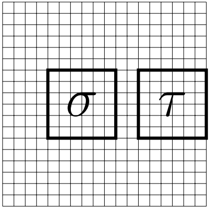
Frontal matrices are not low-rank but in some applications they exhibit low-rank blocks

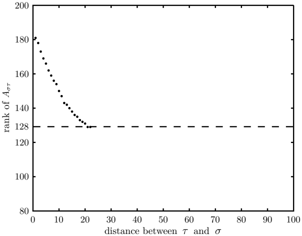



A block $B$ represents the interaction between two subdomains $\sigma$ and $\tau$.
If they have a small diameter and are far away their interaction is weak $\Rightarrow$ rank is low.

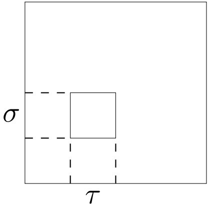
The block-admissibility condition formalizes this intuition:

$$\sigma \times \tau \text{ is admissible } \Leftrightarrow \max\left(\text{diam}\left(\sigma\right), \text{diam}\left(\tau\right)\right) \leq \eta \, \text{dist}\left(\sigma, \tau\right)$$



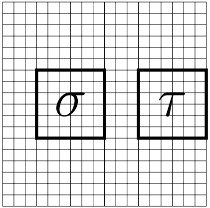
Frontal matrices are not low-rank but in some applications they exhibit low-rank blocks

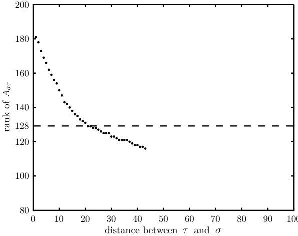



A block $B$ represents the interaction between two subdomains $\sigma$ and $\tau$.
If they have a small diameter and are far away their interaction is weak $\Rightarrow$ rank is low.

The block-admissibility condition formalizes this intuition:

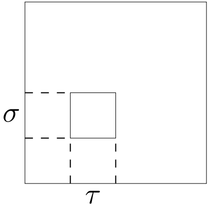
$$\sigma \times \tau \text{ is admissible } \Leftrightarrow \max\left(\text{diam}\left(\sigma\right), \text{diam}\left(\tau\right)\right) \leq \eta \, \text{dist}\left(\sigma, \tau\right)$$



# Low-rank sub-blocks

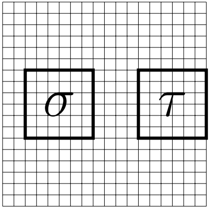
Frontal matrices are not low-rank but in some applications they exhibit low-rank blocks

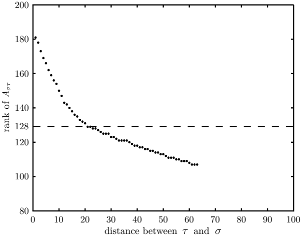



A block $B$ represents the interaction between two subdomains $\sigma$ and $\tau$.
If they have a small diameter and are far away their interaction is weak $\Rightarrow$ rank is low.

The block-admissibility condition formalizes this intuition:

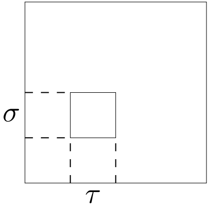
$$\sigma \times \tau \text{ is admissible } \Leftrightarrow \max\left(\text{diam}\left(\sigma\right), \text{diam}\left(\tau\right)\right) \leq \eta \, \text{dist}\left(\sigma, \tau\right)$$



# Low-rank sub-blocks

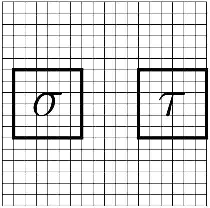
Frontal matrices are not low-rank but in some applications they exhibit low-rank blocks

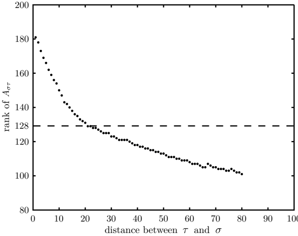



A block $B$ represents the interaction between two subdomains $\sigma$ and $\tau$.
If they have a small diameter and are far away their interaction is weak $\Rightarrow$ rank is low.

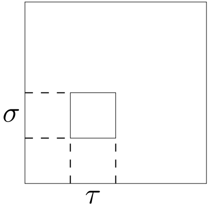
The block-admissibility condition formalizes this intuition:

$$\sigma \times \tau \text{ is admissible } \Leftrightarrow \max\left(\text{diam}\left(\sigma\right), \text{diam}\left(\tau\right)\right) \leq \eta \,\text{dist}\left(\sigma, \tau\right)$$



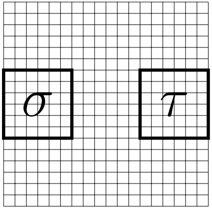
Frontal matrices are not low-rank but in some applications they exhibit low-rank blocks

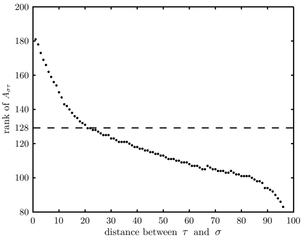



A block $B$ represents the interaction between two subdomains $\sigma$ and $\tau$.
If they have a small diameter and are far away their interaction is weak $\Rightarrow$ rank is low.

The block-admissibility condition formalizes this intuition:

$$\sigma \times \tau \text{ is admissible } \Leftrightarrow \max\left(\text{diam}\left(\sigma\right), \text{diam}\left(\tau\right)\right) \leq \eta \,\text{dist}\left(\sigma, \tau\right)$$

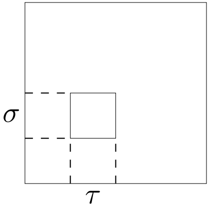


# Low-rank sub-blocks

Frontal matrices are not low-rank but in some applications they exhibit low-rank blocks




A block $B$ represents the interaction between two subdomains $\sigma$ and $\tau$.
If they have a small diameter and are far away their interaction is weak $\Rightarrow$ rank is low.

The block-admissibility condition formalizes this intuition:

$$\sigma \times \tau \text{ is admissible } \Leftrightarrow \max\left(\operatorname{diam}(\sigma), \operatorname{diam}(\tau)\right) \leq \eta \operatorname{dist}(\sigma, \tau)$$

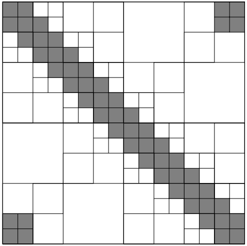
**Geometric**

- ☺ Produces very regular subdomains
- ☹ Not always available

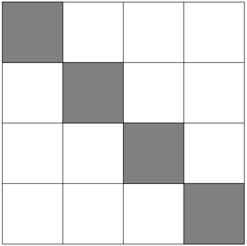
**Algebraic**

- ☹ Produces more irregular subdomains
- ☺ Always available (matrix graph)

# $\mathcal{H}$ and BLR matrices

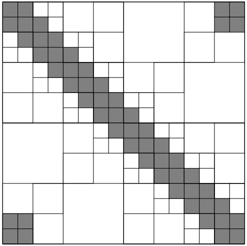


$\mathcal{H}$-matrix

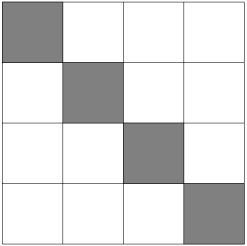


BLR matrix

# $\mathcal{H}$ and BLR matrices



$\mathcal{H}$-matrix

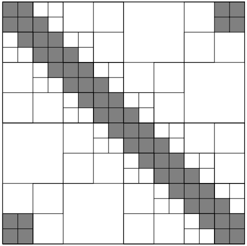


BLR matrix

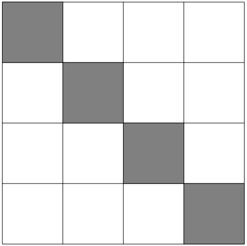
- Theoretical complexity can be as low as $\mathcal{O}(n)$
- Complex, hierarchical structure

- Theoretical complexity?
- Simple structure

# $\mathcal{H}$ and BLR matrices



$\mathcal{H}$-matrix



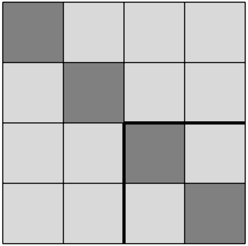
BLR matrix

- Theoretical complexity can be as low as $\mathcal{O}(n)$
- Complex, hierarchical structure

- Theoretical complexity?
- Simple structure

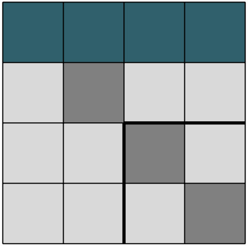
**Find a good comprise between complexity and performance**

# Standard BLR factorization: FSCU



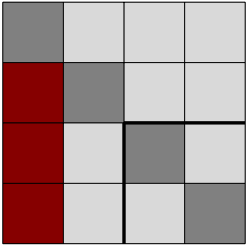
- FSCU

# Standard BLR factorization: FSCU



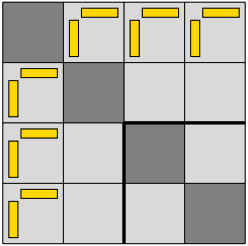
- FSCU (Factor,
- Easy to handle numerical pivoting, a critical feature often lacking in other low-rank solvers

# Standard BLR factorization: FSCU



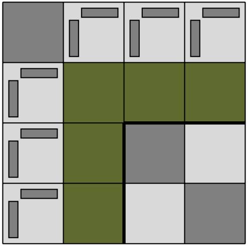
- FSCU (Factor, Solve,
- Easy to handle numerical pivoting, a critical feature often lacking in other low-rank solvers

# Standard BLR factorization: FSCU



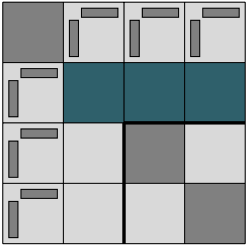
- FSCU (Factor, Solve, Compress,
- Easy to handle numerical pivoting, a critical feature often lacking in other low-rank solvers

# Standard BLR factorization: FSCU



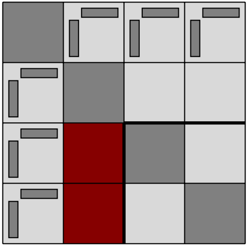
- FSCU (Factor, Solve, Compress, Update)
- Easy to handle numerical pivoting, a critical feature often lacking in other low-rank solvers

# Standard BLR factorization: FSCU



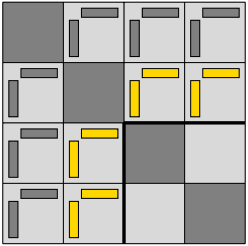
- FSCU (Factor, Solve, Compress, Update)
- Easy to handle numerical pivoting, a critical feature often lacking in other low-rank solvers

# Standard BLR factorization: FSCU



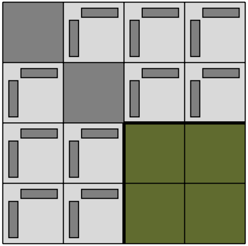
- FSCU (Factor, Solve, Compress, Update)
- Easy to handle numerical pivoting, a critical feature often lacking in other low-rank solvers

# Standard BLR factorization: FSCU



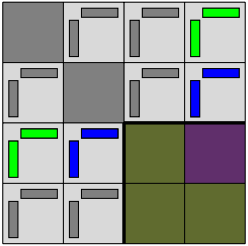
- FSCU (Factor, Solve, Compress, Update)
- Easy to handle numerical pivoting, a critical feature often lacking in other low-rank solvers

# Standard BLR factorization: FSCU

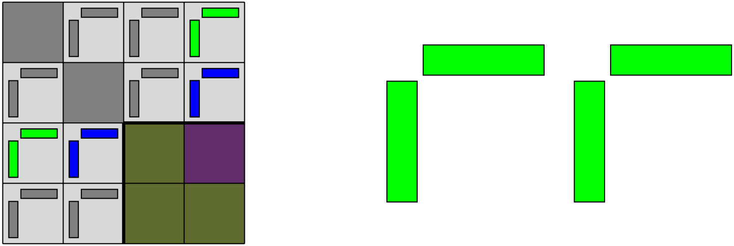


- FSCU (Factor, Solve, Compress, Update)
- Easy to handle numerical pivoting, a critical feature often lacking in other low-rank solvers

# Standard BLR factorization: FSCU

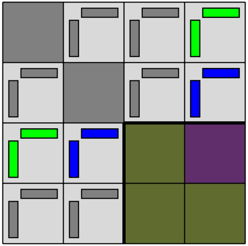


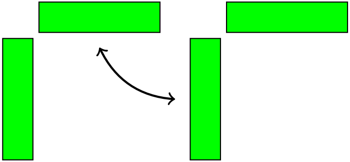
- FSCU (Factor, Solve, Compress, Update)
- Easy to handle numerical pivoting, a critical feature often lacking in other low-rank solvers



- FSCU (Factor, Solve, Compress, Update)
- Easy to handle numerical pivoting, a critical feature often lacking in other low-rank solvers

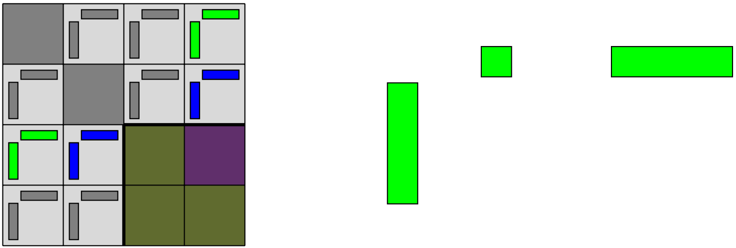
# Standard BLR factorization: FSCU





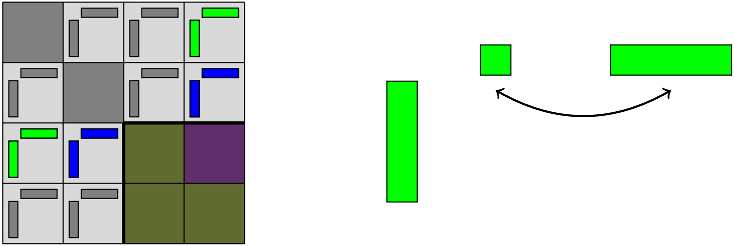
- FSCU (Factor, Solve, Compress, Update)
- Easy to handle numerical pivoting, a critical feature often lacking in other low-rank solvers

# Standard BLR factorization: FSCU



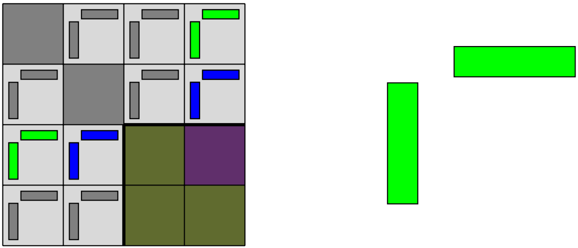
- FSCU (Factor, Solve, Compress, Update)
- Easy to handle numerical pivoting, a critical feature often lacking in other low-rank solvers

# Standard BLR factorization: FSCU



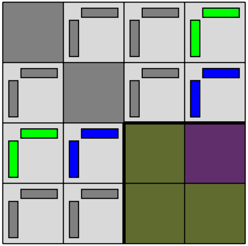
- FSCU (Factor, Solve, Compress, Update)
- Easy to handle numerical pivoting, a critical feature often lacking in other low-rank solvers

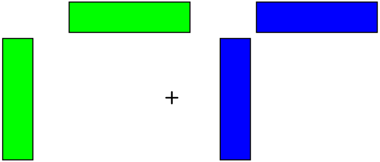
# Standard BLR factorization: FSCU



- FSCU (Factor, Solve, Compress, Update)
- Easy to handle numerical pivoting, a critical feature often lacking in other low-rank solvers

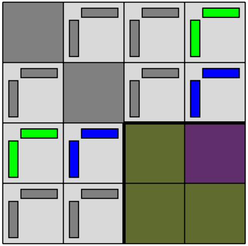
# Standard BLR factorization: FSCU

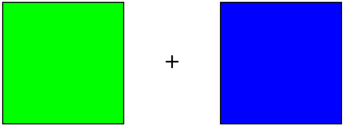





- FSCU (Factor, Solve, Compress, Update)
- Easy to handle numerical pivoting, a critical feature often lacking in other low-rank solvers

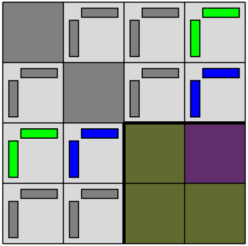
# Standard BLR factorization: FSCU

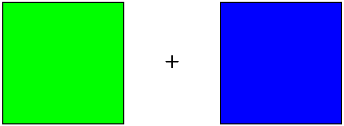





- FSCU (Factor, Solve, Compress, Update)
- Easy to handle numerical pivoting, a critical feature often lacking in other low-rank solvers

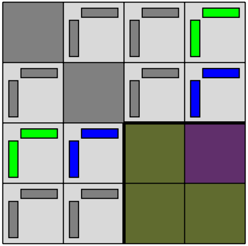
# Standard BLR factorization: FSCU

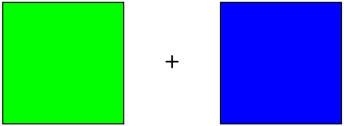




- FSCU (Factor, Solve, Compress, Update)
- Easy to handle numerical pivoting, a critical feature often lacking in other low-rank solvers
- Potential of this variant was studied in
  - Amestoy, Ashcraft, Boiteau, Buttari, L'Excellent, and Weisbecker. *Improving Multifrontal Methods by Means of Block Low-Rank Representations*, SIAM J. Sci. Comput., 2015.

# Standard BLR factorization: FSCU






- FSCU (Factor, Solve, Compress, Update)
- Easy to handle numerical pivoting, a critical feature often lacking in other low-rank solvers
- Potential of this variant was studied in
  - Amestoy, Ashcraft, Boiteau, Buttari, L'Excellent, and Weisbecker. *Improving Multifrontal Methods by Means of Block Low-Rank Representations*, SIAM J. Sci. Comput., 2015.

**...but many open questions remain**

# Open questions

- What is the theoretical complexity of the BLR factorization? Does it hold in the algebraic case?
- How can we get actual performance gains out of the complexity reduction of the BLR factorization?
- Can we design novel variants of the BLR factorization to improve its complexity and performance? Can we do that without sacrificing numerical pivoting?
- How well does the distributed-memory BLR factorization scale and how can we improve its scalability?

# Open questions

- What is the theoretical complexity of the BLR factorization? Does it hold in the algebraic case?
- How can we get actual performance gains out of the complexity reduction of the BLR factorization?
- Can we design novel variants of the BLR factorization to improve its complexity and performance? Can we do that without sacrificing numerical pivoting?
- How well does the distributed-memory BLR factorization scale and how can we improve its scalability?

# Open questions

- What is the theoretical complexity of the BLR factorization? Does it hold in the algebraic case?
- How can we get actual performance gains out of the complexity reduction of the BLR factorization?
- Can we design novel variants of the BLR factorization to improve its complexity and performance? Can we do that without sacrificing numerical pivoting?
- How well does the distributed-memory BLR factorization scale and how can we improve its scalability?

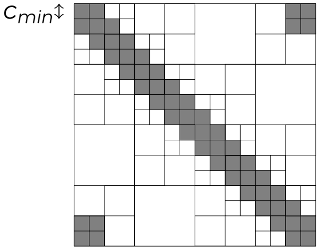
# Open questions

- What is the theoretical complexity of the BLR factorization? Does it hold in the algebraic case?
- How can we get actual performance gains out of the complexity reduction of the BLR factorization?
- Can we design novel variants of the BLR factorization to improve its complexity and performance? Can we do that without sacrificing numerical pivoting?
- How well does the distributed-memory BLR factorization scale and how can we improve its scalability?

Outline of the rest of the presentation:

1. What is the theoretical Complexity of the BLR factorization? Does it hold in the algebraic case?
2. How can we get actual Performance gains out of the complexity reduction of the BLR factorization?
3. Can we design novel Variants of the BLR factorization to improve its complexity and performance? Can we do that without sacrificing numerical pivoting?
4. How well does the Distributed-memory BLR factorization scale and how can we improve its scalability?

# Complexity of the BLR factorization

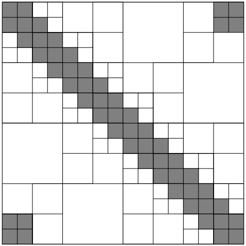



### $\mathcal{H}$-admissibility condition

A partition $\mathcal{P}$ is admissible iff

$$\forall \sigma \times \tau \in \mathcal{P}, \quad \sigma \times \tau \text{ is admissible} \quad \text{or} \quad \min(\#\sigma, \#\tau) \leq c_{min}$$

# $\mathcal{H}$-admissibility and sparsity constant




$c_{sp}$ is the maximal number of blocks of the same size on the same row/column (here, $c_{sp} = 6$)

## $\mathcal{H}$-admissibility condition

A partition $\mathcal{P}$ is admissible iff

$$\forall \sigma \times \tau \in \mathcal{P}, \quad \sigma \times \tau \text{ is admissible} \quad \text{or} \quad \min(\#\sigma, \#\tau) \leq c_{min}$$

The so-called sparsity constant $c_{sp}$ is defined by:

$$c_{sp} = \max(\max_{\sigma} \#\{\tau; \sigma \times \tau \in P\}, \max_{\tau} \#\{\sigma; \sigma \times \tau \in P\})$$

# $\mathcal{H}$ vs. BLR complexity

## Dense factorization complexity

Complexity: $\mathcal{C}_{facto} = \mathcal{O}\left(c_{sp}^2 r_{max}^2 m \log^2 m\right)$

| | |
|---|---|
| $m$ | matrix size |
| $c_{sp}$ | sparsity constant |
| $r_{max}$ | bound on the maximal rank of all blocks |

# $\mathcal{H}$ vs. BLR complexity

## Dense factorization complexity

Complexity: $\mathcal{C}_{facto} = \mathcal{O}\left(c_{sp}^2 r_{max}^2 m \log^2 m\right)$

| | |
|---|---|
| $m$ | matrix size |
| $c_{sp}$ | sparsity constant |
| $r_{max}$ | bound on the maximal rank of all blocks |

| | $\mathcal{H}$ | BLR |
|---|---|---|
| $c_{sp}$ | | |
| $r_{max}$ | | |
| $\mathcal{C}_{facto}$ | | |

# $\mathcal{H}$ vs. BLR complexity

## Dense factorization complexity

Complexity: $\mathcal{C}_{facto} = \mathcal{O}\left(c_{sp}^2 r_{max}^2 m \log^2 m\right)$

| | |
|---|---|
| $m$ | matrix size |
| $c_{sp}$ | sparsity constant |
| $r_{max}$ | bound on the maximal rank of all blocks |

| | $\mathcal{H}$ | BLR |
|---|---|---|
| $c_{sp}$ | $\mathcal{O}(1)^*$ | |
| $r_{max}$ | | |
| $\mathcal{C}_{facto}$ | | |

$^*$*Grasedyck & Hackbusch, 2003*

# $\mathcal{H}$ vs. BLR complexity

## Dense factorization complexity

Complexity: $\mathcal{C}_{facto} = \mathcal{O}\left(c_{sp}^2 r_{max}^2 m \log^2 m\right)$

| | |
|---|---|
| $m$ | matrix size |
| $c_{sp}$ | sparsity constant |
| $r_{max}$ | bound on the maximal rank of all blocks |

| | $\mathcal{H}$ | BLR |
|---|---|---|
| $c_{sp}$ | $\mathcal{O}(1)^*$ | |
| $r_{max}$ | small$^{**}$ | |
| $\mathcal{C}_{facto}$ | | |

* *Grasedyck & Hackbusch, 2003*

** *Bebendorf & Hackbusch, 2003*

# $\mathcal{H}$ vs. BLR complexity

## Dense factorization complexity

Complexity: $\mathcal{C}_{facto} = \mathcal{O}\left(c_{sp}^2 r_{max}^2 m \log^2 m\right)$

| | |
|---|---|
| $m$ | matrix size |
| $c_{sp}$ | sparsity constant |
| $r_{max}$ | bound on the maximal rank of all blocks |

| | $\mathcal{H}$ | BLR |
|---|---|---|
| $c_{sp}$ | $\mathcal{O}(1)^*$ | |
| $r_{max}$ | small** | |
| $\mathcal{C}_{facto}$ | $\mathcal{O}\left(r_{max}^2 m \log^2 m\right)$ | |

*Grasedyck & Hackbusch, 2003

**Bebendorf & Hackbusch, 2003

# $\mathcal{H}$ vs. BLR complexity

## Dense factorization complexity

Complexity: $\mathcal{C}_{facto} = \mathcal{O}\left(c_{sp}^2 r_{max}^2 m \log^2 m\right)$

| | |
|---|---|
| $m$ | matrix size |
| $c_{sp}$ | sparsity constant |
| $r_{max}$ | bound on the maximal rank of all blocks |

| | $\mathcal{H}$ | BLR |
|---|---|---|
| $c_{sp}$ | $\mathcal{O}(1)^*$ | $m/b$ |
| $r_{max}$ | small$^{**}$ | |
| $\mathcal{C}_{facto}$ | $\mathcal{O}\left(r_{max}^2 m \log^2 m\right)$ | |

$^*$*Grasedyck & Hackbusch, 2003*

$^{**}$*Bebendorf & Hackbusch, 2003*

## Dense factorization complexity

**Complexity:** $\mathcal{C}_{facto} = \mathcal{O}\left(c_{sp}^2 r_{max}^2 m \log^2 m\right)$

| | |
|---|---|
| $m$ | matrix size |
| $c_{sp}$ | sparsity constant |
| $r_{max}$ | bound on the maximal rank of all blocks |

| | $\mathcal{H}$ | BLR |
|---|---|---|
| $c_{sp}$ | $\mathcal{O}(1)^*$ | $m/b$ |
| $r_{max}$ | small** | $b$ |
| $\mathcal{C}_{facto}$ | $\mathcal{O}\left(r_{max}^2 m \log^2 m\right)$ | |

**Grasedyck & Hackbusch, 2003*

***Bebendorf & Hackbusch, 2003*

# $\mathcal{H}$ vs. BLR complexity

## Dense factorization complexity

Complexity: $\mathcal{C}_{facto} = \mathcal{O}\left(c_{sp}^2 r_{max}^2 m \log^2 m\right)$

| | |
|---|---|
| $m$ | matrix size |
| $c_{sp}$ | sparsity constant |
| $r_{max}$ | bound on the maximal rank of all blocks |

| | $\mathcal{H}$ | BLR |
|---|---|---|
| $c_{sp}$ | $\mathcal{O}(1)^*$ | $m/b$ |
| $r_{max}$ | small** | $b$ |
| $\mathcal{C}_{facto}$ | $\mathcal{O}\left(r_{max}^2 m \log^2 m\right)$ | $\mathcal{O}\left(m^3 \log^2 m\right)$ |

* *Grasedyck & Hackbusch, 2003*

** *Bebendorf & Hackbusch, 2003*

## Dense factorization complexity

Complexity: $\mathcal{C}_{facto} = \mathcal{O}\left(c_{sp}^2 r_{max}^2 m \log^2 m\right)$

| | |
|---|---|
| $m$ | matrix size |
| $c_{sp}$ | sparsity constant |
| $r_{max}$ | bound on the maximal rank of all blocks |

| | $\mathcal{H}$ | BLR |
|---|---|---|
| $c_{sp}$ | $\mathcal{O}(1)^*$ | $m/b$ |
| $r_{max}$ | small$^{**}$ | $b$ |
| $\mathcal{C}_{facto}$ | $\mathcal{O}\left(r_{max}^2 m \log^2 m\right)$ | $\mathcal{O}\left(m^3 \log^2 m\right)$ |

* *Grasedyck & Hackbusch, 2003*

** *Bebendorf & Hackbusch, 2003*

## BLR: a particular case of $\mathcal{H}$?

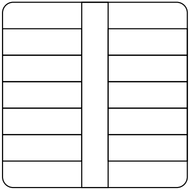
Problem: in $\mathcal{H}$ formalism, the maximal rank of the blocks of a BLR matrix is $r_{max} = b$ (due to the non-admissible blocks)

Solution: bound the rank of the admissible blocks only, and make sure the non-admissible blocks are in small number

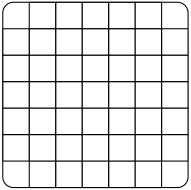
# BLR admissibility condition

## BLR-admissibility condition of a partition $\mathcal{P}$

$$\mathcal{P} \text{ is admissible } \Leftrightarrow \begin{cases} \#\{\sigma, \quad \sigma \times \tau \in \mathcal{P} \text{ is not admissible}\} \leq q \\ \#\{\tau, \quad \sigma \times \tau \in \mathcal{P} \text{ is not admissible}\} \leq q \end{cases}$$



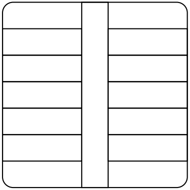
Non-Admissible



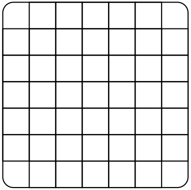
Admissible

## BLR-admissibility condition of a partition $\mathcal{P}$

$$\mathcal{P} \text{ is admissible} \Leftrightarrow \begin{cases} \#\{\sigma, \quad \sigma \times \tau \in \mathcal{P} \text{ is not admissible}\} \leq q \\ \#\{\tau, \quad \sigma \times \tau \in \mathcal{P} \text{ is not admissible}\} \leq q \end{cases}$$



Non-Admissible



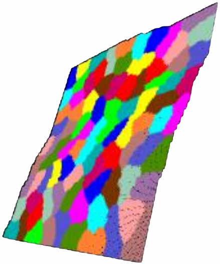
Admissible

## Main result

For any matrix, we can build an admissible $\mathcal{P}$ for $q = \mathcal{O}(1)$, s.t. the maximal rank of the admissible blocks of $A$ is $r = \mathcal{O}\left(r_{max}^{\mathcal{H}}\right)$

- Amestoy, Buttari, L'Excellent, and Mary. *On the Complexity of the Block Low-Rank Multifrontal Factorization*, SIAM J. Sci. Comput., 2017.

# Algebraic clusterings are admissible



Root separator of a $128^3$ Poisson problem clustered with SCOTCH via k-means

- Weisbecker. *Improving multifrontal solvers by means of algebraic Block Low-Rank representations*, PhD thesis.

The BLR-admissibility condition provides a theoretical justification of the intuitive choice to use k-means as clustering

# Complexity of the BLR factorization

## Complexity of the dense BLR factorization

$\mathcal{C}_{facto} = \mathcal{O}\left(rm^3/b + m^2 b\right) = \mathcal{O}\left(r^{1/2} m^{5/2}\right)$ (for $b = \mathcal{O}\left(\sqrt{rm}\right)$)

## Complexity of the sparse multifrontal BLR factorization

In the 3D case (similar analysis possible for 2D):

| | operations (OPC) | factor size (NNZ) |
|---|---|---|
| FR | $\mathcal{O}\left(n^2\right)$ | $\mathcal{O}\left(n^{4/3}\right)$ |
| BLR | $\mathcal{O}\left(n^{5/3} r^{1/2}\right)$ | $\mathcal{O}\left(n \max\left(r^{1/2}, \log n\right)\right)$ |

- Asymptotic complexity reduction...
- ...but still quite far from the $\mathcal{O}(n)$ $\mathcal{H}$-complexity

# Experimental Setting: Matrices

1. **`Poisson`**: $N^3$ grid with a 7-point stencil with $u = 1$ on the boundary $\partial\Omega$

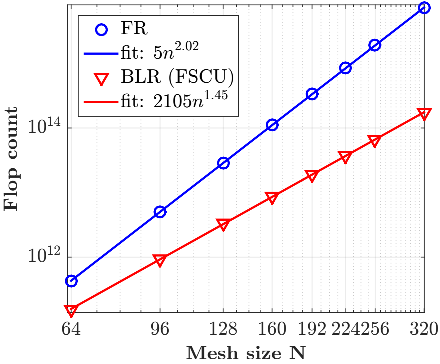
$$\Delta u = f$$

2. **`Helmholtz`**: $N^3$ grid with a 27-point stencil, $\omega$ is the angular frequency, $v(x)$ is the seismic velocity field, and $u(x,\omega)$ is the time-harmonic wavefield solution to the forcing term $s(x,\omega)$.

$$\left(-\Delta - \frac{\omega^2}{v(x)^2}\right) u(x,\omega) = s(x,\omega)$$

$\omega$ is fixed and equal to $4$Hz.

Nested Dissection ordering (geometric)

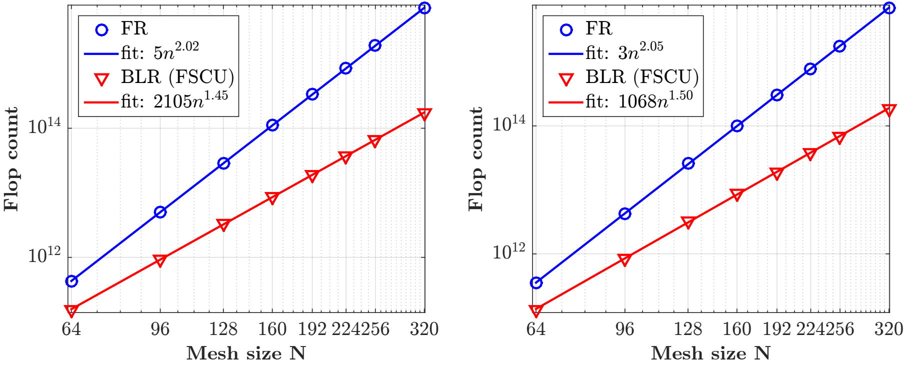



- good agreement with theoretical complexity

# Experimental MF flop complexity: Poisson ($\varepsilon = 10^{-10}$)

Nested Dissection ordering (geometric)

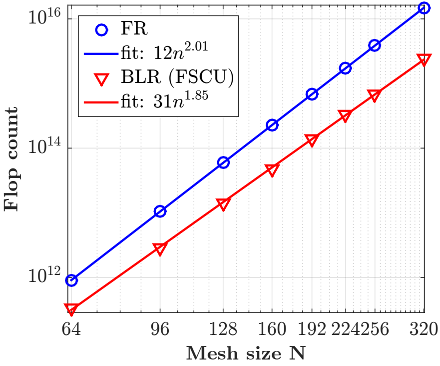
METIS ordering (purely algebraic)



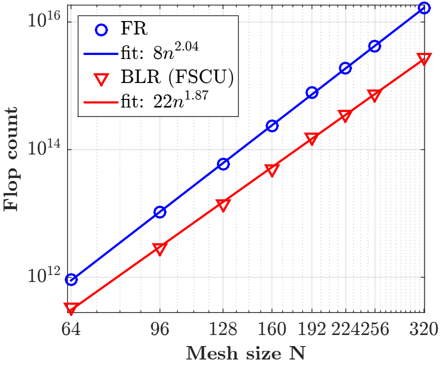

- good agreement with theoretical complexity
- METIS algebraic complexity remains close to geometric ND

# Experimental MF flop complexity: Helmholtz ($\varepsilon = 10^{-4}$)

Nested Dissection ordering (geometric)



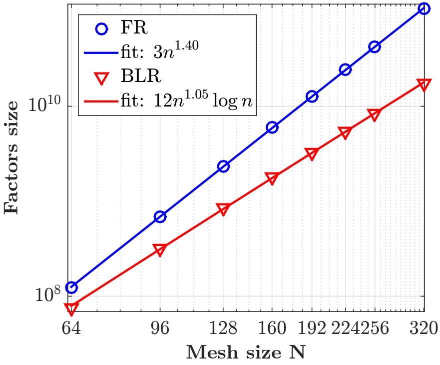

METIS ordering (purely algebraic)



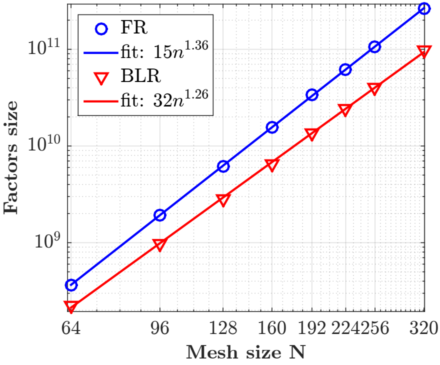

- good agreement with theoretical complexity, under the strong assumption $r = \mathcal{O}(N)$
- METIS algebraic complexity remains close to geometric ND

# Experimental MF complexity: factor size

NNZ
(Poisson)




NNZ
(Helmholtz)




- good agreement with theoretical complexity
- METIS algebraic complexity remains close to geometric ND (not shown)

# Performance of the BLR factorization

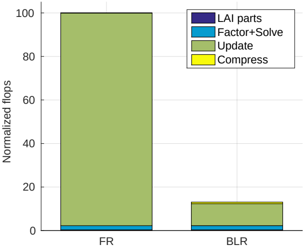
# Shared-memory experimental setting

| application | matrix | arith. | fact. | n | nnz | flops | factor size |
|---|---|---|---|---|---|---|---|
| seismic imaging (SEISCOPE) | 5Hz | c | $LU$ | 2.9M | 70M | 69.5 TF | 61.4 GB |
| | 7Hz | c | $LU$ | 7.2M | 177M | 471.1 TF | 219.6 GB |
| | 10Hz | c | $LU$ | 17.2M | 446M | 2.7 PF | 728.1 GB |
| electromag. modeling (EMGS) | H3 | z | $LDL^T$ | 2.9M | 37M | 57.9 TF | 77.5 GB |
| | H17 | z | $LDL^T$ | 17.4M | 226M | 2.2 PF | 891.1 GB |
| | S3 | z | $LDL^T$ | 3.3M | 43M | 78.0 TF | 94.6 GB |
| | S21 | z | $LDL^T$ | 20.6M | 266M | 3.2 PF | 1.1 TB |
| structural mechanics (EDF) | p8d | d | $LDL^T$ | 1.9M | 81M | 101.0 TF | 52.6 GB |
| | p8ar | d | $LDL^T$ | 3.9M | 159M | 377.5 TF | 129.8 GB |
| | p8cr | d | $LDL^T$ | 7.9M | 321M | 1.6 PF | 341.1 GB |
| | p9ar | d | $LDL^T$ | 5.4M | 209M | 23.6 TF | 40.5 GB |

Experiments were performed on brunch (LIP-ENS Lyon):

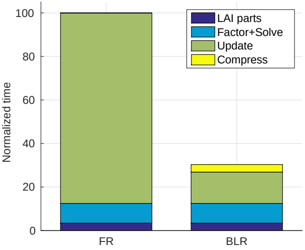
- Four Intel(r) 24-cores Broadwell @ 2.2 GHz
- Peak per core is 35.2 GF/s
- Total memory is 1.5 TB

# Shared-memory performance analysis (matrix S3)



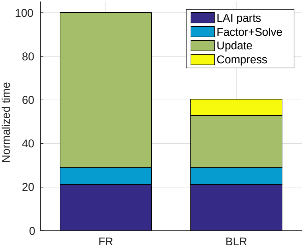

Normalized Flops

**7.7** gain




Normalized Time
(1 thread)

**3.7** gain



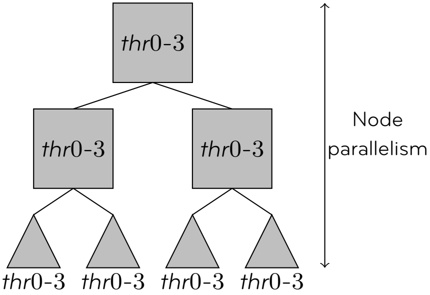

Normalized Time
(24 threads)

**1.7** gain

**7.7** gain in flops only translated to a **1.7** gain in time: why?

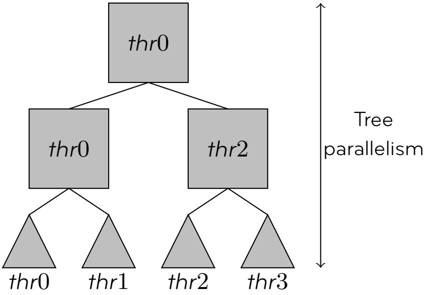
- the higher relative weight of the FR parts (Factor+Solve and LAI parts) limits the global gain
- the Update and Compress steps have a lower speed because of their low granularity and memory-boundedness

# Exploiting tree-based multithreading in MF solvers



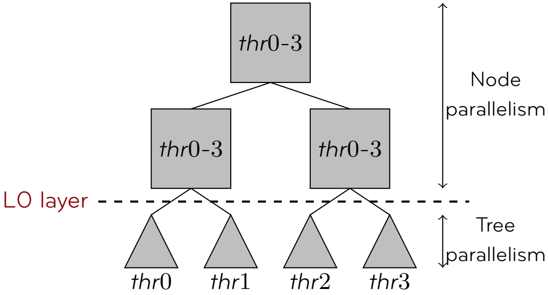

- Node parallelism approach based on OpenMP loops

# Exploiting tree-based multithreading in MF solvers



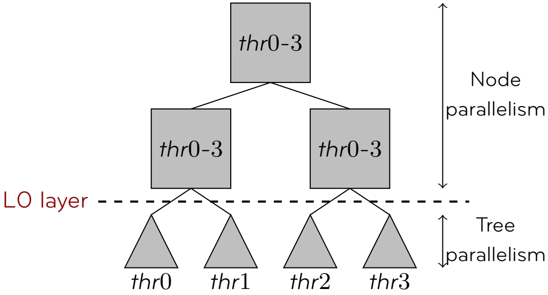

- Node parallelism approach based on OpenMP loops

# Exploiting tree-based multithreading in MF solvers



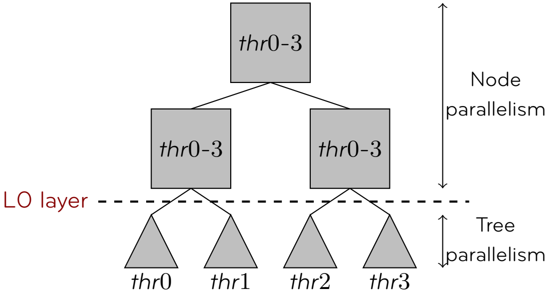

- Node parallelism approach based on OpenMP loops
- Node+tree parallelism approach based on Sid-Lakhdar's PhD
  - L'Excellent and Sid-Lakhdar. *A study of shared-memory parallelism in a multifrontal solver*, Parallel Computing.

# Exploiting tree-based multithreading in MF solvers



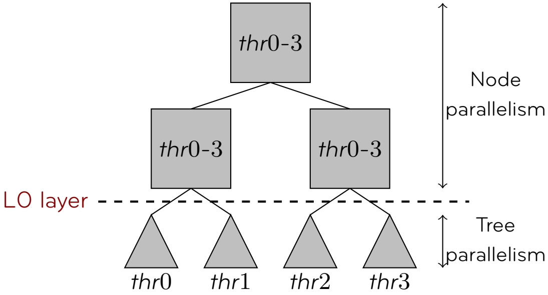

- Node parallelism approach based on OpenMP loops
- Node+tree parallelism approach based on Sid-Lakhdar's PhD
  - L'Excellent and Sid-Lakhdar. *A study of shared-memory parallelism in a multifrontal solver*, Parallel Computing.
- In FR, top of the tree is dominant ⇒ tree MT brings little gain

# Exploiting tree-based multithreading in MF solvers




- Node parallelism approach based on OpenMP loops
- Node+tree parallelism approach based on Sid-Lakhdar's PhD
  - L'Excellent and Sid-Lakhdar. *A study of shared-memory parallelism in a multifrontal solver*, Parallel Computing.
- In FR, top of the tree is dominant ⇒ tree MT brings little gain
- In BLR, bottom of the tree compresses less, becomes important

# Exploiting tree-based multithreading in MF solvers




- Node parallelism approach based on OpenMP loops
- Node+tree parallelism approach based on Sid-Lakhdar's PhD
  - L'Excellent and Sid-Lakhdar. *A study of shared-memory parallelism in a multifrontal solver*, Parallel Computing.
- In FR, top of the tree is dominant ⇒ tree MT brings little gain
- In BLR, bottom of the tree compresses less, becomes important

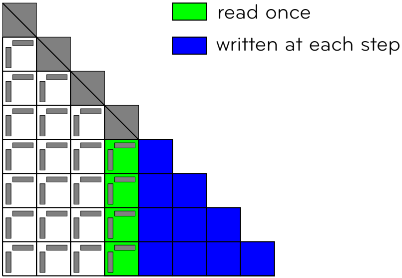
⇒ **1.7** gain becomes **1.9** thanks to tree-based multithreading

# Right-looking Vs. Left-looking analysis (24 threads)

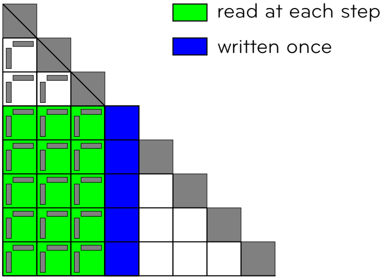
| | FR time | | BLR time | |
|---|---|---|---|---|
| | RL | LL | RL | LL |
| Update | 338 | 336 | 110 | 67 |
| Total | 424 | 421 | 221 | 175 |

# Right-looking Vs. Left-looking analysis (24 threads)

| | FR time | | BLR time | |
|---|---|---|---|---|
| | RL | LL | RL | LL |
| Update | 338 | 336 | 110 | 67 |
| Total | 424 | 421 | 221 | 175 |



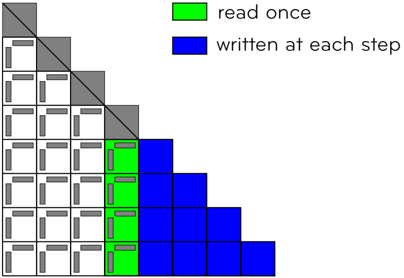

RL factorization



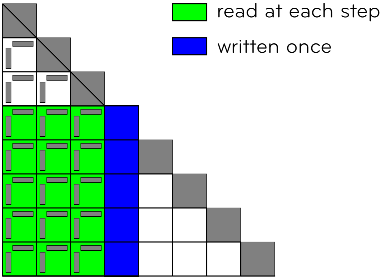

LL factorization

# Right-looking Vs. Left-looking analysis (24 threads)

| | FR time | | BLR time | |
|---|---|---|---|---|
| | RL | LL | RL | LL |
| Update | 338 | 336 | 110 | 67 |
| Total | 424 | 421 | 221 | 175 |



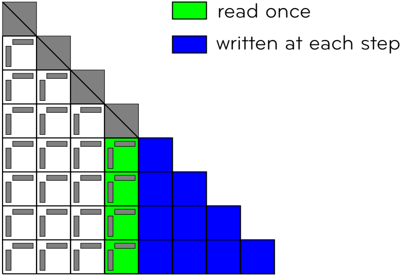

RL factorization



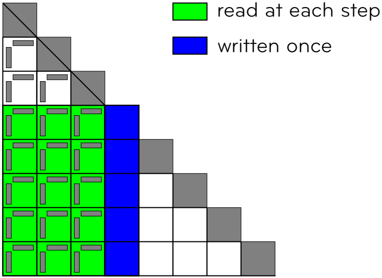

LL factorization

⇒ Lower volume of memory transfers in LL (more critical in MT)

| | FR time | | BLR time | |
|---|---|---|---|---|
| | RL | LL | RL | LL |
| Update | 338 | 336 | 110 | 67 |
| Total | 424 | 421 | 221 | 175 |




RL factorization



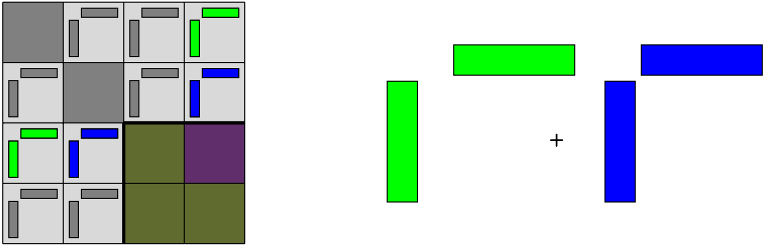

LL factorization

⇒ Lower volume of memory transfers in LL (more critical in MT)

Update is now less memory-bound: **1.9** gain becomes **2.4** in LL

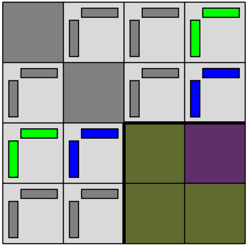
# Variants of the BLR factorization

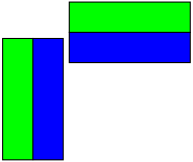
# LUAR variant: accumulation and recompression




- FSCU (Factor, Solve, Compress, Update)
- FSCU+LUAR

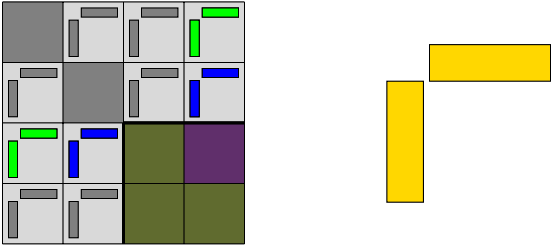
# LUAR variant: accumulation and recompression





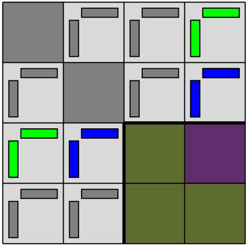
- FSCU (Factor, Solve, Compress, Update)
- FSCU+LUAR
  - Better granularity in Update operations

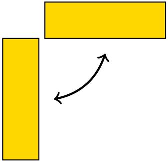
# LUAR variant: accumulation and recompression



- FSCU (Factor, Solve, Compress, Update)
- FSCU+LUAR
  - Better granularity in Update operations
  - Potential recompression

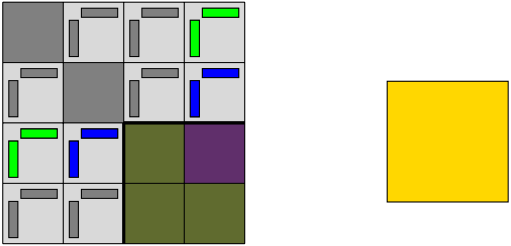
# LUAR variant: accumulation and recompression





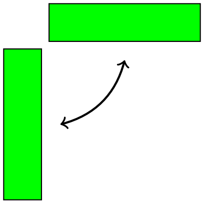
- FSCU (Factor, Solve, Compress, Update)
- FSCU+LUAR
  - Better granularity in Update operations
  - Potential recompression

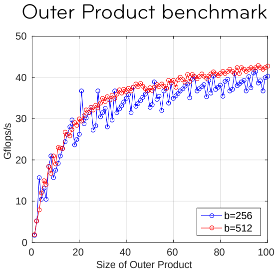
# LUAR variant: accumulation and recompression



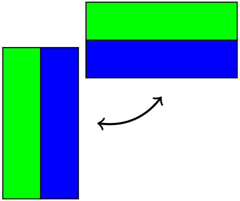
- FSCU (Factor, Solve, Compress, Update)
- FSCU+LUAR
    - Better granularity in Update operations
    - Potential recompression

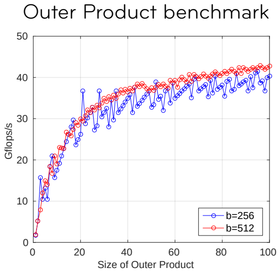
# Performance of Outer Product with LUA(R) (24 threads)





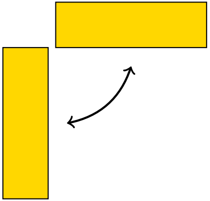

| | | LL |
|---|---|---|
| average size of Outer Product | | 16.5 |
| flops ($\times 10^{12}$) | Outer Product | 3.8 |
| | Total | 10.2 |
| time (s) | Outer Product | 21 |
| | Total | 175 |

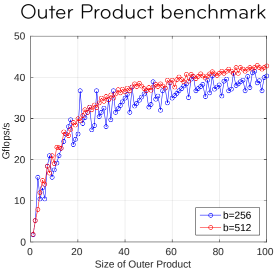





| | | LL | LUA |
|---|---|---|---|
| average size of Outer Product | | 16.5 | 61.0 |
| flops ($\times 10^{12}$) | Outer Product | 3.8 | 3.8 |
| | Total | 10.2 | 10.2 |
| time (s) | Outer Product | 21 | 14 |
| | Total | 175 | 167 |

# Performance of Outer Product with LUA(R) (24 threads)

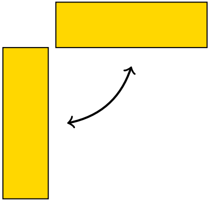


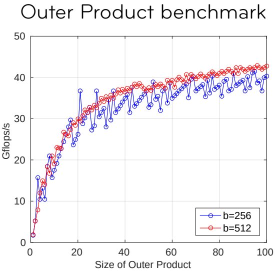



| | | LL | LUA | LUAR* |
|---|---|---|---|---|
| average size of Outer Product | | 16.5 | 61.0 | 32.8 |
| flops ($\times 10^{12}$) | Outer Product | 3.8 | 3.8 | 1.6 |
| | Total | 10.2 | 10.2 | 8.1 |
| time (s) | Outer Product | 21 | 14 | 6 |
| | Total | 175 | 167 | 160 |

* All metrics include the Recompression overhead

# Performance of Outer Product with LUA(R) (24 threads)





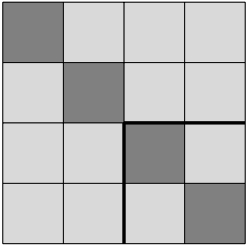

| | | LL | LUA | LUAR* |
|---|---|---|---|---|
| average size of Outer Product | | 16.5 | 61.0 | 32.8 |
| flops ($\times 10^{12}$) | Outer Product | 3.8 | 3.8 | 1.6 |
| | Total | 10.2 | 10.2 | 8.1 |
| time (s) | Outer Product | 21 | 14 | 6 |
| | Total | 175 | 167 | 160 |

* All metrics include the Recompression overhead

Higher granularity and lower flops in Update:

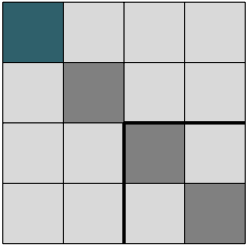
$\Rightarrow$ **2.4** gain becomes **2.6**

# FCSU variant: compress before solve



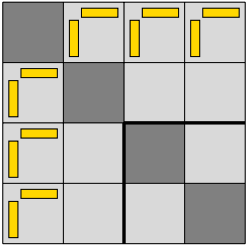
- FSCU (Factor, Solve, Compress, Update)
- FSCU+LUAR
- FCSU(+LUAR)

# FCSU variant: compress before solve



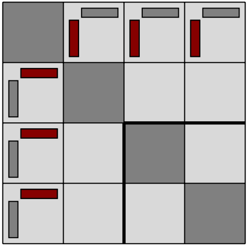
- FSCU (Factor, Solve, Compress, Update)
- FSCU+LUAR
- FCSU(+LUAR)
  - Restricted pivoting

# FCSU variant: compress before solve



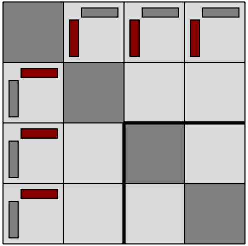
- FSCU (Factor, Solve, Compress, Update)
- FSCU+LUAR
- FCSU(+LUAR)
  - Restricted pivoting

# FCSU variant: compress before solve



- FSCU (Factor, Solve, Compress, Update)
- FSCU+LUAR
- FCSU(+LUAR)
  - Restricted pivoting
  - Low-rank Solve ⇒ flop reduction

# FCSU variant: compress before solve



- FSCU (Factor, Solve, Compress, Update)
- FSCU+LUAR
- FCSU(+LUAR)
  - Restricted pivoting
  - Low-rank Solve ⇒ flop reduction

On matrix S3,
**2.6** gain becomes **3.7**

| | flops (TF) | time (s) | residual |
|---|---|---|---|
| FSCU | 8.2 | 160 | 1.5e-09 |
| FCSU | 4.0 | 111 | 2.7e-09 |

# Variants improve asymptotic complexity

We have theoretically proven that:

| | FSCU | $\rightarrow$ FSCU+LUAR | $\rightarrow$ FCSU+LUAR |
|---|---|---|---|
| dense | $\mathcal{O}\left(m^{5/2}r^{1/2}\right)$ | $\rightarrow \mathcal{O}\left(m^{7/3}r^{2/3}\right)$ | $\rightarrow \mathcal{O}\left(m^{2}r\right)$ |
| sparse (3D) | $\mathcal{O}\left(n^{5/3}r^{1/2}\right)$ | $\rightarrow \mathcal{O}\left(n^{14/9}r^{2/3}\right)$ | $\rightarrow \mathcal{O}\left(n^{4/3}r\right)$ |

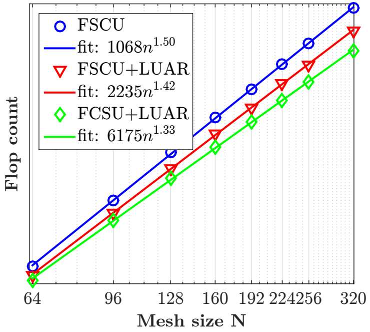
- Amestoy, Buttari, L'Excellent, and Mary. *On the Complexity of the Block Low-Rank Multifrontal Factorization*, SIAM J. Sci. Comput., 2017.

We have theoretically proven that:

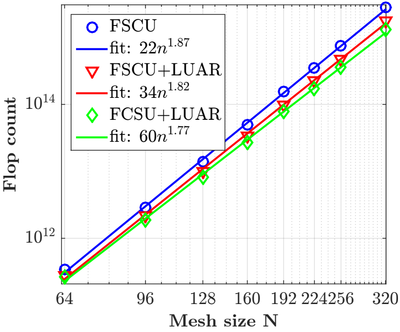
| | FSCU | | FSCU+LUAR | | FCSU+LUAR |
|---|---|---|---|---|---|
| dense | $\mathcal{O}\left(m^{5/2}r^{1/2}\right)$ | $\to$ | $\mathcal{O}\left(m^{7/3}r^{2/3}\right)$ | $\to$ | $\mathcal{O}\left(m^{2}r\right)$ |
| sparse (3D) | $\mathcal{O}\left(n^{5/3}r^{1/2}\right)$ | $\to$ | $\mathcal{O}\left(n^{14/9}r^{2/3}\right)$ | $\to$ | $\mathcal{O}\left(n^{4/3}r\right)$ |

- Amestoy, Buttari, L'Excellent, and Mary. *On the Complexity of the Block Low-Rank Multifrontal Factorization*, SIAM J. Sci. Comput., 2017.

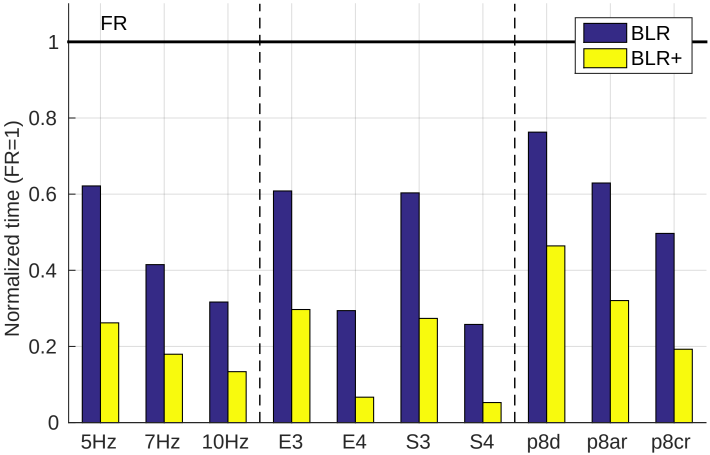
Poisson ($\varepsilon = 10^{-10}$)




Helmholtz ($\varepsilon = 10^{-4}$)

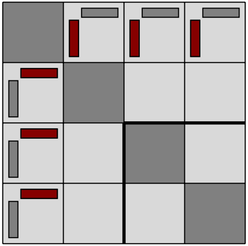


# Multicore performance results (24 threads)



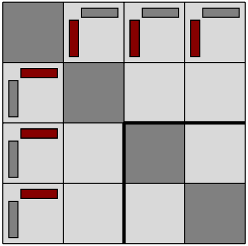

- "BLR": FSCU, right-looking, node only multithreading
- "BLR+": FCSU+LUAR, left-looking, node+tree multithreading


▶ Amestoy, Buttari, L'Excellent, and Mary. *Performance and Scalability of the Block Low-Rank Multifrontal Factorization on Multicore Architectures*, submitted to ACM Trans. Math. Soft., 2017.



- FSCU (Factor, Solve, Compress, Update)
- FSCU+LUAR
- FCSU(+LUAR)
  - Restricted pivoting
  - Low-rank Solve $\Rightarrow$ flop reduction

# The problem with FCSU



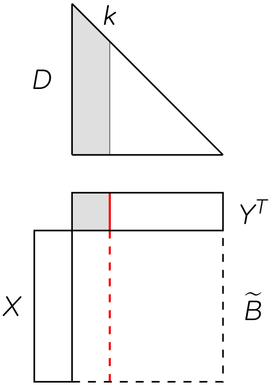
- FSCU (Factor, Solve, Compress, Update)
- FSCU+LUAR
- FCSU(+LUAR)
  - Restricted pivoting ⇒ **not acceptable in many applications**
  - Low-rank Solve ⇒ flop reduction

# Compress before Solve + pivoting: CFSU variant

| FSCU | FCSU | CFSU |
|---|---|---|
| ☺ Standard pivoting | ☹ Restricted pivoting | ☺ Standard pivoting |
| ☹ Compress after Solve | ☺ Compress before Solve | ☺ Compress before Solve |

# Compress before Solve + pivoting: CFSU variant

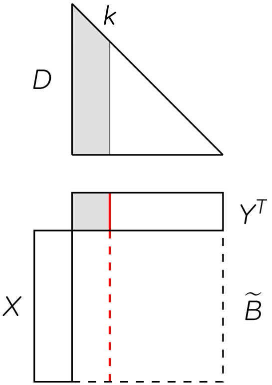
| FSCU | FCSU | CFSU |
|---|---|---|
| ☺ Standard pivoting | ☹ Restricted pivoting | ☺ Standard pivoting |
| ☹ Compress after Solve | ☺ Compress before Solve | ☺ Compress before Solve |



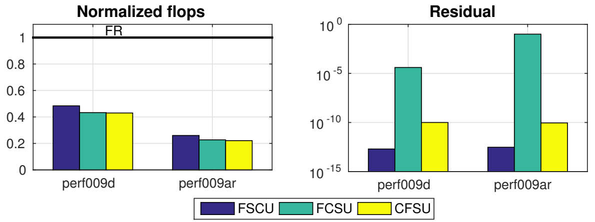

How to assess the quality of pivot $k$?
We need to estimate $\|\widetilde{B}_{:,k}\|_{max}$:
$\|\widetilde{B}_{:,k}\|_{max} \leq \|\widetilde{B}_{:,k}\|_2 = \|XY^T_{k,:}\|_2 = \|Y^T_{k,:}\|_2$,
assuming $X$ is orthonormal (e.g. RRQR, SVD)

# Compress before Solve + pivoting: CFSU variant

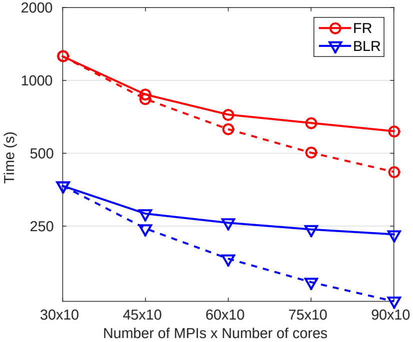
| FSCU | FCSU | CFSU |
|---|---|---|
| ☺ Standard pivoting | ☹ Restricted pivoting | ☺ Standard pivoting |
| ☹ Compress after Solve | ☺ Compress before Solve | ☺ Compress before Solve |




How to assess the quality of pivot $k$?
We need to estimate $\|\widetilde{B}_{:,k}\|_{max}$:
$\|\widetilde{B}_{:,k}\|_{max} \leq \|\widetilde{B}_{:,k}\|_2 = \|XY^T_{k,:}\|_2 = \|Y^T_{k,:}\|_2$,
assuming $X$ is orthonormal (e.g. RRQR, SVD)



# Distributed-memory BLR factorization

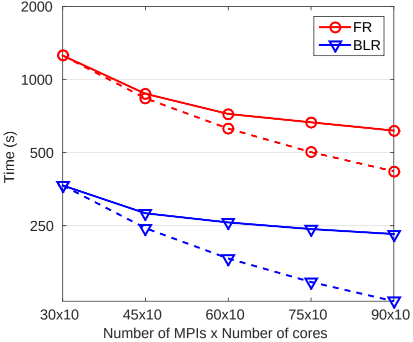



Experiments performed on the eos supercomputer (credits: CALMIP):

- Two Intel(r) 10-cores Ivy Bridge @ 2.8 GHz
- Peak per core is 22.4 GF/s
- 64 GB memory per node
- Infiniband FDR interconnect

How to improve the scalability of the BLR factorization?

# Strong scalability analysis (matrix 10Hz)



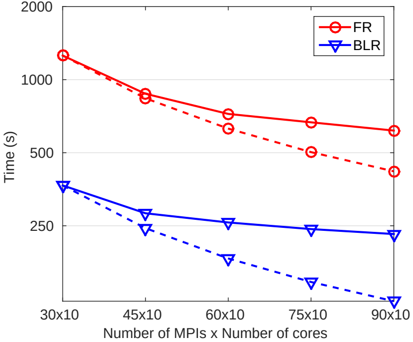

Experiments performed on the eos supercomputer (credits: CALMIP):

- Two Intel(r) 10-cores Ivy Bridge @ 2.8 GHz
- Peak per core is 22.4 GF/s
- 64 GB memory per node
- Infiniband FDR interconnect

How to improve the scalability of the BLR factorization?

- Load imbalance (ratio between most and less loaded processes) increases from $1.3$ (FR) to $2.6$ (BLR)
  $\Rightarrow$ we devised some heuristics showing promising gains

# Strong scalability analysis (matrix 10Hz)



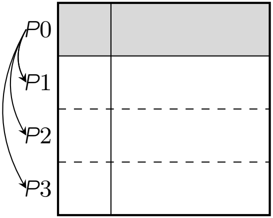

Experiments performed on the eos supercomputer (credits: CALMIP):

- Two Intel(r) 10-cores Ivy Bridge @ 2.8 GHz
- Peak per core is 22.4 GF/s
- 64 GB memory per node
- Infiniband FDR interconnect

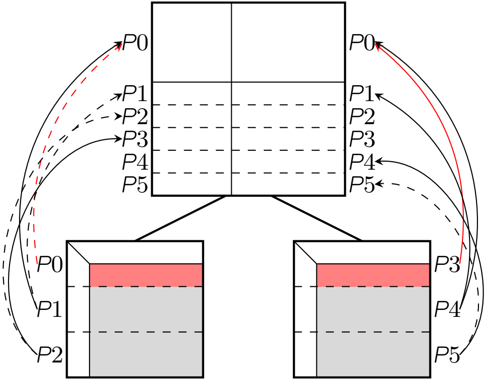
How to improve the scalability of the BLR factorization?

- Load imbalance (ratio between most and less loaded processes) increases from $1.3$ (FR) to $2.6$ (BLR)
  $\Rightarrow$ we devised some heuristics showing promising gains
- Flops reduced by $12.8$ but volume of communications only by $2.2 \Rightarrow$ higher relative weight of communications

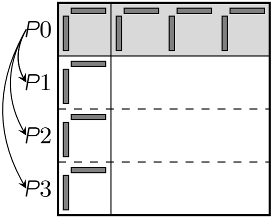
# Type of messages



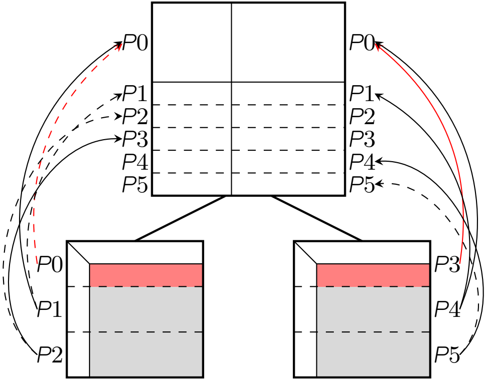

*LU* messages




CB messages



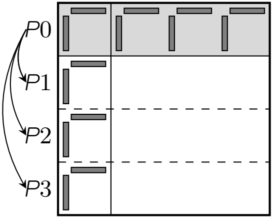

$LU$ messages



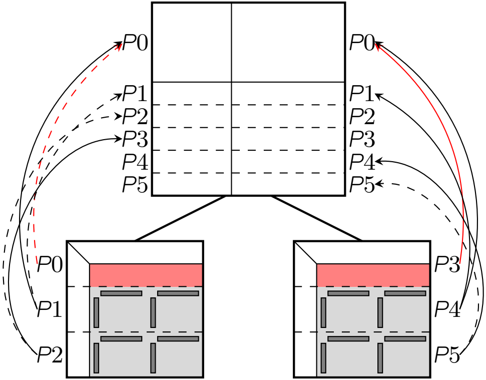

CB messages

- Volume of $LU$ messages is reduced by compressing the factors
  - ☺ Reduces operation count, communications, and memory consumption

# Type of messages



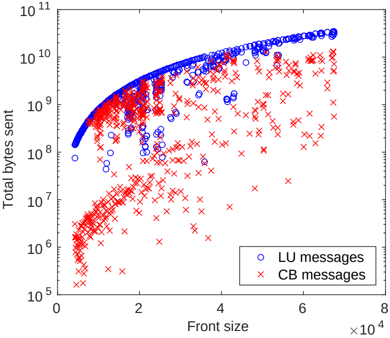

$LU$ messages




CB messages

- Volume of $LU$ messages is reduced by compressing the factors
  - ☺ Reduces operation count, communications, and memory consumption
- Volume of CB messages can be reduced by compressing the CB
  - ☺ Reduces communications and memory consumption
  - ☹ Increases operation count unless assembly is done in LR

# Communication analysis

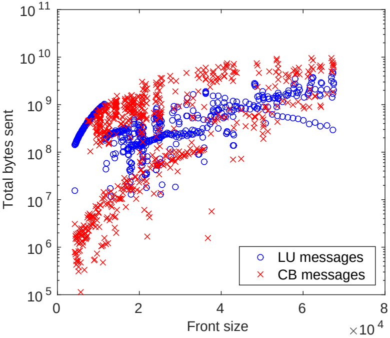



- FR case: *LU* messages dominate

Theoretical communication bounds

| | $\mathcal{W}_{LU}$ | $\mathcal{W}_{CB}$ | $\mathcal{W}_{tot}$ |
|---|---|---|---|
| FR | $\mathcal{O}\left(n^{4/3}p\right)$ | $\mathcal{O}\left(n^{4/3}\right)$ | $\mathcal{O}\left(n^{4/3}p\right)$ |

# Communication analysis

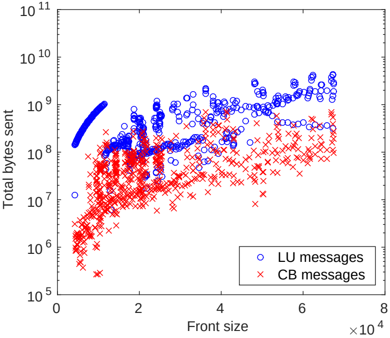



- FR case: *LU* messages dominate
- BLR case: CB messages dominate $\Rightarrow$ underwhelming reduction of communications

Theoretical communication bounds

| | $\mathcal{W}_{LU}$ | $\mathcal{W}_{CB}$ | $\mathcal{W}_{tot}$ |
|---|---|---|---|
| FR | $\mathcal{O}\left(n^{4/3}p\right)$ | $\mathcal{O}\left(n^{4/3}\right)$ | $\mathcal{O}\left(n^{4/3}p\right)$ |
| BLR ($CB_{FR}$) | $\mathcal{O}\left(nr^{1/2}p\right)$ | $\mathcal{O}\left(n^{4/3}\right)$ | $\mathcal{O}\left(nr^{1/2}p + n^{4/3}\right)$ |

# Communication analysis




- FR case: *LU* messages dominate
- BLR case: CB messages dominate $\Rightarrow$ underwhelming reduction of communications

$\Rightarrow$ CB compression allows for truly reducing the communications

Theoretical communication bounds

| | $\mathcal{W}_{LU}$ | $\mathcal{W}_{CB}$ | $\mathcal{W}_{tot}$ |
|---|---|---|---|
| FR | $\mathcal{O}\left(n^{4/3}p\right)$ | $\mathcal{O}\left(n^{4/3}\right)$ | $\mathcal{O}\left(n^{4/3}p\right)$ |
| BLR ($CB_{FR}$) | $\mathcal{O}\left(nr^{1/2}p\right)$ | $\mathcal{O}\left(n^{4/3}\right)$ | $\mathcal{O}\left(nr^{1/2}p + n^{4/3}\right)$ |
| BLR ($CB_{LR}$) | $\mathcal{O}\left(nr^{1/2}p\right)$ | $\mathcal{O}\left(nr^{1/2}\right)$ | $\mathcal{O}\left(nr^{1/2}p\right)$ |

# Results on very large problems (1/2)

Performance impact of the CB compression:
illustration on very large problems from SEISCOPE
(Helmholtz equation, unsymmetric complex):

| | | | |
|---|---|---|---|
| matrix | 10Hz | 15Hz | 20Hz |
| order | 17 M | 58 M | 130 M |
| cores | 900 Ivy Bridge | 900 Ivy Bridge | 2,400 Haswell |
| computer | eos (CALMIP) | eos (CALMIP) | occigen (CINES) |
| factor flops (FR) | 2.6 PF | 29.6 PF | 150.0 PF |
| ⇒ BLR ($CB_{FR}$) | 0.1 PF (5.3%) | 1.0 PF (3.3%) | 3.6 PF (2.4%) |
| ⇒ BLR ($CB_{LR}$) | 0.2 PF (6.1%) | 1.1 PF (3.7%) | 3.9 PF (2.6%) |
| factor time (FR) | 601 | 5,206 | n/a |
| ⇒ BLR ($CB_{FR}$) | 123 (4.9) | 838 (6.2) | 1,665 |
| ⇒ BLR ($CB_{LR}$) | 213 (2.8) | 856 (6.1) | 2,641 |
| $CB_{LR}$ time impact | +73% | +2% | +58% |
| comm. volume (FR) | 5.3 TB | 29.6 TB | n/a |
| comm. volume ($CB_{FR}$) | 1.7 TB (3.2) | 13.3 TB ( 2.2) | 79.8 TB |
| comm. volume ($CB_{LR}$) | 0.6 TB (9.1) | 1.2 TB (23.2) | 8.6 TB |

⇒ CB compression becomes increasingly critical?

# Results on very large problems (1/2)

Performance impact of the CB compression:
illustration on very large problems from SEISCOPE
(Helmholtz equation, unsymmetric complex):

| | | | |
|---|---|---|---|
| matrix | 10Hz | 15Hz | 20Hz |
| order | 17 M | 58 M | 130 M |
| cores | 900 Ivy Bridge | 900 Ivy Bridge | 2,400 Haswell |
| computer | eos (CALMIP) | eos (CALMIP) | occigen (CINES) |
| factor flops (FR) | 2.6 PF | 29.6 PF | 150.0 PF |
| ⇒ BLR ($CB_{FR}$) | 0.1 PF (5.3%) | 1.0 PF (3.3%) | 3.6 PF (2.4%) |
| ⇒ BLR ($CB_{LR}$) | 0.2 PF (6.1%) | 1.1 PF (3.7%) | 3.9 PF (2.6%) |
| factor time (FR) | 601 | 5,206 | n/a |
| ⇒ BLR ($CB_{FR}$) | 123 (4.9) | 838 (6.2) | 1,665 |
| ⇒ BLR ($CB_{LR}$) | 213 (2.8) | 856 (6.1) | 2,641 |
| $CB_{LR}$ time impact | +73% | +2% | +58% |
| comm. volume (FR) | 5.3 TB | 29.6 TB | n/a |
| comm. volume ($CB_{FR}$) | 1.7 TB (3.2) | 13.3 TB ( 2.2) | 79.8 TB |
| comm. volume ($CB_{LR}$) | 0.6 TB (9.1) | 1.2 TB (23.2) | 8.6 TB |

⇒ CB compression becomes increasingly critical?

# Results on very large problems (1/2)

Performance impact of the CB compression:
illustration on very large problems from SEISCOPE
(Helmholtz equation, unsymmetric complex):

| | | | |
|---|---|---|---|
| matrix | 10Hz | 15Hz | 20Hz |
| order | 17 M | 58 M | 130 M |
| cores | 900 Ivy Bridge | 900 Ivy Bridge | 2,400 Haswell |
| computer | eos (CALMIP) | eos (CALMIP) | occigen (CINES) |
| factor flops (FR) | 2.6 PF | 29.6 PF | 150.0 PF |
| ⇒ BLR ($CB_{FR}$) | 0.1 PF (5.3%) | 1.0 PF (3.3%) | 3.6 PF (2.4%) |
| ⇒ BLR ($CB_{LR}$) | 0.2 PF (6.1%) | 1.1 PF (3.7%) | 3.9 PF (2.6%) |
| factor time (FR) | 601 | 5,206 | n/a |
| ⇒ BLR ($CB_{FR}$) | 123 (4.9) | 838 (6.2) | 1,665 |
| ⇒ BLR ($CB_{LR}$) | 213 (2.8) | 856 (6.1) | 2,641 |
| $CB_{LR}$ time impact | +73% | +2% | +58% |
| comm. volume (FR) | 5.3 TB | 29.6 TB | n/a |
| comm. volume ($CB_{FR}$) | 1.7 TB (3.2) | 13.3 TB ( 2.2) | 79.8 TB |
| comm. volume ($CB_{LR}$) | 0.6 TB (9.1) | 1.2 TB (23.2) | 8.6 TB |

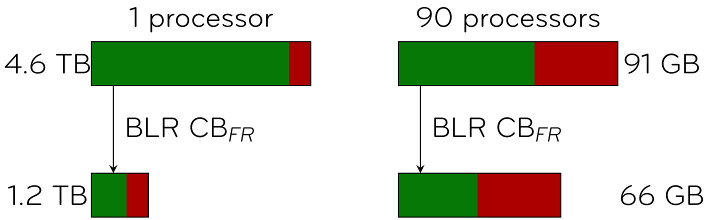
⇒ CB compression becomes increasingly critical?

# Results on very large problems (2/2)

Memory consumption on matrix 15Hz: **factors** + **active memory**
(**CB** + **active front**)

# Results on very large problems (2/2)

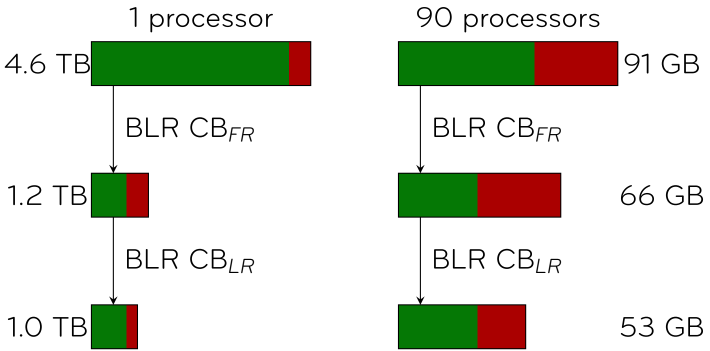
Memory consumption on matrix 15Hz: **factors** + **active memory** (**CB** + **active front**)




- Factors compression (19% of FR) leads to important gains, but the BLR solver inherits the poor scalability of the active memory

# Results on very large problems (2/2)

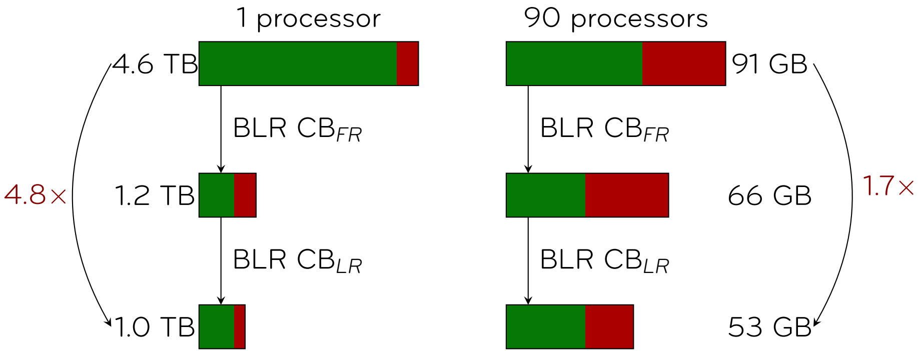
Memory consumption on matrix 15Hz: **factors** + **active memory** (**CB** + **active front**)




- Factors compression (19% of FR) leads to important gains, but the BLR solver inherits the poor scalability of the active memory
- CB compression (7% of FR) slightly attenuates this issue

# Results on very large problems (2/2)

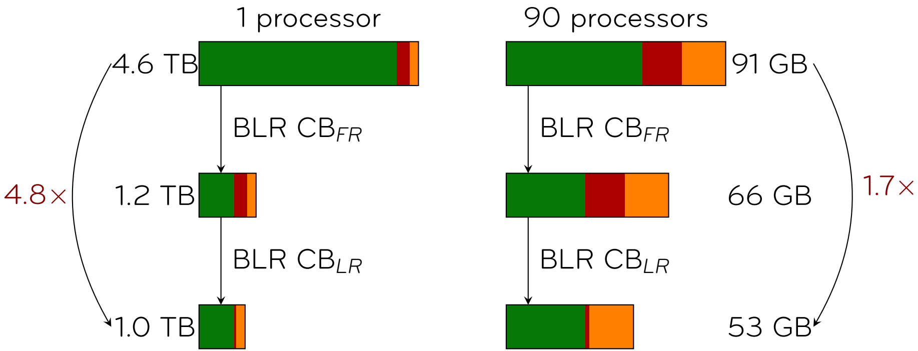
Memory consumption on matrix 15Hz: **factors** + **active memory** (**CB** + **active front**)




- Factors compression (19% of FR) leads to important gains, but the BLR solver inherits the poor scalability of the active memory
- CB compression (7% of FR) slightly attenuates this issue

# Results on very large problems (2/2)

Memory consumption on matrix 15Hz: **factors** + **active memory** (**CB** + **active front**)




- Factors compression (19% of FR) leads to important gains, but the BLR solver inherits the poor scalability of the active memory
- CB compression (7% of FR) slightly attenuates this issue
- Storage for the active front becomes critical

# Conclusion

# Summary of the work presented in this talk

## Asymptotic complexity reduction (Chap. 4)

Theoretical proof of the asymptotic complexity reduction: $\mathcal{O}(n \log n)$ memory and $\mathcal{O}(n^{5/3})$ operations for standard BLR variant

## Efficient and scalable algorithms (Chap. 5 and 6)

Efficient and scalable BLR factorization algorithms for shared- and distributed-memory systems, to achieve actual performance gains

## Novel variants (Chap. 2)

Novel variants to improve complexity (down to $\mathcal{O}(n^{4/3})$ operations) and performance of the BLR factorization, without sacrificing numerical pivoting

# Summary of the work *not* presented in this talk

## Recompression strategies (Chap. 3)

Analysis and comparison of several strategies to add and recompress low-rank updates, showing there is a trade-off between achieved rank and recompression overhead (with asymptotic difference between strategies)

## Applicative case-studies (Chap. 7)

Two case-studies from industrial applications (FWI and CSEM) based on frequency-domain inversion, showing that BLR direct solver is competitive with iterative or time-domain approaches

## Comparison with the HSS solver STRUMPACK (Chap. 8)

Comparison on real-life problems suggests that BLR works best as an accurate low-rank direct solver, while HSS favors more aggressive compression to build fast preconditioners

# Main perspectives (Chap. 9)

## Analysis and solution phases

The other two phases will become of growing importance:

- Analysis time can represent up to $50\%$ of the BLR factorization time
- Solution time is dominant for applications with multiple RHS
  $\Rightarrow$ how to translate factors size reduction into actual performance gains?

## Improving the memory scalability

- Active front becomes dominant and limits memory scalability:
  - Switch to fully-structured (matrix-free) implementation?
  - Panel by panel allocation and compression
- Memory aware mappings: map critical fronts on more processes to improve memory scalability

# References

## Publications

On complexity (Chapter 4):

- Amestoy, Buttari, L'Excellent, and Mary. *On the Complexity of the Block Low-Rank Multifrontal Factorization*, SIAM J. Sci. Comput., 2017.

On multicore performance (Chapter 5):

- Amestoy, Buttari, L'Excellent, and Mary. *Performance and Scalability of the Block Low-Rank Multifrontal Factorization on Multicore Architectures*, submitted to ACM Trans. Math. Soft., 2017.

On the seismic application (Section 7.1):

- Amestoy, Brossier, Buttari, L'Excellent, Mary, Métivier, Miniussi, and Operto. *Fast 3D frequency-domain full waveform inversion with a parallel Block Low-Rank multifrontal direct solver: application to OBC data from the North Sea*, Geophysics, 2016.

On the electromagnetic application (Section 7.2):

- Shantsev, Jaysaval, de la Kethulle de Ryhove, Amestoy, Buttari, L'Excellent, and Mary. *Large-scale 3D EM modeling with a Block Low-Rank multifrontal direct solver*, Geophysical Journal International, 2017.

## Software

- MUMPS 5.1.2

# Acknowledgments

Thank you for your attention


## Computing centers

We thank our computing centers for providing access to the machines:

- brunch (LIP)
- eos (CALMIP)
- occigen (CINES)

## Applicative collaborators

We thank our applicative collaborators for providing access to matrices from:

- seismic imaging (SEISCOPE)
- electromagnetic modeling (EMGS)
- structural mechanics (EDF)