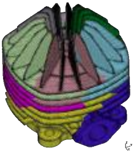
# Block Low-Rank multifrontal sparse direct solvers

P. Amestoy[*,1] A. Buttari[*,2] J.-Y. L'Excellent[†,3] T. Mary[*,4]

[*]Université de Toulouse [†]ENS Lyon
[1]INPT-IRIT [2]CNRS-IRIT [3]INRIA-LIP [4]UPS-IRIT

Mathias 2017, 25-27 Oct. 2017, Paris

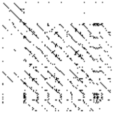
# Introduction



Discretization of a physical problem
(e.g. Code_Aster, finite elements)

$\Downarrow$

$$\mathbf{A}\ \mathbf{X} = \mathbf{B}$$



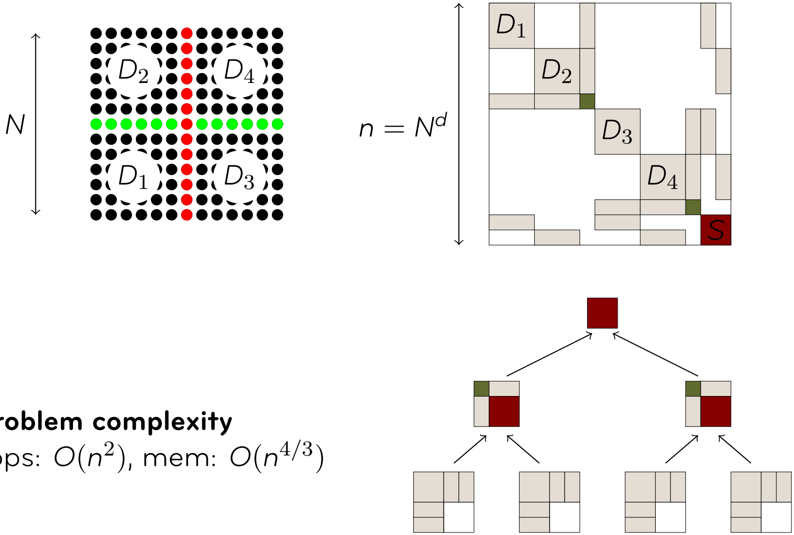
**A** large and sparse, **B** dense or sparse
Sparse direct methods : $\mathbf{A} = \mathbf{LU}$ ($\mathbf{LDL^T}$)

*Often a significant part of simulation cost*

**Objective discussed in this presentation:**
**how to reduce the cost of sparse direct solvers?**

*Focus on large-scale applications and architectures*

# Multifrontal Factorization with Nested Dissection

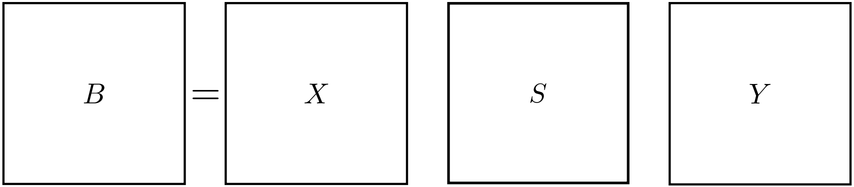



**3D problem complexity**

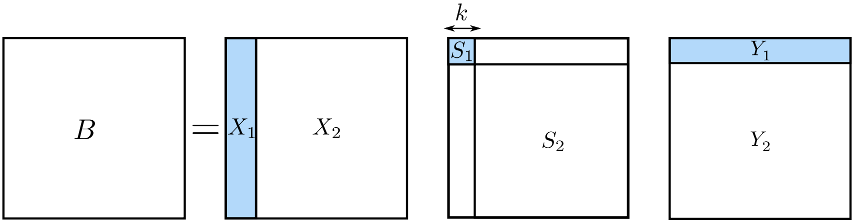
→ Flops: $O(n^2)$, mem: $O(n^{4/3})$

# Low-rank matrices

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 $B = XSY$:

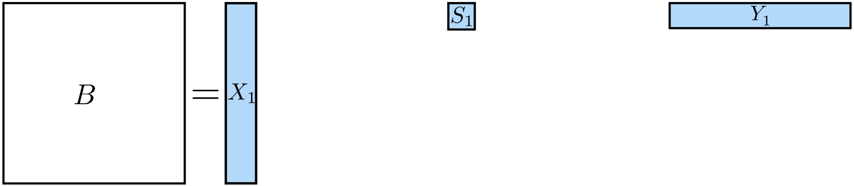



$$B = X_1 S_1 Y_1 + X_2 S_2 Y_2 \quad \text{with} \quad S_1(k,k) = \sigma_k > \varepsilon,\ S_2(1,1) = \sigma_{k+1} \leq \varepsilon$$

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 $B = XSY$:



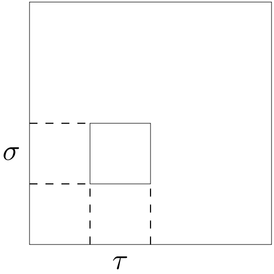

$B = X_1 S_1 Y_1 + X_2 S_2 Y_2$ with $S_1(k,k) = \sigma_k > \varepsilon,\ S_2(1,1) = \sigma_{k+1} \leq \varepsilon$

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 matrix formats

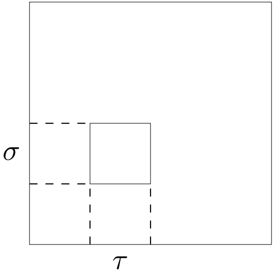
Frontal matrices are not low-rank but in some applications they exhibit low-rank blocks



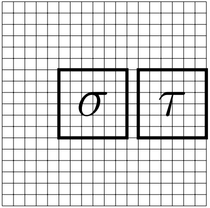

A block $B$ represents the interaction between two subdomains $\sigma$ and $\tau$.
If they have a small diameter and are far away their interaction is weak $\Rightarrow$ rank is low.

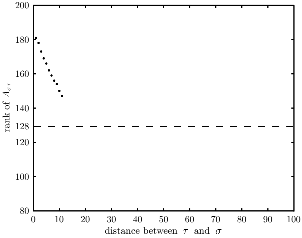
# Low-rank matrix formats

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

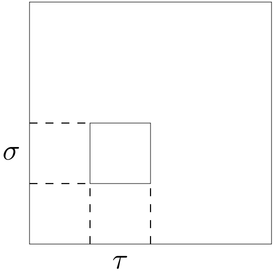



A block $B$ represents the interaction between two subdomains $\sigma$ and $\tau$.
If they have a small diameter and are far away their interaction is weak $\Rightarrow$ rank is low.

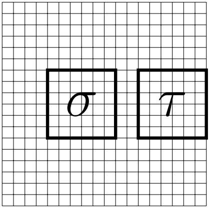


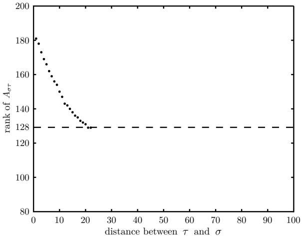



Frontal matrices are not low-rank but in some applications they exhibit low-rank blocks

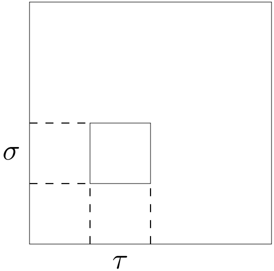



A block $B$ represents the interaction between two subdomains $\sigma$ and $\tau$.
If they have a small diameter and are far away their interaction is weak $\Rightarrow$ rank is low.

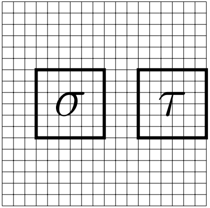


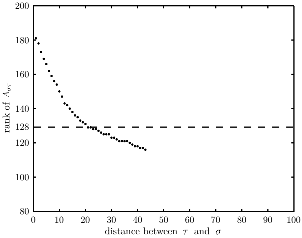



Frontal matrices are not low-rank but in some applications they exhibit low-rank blocks

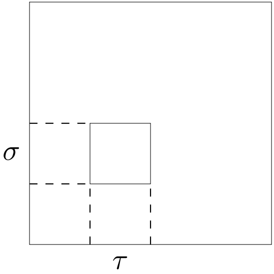



A block $B$ represents the interaction between two subdomains $\sigma$ and $\tau$.
If they have a small diameter and are far away their interaction is weak $\Rightarrow$ rank is low.

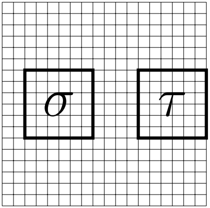


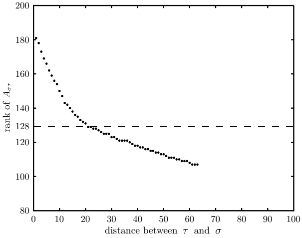



Frontal matrices are not low-rank but in some applications they exhibit low-rank blocks

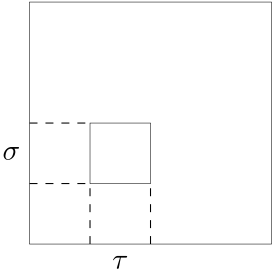



A block $B$ represents the interaction between two subdomains $\sigma$ and $\tau$.
If they have a small diameter and are far away their interaction is weak $\Rightarrow$ rank is low.

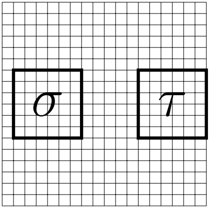


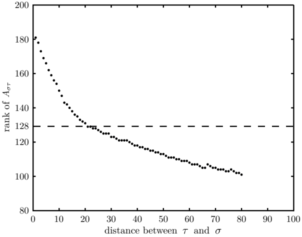



Frontal matrices are not low-rank but in some applications they exhibit low-rank blocks

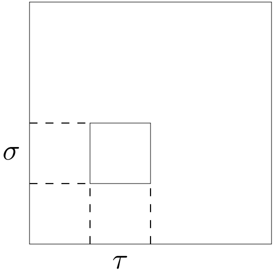



A block $B$ represents the interaction between two subdomains $\sigma$ and $\tau$.
If they have a small diameter and are far away their interaction is weak $\Rightarrow$ rank is low.

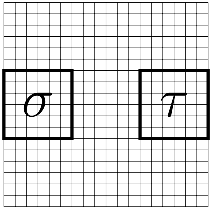


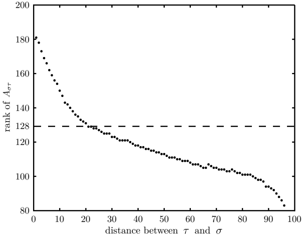



Frontal matrices are not low-rank but in some applications they exhibit low-rank blocks

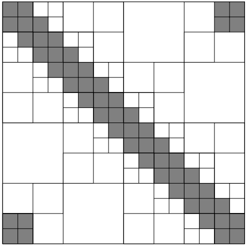



A block $B$ represents the interaction between two subdomains $\sigma$ and $\tau$.
If they have a small diameter and are far away their interaction is weak $\Rightarrow$ rank is low.

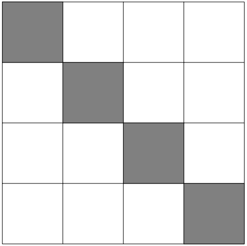


# $\mathcal{H}$ and BLR matrices

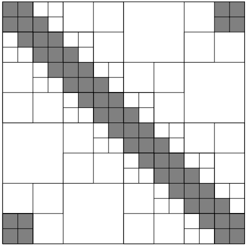


$\mathcal{H}$-matrix

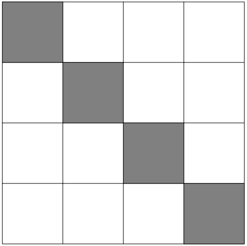


BLR matrix

# $\mathcal{H}$ and BLR matrices



$\mathcal{H}$-matrix

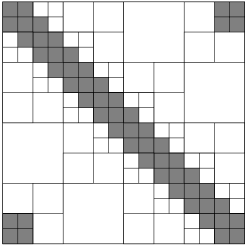


BLR matrix

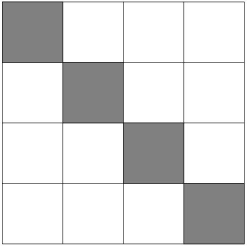
- Theoretical complexity can be as low as $O(n)$
- Complex, hierarchical structure

- Theoretical complexity can be as low as $O(n^{4/3})$
- Simple structure

# $\mathcal{H}$ and BLR matrices



$\mathcal{H}$-matrix



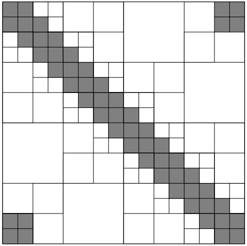
BLR matrix

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

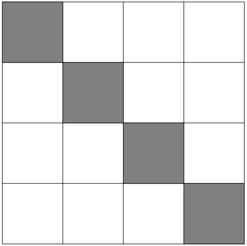
- Theoretical complexity can be as low as $O(n^{4/3})$
- Simple structure

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

# $\mathcal{H}$ and BLR matrices



$\mathcal{H}$-matrix



BLR matrix

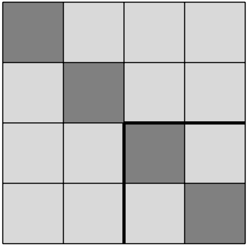
- Theoretical complexity can be as low as $O(n)$
- Complex, hierarchical structure

- Theoretical complexity can be as low as $O(n^{4/3})$
- Simple structure

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

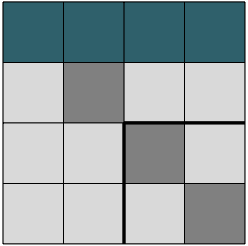
$\Rightarrow$ Ongoing collaboration with STRUMPACK team (LBNL) to compare BLR and hierarchical formats

# Standard BLR factorization: FSCU



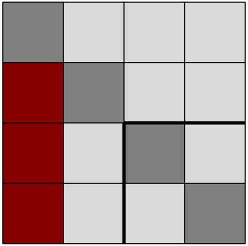
- FSCU

# Standard BLR factorization: FSCU



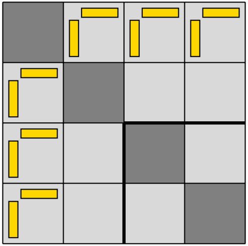
- FSCU (Factor,
- Easy to handle numerical pivoting, a critical feature often lacking in other low-rank solvers

# Standard BLR factorization: FSCU



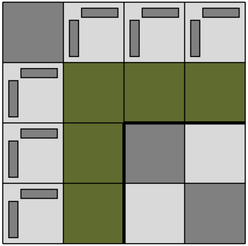
- FSCU (Factor, Solve,
- Easy to handle numerical pivoting, a critical feature often lacking in other low-rank solvers

# Standard BLR factorization: FSCU



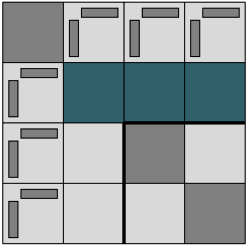
- FSCU (Factor, Solve, Compress,
- Easy to handle numerical pivoting, a critical feature often lacking in other low-rank solvers

# Standard BLR factorization: FSCU



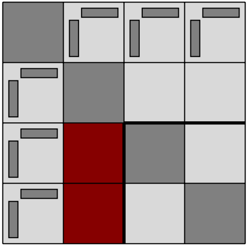
- FSCU (Factor, Solve, Compress, Update)
- Easy to handle numerical pivoting, a critical feature often lacking in other low-rank solvers

# Standard BLR factorization: FSCU



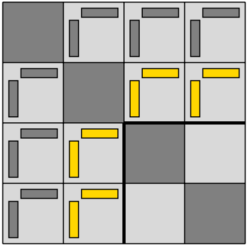
- FSCU (Factor, Solve, Compress, Update)
- Easy to handle numerical pivoting, a critical feature often lacking in other low-rank solvers

# Standard BLR factorization: FSCU



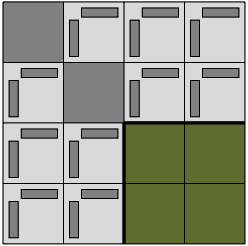
- FSCU (Factor, Solve, Compress, Update)
- Easy to handle numerical pivoting, a critical feature often lacking in other low-rank solvers

# Standard BLR factorization: FSCU



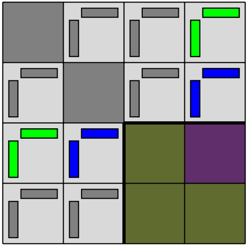
- FSCU (Factor, Solve, Compress, Update)
- Easy to handle numerical pivoting, a critical feature often lacking in other low-rank solvers

# Standard BLR factorization: FSCU



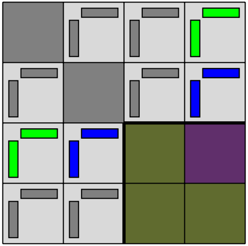
- FSCU (Factor, Solve, Compress, Update)
- Easy to handle numerical pivoting, a critical feature often lacking in other low-rank solvers

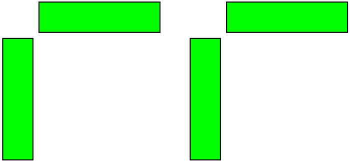
# Standard BLR factorization: FSCU



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

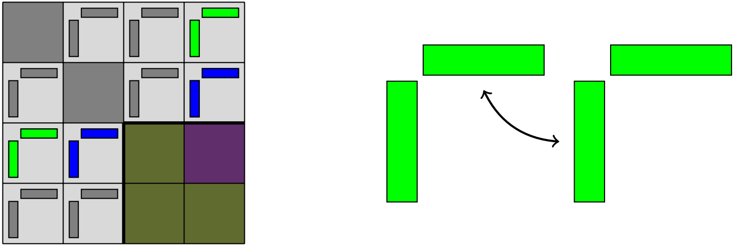
# Standard BLR factorization: FSCU





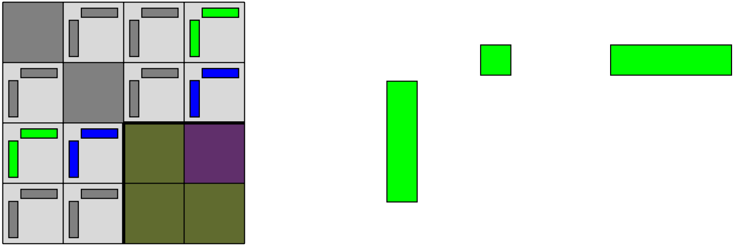
- FSCU (Factor, Solve, Compress, Update)
- Easy to handle numerical pivoting, a critical feature often lacking in other low-rank solvers

# Standard BLR factorization: FSCU



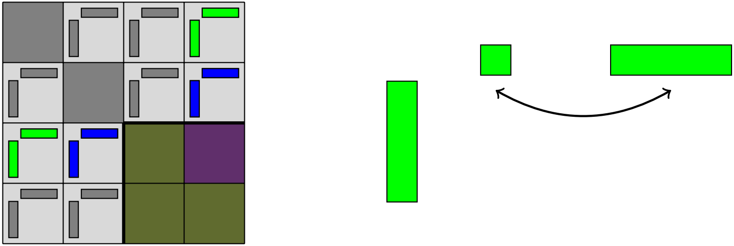
- FSCU (Factor, Solve, Compress, Update)
- Easy to handle numerical pivoting, a critical feature often lacking in other low-rank solvers

# Standard BLR factorization: FSCU



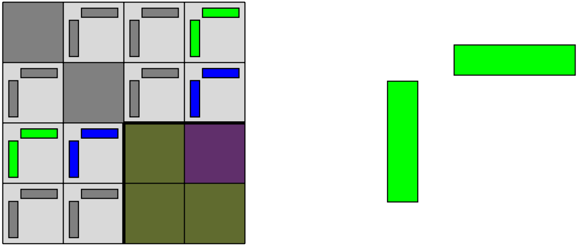
- FSCU (Factor, Solve, Compress, Update)
- Easy to handle numerical pivoting, a critical feature often lacking in other low-rank solvers

# Standard BLR factorization: FSCU



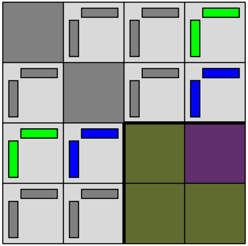
- FSCU (Factor, Solve, Compress, Update)
- Easy to handle numerical pivoting, a critical feature often lacking in other low-rank solvers

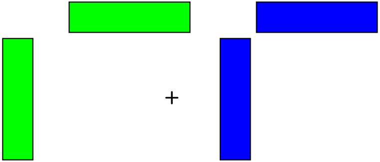
# Standard BLR factorization: FSCU



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

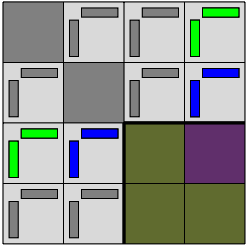
# Standard BLR factorization: FSCU

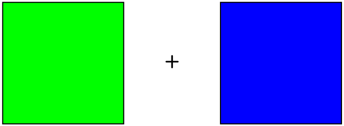




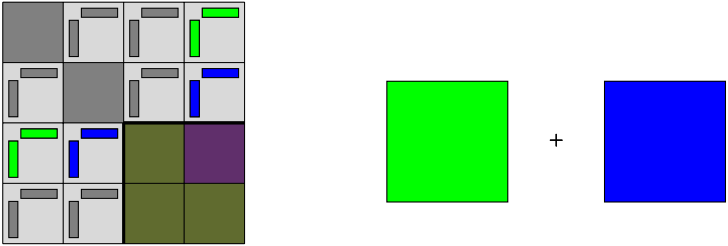

- FSCU (Factor, Solve, Compress, Update)
- Easy to handle numerical pivoting, a critical feature often lacking in other low-rank solvers

# Standard BLR factorization: FSCU



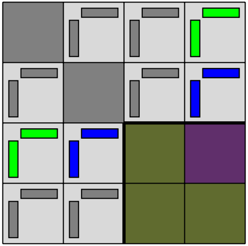


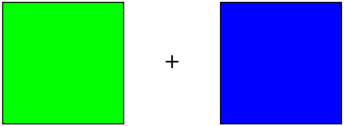
- FSCU (Factor, Solve, Compress, Update)
- Easy to handle numerical pivoting, a critical feature often lacking in other low-rank solvers



- 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



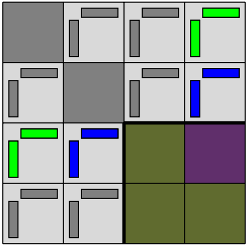


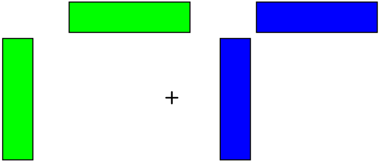
- FSCU (Factor, Solve, Compress, Update)
- Easy to handle numerical pivoting, a critical feature often lacking in other low-rank solvers
- Potential of this variant was studied in
  - Amestoy, Ashcraft, Boiteau, Buttari, L'Excellent, and Weisbecker. *Improving Multifrontal Methods by Means of Block Low-Rank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

**...but it had much room for improvement**

# Novel variants to improve the BLR factorization

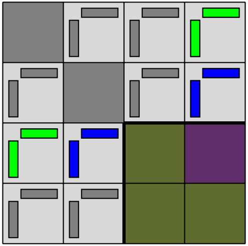
# LUAR variant: accumulation and recompression

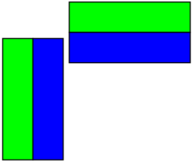





- FSCU (Factor, Solve, Compress, Update)
- FSCU+LUAR

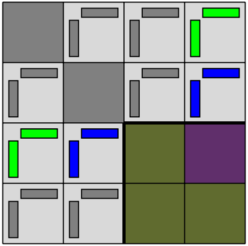
# LUAR variant: accumulation and recompression

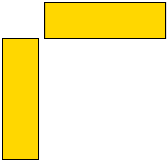




- FSCU (Factor, Solve, Compress, Update)
- FSCU+LUAR
  - Better granularity in Update operations

# LUAR variant: accumulation and recompression

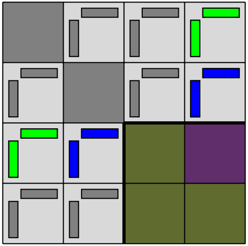


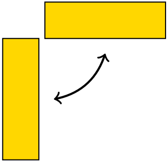


- FSCU (Factor, Solve, Compress, Update)
- FSCU+LUAR
  - Better granularity in Update operations
  - Potential recompression ⇒ complexity reduction: $O(n^{\frac{5}{3}}) \rightarrow O(n^{\frac{14}{9}})$

    ⇒ Collaboration with LSTC to design efficient recompression strategies

# LUAR variant: accumulation and recompression

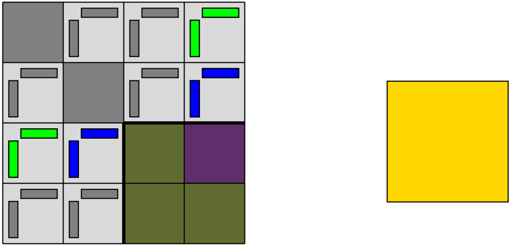




- FSCU (Factor, Solve, Compress, Update)
- FSCU+LUAR
  - Better granularity in Update operations
  - Potential recompression ⇒ complexity reduction: $O(n^{\frac{5}{3}}) \rightarrow O(n^{\frac{14}{9}})$

    ⇒ Collaboration with LSTC to design efficient recompression strategies

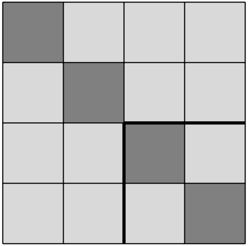
# LUAR variant: accumulation and recompression



- FSCU (Factor, Solve, Compress, Update)
- FSCU+LUAR
  - Better granularity in Update operations
  - Potential recompression ⇒ complexity reduction: $O(n^{\frac{5}{3}}) \to O(n^{\frac{14}{9}})$

    ⇒ Collaboration with LSTC to design efficient recompression strategies

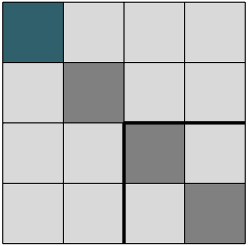
# FCSU variant: compress before solve



- FSCU (Factor, Solve, Compress, Update)
- FSCU+LUAR
  - Better granularity in Update operations
  - Potential recompression ⇒ complexity reduction: $O(n^{\frac{5}{3}}) \rightarrow O(n^{\frac{14}{9}})$

    ⇒ Collaboration with LSTC to design efficient recompression strategies
- FCSU(+LUAR)

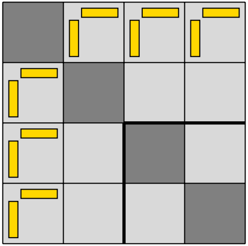
# FCSU variant: compress before solve



- FSCU (Factor, Solve, Compress, Update)
- FSCU+LUAR
  - Better granularity in Update operations
  - Potential recompression ⇒ complexity reduction: $O(n^{\frac{5}{3}}) \to O(n^{\frac{14}{9}})$

    ⇒ Collaboration with LSTC to design efficient recompression strategies
- FCSU(+LUAR)
  - Restricted pivoting, e.g. to diagonal blocks

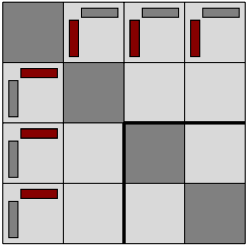
# FCSU variant: compress before solve



- FSCU (Factor, Solve, Compress, Update)
- FSCU+LUAR
  - Better granularity in Update operations
  - Potential recompression ⇒ complexity reduction: $O(n^{\frac{5}{3}}) \rightarrow O(n^{\frac{14}{9}})$

    ⇒ Collaboration with LSTC to design efficient recompression strategies
- FCSU(+LUAR)
  - Restricted pivoting, e.g. to diagonal blocks

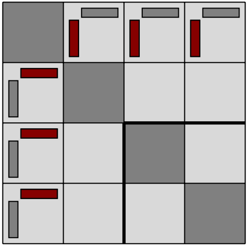
# FCSU variant: compress before solve



- FSCU (Factor, Solve, Compress, Update)
- FSCU+LUAR
  - Better granularity in Update operations
  - Potential recompression ⇒ complexity reduction: $O(n^{\frac{5}{3}}) \to O(n^{\frac{14}{9}})$

    ⇒ Collaboration with LSTC to design efficient recompression strategies
- FCSU(+LUAR)
  - Restricted pivoting, e.g. to diagonal blocks
  - Low-rank Solve ⇒ complexity reduction: $O(n^{\frac{14}{9}}) \to O(n^{\frac{4}{3}})$

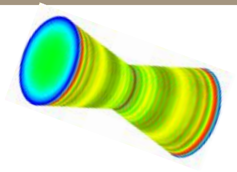
# FCSU variant: compress before solve

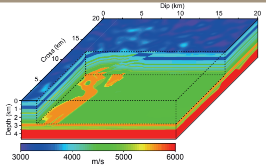


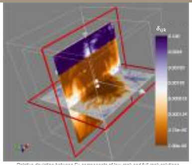
- FSCU (Factor, Solve, Compress, Update)
- FSCU+LUAR
  - Better granularity in Update operations
  - Potential recompression ⇒ complexity reduction: $O(n^{\frac{5}{3}}) \rightarrow O(n^{\frac{14}{9}})$

    ⇒ Collaboration with LSTC to design efficient recompression strategies
- FCSU(+LUAR)
  - Restricted pivoting, e.g. to diagonal blocks ⇒ not acceptable in many applications ⇒ encouraging results with new variant compatible with pivoting
  - Low-rank Solve ⇒ complexity reduction: $O(n^{\frac{14}{9}}) \rightarrow O(n^{\frac{4}{3}})$

# Performance and scalability of the BLR factorization





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 24 Haswell cores:

| application | factorization time (s) MUMPS | BLR | BLR+ | ratio |
|---|---|---|---|---|
| structural | 2066.9 | 1129.0 | 377.9 | 5.5 |
| seismic | 5649.5 | 1998.8 | 773.7 | 7.3 |
| electromag. | 13842.7 | 3702.9 | 736.1 | 18.8 |

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

# Distributed-memory performance results

- Volume of communications is reduced less than flops ⇒ higher relative weight of communications
- Low-rank compression cannot be predicted ⇒ load unbalance increases

# Distributed-memory performance results

- Volume of communications is reduced less than flops ⇒ higher relative weight of communications
- Low-rank compression cannot be predicted ⇒ load unbalance increases

⇒ Ongoing work to design strategies to overcome these issues

Results on 900 Ivy Bridge cores:

| application | factorization time (s) MUMPS | BLR | BLR+ | ratio |
|---|---|---|---|---|
| structural | 263.0 | 156.9 | 104.9 | 2.5 |
| seismic | 600.9 | 231.2 | 123.4 | 4.9 |
| electromag. | 1242.6 | 454.3 | 233.8 | 5.3 |

# Result on a very large problem

Result on matrix 15Hz (order $58 \times 10^6$, nnz $1.5 \times 10^9$)
on 900 cores:

| | flops (PF) | factors size (TB) | memory (GB) avg. | memory (GB) max. | elapsed time (s) ana. | elapsed time (s) fac. | elapsed time (s) sol. |
|---|---|---|---|---|---|---|---|
| MUMPS | 29.6 | 3.7 | 103 | 120 | OOM | OOM | OOM |
| BLR | 1.3 | 0.7 | 37 | 57 | 437 | 856 | 0.2/RHS |
| ratio | 22.9 | 5.1 | 2.8 | 2.3 | | | |

$\Rightarrow$ this result opens promising perspectives for
frequency-domain inversion with low-rank direct solver
even at high frequencies

# Conclusion

# References and acknowledgements

## Publications


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

## Acknowledgements


- LIP and CALMIP for providing access to the machines
- EMGS, SEISCOPE, and EDF for providing the matrices
- MUMPS consortium (EDF, Altair, Michelin, LSTC, Siemens, ESI, Total, FFT, Safran, LBNL)

Thanks!
Questions?

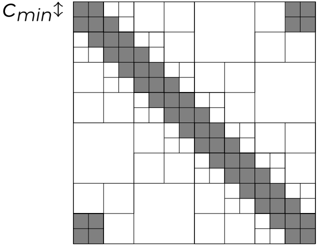
# Backup Slides

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

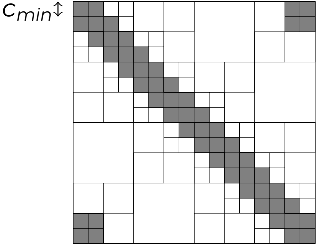
Until recently, BLR complexity was unknown.
Can we use $\mathcal{H}$ theory on BLR matrices?

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

Until recently, BLR complexity was unknown.
Can we use $\mathcal{H}$ theory on BLR matrices?



Until recently, BLR complexity was unknown.
Can we use $\mathcal{H}$ theory on BLR matrices?

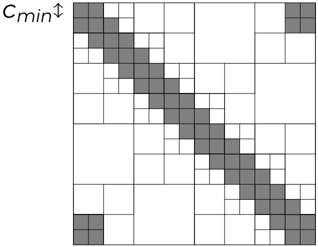



Complexity mainly depends on $r_{max}$, the maximal rank of the blocks

With $\mathcal{H}$ partitioning, $r_{max}$ is small

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

Until recently, BLR complexity was unknown.
Can we use $\mathcal{H}$ theory on BLR matrices?




Complexity mainly depends on $r_{max}$, the maximal rank of the blocks

With $\mathcal{H}$ partitioning, $r_{max}$ is small

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

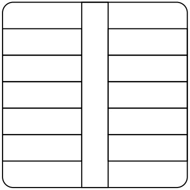
Problem: in $\mathcal{H}$ formalism, the maxrank of the blocks of a BLR matrix is $r_{max} = b$ (due to the non-admissible blocks)

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

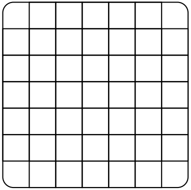
# Complexity of dense BLR factorization

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

$$\mathcal{P} \text{ is admissible } \Leftrightarrow N_{na} = \#\{\sigma \times \tau \in \mathcal{P}, \ \ \sigma \times \tau \text{ is not admissible}\} \leq q$$



Non-Admissible



Admissible

# Complexity of dense BLR factorization

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

$\mathcal{P}$ is admissible $\Leftrightarrow N_{na} = \#\{\sigma \times \tau \in \mathcal{P},\ \ \sigma \times \tau \text{ is not admissible}\} \leq q$


## Main result

There exists an admissible $\mathcal{P}$ for $q = O(1)$, s.t. the maxrank of the admissible blocks of $A$ is $r = O(r_{max}^{\mathcal{H}})$.
The complexity of the factorization of a dense matrix of order $m$ is thus:
$\mathcal{C}_{facto} = O(r^2 m^3/b^2 + mb^2q^2) = O(r^2m^3/b^2 + mb^2) = O(rm^2)$ (for $b = O(\sqrt{rm})$)

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

# Complexity experiments: problems

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

$$\Delta u = f$$

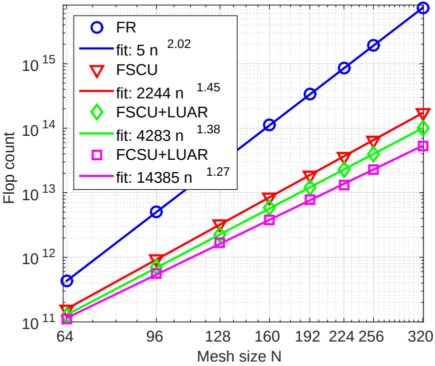
2. **`Helmholtz`**: $N^3$ grid with a 27-point stencil, $\omega$ is the angular frequency, $v(x)$ is the seismic velocity field, and $u(x, \omega)$ is the time-harmonic wavefield solution to the forcing term $s(x, \omega)$.

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

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

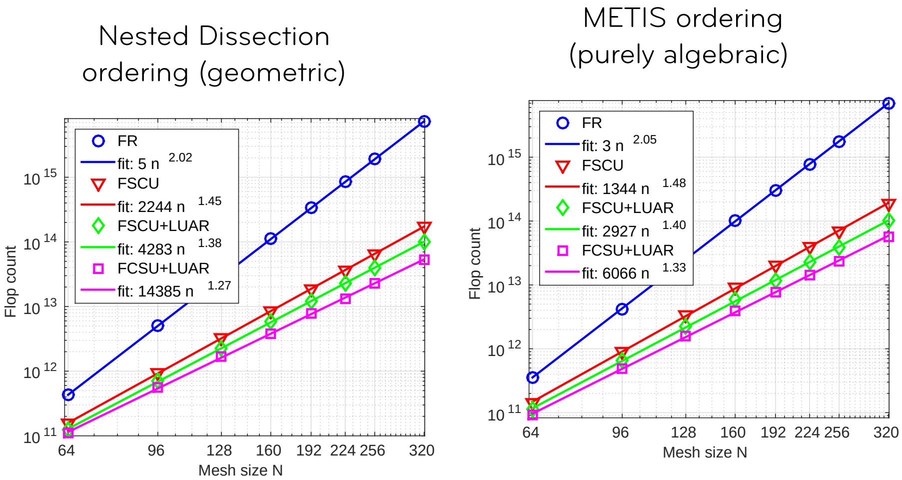
# Experimental MF flop complexity: Poisson ($\varepsilon = 10^{-10}$)

Nested Dissection ordering (geometric)




- good agreement with theoretical complexity ($O(n^2)$, $O(n^{1.67})$, $O(n^{1.55})$, and $O(n^{1.33})$)

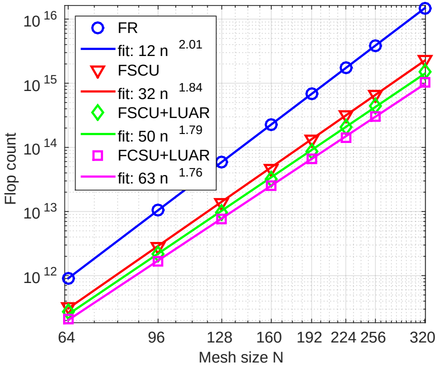
# Experimental MF flop complexity: Poisson ($\varepsilon = 10^{-10}$)



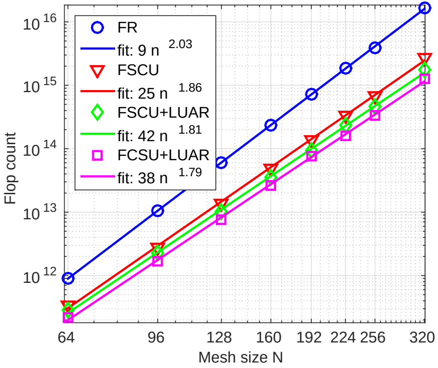

- good agreement with theoretical complexity ($O(n^2)$, $O(n^{1.67})$, $O(n^{1.55})$, and $O(n^{1.33})$)
- remains close to ND complexity with METIS ordering

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

Nested Dissection ordering (geometric)

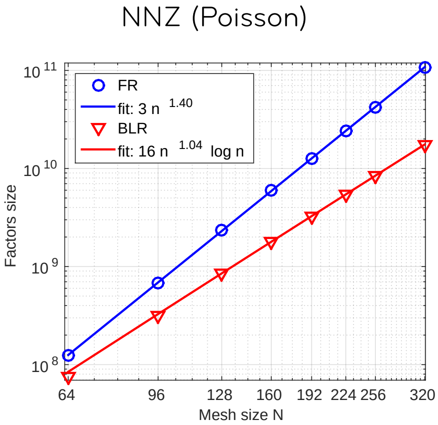


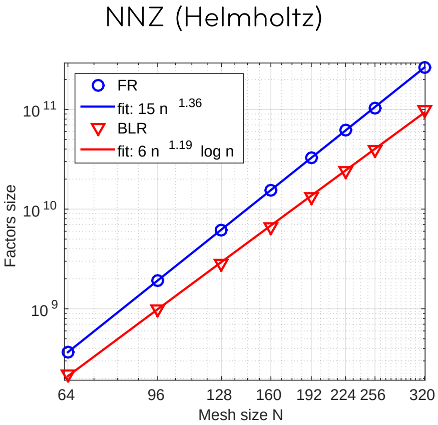

METIS ordering (purely algebraic)




- good agreement with theoretical complexity ($O(n^2)$, $O(n^{1.83})$, $O(n^{1.78})$, and $O(n^{1.67})$)
- remains close to ND complexity with METIS ordering

# Experimental MF complexity: factor size




- good agreement with theoretical complexity (FR: $O(n^{1.33})$; BLR: $O(n \log n)$ and $O(n^{1.17} \log n)$)

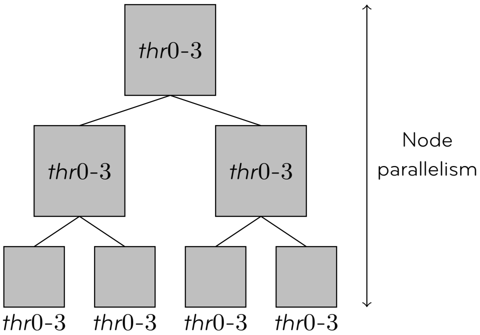
# Experimental Setting: Machines

Experiments are done on the shared-memory machines of the LIP laboratory of Lyon:

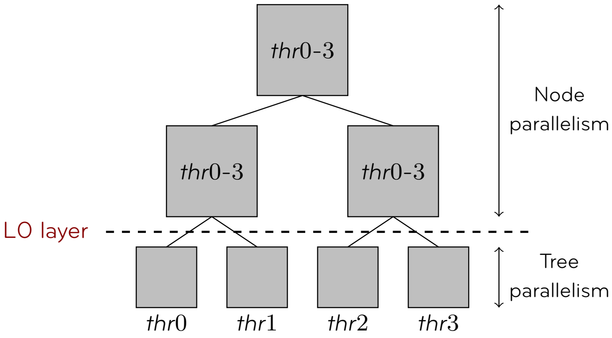
1. **brunch**
   - Four Intel(r) 24-cores Broadwell @ 2,2 GHz
   - Peak per core is 35.2 GF/s
   - Total memory is 1.5 TB

2. **grunch**
   - Two Intel(r) 14-cores Haswell @ 2,3 GHz
   - Peak per core is 36.8 GF/s
   - Total memory is 768 GB

# Exploiting tree-based multithreading in MF solvers



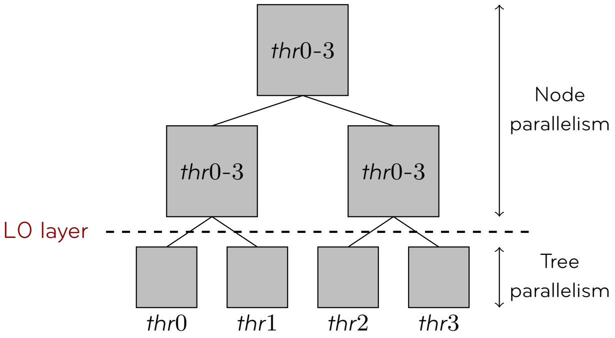
# Exploiting tree-based multithreading in MF solvers




- Work based on W. M. Sid-Lakhdar's PhD thesis
  - L0 layer computed with a variant of the Geist-Ng algorithm
  - NUMA-aware implementation
  - use of Idle Core Recycling technique (variant of work-stealing)

► L'Excellent and Sid-Lakhdar. *A study of shared-memory parallelism in a multifrontal solver*, Parallel Computing.

# Exploiting tree-based multithreading in MF solvers

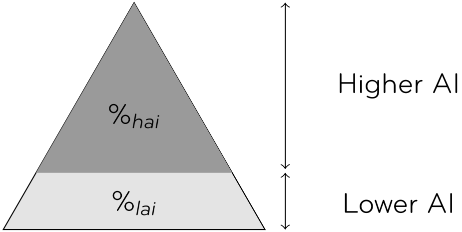



- Work based on W. M. Sid-Lakhdar's PhD thesis
  - L0 layer computed with a variant of the Geist-Ng algorithm
  - NUMA-aware implementation
  - use of Idle Core Recycling technique (variant of work-stealing)

▶ L'Excellent and Sid-Lakhdar. *A study of shared-memory parallelism in a multifrontal solver*, Parallel Computing.

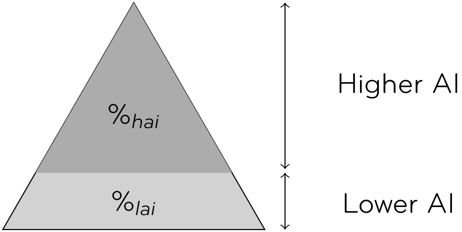
⇒ how big an impact can tree-based multithreading make?

# Impact of tree-based multithreading on BLR



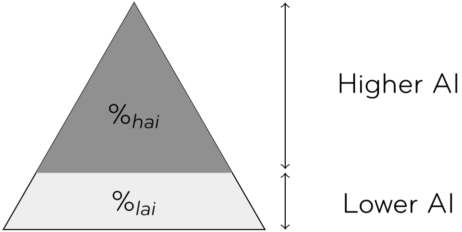

| | 24 threads | | 24 threads + tree MT | |
|---|---|---|---|---|
| | time | $\%_{lai}$ | time | $\%_{lai}$ |
| FR BLR | 509 | 21% | | |

# Impact of tree-based multithreading on BLR



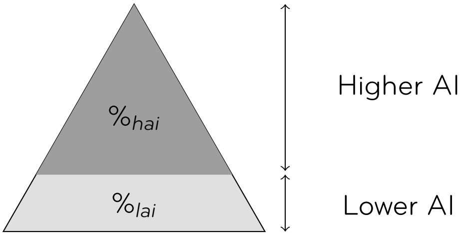

| | 24 threads | | 24 threads + tree MT | |
|---|---|---|---|---|
| | time | $\%_{lai}$ | time | $\%_{lai}$ |
| FR | 509 | 21% | | |
| BLR | 307 | 35% | | |

# Impact of tree-based multithreading on BLR




| | 24 threads | | 24 threads + tree MT | |
|---|---|---|---|---|
| | time | $\%_{lai}$ | time | $\%_{lai}$ |
| FR | 509 | 21% | 424 | 13% |
| BLR | 307 | 35% | | |

# Impact of tree-based multithreading on BLR




| | 24 threads | | 24 threads + tree MT | |
|---|---|---|---|---|
| | time | $\%_{lai}$ | time | $\%_{lai}$ |
| FR | 509 | 21% | 424 | 13% |
| BLR | 307 | 35% | 221 | 24% |

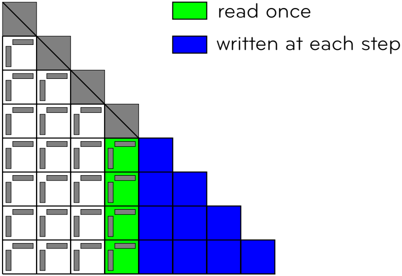
⇒ 1.7 gain becomes 1.9 thanks to tree-based MT

# Right Looking Vs. Left-Looking analysis

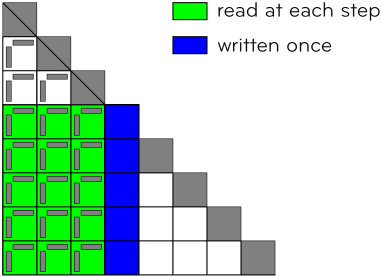
| | | FR | | BLR | |
|---|---|---|---|---|---|
| | | RL | LL | RL | LL |
| 1 thread | Update | 6467 | | 1064 | |
| | Total | 7390 | | 2242 | |
| 24 threads | Update | 338 | 336 | 110 | 67 |
| | Total | 424 | 421 | 221 | 175 |

# Right Looking Vs. Left-Looking analysis

| | | FR | | BLR | |
|---|---|---|---|---|---|
| | | RL | LL | RL | LL |
| 1 thread | Update | 6467 | | 1064 | |
| | Total | 7390 | | 2242 | |
| 24 threads | Update | 338 | 336 | 110 | 67 |
| | Total | 424 | 421 | 221 | 175 |



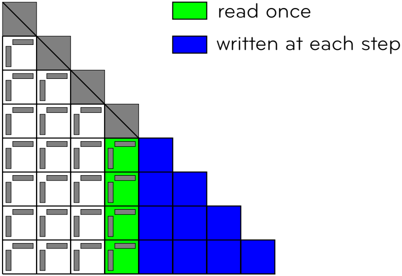
RL factorization



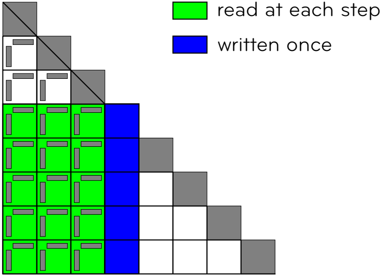
LL factorization

# Right Looking Vs. Left-Looking analysis

| | | FR | | BLR | |
|---|---|---|---|---|---|
| | | RL | LL | RL | LL |
| 1 thread | Update | 6467 | | 1064 | |
| | Total | 7390 | | 2242 | |
| 24 threads | Update | 338 | 336 | 110 | 67 |
| | Total | 424 | 421 | 221 | 175 |



RL factorization

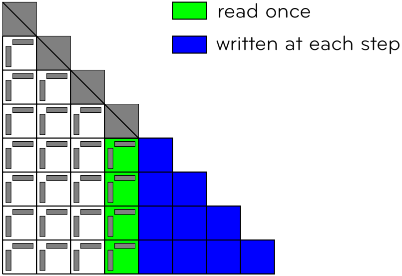


LL factorization

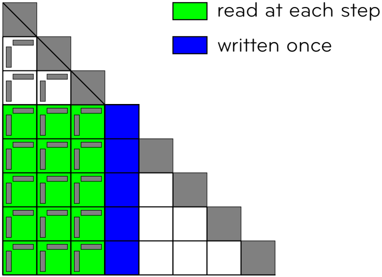
$\Rightarrow$ Lower volume of memory transfers in LL (more critical in MT)

# Right Looking Vs. Left-Looking analysis

| | | FR | | BLR | |
|---|---|---|---|---|---|
| | | RL | LL | RL | LL |
| 1 thread | Update | 6467 | | 1064 | |
| | Total | 7390 | | 2242 | |
| 24 threads | Update | 338 | 336 | 110 | 67 |
| | Total | 424 | 421 | 221 | 175 |




RL factorization

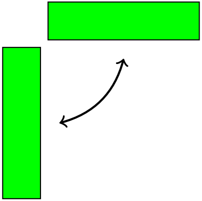



LL factorization

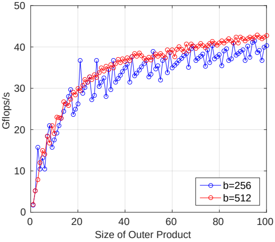
⇒ Lower volume of memory transfers in LL (more critical in MT)

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

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



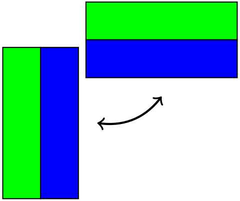
Double complex (z) performance benchmark of Outer Product

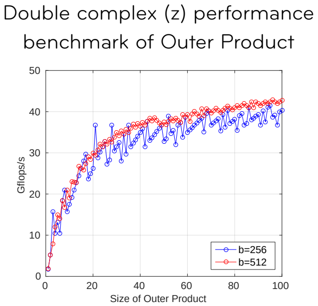



| | | LL | LUA | LUAR* |
|---|---|---|---|---|
| average size of Outer Product | | 16.5 | 61.0 | 32.8 |
| flops ($\times 10^{12}$) | Outer Product | 3.76 | 3.76 | 1.59 |
| | Total | 10.19 | 10.19 | 8.15 |
| time (s) | Outer Product | 21 | 14 | 6 |
| | Total | 175 | 167 | 160 |

* All metrics include the Recompression overhead

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

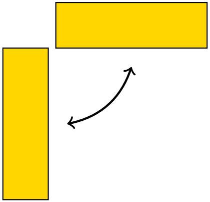




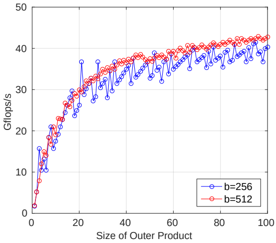

| | | LL | LUA | LUAR* |
|---|---|---|---|---|
| average size of Outer Product | | 16.5 | 61.0 | 32.8 |
| flops ($\times 10^{12}$) | Outer Product | 3.76 | 3.76 | 1.59 |
| | Total | 10.19 | 10.19 | 8.15 |
| time (s) | Outer Product | 21 | 14 | 6 |
| | Total | 175 | 167 | 160 |

* All metrics include the Recompression overhead

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



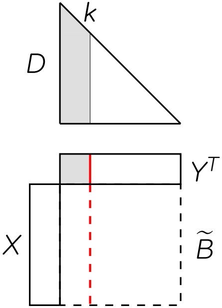
Double complex (z) performance benchmark of Outer Product




| | | LL | LUA | LUAR* |
|---|---|---|---|---|
| average size of Outer Product | | 16.5 | 61.0 | 32.8 |
| flops ($\times 10^{12}$) | Outer Product | 3.76 | 3.76 | 1.59 |
| | Total | 10.19 | 10.19 | 8.15 |
| time (s) | Outer Product | 21 | 14 | 6 |
| | Total | 175 | 167 | 160 |

* All metrics include the Recompression overhead

# Compress before Solve + pivoting: CFSU variant




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).

| matrix | residual | | | flops (% FR) | | |
|---|---|---|---|---|---|---|
| | FSCU | FCSU | CFSU | FSCU | FCSU | CFSU |
| af_shell10 | 2e-06 | 5e-06 | 4e-06 | 29.9 | 22.7 | 22.7 |
| Lin | 4e-05 | 4e-05 | 4e-05 | 24.0 | 18.5 | 18.5 |
| mario002 | 2e-06 | fail | 1e-06 | 82.8 | – | 72.2 |
| perf009ar | 3e-13 | 1e-01 | 9e-11 | 26.0 | 22.7 | 22.1 |