Complexity and performance of the Block Low-Rank multifrontal factorization and its variants

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

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

SIAM PP’16, Paris Apr. 12-15
Introduction
Multifrontal (Duff ’83) with Nested Dissection (George ’73)

\[ n = N^d \]

SIAM PP’16, Paris Apr. 12-15
Multifrontal (Duff ‘83) with Nested Dissection (George ‘73)

$N = N^d$

3D problem cost $\propto$

$\rightarrow$ Flops: $O(n^2)$, mem: $O(n^{4/3})$
Our hope is to find a good compromise between theoretical complexity and performance/usability.
Our hope is to find a good compromise between theoretical complexity and performance/usability.
A block $B$ represents the interaction between two subdomains. If they have a small diameter and are far away their interaction is weak $\Rightarrow$ rank is low.
A block $B$ represents the interaction between two subdomains. If they have a small diameter and are far away their interaction is weak $\Rightarrow$ rank is low.

$$\tilde{B} = XY^T$$ such that $\text{rank}(\tilde{B}) = k_\varepsilon$ and $\|B - \tilde{B}\| \leq \varepsilon$

If $k_\varepsilon \ll \text{size}(B)$ $\Rightarrow$ memory and flops can be reduced with a controlled loss of accuracy ($\leq \varepsilon$)
A block $B$ represents the interaction between two subdomains. If they have a small diameter and are far away their interaction is weak $\Rightarrow$ rank is low.

$$\tilde{B} = XY^T \text{ such that } \text{rank}(\tilde{B}) = k_\varepsilon \text{ and } \|B - \tilde{B}\| \leq \varepsilon$$

If $k_\varepsilon \ll \text{size}(B) \Rightarrow$ memory and flops can be reduced with a controlled loss of accuracy ($\leq \varepsilon$)
\( \mathcal{H} \) and BLR matrices

- \( \mathcal{H} \)-matrix
  - Leads to very low theoretical complexity
  - Complex, hierarchical structure

- BLR matrix
  - Simple structure
  - Theoretical complexity?
\( \mathcal{H} \) and BLR matrices

\( \mathcal{H} \)-matrix

BLR matrix

⇒ Our hope is to find a good compromise between theoretical complexity and performance/usability
Questions that will be answered in this talk

- Is the complexity of the BLR factorization asymptotically better than the full-rank one? (i.e., in $O(n^\alpha)$, with $\alpha < 2$ and where $n$ is the number of unknowns)

- What are the different variants of the BLR factorization? Do they improve its complexity?

- How well does the complexity improvement translate into a performance gain?

- How parallel is the BLR factorization? What about its variants?
Variants of the BLR factorization
Variants of the BLR LU factorization

- FSCU
Variants of the BLR LU factorization

- FSCU (Factor, Solve, Compress, Update)
Variants of the BLR LU factorization

- FSCU (Factor, Solve,
Variants of the BLR LU factorization

- FSCU (Factor, Solve, Compress,
Variants of the BLR LU factorization

- FSCU (Factor, Solve, Compress, Update)
Variants of the BLR LU factorization

- FSCU (Factor, Solve, Compress, Update)
Variants of the BLR LU factorization

- FSCU (Factor, Solve, Compress, Update)
Variants of the BLR LU factorization

- FSCU (Factor, Solve, Compress, Update)
Variants of the BLR LU factorization

- FSCU (Factor, Solve, Compress, Update)
Variants of the BLR LU factorization

- FSCU (Factor, Solve, Compress, Update)
Variants of the BLR LU factorization

- FSCU (Factor, Solve, Compress, Update)
Variants of the BLR LU factorization

- FSCU (Factor, Solve, Compress, Update)
Variants of the BLR LU factorization

- **FSCU** (Factor, Solve, Compress, Update)
Variants of the BLR LU factorization

- FSCU (Factor, Solve, Compress, Update)
- FSCU+LUA
Variants of the BLR LU factorization

- FSCU (Factor, Solve, Compress, Update)
- FSCU+LUA
Variants of the BLR LU factorization

- FSCU (Factor, Solve, Compress, Update)
- FSCU+LUA
Variants of the BLR LU factorization

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

SIAM PP’16, Paris Apr. 12-15
Variants of the BLR LU factorization

- FSCU (Factor, Solve, Compress, Update)
- FSCU+LUA
  - More natural in Left-looking
Variants of the BLR LU factorization

- FSCU (Factor, Solve, Compress, Update)
- FSCU+LUA
  - More natural in Left-looking
Variants of the BLR LU factorization

- FSCU (Factor, Solve, Compress, Update)
- FSCU+LUA
  - More natural in Left-looking
Variants of the BLR LU factorization

- FSCU (Factor, Solve, Compress, Update)
- FSCU+LUA
  - More natural in Left-looking
Variants of the BLR LU factorization

- FSCU (Factor, Solve, Compress, Update)
- FSCU+LUA
  - More natural in Left-looking
Variants of the BLR LU factorization

- FSCU (Factor, Solve, Compress, Update)
- FSCU+LUA
  - More natural in Left-looking
Variants of the BLR LU factorization

<table>
<thead>
<tr>
<th>FSCU (Factor, Solve, Compress, Update)</th>
</tr>
</thead>
<tbody>
<tr>
<td>FSCU+LUA</td>
</tr>
<tr>
<td>- More natural in Left-looking</td>
</tr>
</tbody>
</table>
Variants of the BLR LU factorization

- FSCU (Factor, Solve, Compress, Update)
- FSCU+LUA
  - More natural in Left-looking
Variants of the BLR LU factorization

- **FSCU (Factor, Solve, Compress, Update)**
- **FSCU+LUA**
  - More natural in Left-looking
Variants of the BLR LU factorization

- FSCU (Factor, Solve, Compress, Update)
- FSCU+LUA
  - More natural in Left-looking
Variants of the BLR LU factorization

- FSCU (Factor, Solve, Compress, Update)
- FSCU+LUA
  - More natural in Left-looking
Variants of the BLR LU factorization

- FSCU (Factor, Solve, Compress, Update)
- FSCU+LUA
  - More natural in Left-looking
Variants of the BLR LU factorization

- FSCU (Factor, Solve, Compress, Update)
- FSCU+LUA
  - More natural in Left-looking
  - Better granularity in update operations
Variants of the BLR LU factorization

- FSCU (Factor, Solve, Compress, Update)
- FSCU+LUA
  - More natural in Left-looking
  - Better granularity in update operations
  - Potential recompression
Variants of the BLR LU factorization

- **FSCU** (Factor, Solve, Compress, Update)
- **FSCU+LUA**
  - More natural in Left-looking
  - Better granularity in update operations
  - Potential recompression
Variants of the BLR LU factorization

- FSCU (Factor, Solve, Compress, Update)
- FSCU+LUA
  - More natural in Left-looking
  - Better granularity in update operations
  - Potential recompression
Variants of the BLR LU factorization

- **FSCU** (Factor, Solve, Compress, Update)
- **FSCU+LUA**
  - More natural in Left-looking
  - Better granularity in update operations
  - Potential recompression
Variants of the BLR LU factorization

- FSCU (Factor, Solve, Compress, Update)
- FSCU+LUA
  - More natural in Left-looking
  - Better granularity in update operations
  - Potential recompression
Variants of the BLR LU factorization

- **FSCU (Factor, Solve, Compress, Update)**
- **FSCU+LUA**
  - More natural in Left-looking
  - Better granularity in update operations
  - Potential recompression
- **FCSU(+LUA)**
  - Restricted pivoting, e.g. to diagonal blocks
Variants of the BLR LU factorization

- FSCU (Factor, Solve, Compress, Update)
- FSCU+LUA
  - More natural in Left-looking
  - Better granularity in update operations
  - Potential recompression
- FCSU(+LUA)
  - Restricted pivoting, e.g. to diagonal blocks
Variants of the BLR LU factorization

- FSCU (Factor, Solve, Compress, Update)
- FSCU+LUA
  - More natural in Left-looking
  - Better granularity in update operations
  - Potential recompression
- FCSU(+LUA)
  - Restricted pivoting, e.g. to diagonal blocks
Variants of the BLR LU factorization

- FSCU (Factor, Solve, Compress, Update)
- FSCU+LUA
  - More natural in Left-looking
  - Better granularity in update operations
  - Potential recompression
- FCSU(+LUA)
  - Restricted pivoting, e.g. to diagonal blocks
  - Low-rank Solve
  - Better ratio BLAS-3/BLAS-2 in Solve
Variants of the BLR LU factorization

- FSCU (Factor, Solve, Compress, Update)
- FSCU+LUA
  - More natural in Left-looking
  - Better granularity in update operations
  - Potential recompression
- FCSU(+LUA)
  - Restricted pivoting, e.g. to diagonal blocks
  - Low-rank Solve
  - Better ratio BLAS-3/BLAS-2 in Solve
Variants of the BLR LU factorization

- FSCU (Factor, Solve, Compress, Update)
- FSCU+LUA
  - More natural in Left-looking
  - Better granularity in update operations
  - Potential recompression
- FCSU(+LUA)
  - Restricted pivoting, e.g. to diagonal blocks
  - Low-rank Solve
  - Better ratio BLAS-3/BLAS-2 in Solve
Variants of the BLR LU factorization

- **FSCU (Factor, Solve, Compress, Update)**
- **FSCU+LUA**
  - More natural in Left-looking
  - Better granularity in update operations
  - Potential recompression
- **FCSU(+LUA)**
  - Restricted pivoting, e.g. to diagonal blocks
  - Low-rank Solve
  - Better ratio BLAS-3/BLAS-2 in Solve
Variants of the BLR LU factorization

- **FSCU** (Factor, Solve, Compress, Update)
- **FSCU**+LUA
  - More natural in Left-looking
  - Better granularity in update operations
  - Potential recompression
- **FCSU**(+LUA)
  - Restricted pivoting, e.g. to diagonal blocks
  - Low-rank Solve
  - Better ratio BLAS-3/BLAS-2 in Solve
Variants of the BLR LU factorization

- FSCU (Factor, Solve, Compress, Update)
- FSCU+LUA
  - More natural in Left-looking
  - Better granularity in update operations
  - Potential recompression
- FCSU(+LUA)
  - Restricted pivoting, e.g. to diagonal blocks
  - Low-rank Solve
  - Better ratio BLAS-3/BLAS-2 in Solve
Variants of the BLR LU factorization

- FSCU (Factor, Solve, Compress, Update)
- FSCU+LUA
  - More natural in Left-looking
  - Better granularity in update operations
  - Potential recompression
- FCSU(+LUA)
  - Restricted pivoting, e.g. to diagonal blocks
  - Low-rank Solve
  - Better ratio BLAS-3/BLAS-2 in Solve
Variants of the BLR LU factorization

- **FSCU** (Factor, Solve, Compress, Update)
- **FSCU+LUA**
  - More natural in Left-looking
  - Better granularity in update operations
  - Potential recompression
- **FCSU(+LUA)**
  - Restricted pivoting, e.g. to diagonal blocks
  - Low-rank Solve
  - Better ratio BLAS-3/BLAS-2 in Solve
Variants of the BLR LU factorization

- **FSCU** (Factor, Solve, Compress, Update)
- **FSCU**+LUA
  - More natural in Left-looking
  - Better granularity in update operations
  - Potential recompression
- **FCSU**(+LUA)
  - Restricted pivoting, e.g. to diagonal blocks
  - Low-rank Solve
  - Better ratio BLAS-3/BLAS-2 in Solve
Variants of the BLR LU factorization

- FSCU (Factor, Solve, Compress, Update)
- FSCU+LUA
  - More natural in Left-looking
  - Better granularity in update operations
  - Potential recompression
- FCSU(+LUA)
  - Restricted pivoting, e.g. to diagonal blocks
  - Low-rank Solve
  - Better ratio BLAS-3/BLAS-2 in Solve
### Variants of the BLR LU factorization

<table>
<thead>
<tr>
<th>FSCU (Factor, Solve, Compress, Update)</th>
</tr>
</thead>
<tbody>
<tr>
<td>FSCU+LUA</td>
</tr>
<tr>
<td>- More natural in Left-looking</td>
</tr>
<tr>
<td>- Better granularity in update operations</td>
</tr>
<tr>
<td>- Potential recompression</td>
</tr>
<tr>
<td>FCSU(+LUA)</td>
</tr>
<tr>
<td>- Restricted pivoting, e.g. to diagonal blocks</td>
</tr>
<tr>
<td>- Low-rank Solve</td>
</tr>
<tr>
<td>- Better ratio BLAS-3/BLAS-2 in Solve</td>
</tr>
</tbody>
</table>

**SIAM PP’16, Paris Apr. 12-15**
Variants of the BLR LU factorization

- FSCU (Factor, Solve, Compress, Update)
- FSCU+LUA
  - More natural in Left-looking
  - Better granularity in update operations
  - Potential recompression
- FCSU(+LUA)
  - Restricted pivoting, e.g. to diagonal blocks
  - Low-rank Solve
  - Better ratio BLAS-3/BLAS-2 in Solve
  - With LUA, no need to decompress accumulators
Variants of the BLR LU factorization

- FSCU (Factor, Solve, Compress, Update)
- FSCU+LUA
  - More natural in Left-looking
  - Better granularity in update operations
  - Potential recompression
- FCSU(+LUA)
  - Restricted pivoting, e.g. to diagonal blocks
  - Low-rank Solve
  - Better ratio BLAS-3/BLAS-2 in Solve
  - With LUA, no need to decompress accumulators
Complexity of the BLR factorization

Today, regarding the complexity, we focus on:
- Presenting some important properties of the BLR complexity
- Validating these properties experimentally
### Complexity of multifrontal BLR factorization

<table>
<thead>
<tr>
<th></th>
<th>operations (OPC)</th>
<th>factor size (NNZ)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>$r = O(1)$</td>
<td>$r = O(1)$</td>
</tr>
<tr>
<td>$r = O(n^{\frac{1}{3}})$</td>
<td>$O(n^2)$</td>
<td>$O(n^3)$</td>
</tr>
<tr>
<td>$r = O(n^{\frac{1}{3}})$</td>
<td>$O(n^{\frac{11}{6}})$</td>
<td>$O(n \log n)$</td>
</tr>
<tr>
<td>$r = O(n^{\frac{1}{3}})$</td>
<td>$O(n^{\frac{14}{9}})$</td>
<td>$O(n \log n)$</td>
</tr>
<tr>
<td>$r = O(n^{\frac{1}{3}})$</td>
<td>$O(n^{\frac{4}{3}} \log n)$</td>
<td>$O(n \log n)$</td>
</tr>
<tr>
<td>$r = O(n^{\frac{1}{3}})$</td>
<td>$O(n^{\frac{7}{6}})$</td>
<td>$O(n^{\frac{7}{6}})$</td>
</tr>
</tbody>
</table>

in the 3D case (similar analysis possible for 2D)

**Important properties:**

- The complexity of the standard BLR variant (FSCU) has a lower exponent than the full-rank one.

- Each variant further improves the complexity, with the best one (FCSU+LUA) being not so far from the $\mathcal{H}$-case.

- These properties hold for different rank bound assumptions, e.g. $r = O(1)$ or $r = O(N) = O(n^{\frac{1}{3}})$.
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 4Hz.
Experimental MF complexity: operations

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

<table>
<thead>
<tr>
<th>Mesh size N</th>
<th>Flop count</th>
</tr>
</thead>
<tbody>
<tr>
<td>64</td>
<td>$10^{10}$</td>
</tr>
<tr>
<td>96</td>
<td>$10^{11}$</td>
</tr>
<tr>
<td>128</td>
<td>$10^{12}$</td>
</tr>
<tr>
<td>160</td>
<td>$10^{13}$</td>
</tr>
<tr>
<td>192</td>
<td>$10^{14}$</td>
</tr>
<tr>
<td>224</td>
<td>$10^{15}$</td>
</tr>
<tr>
<td>256</td>
<td>$10^{16}$</td>
</tr>
</tbody>
</table>

- FR fit: $3n^{2.05}$
- FSCU fit: $1231n^{1.49}$
- FSCU+LUA fit: $2779n^{1.41}$
- FCSU+LUA fit: $5674n^{1.33}$

### OPC (Helmholtz, $\varepsilon = 10^{-5}$)

<table>
<thead>
<tr>
<th>Mesh size N</th>
<th>Flop count</th>
</tr>
</thead>
<tbody>
<tr>
<td>64</td>
<td>$10^{10}$</td>
</tr>
<tr>
<td>96</td>
<td>$10^{12}$</td>
</tr>
<tr>
<td>128</td>
<td>$10^{13}$</td>
</tr>
<tr>
<td>160</td>
<td>$10^{14}$</td>
</tr>
<tr>
<td>192</td>
<td>$10^{15}$</td>
</tr>
<tr>
<td>224</td>
<td>$10^{16}$</td>
</tr>
<tr>
<td>256</td>
<td>$10^{17}$</td>
</tr>
</tbody>
</table>

- FR fit: $10n^{2.02}$
- FSCU fit: $35n^{1.85}$
- FSCU+LUA fit: $61n^{1.80}$
- FCSU+LUA fit: $20n^{1.68} \log n$

- good agreement with theoretical complexity
Experimental MF complexity: operations

 OPC (Poisson, \( \varepsilon = 10^{-6} \))

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

- good agreement with theoretical complexity
- \( \varepsilon \) only plays a role in the constant factor
Performance results
1. **Distributed memory** experiments are done on the **eos** supercomputer at the CALMIP center of Toulouse (grant 2014-P0989):
   - 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

2. **Shared memory** experiments are done on **grunch** at the LIP laboratory of Lyon:
   - Two Intel(r) 14-cores Haswell @ 2.3 GHz
   - Peak per core is 36.8 GF/s
   - Total memory is 768 GB
Scalability of the BLR factorization (distributed)

Poisson ($\varepsilon = 10^{-6}, N = 192$)

- MPI+OpenMP parallelism (10 threads/MPI process, 1 MPI/node)

- each time the number of processes doubles, speedup of $\sim 1.6$
- both FR and BLR scale reasonably well
- gain due to BLR remains constant

Helmholtz ($\varepsilon = 10^{-4}, N = 192$)
Gains due to BLR (distributed, MPI+OpenMP)

Poisson ($\varepsilon = 10^{-6}$)

- gains increase with problem size
- gain in flops does not fully translate into gain in time
- multithreaded efficiency lower in LR than in FR
- same remarks apply to Helmoltz, to a lesser extent
Gains due to BLR (distributed, MPI+OpenMP)

- gains increase with problem size
- gain in flops does not fully translate into gain in time
- multithreaded efficiency lower in LR than in FR
- same remarks apply to Helmholtz, to a lesser extent

⇒ improve multithreading with variants
Focus on the Update step (which includes the Decompress)

<table>
<thead>
<tr>
<th></th>
<th>1 thread</th>
<th></th>
<th>28 threads</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>RL</td>
<td>LL</td>
<td>RL</td>
<td>LL</td>
</tr>
<tr>
<td>Poisson (N = 256)</td>
<td>FR</td>
<td>62294s</td>
<td>3772s</td>
<td>65208s</td>
</tr>
<tr>
<td></td>
<td>BLR</td>
<td>2516s</td>
<td>662s</td>
<td>1544s</td>
</tr>
<tr>
<td>Helmholtz (N = 256)</td>
<td>FR</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>BLR</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Right Looking Vs. Left-Looking (shared)

Focus on the Update step (which includes the Decompress)

<table>
<thead>
<tr>
<th></th>
<th>1 thread</th>
<th>28 threads</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>RL</td>
<td>LL</td>
</tr>
<tr>
<td>Poisson (N = 256) FR BLR</td>
<td>62294s</td>
<td>65208s</td>
</tr>
<tr>
<td></td>
<td>2516s</td>
<td>1544s</td>
</tr>
<tr>
<td>Helmholtz (N = 256) FR BLR</td>
<td>9862s</td>
<td>10234s</td>
</tr>
</tbody>
</table>
Focus on the Update step (which includes the Decompress)

<table>
<thead>
<tr>
<th></th>
<th>1 thread</th>
<th></th>
<th>28 threads</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>RL</td>
<td>LL</td>
<td>RL</td>
</tr>
<tr>
<td>Poisson (N = 256)</td>
<td>FR</td>
<td>62294s 65208s</td>
<td>3772s</td>
</tr>
<tr>
<td></td>
<td>BLR</td>
<td>2516s 1544s</td>
<td>662s 183s</td>
</tr>
<tr>
<td>Helmholtz (N = 256)</td>
<td>FR</td>
<td>9862s 10234s</td>
<td></td>
</tr>
<tr>
<td></td>
<td>BLR</td>
<td>1694s 1435s</td>
<td></td>
</tr>
</tbody>
</table>

- in RL: FR (green) block is accessed many times; LR (blue) blocks are accessed once
- in LL: FR (green) block is accessed once; LR (blue) blocks are accessed many times
Focus on the Update step (which includes the Decompress)

<table>
<thead>
<tr>
<th></th>
<th>1 thread</th>
<th></th>
<th>28 threads</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>RL</td>
<td>LL</td>
<td>RL</td>
</tr>
<tr>
<td>Poisson (N = 256)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>FR</td>
<td>62294s</td>
<td>65208s</td>
<td>3772s</td>
</tr>
<tr>
<td>BLR</td>
<td>2516s</td>
<td>1544s</td>
<td>662s</td>
</tr>
<tr>
<td>Helmholtz (N = 256)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>FR</td>
<td>9862s</td>
<td>10234s</td>
<td>1694s</td>
</tr>
<tr>
<td>BLR</td>
<td>1694s</td>
<td>1435s</td>
<td></td>
</tr>
</tbody>
</table>

- in RL: FR (green) block is accessed many times; LR (blue) blocks are accessed once
- in LL: FR (green) block is accessed once; LR (blue) blocks are accessed many times

⇒ lower volume of memory transfers (more critical in multithreaded)
Focus on the Update step (which includes the Decompress)

<table>
<thead>
<tr>
<th></th>
<th>1 thread</th>
<th></th>
<th>28 threads</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>RL</td>
<td>LL</td>
<td>RL</td>
</tr>
<tr>
<td>Poisson (N = 256)</td>
<td>FR 62294s</td>
<td>65208s</td>
<td>3772s</td>
</tr>
<tr>
<td></td>
<td>BLR 2516s</td>
<td>1544s</td>
<td>662s</td>
</tr>
<tr>
<td>Helmholtz (N = 256)</td>
<td>FR 9862s</td>
<td>10234s</td>
<td></td>
</tr>
<tr>
<td></td>
<td>BLR 1694s</td>
<td>1435s</td>
<td></td>
</tr>
</tbody>
</table>

- in RL: FR (green) block is accessed **many times**; LR (blue) blocks are accessed **once**
- in LL: FR (green) block is accessed **once**; LR (blue) blocks are accessed **many times**

⇒ **lower volume of memory transfers** (more critical in multithreaded)

⇒ the **Decompress** part (135s) remains the bottleneck of the Update (183s)
Performance of LUA (shared, 28 threads)

Double precision (d) performance benchmark of Decompress

<table>
<thead>
<tr>
<th></th>
<th>Poisson ($N = 256$)</th>
<th>Helmholtz ($N = 256$)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>LL</td>
<td>LUA</td>
</tr>
<tr>
<td>Flops in Update ($\times 10^{13}$)</td>
<td>1.0</td>
<td>1.0</td>
</tr>
<tr>
<td>Avg. decompress size</td>
<td>3.8</td>
<td>27.1</td>
</tr>
<tr>
<td>Time in Update</td>
<td>183s</td>
<td>87s</td>
</tr>
<tr>
<td>% of peak reached</td>
<td>5%</td>
<td>11%</td>
</tr>
</tbody>
</table>

* All metrics include the Recompression overhead
### Performance of LUA (shared, 28 threads)

#### Double precision (d) performance benchmark of Decompress

<table>
<thead>
<tr>
<th>Decompress Size</th>
<th>Gflops/s</th>
<th>b=256</th>
<th>b=512</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>10</td>
<td>5</td>
<td>5</td>
<td>5</td>
</tr>
<tr>
<td>20</td>
<td>10</td>
<td>10</td>
<td>10</td>
</tr>
<tr>
<td>30</td>
<td>15</td>
<td>15</td>
<td>15</td>
</tr>
<tr>
<td>40</td>
<td>20</td>
<td>20</td>
<td>20</td>
</tr>
<tr>
<td>50</td>
<td>25</td>
<td>25</td>
<td>25</td>
</tr>
</tbody>
</table>

#### Metrics

<table>
<thead>
<tr>
<th></th>
<th>Poisson (N = 256)</th>
<th>Helmholtz (N = 256)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>LL</td>
<td>LUA +Rec.*</td>
</tr>
<tr>
<td>Flops in Update (\times 10^{13})</td>
<td>1.0</td>
<td>1.0</td>
</tr>
<tr>
<td>Avg. decompress size</td>
<td>3.8</td>
<td>27.1</td>
</tr>
<tr>
<td>Time in Update</td>
<td>183s</td>
<td>87s</td>
</tr>
<tr>
<td>% of peak reached</td>
<td>5%</td>
<td>11%</td>
</tr>
</tbody>
</table>

* All metrics include the Recompression overhead

---

20/25

SIAM PP’16, Paris Apr. 12-15
Performance of LUA (shared, 28 threads)

Double precision (d) performance benchmark of Decompress

<table>
<thead>
<tr>
<th>Decompress Size</th>
<th>Gflops/s</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>10</td>
<td>5</td>
</tr>
<tr>
<td>20</td>
<td>10</td>
</tr>
<tr>
<td>30</td>
<td>15</td>
</tr>
<tr>
<td>40</td>
<td>20</td>
</tr>
<tr>
<td>50</td>
<td>25</td>
</tr>
</tbody>
</table>

### Data

- **Poisson** ($N = 256$)
  - LL
  - Flops in Update ($\times 10^{13}$): 1.0
  - Avg. decompress size: 3.8
  - Time in Update: 183s
  - % of peak reached: 5%

- **Helmholtz** ($N = 256$)
  - LL
  - Flops in Update ($\times 10^{13}$): 43
  - Avg. decompress size: 31.3
  - Time in Update: 1435s
  - % of peak reached: 59%

- **LUA**
  - Flops in Update ($\times 10^{13}$): 1.0
  - Avg. decompress size: 27.1
  - Time in Update: 87s
  - % of peak reached: 11%

- **LUA + Rec.***
  - Flops in Update ($\times 10^{13}$): 0.58
  - Avg. decompress size: 12.7
  - Time in Update: 110s
  - % of peak reached: 5%

### Notes

- *All metrics include the Recompression overhead*

![Graph showing performance of Decompress](image-url)
Performance of BLR variants (shared, 28 threads)

Poisson ($\varepsilon = 10^{-6}, N = 256$)

Helmholtz ($\varepsilon = 10^{-3}, N = 256$)

- Non-computational time ($\sim 300s$) is not included $\Rightarrow$ addressed in MPI by tree parallelism and in OpenMP by W. Sid-Lakhdar's PhD thesis work (2014)
Performance of BLR variants (shared, 28 threads)

Poisson ($\varepsilon = 10^{-6}, N = 256$)

Helmholtz ($\varepsilon = 10^{-3}, N = 256$)

- Non-computational time ($\sim 300s$) is not included $\Rightarrow$ addressed in MPI by tree parallelism and in OpenMP by W. Sid-Lakhdar’s PhD thesis work (2014)
- FCSU: Factor+Solve greatly reduced
Performance of BLR variants (shared, 28 threads)

Poisson ($\varepsilon = 10^{-6}, N = 256$)

Time (s)

- FSCU-RL
- FCSU-RL
- FCSU-LL

Helmholtz ($\varepsilon = 10^{-3}, N = 256$)

Time (s)

- FSCU-RL
- FCSU-RL
- FCSU-LL

- Non-computational time ($\sim 300s$) is not included $\Rightarrow$ addressed in MPI by tree parallelism and in OpenMP by W. Sid-Lakhdar’s PhD thesis work (2014)
- FCSU: Factor+Solve greatly reduced
- LL: Update reduced thanks to lower volume of communications
Performance of BLR variants (shared, 28 threads)

Poisson ($\varepsilon = 10^{-6}, N = 256$)

- Non-computational time ($\sim 300s$) is not included ⇒ addressed in MPI by tree parallelism and in OpenMP by W. Sid-Lakhdar's PhD thesis work (2014)
- FCSU: Factor+Solve greatly reduced
- LL: Update reduced thanks to lower volume of communications
- LUA: Update (Decompress) reduced thanks to better granularities

Helmholtz ($\varepsilon = 10^{-3}, N = 256$)
Performance of BLR variants (shared, 28 threads)

Poisson ($\epsilon = 10^{-6}, N = 256$)

Helmholtz ($\epsilon = 10^{-3}, N = 256$)

- Non-computational time ($\sim 300s$) is not included $\Rightarrow$ addressed in MPI by tree parallelism and in OpenMP by W. Sid-Lakhdar’s PhD thesis work (2014)
- FCSU: Factor+Solve greatly reduced
- LL: Update reduced thanks to lower volume of communications
- LUA: Update (Decompress) reduced thanks to better granularities
- Recompression: potential flop reduction not translated into a time gain yet
Conclusion and perspectives
Complexity results

- Theoretical complexity of the BLR (multifrontal) factorization is asymptotically better than FR.
- Studied BLR variants to further reduce complexity by achieving higher compression.
- Numerical experiments show experimental complexity in agreement with theoretical one.

Performance results

- BLR variants possess better properties (efficiency, granularity, volume of communications, number of operations) ⇒ leads to a considerable speedup w.r.t. standard BLR variant...
- ...which itself achieves up to 4.7 (Poisson) and 2.7 (Helmholtz) speedup w.r.t. FR.
Perspectives

- Implementation and performance analysis of the BLR variants in distributed memory (MPI+OpenMP parallelism)
- Efficient strategies to recompress accumulators (cf. J. Anton’s talk)
- Pivoting strategies compatible with the BLR variants
- Influence of the BLR variants on the accuracy of the factorization
Perspectives

- Implementation and performance analysis of the BLR variants in distributed memory (MPI+OpenMP parallelism)
- Efficient strategies to recompress accumulators (cf. J. Anton’s talk)
- Pivoting strategies compatible with the BLR variants
- Influence of the BLR variants on the accuracy of the factorization

Acknowledgements

- CALMIP, BULL and LIP for providing access to the machines
- SEISCOPE for providing the Helmholtz Generator
- LSTC members for scientific discussions
Thanks!
Questions?
Backup Slides
Accumulator recompression

Weight recompression on \( f(C_i) \)

With absolute threshold " \( C_i \) can be compressed separately

Redundancy recompression on \( f(Q_i) \)

Bigger recompression overhead, when is it worth it?
Accumulator recompression

Weight recompression on $f(C_i)$ with absolute threshold 

Redundancy recompression on $f(Q_i)$

Bigger recompression overhead, when is it worth it?
Accumulator recompression

Weight recompression on $f(C_i)$ with absolute threshold "$C_i$" can be compressed separately.

Redundancy recompression on $f(Q_i)$ gives bigger recompression overhead, when is it worth it?
Accumulator recompression

Weight recompression on $Q$ (with absolute threshold $T$), each $C_i$ can be compressed separately.

Redundancy recompression on $Q^T$.

Bigger recompression overhead, when is it worth it?
Accumulator recompression

Weight recompression on $	ext{fig}_i$ with absolute threshold "$	ext{C}_i$ can be compressed separately."

Redundancy recompression on $	ext{fig}_i$. Bigger recompression overhead, when is it worth it?
Accumulator recompression

- Weight recompression on \( \{C_i\}_i \)
  \( \Rightarrow \) With absolute threshold \( \varepsilon \), each \( C_i \) can be compressed separately

- Redundancy recompression on \( \{Q_i\}_i \)
  \( \Rightarrow \) Bigger recompression overhead, when is it worth it?
Experimental MF complexity: entries in factor

- good agreement with theoretical complexity
- $\varepsilon$ only plays a role in the constant factor