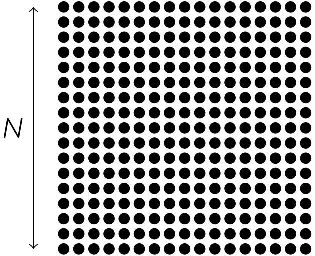
Improving multifrontal solvers by means of Block Low-Rank approximations

The MUMPS team
INP-IRIT, INRIA-LIP, Université de Bordeaux, CNRS-IRIT

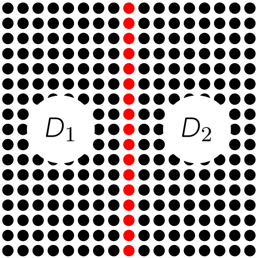
Workshop on fast solvers, Toulouse, June 24-26, 2015

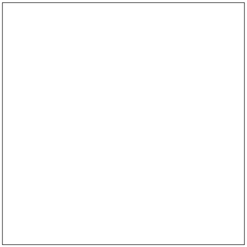
The Multifrontal method



2D problem cost $\propto$

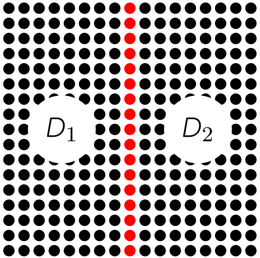
Flops:$\mathcal{O}(N^6)$, mem:$\mathcal{O}(N^4)$

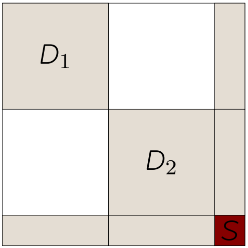




2D problem cost $\propto$
Flops:$\mathcal{O}(N^6)$, mem:$\mathcal{O}(N^4)$

MF (Duff'83) ND (George'73)

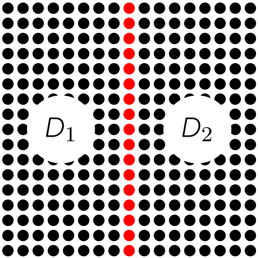


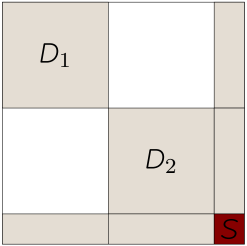


2D problem cost $\propto$

Flops:$\mathcal{O}(N^6)$, mem:$\mathcal{O}(N^4)$

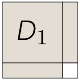
MF (Duff'83) ND (George'73)

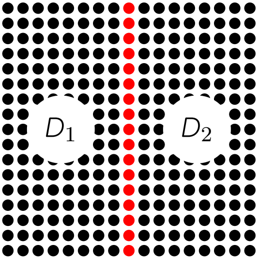


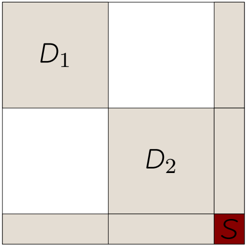


2D problem cost $\propto$

Flops:$\mathcal{O}(N^6)$, mem:$\mathcal{O}(N^4)$

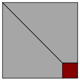


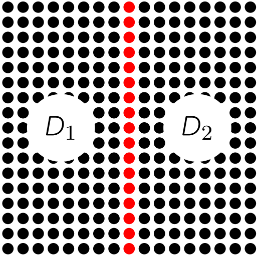


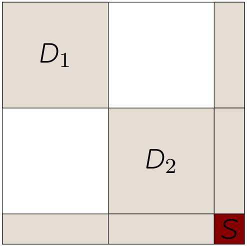


2D problem cost $\propto$

Flops:$\mathcal{O}(N^6)$, mem:$\mathcal{O}(N^4)$

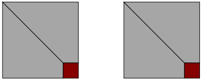


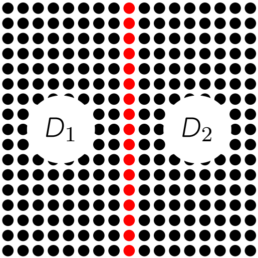


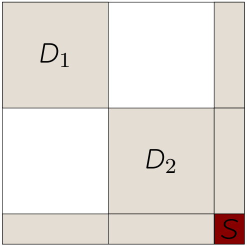


2D problem cost $\propto$

Flops:$\mathcal{O}(N^6)$, mem:$\mathcal{O}(N^4)$

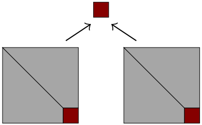




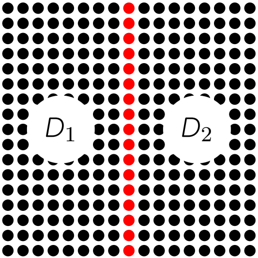


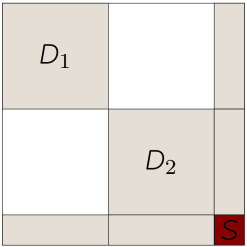
2D problem cost $\propto$

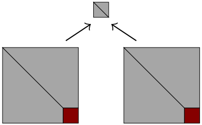
Flops:$\mathcal{O}(N^6)$, mem:$\mathcal{O}(N^4)$



MF (Duff'83) ND (George'73)



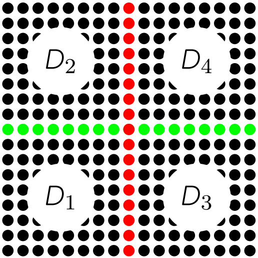


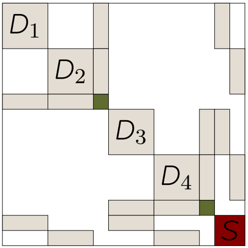


2D problem cost $\propto$

Flops:$\mathcal{O}(N^6)$, mem:$\mathcal{O}(N^4)$

$\rightarrow$ Flops:$\mathcal{O}(N^6/8)$, mem:$\mathcal{O}(N^4/2)$



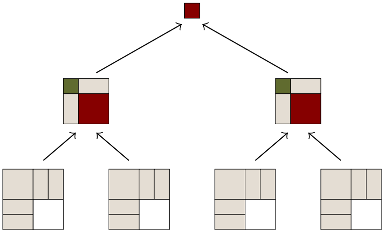


2D problem cost $\propto$

Flops:$\mathcal{O}(N^6)$, mem:$\mathcal{O}(N^4)$

$\rightarrow$ Flops:$\mathcal{O}(N^6/8)$, mem:$\mathcal{O}(N^4/2)$

$\rightarrow$ Flops:$\mathcal{O}(N^3)$, mem:$\mathcal{O}(N^2 log(N))$

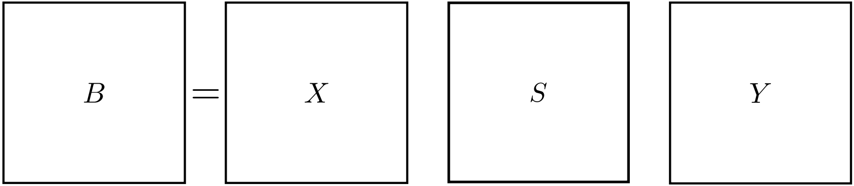


3D problem cost $\propto$

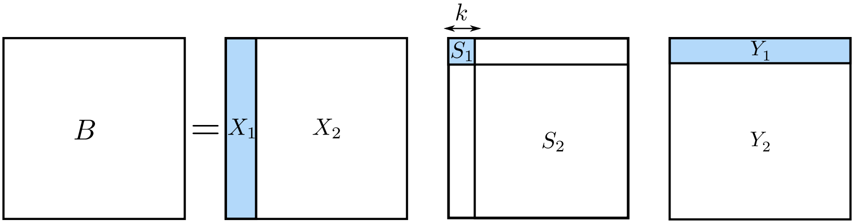
$\rightarrow$ Flops:$\mathcal{O}(N^6)$, mem:$\mathcal{O}(N^4)$

Low-Rank property

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

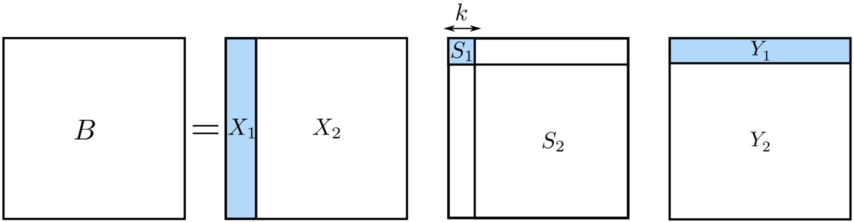


Take a dense matrix B of size $n \times n$ and compute its SVD $B = XSY$:



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

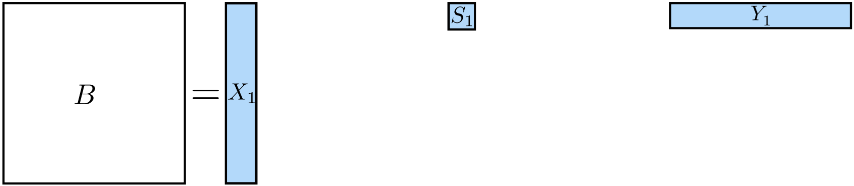
Take a dense matrix B of size $n \times n$ and compute its SVD $B = XSY$:



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

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

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



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

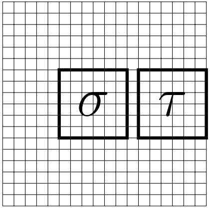
If $\tilde{B} = X_1 S_1 Y_1$ then $\|B - \tilde{B}\|_2 = \|X_2 S_2 Y_2\|_2 = \sigma_{k+1} \le \varepsilon$

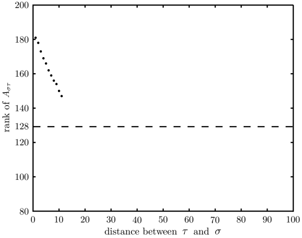
If the singular values of B decay very fast (e.g. exponentially) then $k \ll n$ even for very small ε (e.g. 10^{-14}) $\Rightarrow$ memory and CPU consumption can be reduced considerably with a controlled loss of accuracy ($\le \varepsilon$) if $\tilde{B}$ is used instead of B

Can we exploit low-rankness in frontal matrices?

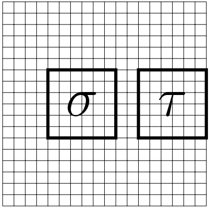
Frontal matrices are usually not low-rank but in many applications they exhibit low-rank blocks.
A block represents the interaction between two subdomains σ and τ. If they have a small diameter and are far away the interaction is weak $\Rightarrow$ rank is low

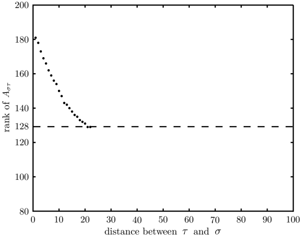
Frontal matrices are usually not low-rank but in many applications they exhibit low-rank blocks.
A block represents the interaction between two subdomains σ and τ. If they have a small diameter and are far away the interaction is weak $\Rightarrow$ rank is low





Frontal matrices are usually not low-rank but in many applications they exhibit low-rank blocks.
A block represents the interaction between two subdomains σ and τ. If they have a small diameter and are far away the interaction is weak $\Rightarrow$ rank is low

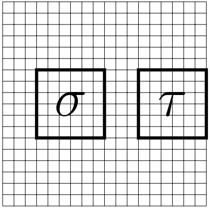


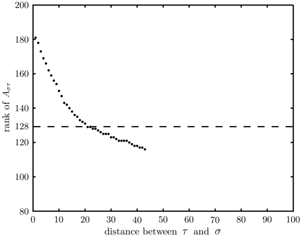


Can we exploit low-rankness in frontal matrices?

Frontal matrices are usually not low-rank but in many applications they exhibit low-rank blocks.

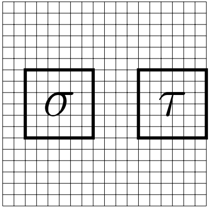
A block represents the interaction between two subdomains σ and τ. If they have a small diameter and are far away the interaction is weak $\Rightarrow$ rank is low

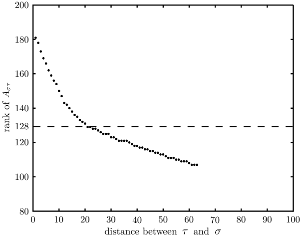




Can we exploit low-rankness in frontal matrices?

Frontal matrices are usually not low-rank but in many applications they exhibit low-rank blocks.
A block represents the interaction between two subdomains σ and τ. If they have a small diameter and are far away the interaction is weak $\Rightarrow$ rank is low

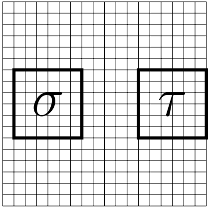


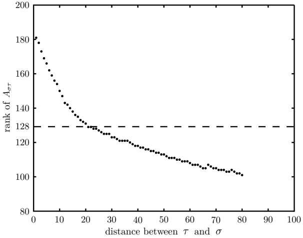


Can we exploit low-rankness in frontal matrices?

Frontal matrices are usually not low-rank but in many applications they exhibit low-rank blocks.

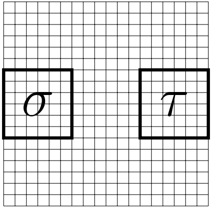
A block represents the interaction between two subdomains σ and τ. If they have a small diameter and are far away the interaction is weak $\Rightarrow$ rank is low

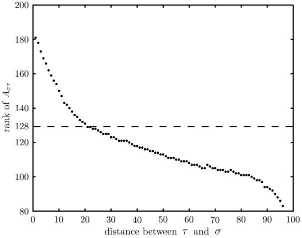




Can we exploit low-rankness in frontal matrices?

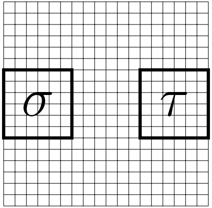
Frontal matrices are usually not low-rank but in many applications they exhibit low-rank blocks.
A block represents the interaction between two subdomains σ and τ. If they have a small diameter and are far away the interaction is weak $\Rightarrow$ rank is low

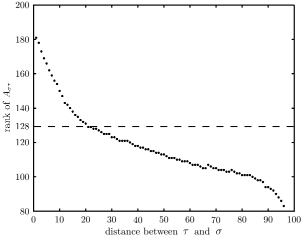




Can we exploit low-rankness in frontal matrices?

Frontal matrices are usually not low-rank but in many applications they exhibit low-rank blocks.
A block represents the interaction between two subdomains σ and τ. If they have a small diameter and are far away the interaction is weak $\Rightarrow$ rank is low



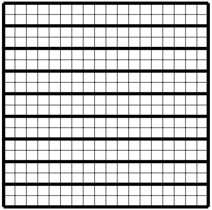


1. compute a clustering of your domain (mesh)
2. permute the matrix accordingly
3. enjoy low-rankness

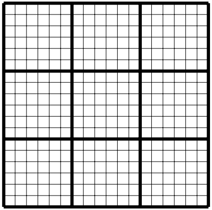
Clustering

Clustering

We aim at a clustering which is such that each frontal matrix has a maximum of low-rank blocks.
If the geometry of the domain, and of the separators is known, the task would be relatively simple



large diameters
small distances



small diameters
large distances

- maximize the relative distance between clusters
- minimize their diameter...
- but not too much to achieve an acceptable BLAS efficiency

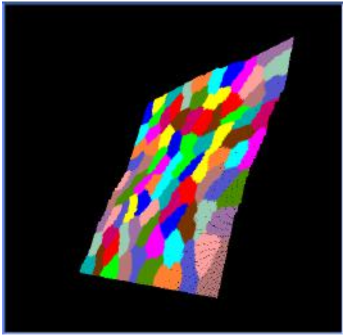
Algebraic clustering/blocking

In a purely algebraic context, we don't have the luxury of knowing the geometry because we only know the matrix
→ use the adjacency graph instead of the domain geometry

For all the separators
- extract the adjacency graph
- extend it with halo
- pass it to a partitioning tool

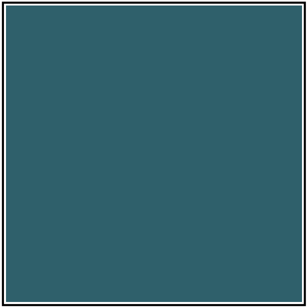
End for

SCOTCH-partitioned SCOTCH separator on a cubic domain of size $N = 128$ →

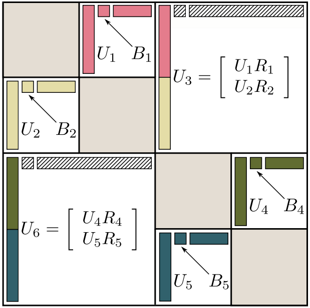


Low-rank formats

Once the blocking is defined, several low-rank formats are possible.

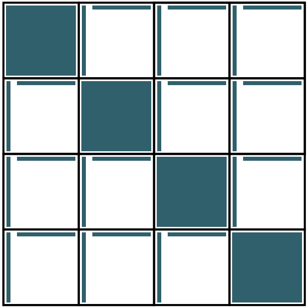


Once the blocking is defined, several low-rank formats are possible.
Some have a hierarchical format ($\mathcal{H}$, $\mathcal{H}^2$, HSS, HODLR, ...)



- Leads to very low complexity (fact. is $\sim O(n)$, with a big constant).
- Complex, hierarchical structure.
- Relatively inefficient and expensive SVD/RRQR...(very T&S blocks), unless randomization or low-rank assembly is used.
- Parallelism is difficult to exploit.

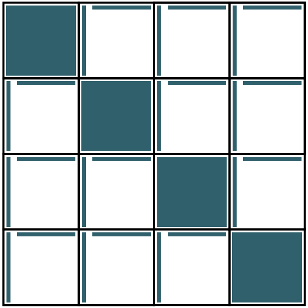
Once the blocking is defined, several low-rank formats are possible.
Another one (ours) is Block Low-Rank



- Very simple structure (very little logic to handle).
- Cheap SVD/RRQR.
- Completely parallel.
- Complexity is a question under investigation.

Low-rank approximations – representations

Once the blocking is defined, several low-rank formats are possible.
Another one (ours) is Block Low-Rank



- Very simple structure (very little logic to handle).
- Cheap SVD/RRQR.
- Completely parallel.
- Complexity is a question under investigation.

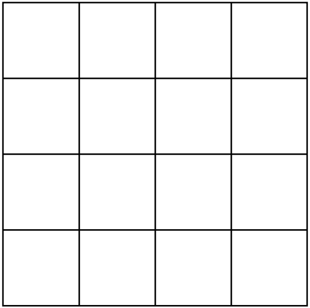
We believe Block Low-Rank (BLR) aims at a good compromise between complexity and performance/usability.

Factorization

BLR LU factorization

task	operation type	full-rank	low-rank
Factor (F)	$B = LU^T$	$(2/3)b^3$	$(2/3)b^3$
Solve (S)	$B = X(YL^{-1})$	b^3	rb^2
Compress (C)	$B = XY$	---	rb^2
Update (U)	$B = B - X_1(Y_1X_2)Y_2$	$2b^3$	rb^2

(b=block size, r=rank)



_GETRF

_TRSM

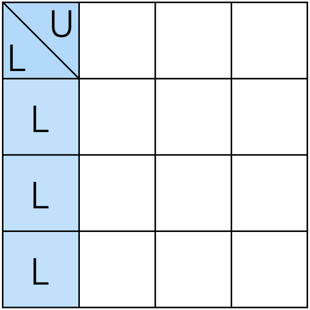
_GEQP3/_GESVD

_GEMM

BLR LU factorization

task	operation type	full-rank	low-rank
Factor (F)	$B = LU^T$	$(2/3)b^3$	$(2/3)b^3$
Solve (S)	$B = X(YL^{-1})$	b^3	rb^2
Compress (C)	$B = XY$	---	rb^2
Update (U)	$B = B - X_1(Y_1X_2)Y_2$	$2b^3$	rb^2

(b=block size, r=rank)

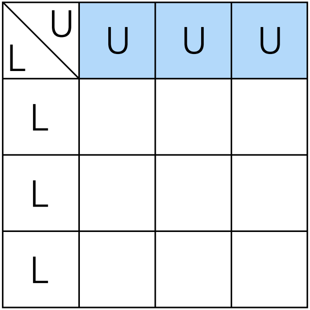


- ▶ _GETRF
- _TRSM
- _GEQP3/_GESVD
- _GEMM

BLR LU factorization

task	operation type	full-rank	low-rank
Factor (F)	$B = LU^T$	$(2/3)b^3$	$(2/3)b^3$
Solve (S)	$B = X(YL^{-1})$	b^3	rb^2
Compress (C)	$B = XY$	---	rb^2
Update (U)	$B = B - X_1(Y_1X_2)Y_2$	$2b^3$	rb^2

(b=block size, r=rank)



_GETRF

▶ _TRSM

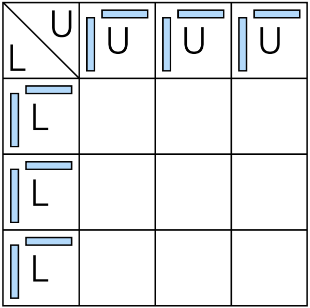
_GEQP3/_GESVD

_GEMM

BLR LU factorization

task	operation type	full-rank	low-rank
Factor (F)	$B = LU^T$	$(2/3)b^3$	$(2/3)b^3$
Solve (S)	$B = X(YL^{-1})$	b^3	rb^2
Compress (C)	$B = XY$	---	rb^2
Update (U)	$B = B - X_1(Y_1X_2)Y_2$	$2b^3$	rb^2

(b=block size, r=rank)



_GETRF

_TRSM

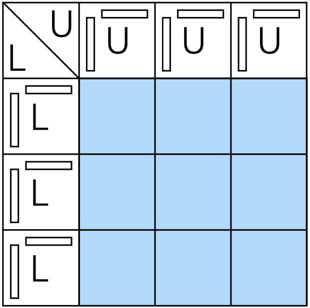
▶ _GEQP3/_GESVD

_GEMM

BLR LU factorization

task	operation type	full-rank	low-rank
Factor (F)	$B = LU^T$	$(2/3)b^3$	$(2/3)b^3$
Solve (S)	$B = X(YL^{-1})$	b^3	rb^2
Compress (C)	$B = XY$	---	rb^2
Update (U)	$B = B - X_1(Y_1X_2)Y_2$	$2b^3$	rb^2

(b=block size, r=rank)



_GETRF

_TRSM

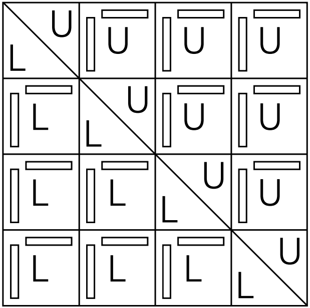
_GEQP3/_GESVD

▶ _GEMM

BLR LU factorization

task	operation type	full-rank	low-rank
Factor (F)	$B = LU^T$	$(2/3)b^3$	$(2/3)b^3$
Solve (S)	$B = X(YL^{-1})$	b^3	rb^2
Compress (C)	$B = XY$	---	rb^2
Update (U)	$B = B - X_1(Y_1X_2)Y_2$	$2b^3$	rb^2

(b=block size, r=rank)



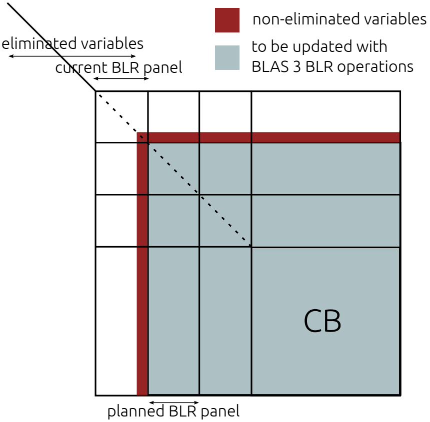
_GETRF

_TRSM

_GEQP3/_GESVD

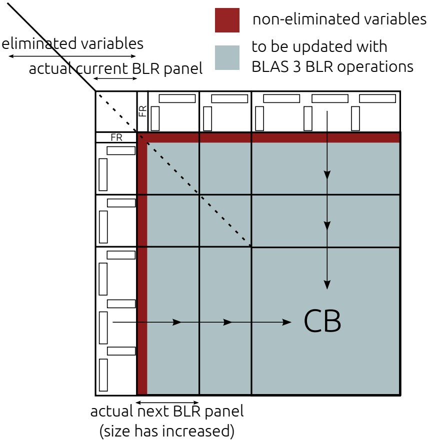
_GEMM

Threshold partial pivoting with BLR



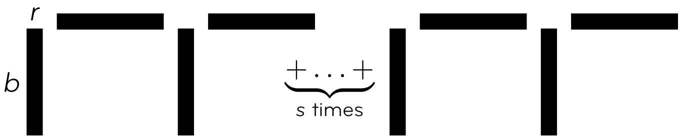
Pivots are delayed panelwise and eventually to the parent node

Threshold partial pivoting with BLR



Pivots are delayed panelwise and eventually to the parent node

Complexity of the BLR factorization

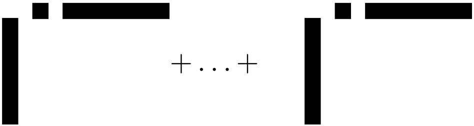


update without LUA

```
for k = 1, s do
    compute_middle_block
    multiply_middle_block
    decompress
    FR sum
end for
```

update with LUA

```
for k = 1, s do
    compute_middle_block
    multiply_middle_block
    LR sum
end for
decompress
```



update without LUA

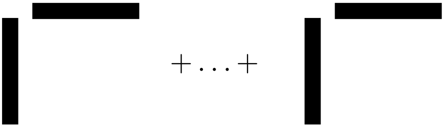
```
for k = 1, s do
    compute_middle_block
    multiply_middle_block
    decompress
    FR sum
end for
```

$O(sbr^2)$

update with LUA

```
for k = 1, s do
    compute_middle_block
    multiply_middle_block
    LR sum
end for
decompress
```

Low-rank Updates Accumulation (LUA)



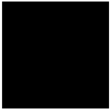
update without LUA

for $k = 1, s$ **do**
    `compute_middle_block`
    `multiply_middle_block`
    `decompress`
    `FR sum`
end for

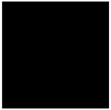
$O(sbr^2)$

update with LUA

for $k = 1, s$ **do**
    `compute_middle_block`
    `multiply_middle_block`
    `LR sum`
end for
`decompress`



$+ \ldots +$



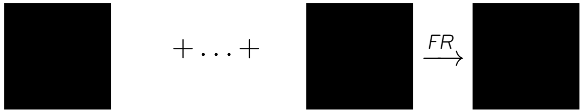
update without LUA

```
for k = 1, s do
    compute_middle_block
    multiply_middle_block
    decompress
    FR sum
end for
```

$O(sbr^2) + O(sb^2r)$

update with LUA

```
for k = 1, s do
    compute_middle_block
    multiply_middle_block
    LR sum
end for
decompress
```



update without LUA

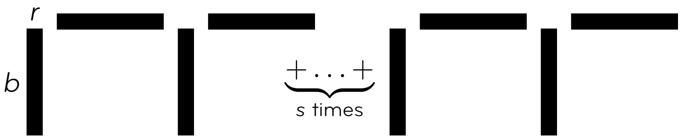
```
for k = 1, s do
    compute_middle_block
    multiply_middle_block
    decompress
    FR sum
end for
```

$$O(sbr^2) + O(sb^2r)$$

update with LUA

```
for k = 1, s do
    compute_middle_block
    multiply_middle_block
    LR sum
end for
decompress
```

Low-rank Updates Accumulation (LUA)



update without LUA

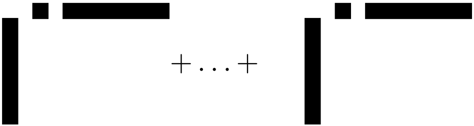
```
for k = 1, s do
    compute_middle_block
    multiply_middle_block
    decompress
    FR sum
end for
```

$O(sbr^2) + O(sb^2r)$

update with LUA

```
for k = 1, s do
    compute_middle_block
    multiply_middle_block
    LR sum
end for
decompress
```

Low-rank Updates Accumulation (LUA)



update without LUA

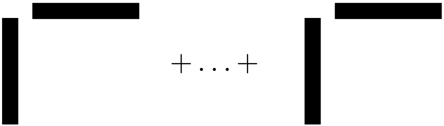
```
for k = 1, s do
    compute_middle_block
    multiply_middle_block
    decompress
    FR sum
end for
```

$$O(sbr^2) + O(sb^2r)$$

update with LUA

```
for k = 1, s do
    compute_middle_block
    multiply_middle_block
    LR sum
end for
decompress
```

$$O(sbr^2)$$



update without LUA

```
for k = 1, s do
    compute_middle_block
    multiply_middle_block
    decompress
    FR sum
end for
```

$O(sbr^2) + O(sb^2r)$

update with LUA

```
for k = 1, s do
    compute_middle_block
    multiply_middle_block
    LR sum
end for
decompress
```

$O(sbr^2)$

update without LUA

```
for k = 1, s do
    compute_middle_block
    multiply_middle_block
    decompress
    FR sum
end for
```

$$O(sbr^2) + O(sb^2r)$$

update with LUA

```
for k = 1, s do
    compute_middle_block
    multiply_middle_block
    LR sum
end for
decompress
```

$$O(sbr^2)$$

Low-rank Updates Accumulation (LUA)

update without LUA

for $k = 1, s$ **do**
```
    compute_middle_block
    multiply_middle_block
    decompress
    FR sum
```
end for

$$O(sbr^2) + O(sb^2r)$$

update with LUA

for $k = 1, s$ **do**
```
    compute_middle_block
    multiply_middle_block
    LR sum
```
end for
```
decompress
```

$$O(sbr^2) + O(b^2r)$$

update without LUA

```
for k = 1, s do
    compute_middle_block
    multiply_middle_block
    decompress
    FR sum
end for
```

$O(sbr^2) + O(sb^2r)$

update with LUA

```
for k = 1, s do
    compute_middle_block
    multiply_middle_block
    LR sum
end for
decompress
```

$O(sbr^2) + O(b^2r)$

Complexity of BLR LU factorization

Depending on when and how the compression is done, different variants are possible with different theoretical complexity:

	operations		memory	
	$r = O(1)$	$r = O(N)$	$r = O(1)$	$r = O(N)$
FR	$O(n^2)$	$O(n^2)$	$O(n^{\frac{4}{3}})$	$O(n^{\frac{4}{3}})$
BLR FSCU	$O(n^{\frac{5}{3}})$	$O(n^{\frac{11}{6}})$	$O(n \log n)$	$O(n^{\frac{4}{3}})$
BLR FCSU	$O(n^{\frac{14}{9}})$	$O(n^{\frac{16}{9}})$	$O(n \log n)$	$O(n^{\frac{4}{3}})$
BLR FSCU+LUA	$O(n^{\frac{14}{9}})$	$O(n^{\frac{16}{9}})$	$O(n \log n)$	$O(n^{\frac{4}{3}})$
BLR FCSU+LUA	$O(n^{\frac{4}{3}})$	$O(n^{\frac{5}{3}} \log n)$	$O(n \log n)$	$O(n^{\frac{4}{3}})$
$\mathcal{H}$	$O(n^{\frac{4}{3}})$	$O(n^{\frac{5}{3}})$	$O(n)$	$O(n^{\frac{7}{6}})$

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

If updates are accumulated and applied at once (LUA), a further reduction can be achieved which leads to the same theoretical complexity as HSS.

Experimental results

Experimental MF complexity

Setting:

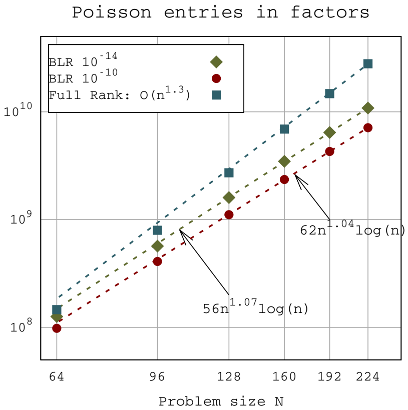
1. **`Poisson`**: N^3 grid with a 7-point stencil with $u = 1$ on the boundary $\partial\Omega$

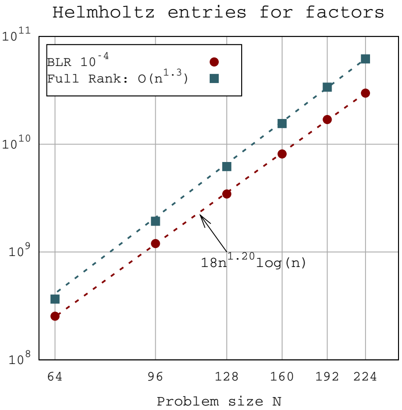
$$\Delta u = f$$

2. **`Helmholtz`**: N^3 grid with a 27-point stencil, ω 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 $s(x, \omega)$.

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

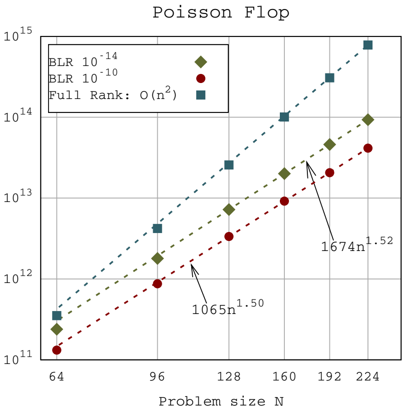
Experimental MF complexity: entries in factor

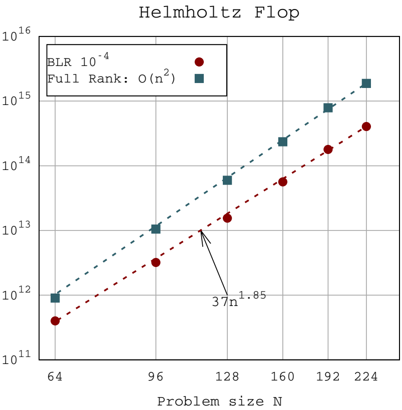




- ε only plays a role in the constant factor
- good agreement with theory
- for Poisson a factor ~ 3 gain with almost no loss of accuracy

Experimental MF complexity: operations





- ε only plays a role in the constant factor
- good agreement with theory
- for Poisson a factor ~ 9 gain with almost no loss of accuracy

Application to frequency-domain seismic modeling

- Credits: SEISCOPE project
- 3D VTI visco-acoustic Valhall model
- VTI visco-acoustic Helmholtz equation

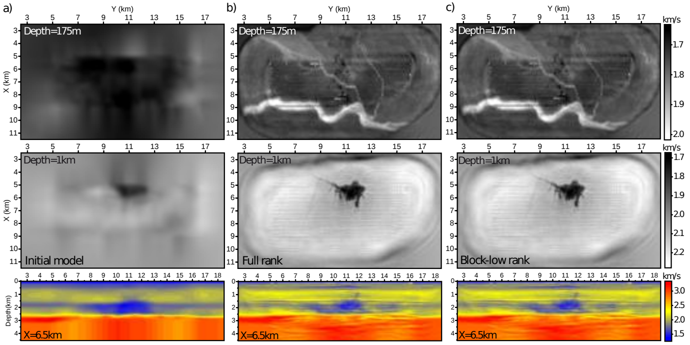
Freq.	n	nnz	factors	flops	time	cores
5Hz	3M	70M	2.5GB	6.5E+13	80s	240
7Hz	7M	177M	6.4GB	4.1E+14	323s	320
10Hz	17M	446M	10.5GB	2.6E+15	1117s	680

Full-rank statistics

Experiments are done on the LICALLO supercomputer at the OCA mesocenter:

- Two Intel(r) 10-cores Ivy Bridge 2,5 GHz and 64 GB memory
- Peak per core is 20.0 GF/s
- Infiniband FDR interconnect

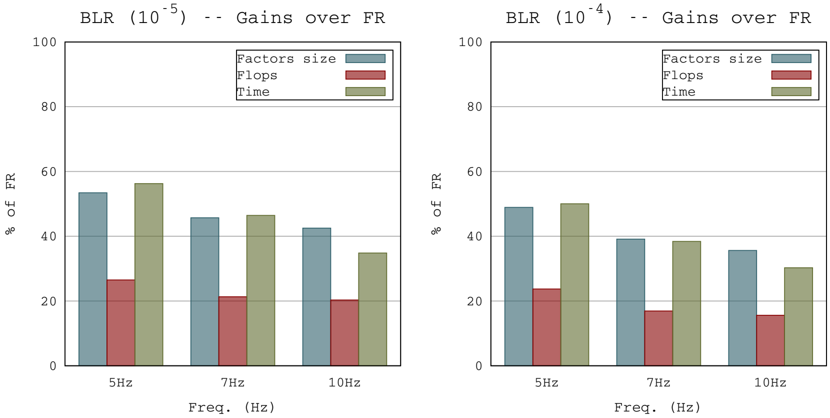
Application to frequency-domain seismic modeling



$7Hz$ problem with single-precision on 320 cores:

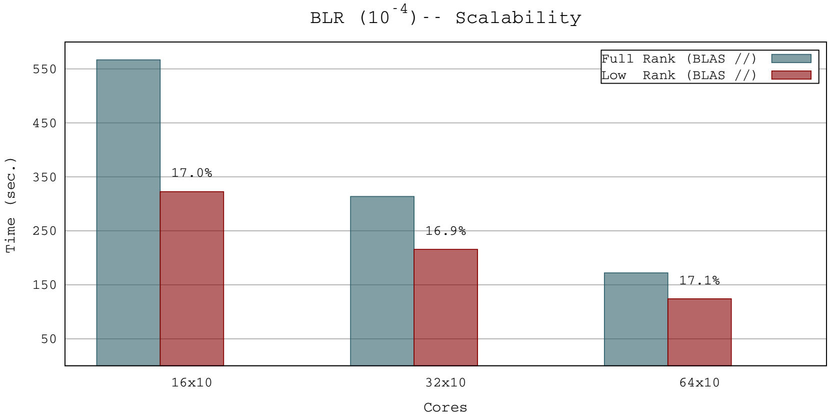
- each row is a different section of the domain
- first column: initial model obtained with traveltime tomography
- second column: FWI solution computed with FR-MUMPS
- third column: FWI solution computed with BLR-MUMPS ($\varepsilon = 10^{-5}$)

Application to frequency-domain seismic modeling



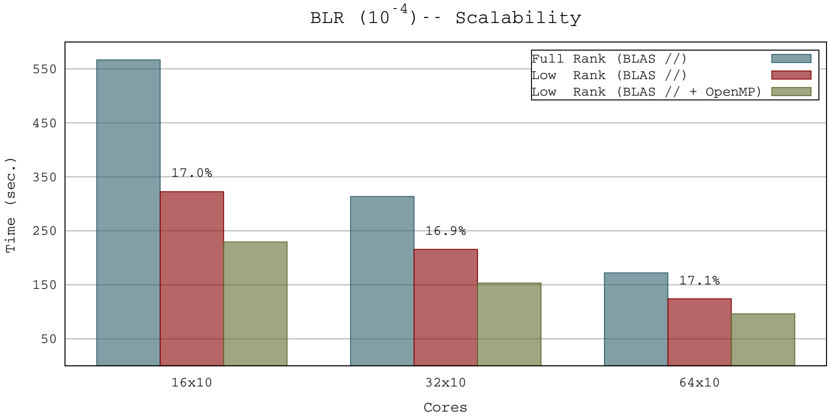
Gains in execution time do not match those in Flops because of the weaker efficiency of BLAS kernels due to the small granularity.

Application to frequency-domain seismic modeling



Due to the small size of blocks, multithreaded BLAS is inefficient.

Application to frequency-domain seismic modeling



Due to the small size of blocks, multithreaded BLAS is inefficient. We have added OpenMP directives to exploit multicores on BLR computations

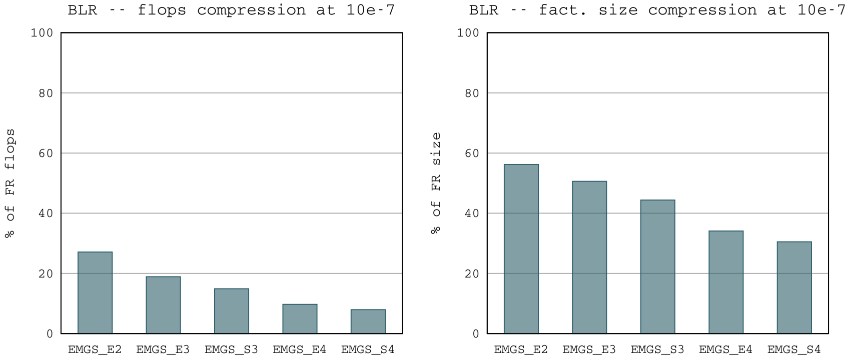
Matrices from EMGS (Norway). All matrices are complex and solved in double-precision

Mat.	n	nnz	factors	flops
EMGS_E2	0.9 M	12M	16GB	6.1e+12
EMGS_E3	2.9 M	37M	76GB	5.6e+13
EMGS_S3	3.3 M	43M	92GB	7.5e+13
EMGS_E4	17.4 M	226M	897GB	2.1e+15
EMGS_S4	20.6 M	266M	1122GB	3.0e+15

Experiments are done on the EOS supercomputer at the CALMIP center of Toulouse (grant 2014-P0989):

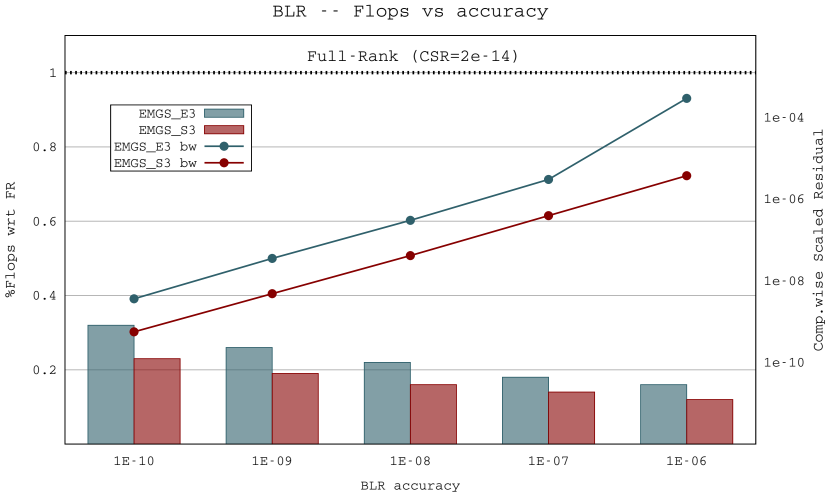
- Two Intel(r) 10-cores Ivy Bridge 2,8 GHz and 64 GB memory
- Peak per core is 22.4 GF/s
- Infiniband FDR interconnect

Application to Electromagnetism



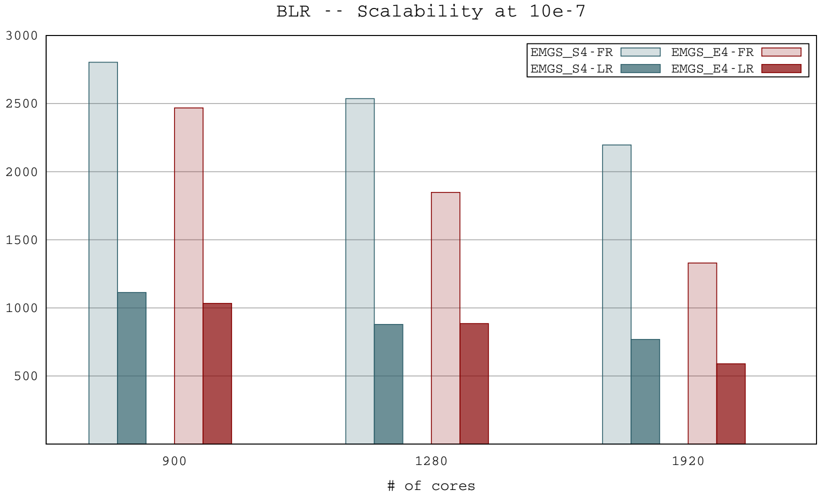
- Gains increase with the size of the problem
- Global memory is reduced more than just factors
- Compression overhead is included

Application to Electromagnetism



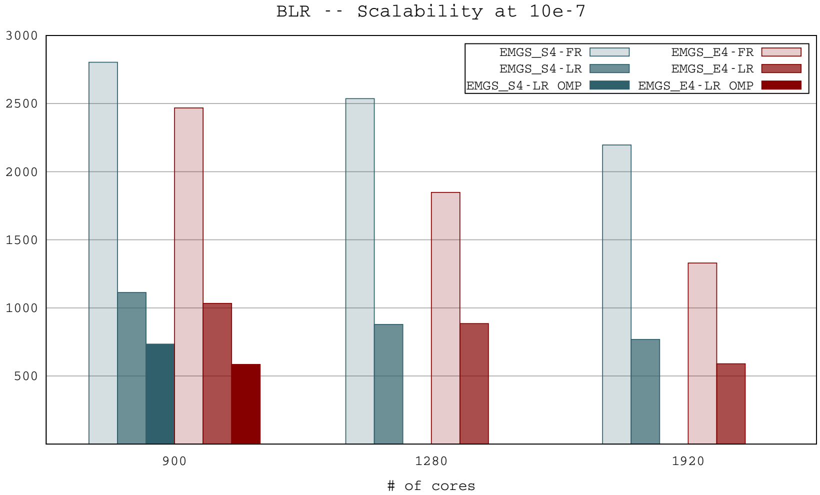
- compression improves, accuracy deteriorates as ε increases
- good agreement between ε and solution accuracy

Application to Electromagnetism



- smaller BLAS granularity (lower seq. and m.threaded speed)
- a factor ~ 2.5 out of ~ 10

Application to Electromagnetism



- smaller BLAS granularity (lower seq. and m.threaded speed)
- a factor ~ 4.2 out of ~ 10 thanks to OpenMP

Performance analysis

FR and BLR ($\varepsilon = 10^{-6}$) factorization of Geo_1438 matrix

Matrix	n	nnz	factors	flops
Geo_1438	1.4 M	60M	41GB	3.8e+13

% of FR ops is 6.8%
Peak per core is 22.4 GF/s

	1 thread		10 threads	
FR Facto	1859.0s	(19.2 GF/s)	207.6s	(16.1 GF/s)
LR Facto	301.7s	(6.8 GF/s)	60.3s	(4.3 GF/s)
Assembly+Stack	105.9s		28.1s	

- Weight of assembly and memory copies becomes considerable in LR and in multithreaded context
- FR vs. BLR: speedup of 6.2 out of 14.7 in sequential
- 1 $\rightarrow$ 10 threads: speedup of 5 out of 10 (to compare to 9 in FR)

FR and BLR ($\varepsilon = 10^{-6}$) factorization of Geo_1438 matrix

	Operation	1 thread time	1 thread time%	1 thread GF/s	10 threads time	10 threads time%	10 threads GF/s
FR	Panel+TRSM	92.2s	5.0%	13.5	17.7s	8.5%	7.0
	Compress	0.0s	0.0%	-	0.0s	0.0%	-
	Update	1748.0s	94.0%	20.7	184.2s	88.8%	19.7
	Bottom Fronts	18.5s	1.0%	16.1	5.6s	2.7%	5.3
	FR Total	1859.0s	100.0%	19.2	207.6s	100.0%	16.1
LR	Panel+TRSM	92.2s	30.6%	13.5	17.7s	29.3%	7.0
	Compress	67.2s	22.3%	3.2	12.0s	20.0%	1.8
	Update	123.6s	41.0%	6.5	24.6s	40.9%	3.3
	Bottom Fronts	18.5s	6.1%	16.1	5.6s	9.3%	5.4
	LR Total	301.7s	100.0%	6.8	60.3s	100.0%	4.3

FR and BLR ($\varepsilon = 10^{-6}$) factorization of Geo_1438 matrix

	Operation	1 thread time	1 thread time%	1 thread GF/s	10 threads time	10 threads time%	10 threads GF/s
FR	Panel+TRSM	92.2s	5.0%	13.5	17.7s	8.5%	7.0
	Compress	0.0s	0.0%	-	0.0s	0.0%	-
	Update	1748.0s	94.0%	20.7	184.2s	88.8%	19.7
	Bottom Fronts	18.5s	1.0%	16.1	5.6s	2.7%	5.3
	FR Total	1859.0s	100.0%	19.2	207.6s	100.0%	16.1
LR	Panel+TRSM	92.2s	30.6%	13.5	17.7s	29.3%	7.0
	Compress	67.2s	22.3%	3.2	12.0s	20.0%	1.8
	Update	123.6s	41.0%	6.5	24.6s	40.9%	3.3
	Bottom Fronts	18.5s	6.1%	16.1	5.6s	9.3%	5.4
	LR Total	301.7s	100.0%	6.8	60.3s	100.0%	4.3

FR and BLR ($\varepsilon = 10^{-6}$) factorization of Geo_1438 matrix

	Operation	1 thread time	1 thread time%	1 thread GF/s	10 threads time	10 threads time%	10 threads GF/s
FR	Panel+TRSM	92.2s	5.0%	13.5	17.7s	8.5%	7.0
	Compress	0.0s	0.0%	-	0.0s	0.0%	-
	Update	1748.0s	94.0%	20.7	184.2s	88.8%	19.7
	Bottom Fronts	18.5s	1.0%	16.1	5.6s	2.7%	5.3
	FR Total	1859.0s	100.0%	19.2	207.6s	100.0%	16.1
LR	Panel+TRSM	92.2s	30.6%	13.5	17.7s	29.3%	7.0
	Compress	67.2s	22.3%	3.2	12.0s	20.0%	1.8
	Update	123.6s	41.0%	6.5	24.6s	40.9%	3.3
	Bottom Fronts	18.5s	6.1%	16.1	5.6s	9.3%	5.4
	LR Total	301.7s	100.0%	6.8	60.3s	100.0%	4.3

FR and BLR ($\varepsilon = 10^{-6}$) factorization of Geo_1438 matrix

	Operation	1 thread time	1 thread time%	1 thread GF/s	10 threads time	10 threads time%	10 threads GF/s
FR	Panel+TRSM	92.2s	5.0%	13.5	17.7s	8.5%	7.0
	Compress	0.0s	0.0%	-	0.0s	0.0%	-
	Update	1748.0s	94.0%	20.7	184.2s	88.8%	19.7
	Bottom Fronts	18.5s	1.0%	16.1	5.6s	2.7%	5.3
	FR Total	1859.0s	100.0%	19.2	207.6s	100.0%	16.1
LR	Panel+TRSM	92.2s	30.6%	13.5	17.7s	29.3%	7.0
	Compress	67.2s	22.3%	3.2	12.0s	20.0%	1.8
	Update	123.6s	41.0%	6.5	24.6s	40.9%	3.3
	Bottom Fronts	18.5s	6.1%	16.1	5.6s	9.3%	5.4
	LR Total	301.7s	100.0%	6.8	60.3s	100.0%	4.3

FR and BLR ($\varepsilon = 10^{-6}$) factorization of Geo_1438 matrix

	Operation	1 thread time	1 thread time%	1 thread GF/s	10 threads time	10 threads time%	10 threads GF/s
FR	Panel+TRSM	92.2s	5.0%	13.5	17.7s	8.5%	7.0
	Compress	0.0s	0.0%	-	0.0s	0.0%	-
	Update	1748.0s	94.0%	20.7	184.2s	88.8%	19.7
	Bottom Fronts	18.5s	1.0%	16.1	5.6s	2.7%	5.3
	FR Total	1859.0s	100.0%	19.2	207.6s	100.0%	16.1
LR	Panel+TRSM	92.2s	30.6%	13.5	17.7s	29.3%	7.0
	Compress	67.2s	22.3%	3.2	12.0s	20.0%	1.8
	Update	123.6s	41.0%	6.5	24.6s	40.9%	3.3
	Bottom Fronts	18.5s	6.1%	16.1	5.6s	9.3%	5.4
	LR Total	301.7s	100.0%	6.8	60.3s	100.0%	4.3

- Compress has poor efficiency

FR and BLR ($\varepsilon = 10^{-6}$) factorization of Geo_1438 matrix

	Operation	1 thread time	1 thread time%	1 thread GF/s	10 threads time	10 threads time%	10 threads GF/s
	Panel+TRSM	92.2s	5.0%	13.5	17.7s	8.5%	7.0
	Compress	0.0s	0.0%	-	0.0s	0.0%	-
FR	Update	1748.0s	94.0%	20.7	184.2s	88.8%	19.7
	Bottom Fronts	18.5s	1.0%	16.1	5.6s	2.7%	5.3
	FR Total	1859.0s	100.0%	19.2	207.6s	100.0%	16.1
	Panel+TRSM	92.2s	30.6%	13.5	17.7s	29.3%	7.0
	Compress	67.2s	22.3%	3.2	12.0s	20.0%	1.8
LR	Update	123.6s	41.0%	6.5	24.6s	40.9%	3.3
	Bottom Fronts	18.5s	6.1%	16.1	5.6s	9.3%	5.4
	LR Total	301.7s	100.0%	6.8	60.3s	100.0%	4.3

- Compress has poor efficiency
- Updates are much slower in LR because of smaller granularity

FR and BLR ($\varepsilon = 10^{-6}$) factorization of Geo_1438 matrix

	Operation	1 thread time	1 thread time%	1 thread GF/s	10 threads time	10 threads time%	10 threads GF/s
FR	Panel+TRSM	92.2s	5.0%	13.5	17.7s	8.5%	7.0
	Compress	0.0s	0.0%	-	0.0s	0.0%	-
	Update	1748.0s	94.0%	20.7	184.2s	88.8%	19.7
	Bottom Fronts	18.5s	1.0%	16.1	5.6s	2.7%	5.3
	FR Total	1859.0s	100.0%	19.2	207.6s	100.0%	16.1
LR	Panel+TRSM	92.2s	30.6%	13.5	17.7s	29.3%	7.0
	Compress	67.2s	22.3%	3.2	12.0s	20.0%	1.8
	Update	123.6s	41.0%	6.5	24.6s	40.9%	3.3
	Bottom Fronts	18.5s	6.1%	16.1	5.6s	9.3%	5.4
	LR Total	301.7s	100.0%	6.8	60.3s	100.0%	4.3

- Compress has poor efficiency
- Updates are much slower in LR because of smaller granularity

FR and BLR ($\varepsilon = 10^{-6}$) factorization of Geo_1438 matrix

	Operation	1 thread time	1 thread time%	1 thread GF/s	10 threads time	10 threads time%	10 threads GF/s
FR	Panel+TRSM	92.2s	5.0%	13.5	17.7s	8.5%	7.0
	Compress	0.0s	0.0%	-	0.0s	0.0%	-
	Update	1748.0s	94.0%	20.7	184.2s	88.8%	19.7
	Bottom Fronts	18.5s	1.0%	16.1	5.6s	2.7%	5.3
	FR Total	1859.0s	100.0%	19.2	207.6s	100.0%	16.1
LR	Panel+TRSM	92.2s	30.6%	13.5	17.7s	29.3%	7.0
	Compress	67.2s	22.3%	3.2	12.0s	20.0%	1.8
	Update	123.6s	41.0%	6.5	24.6s	40.9%	3.3
	Bottom Fronts	18.5s	6.1%	16.1	5.6s	9.3%	5.4
	LR Total	301.7s	100.0%	6.8	60.3s	100.0%	4.3

- Compress has poor efficiency
- Updates are much slower in LR because of smaller granularity
- Multithreading bottom fronts: factor 3.3 out of 10 $\rightarrow$ W. Sid-Lakhdar's thesis

FR and BLR ($\varepsilon = 10^{-6}$) factorization of Geo_1438 matrix

	Operation	1 thread time	1 thread time%	1 thread GF/s	10 threads time	10 threads time%	10 threads GF/s
FR	Panel+TRSM	92.2s	5.0%	13.5	17.7s	8.5%	7.0
	Compress	0.0s	0.0%	-	0.0s	0.0%	-
	Update	1748.0s	94.0%	20.7	184.2s	88.8%	19.7
	Bottom Fronts	18.5s	1.0%	16.1	5.6s	2.7%	5.3
	FR Total	1859.0s	100.0%	19.2	207.6s	100.0%	16.1
LR	Panel+TRSM	92.2s	30.6%	13.5	17.7s	29.3%	7.0
	Compress	67.2s	22.3%	3.2	12.0s	20.0%	1.8
	Update	123.6s	41.0%	6.5	24.6s	40.9%	3.3
	Bottom Fronts	18.5s	6.1%	16.1	5.6s	9.3%	5.4
	LR Total	301.7s	100.0%	6.8	60.3s	100.0%	4.3

- Compress has poor efficiency
- Updates are much slower in LR because of smaller granularity
- Multithreading bottom fronts: factor 3.3 out of 10 → W. Sid-Lakhdar's thesis
- Multithreading compress: factor 5.6 out of 10

FR and BLR ($\varepsilon = 10^{-6}$) factorization of Geo_1438 matrix

	Operation	1 thread time	1 thread time%	1 thread GF/s	10 threads time	10 threads time%	10 threads GF/s
FR	Panel+TRSM	92.2s	5.0%	13.5	17.7s	8.5%	7.0
	Compress	0.0s	0.0%	-	0.0s	0.0%	-
	Update	1748.0s	94.0%	20.7	184.2s	88.8%	19.7
	Bottom Fronts	18.5s	1.0%	16.1	5.6s	2.7%	5.3
	FR Total	1859.0s	100.0%	19.2	207.6s	100.0%	16.1
LR	Panel+TRSM	92.2s	30.6%	13.5	17.7s	29.3%	7.0
	Compress	67.2s	22.3%	3.2	12.0s	20.0%	1.8
	Update	123.6s	41.0%	6.5	24.6s	40.9%	3.3
	Bottom Fronts	18.5s	6.1%	16.1	5.6s	9.3%	5.4
	LR Total	301.7s	100.0%	6.8	60.3s	100.0%	4.3

- Compress has poor efficiency
- Updates are much slower in LR because of smaller granularity
- Multithreading bottom fronts: factor 3.3 out of 10 → W. Sid-Lakhdar's thesis
- Multithreading compress: factor 5.6 out of 10
- Multithreading LR update: factor 5 out of 10 (compare to 9.5 in FR)

FR and BLR ($\varepsilon = 10^{-6}$) factorization of Geo_1438 matrix

	Operation	1 thread time	1 thread time%	1 thread GF/s	10 threads time	10 threads time%	10 threads GF/s
FR	Panel+TRSM	92.2s	5.0%	13.5	17.7s	8.5%	7.0
	Compress	0.0s	0.0%	-	0.0s	0.0%	-
	Update	1748.0s	94.0%	20.7	184.2s	88.8%	19.7
	Bottom Fronts	18.5s	1.0%	16.1	5.6s	2.7%	5.3
	FR Total	1859.0s	100.0%	19.2	207.6s	100.0%	16.1
LR	Panel+TRSM	92.2s	30.6%	13.5	17.7s	29.3%	7.0
	Compress	67.2s	22.3%	3.2	12.0s	20.0%	1.8
	Update	123.6s	41.0%	6.5	24.6s	40.9%	3.3
	Bottom Fronts	18.5s	6.1%	16.1	5.6s	9.3%	5.4
	LR Total	301.7s	100.0%	6.8	60.3s	100.0%	4.3

- Compress has poor efficiency
- Updates are much slower in LR because of smaller granularity
- Multithreading bottom fronts: factor 3.3 out of 10 → W. Sid-Lakhdar's thesis
- Multithreading compress: factor 5.6 out of 10
- Multithreading LR update: factor 5 out of 10 (compare to 9.5 in FR)

Thanks!

Questions?