MUMPS User Days

5th edition

MUMPS group

CERFACS, CNRS, ENS-Lyon, LIP6, INRIA, INPT, Mumps Tech, University of Bordeaux

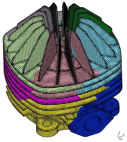
Paris-Sorbonne Univ., June 22-23, 2023

MUMPS overview and recent features

Keywords:
performance and accurary, data sparsity and mixed precision, computer architectures, MUMPS features, MUMPS for iterative solvers

(MUMPS a free software supported by industrials and public research)

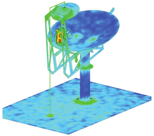
Outline



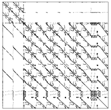
Code Aster (EDF)

Wide range of applications

(e.g. structural analysis, geoscience, electromagnetism, circuit simulation, finite element and optimization ...)



FEKO-EM (Altair)



$\Rightarrow$ Solve $\mathbf{AX} = \mathbf{B}$, with $\mathbf{A}$ a sparse matrix $\Rightarrow$

critical step in HPC simulations

Sparse direct linear solvers

Factor $\mathbf{A} = \mathbf{LU}$; Solve: $\mathbf{LY} = \mathbf{B}$, then $\mathbf{UX} = \mathbf{Y}$

Method of choice for its accuracy and robustness

Outline

MUMPS: a MUltifrontal Massively Parallel Solver

A robust package with a wide range of features using direct methods for solving

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

where **A** is a large sparse matrix, and **B** is dense or sparse

Co-developed by

CERFACS, CNRS, ENS Lyon, INPT, Inria, Univ. Bordeaux

and, since 2019, *Mumps Technologies*

MUMPS: brief history of a free software

- Multifrontal approach: Schreiber'82; Duff, Reid'83
- 1996-1999: MUMPS started in Toulouse (EU LTR project (PARASOL) inspired from a shared memory research code
- 2000: First "public domain" version of MUMPS, – `http://mumps-solver.org`
- 2014-2019: Consortium of MUMPS users
  Founding members: *CERFACS, INPT, Inria, ENS-Lyon, Bordeaux University*; Members: EDF, Altair, Michelin, LSTC (USA), SISW-Siemens (Belgium), ESI, Total, FFT/MSC Soft. (Belgium), SAFRAN, Lawrence Berkeley Nat. Lab. (USA)
- 2015: MUMPS 5.0.0, first CeCILL-C version of MUMPS
- 2019: Creation of Mumps technologies SAS to ensure software sustainability and development of MUMPS solver – `http://mumps-tech.com`
- 2023: Fifth edition of MUMPS User Days Meeting

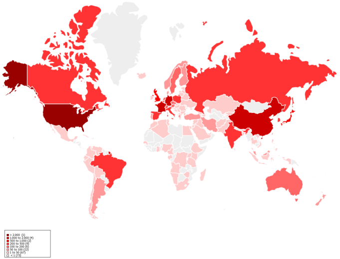
The MUMPS solver – http://mumps-solver.org

- Free software package, state-of-the-art in its field (fed by the research (13 theses)
- Original approach to parallelism: adaptable to the evolution of the machines

Users: developers of simulation software

- Used worldwide by industrials/academics
- Part of commercial and open-source packages
- User community (3 explicit software requests/day)
  - Users days every 3-4 years
  - $\simeq$ 800 users emails per year
  - mumps-users: $\simeq$ 600 subscribers

Map of the download requests



- Latest release: MUMPS 5.6.1 July 2023,
  $\approx$ 250 000 lines of C and Fortran code
- License: CeCILL-C

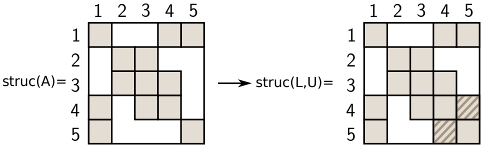
Partners supporting the MUMPS solver (Gold subscription to Mumps Technologies):

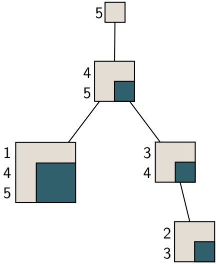
ALTAIR
AIRBUS
Ansys
CINES Centre Informatique National de l'Enseignement Supérieur
EDF
esi get it right®
HEXAGON
MICHELIN A BETTER WAY FORWARD
SAFRAN
SIEMENS
TotalEnergies

Solution of $\mathbf{AX} = \mathbf{B}$ *performed in 3 phases:*

($\mathbf{A}$ $n \times n$ sparse matrix with NZ non-zeros)

1. **analysis**, on the graph of $\mathbf{A}$
   - build ordering (METIS, SCOTCH, parMETIS, pt-SCOTCH, ...)
   - prepare factorization, build elimination tree





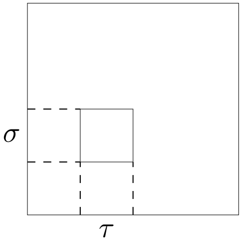
2. **numerical factorization**, decompose $\mathbf{A} = \mathbf{LU}$
   - work on dense matrices following elimination tree
   - stability relies on numerical pivoting
3. **solve**, forward and backward substitutions $\mathbf{LY} = \mathbf{X}$, $\mathbf{UX} = \mathbf{Y}$

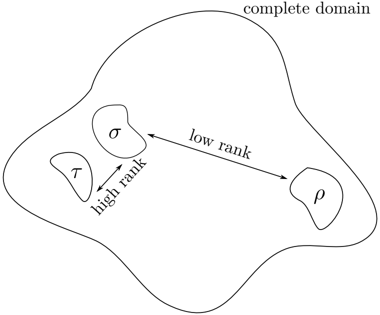
Outline

Table of Contents

Data sparsity

In some applications the matrices exhibit low-rank blocks





A block B represents the interaction
between two subdomains σ and τ.
Small diameter and far away $\Rightarrow$ low numerical rank.

$\Rightarrow$ Many representations: Recursive $\mathcal{H}, \mathcal{H}^2$, HSS, HODLR, BLR ...

Block Low-Rank general context and main features

- Approximate factorization $\mathbf{A} \approx \mathbf{L}_\varepsilon \mathbf{U}_\varepsilon$ at accuracy ε controlled by the user

Block Low-Rank[1] (BLR)

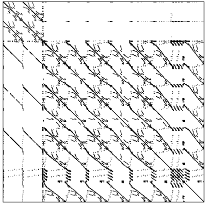
- Flat and simple format
  - Robust algebraic solver, compatible with the numerical features of a general solver
  - Backward stability proved by Higham and Mary (2021) IMA J. Num. Ana.[2]
- Reduced complexity, on a 3D regular problem of size $n = N^3$
  - Operations during facto. : Full-Rank, $\mathcal{O}\left(N^6\right) \rightarrow$ BLR, $\mathcal{O}\left(N^4\right)$
  - Size of LU factors : Full-Rank, $\mathcal{O}\left(N^4\right) \rightarrow$ BLR, $\mathcal{O}\left(N^3 \log N\right)$

Work supported by PhD theses: C. Weisbecker (2010-2013, EDF grant) T. Mary (2014-2017), B. Vieublé (2019-2022) and M. Gerest (2020-, EDF grant)

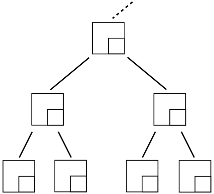
[1] *See publications in SIAM J. Sci. Comput. or ACM Trans. Math. Soft.: "Improving multifrontal methods by means of block low-rank representations" (2015), " On the complexity of the Block Low-Rank multifrontal factorization" (2017), "Bridging the gap between flat and hierarchical low-rank matrix formats: the multilevel BLR format" (2019), "Performance and Scalability of the Block Low-Rank Multifrontal Factorization on Multicore Architectures" (2018)*

[2] https://doi.org/10.1093/imanum/drab020

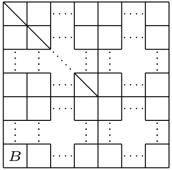
Block Low-Rank Multifrontal feature: principle



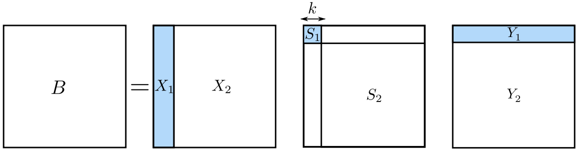
$\Rightarrow$



Elimination tree



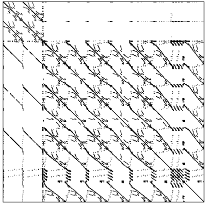
$\Rightarrow$



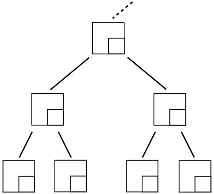
Singular value decomposition (SVD) of each block B

$\Rightarrow B = X_1 S_1 Y_1 + X_2 S_2 Y_2$

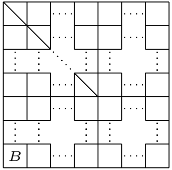
Block Low-Rank Multifrontal feature: principle



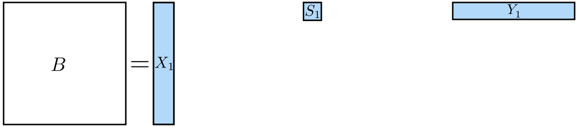
$\Rightarrow$



Elimination tree



$\Rightarrow$

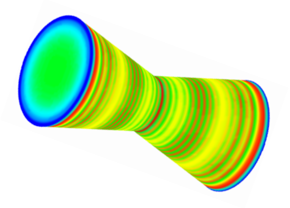


rank $k(\varepsilon)$: $B = X_1 S_1 Y_1 \cancel{+ X_2 S_2 Y_2}$

$\|E\|_2 = \|X_2 S_2 Y_2\|_2 = \sigma_{k+1} \leq \varepsilon$

Full-Rank/Block Low-Rank: flops ratio versus time ratio

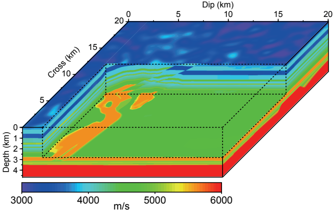
Required accuracy: 10^{-9}



Structural mechanics, $n = 8M$

Flop Ratio=17

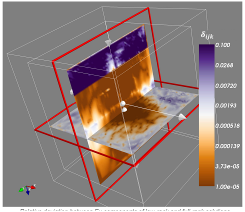
Required accuracy: 10^{-3}



Seismic imaging, $n = 17M$

Flop Ratio=27

Required accuracy: 10^{-7}

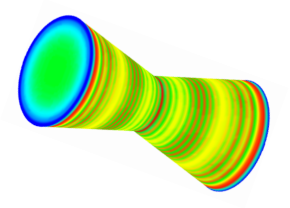


Electromagnetism, $n = 21M$

Flop Ratio=65

Full-Rank/Block Low-Rank: flops ratio versus time ratio

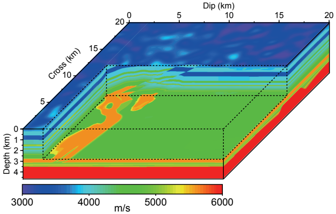
Required accuracy: 10^{-9}



Structural mechanics, $n = 8M$
Flop Ratio=17

→ **Time Ratio= 6**

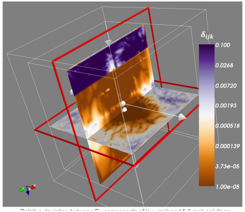
Required accuracy: 10^{-3}



Seismic imaging, $n = 17M$
Flop Ratio=27

→ **Time Ratio= 7**

Required accuracy: 10^{-7}



Electromagnetism, $n = 21M$
Flop Ratio=65

→ **Time Ratio=19**

Converting flop reduction into performance gains is not straightforward[a]

[a] *Amestoy, Buttari, L'Excellent, Mary, Performance and Scalability of the BLR Multifrontal Factorization on Multicores, ACM Trans. on Math. Soft. 2018*

Low precision arithmetics

	Signif. bits (t)	Exp.	Rang	$u = 2^{-t}$
fp64	53	11	$10^{\pm 308}$	1×10^{-16}
fp32	24	8	$10^{\pm 38}$	6×10^{-8}
fp16	11	5	$10^{\pm 5}$	5×10^{-4}
bfloat16	8	8	$10^{\pm 38}$	4×10^{-3}
fp8 (e4m3)	4	4	$10^{\pm 2}$	6×10^{-2}
fp8 (e5m2)	3	5	$10^{\pm 5}$	1×10^{-1}

Opportunities to:

- Reduce storage, data movement, and communication
- Increase speed and reduce energy
- However, reduce range and accuracy: low precision $\equiv$ low accuracy

$\rightarrow$ Motivation to use mixed precision algorithms

Mixed precision iterative refinement: algorithm

Factorize $A = LU$ **in low precision (ex: fp32)**
Solve $Ax_1 = b$ via $x_1 = U^{-1}(L^{-1}b)$ **in low precision (ex: fp32)**
repeat
  $r_i = b - Ax_i$ **in high precision (ex: fp64)**
  Solve $Ad_i = r_i$ via $d_i = U^{-1}(L^{-1}r_i)$ **in low precision (ex: fp32)**
  $x_{i+1} = x_i + d_i$ **in high precision (ex: fp64)**
until converged

Can achieve fp64 accuracy while mostly using fp32 precision
Works for moderately ill-conditioned problems; for very ill-conditioned ones, can use LU factors to precondition a more robust Krylov solver[3]

[3]See Amestoy et al. (2021), https://hal.archives-ouvertes.fr/hal-03190686

Mixed precision iterative refinement: results

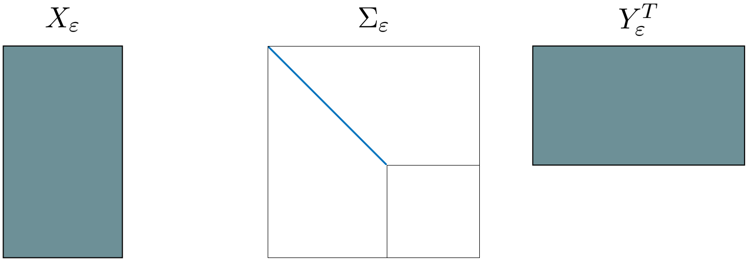
ElectroPhys10M matrix
$n = 10M$, well conditioned matrix, 4 MPI $\times$ 18 threads

	Time (s)	Memory (GB)	Backward error
DMUMPS	265	272	10^{-16}
SMUMPS+IR	154	138	10^{-16}
SMUMPS+BLR($\varepsilon = 10^{-6}$)+IR	65	78	10^{-16}

See Amestoy et al. (2023) ACM TOMS[4] for more results,
including ill-conditioned matrices

[4] https://doi.org/10.1145/3582493

Mixed precision low-Rank approximation: algorithm[5]

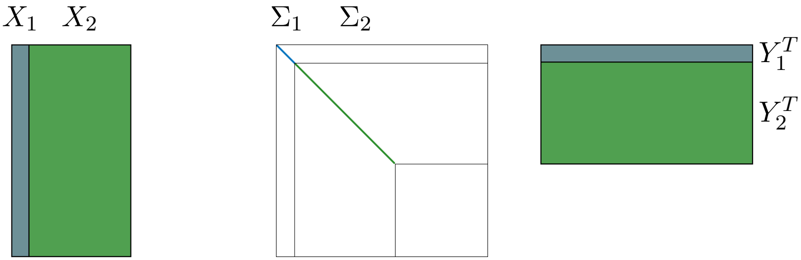


Truncated SVD

- $B = \sum_{k=1}^{r} x_k \sigma_k y_k^T$, with r such that
- $\|B - X_\varepsilon \Sigma_\varepsilon Y_\varepsilon^T\| \leq \varepsilon \|A\|$

[5] work supported by EDF

Mixed precision low-Rank approximation: algorithm[6]



Truncated SVD with 2-precision formats (fp64, fp32)

- Can convert X_2 and Y_2 to single precision
- Criterion for storing columns x_i, y_i in precision fp32: $\sigma_i \leq \frac{\varepsilon}{10 u_s} \|A\|$, with $u_s = 6 \times 10^{-8}$
- $\|B - X \Sigma Y^T\| \lesssim 1.2 \varepsilon \|A\| \Rightarrow$ preserved accuracy[5]

[5] see Amestoy et al. 2022, IMA JNA https://doi.org/10.1093/imanum/drac037

[6] work supported by EDF

Mixed precision low-rank approximation: results

thmgaz matrix
$(n = 8M)$

	Factor size	Total memory	Factorization time	Solve time	Backward error
BLR double	95	131	85	0.45	6×10^{-14}
BLR mixed	59	105	93	0.35	6×10^{-14}

$\Rightarrow$ Memory and solve time reductions, with preserved accuracy
Ongoing work on accelerating factorization

- BLR can be used to compress the working memory (so called Contribution Blocks (CB))

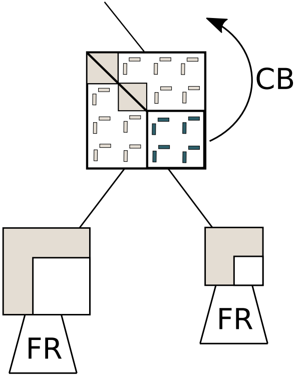
  ... at the cost of an increase in flops

  ... but with a reduction of the volume of communication

  Example on FWI at 13Hz (Helmholtz)

	Time (s)	Memory (GB)
FR	2131	4711
BLR	731	3258
BLR+compressCB	834	2623

- Extra compression possible using Mixed precision on such BLR blocks



More during the workshop related to BLR and/or mixed precision

- Bastien Vieublé, *A new backward error analysis framework for GMRES and its application to GMRES preconditioned with MUMPS in mixed precision*
- Matthieu Gerest, *Solving linear systems efficiently using BLR compression in mixed precision*
- Roméo Molina, *Adaptive precision algorithms for SpMV and iterative solvers*

Table of Contents

Analysis phase: main issues

Cost of analysis can be significant with respect to numerical phases

- TIME for ordering, symbolic factorization and BLR clustering
- QUALITY of analysis (impact on numerical factorization)
- MEMORY usage of sequential analysis

Tracks of improvement that have been followed

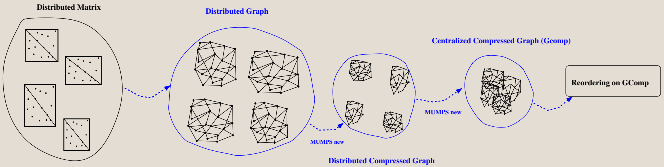
- Work with compressed graph → TIME, MEMORY
- BLR clustering on distributed [compressed] graphs → TIME, MEMORY
- Faster symbolic factorization → TIME and QUALITY, adapt to our needs the algorithm by [Gilbert, Ng, Peyton, SIAM J. Matrix Anal. Appl., 1994] since only column count needed

while following advances on graph partitioners →
see talk of F. Pellegrini "What to expect of a 30 y.o. Scotch"

Analysis by blocks

Objective: Exploit block structure of sparse matrices to reduce time and memory footprint of analysis phase
(k dofs per block $\Rightarrow$ k (k^2) less vertices (edges) to handle!)

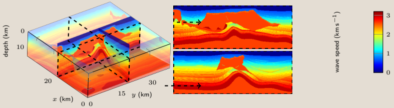
Block format $\rightarrow$ compressed graph G_{comp}



Reduce time+memory: $NZ_G \rightarrow NZ_{Gcomp} = \frac{1}{k^2} NZ_G$ (for k dofs per block)

Analysis by blocks, performance analysis

- **Time-harmonic wave problems** using Hybridizable Discontinuous Galerkin discretization [7] and p-adaptivity HDG Matrix 65k-2hz, general symmetric of order $n = 10M$, $NZ_{G(A+A^T)} = 5\,566M$, factorization on 360 cores (Olympe)



Elastic SEAM model, S-wave speed model. At 2Hz, the wavelength is between 300 and 1500 m.

- **Application in structural mechanics**, Matrix EngineAssy5M_32 unsymmetric of order $n = 4.6M$, $NZ_{G(A+A^T)} = 94M$, factorization on 64 cores (Olympe)

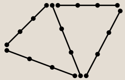
	HDG Matrix 65k-2hz	EngineAssy5M_32
	Analysis by blocks	
Time (sec) for	OFF	OFF
Analysis	337	90
Factorization	98	38

[7] M. Bonnasse-Gahot, H. Calandra, J. Diaz, and S. Lanteri, Hybridizable discontinuous Galerkin method for the 2D frequency-domain elastic wave equations, Geophysical Journal International, 213 (2017), pp. 637-659.

Analysis by blocks, performance analysis

- **Time-harmonic wave problems** using Hybridizable Discontinuous Galerkin discretization and p-adaptivity HDG Matrix 65k-2hz, general symmetric of order $n = 10M$, $NZ_{G(A+A^T)} = 5\ 566M$, factorization on 360 cores (Olympe)

  dofs on cell faces have identical adjacencies and form blocks in the matrix

  $$n/n_{comp} = 24 \longrightarrow NZ_G/NZ_{Gcomp} = 699 > 24^2$$

- **Application in structural mechanics**, Matrix EngineAssy5M_32 unsymmetric of order $n = 4.6M$, $NZ_{G(A+A^T)} = 94M$, factorization on 64 cores (Olympe)
  average number of dofs per grid point is 3

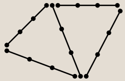
  $$n/n_{comp} = 3 \qquad \longrightarrow \qquad NZ_G/NZ_{Gcomp} = 3^2$$

	HDG Matrix 65k-2hz	EngineAssy5M_32
	Analysis by blocks	
Time (sec) for	OFF	OFF
Analysis	337	90
Factorization	98	38

Analysis by blocks, performance analysis

- **Time-harmonic wave problems** using Hybridizable Discontinuous Galerkin discretization and p-adaptivity HDG Matrix 65k-2hz, general symmetric of order $n = 10M$, $NZ_{G(A+A^T)} = 5\,566M$, factorization on 360 cores (Olympe)

  dofs on cell faces have identical adjacencies and form blocks in the matrix

  $$n/n_{comp} = 24 \longrightarrow NZ_G/NZ_{Gcomp} = 699 > 24^2$$

- **Application in structural mechanics**, Matrix EngineAssy5M_32 unsymmetric of order $n = 4.6M$, $NZ_{G(A+A^T)} = 94M$, factorization on 64 cores (Olympe)

  average number of dofs per grid point is 3

  $$n/n_{comp} = 3 \qquad \longrightarrow \qquad NZ_G/NZ_{Gcomp} = 3^2$$

	HDG Matrix 65k-2hz		EngineAssy5M_32	
	Analysis by blocks			
Time (sec) for	OFF	ON	OFF	ON
Analysis	337	**7**	90	**44**
Factorization	98	98	38	38

Table of Contents

Evolution of Schur feature

- Schur feature, ICNTL(19)=1
  compute $\mathbf{S} = \mathbf{A}_{2,2} - \mathbf{A}_{2,1}\mathbf{A}_{1,1}^{-1}\mathbf{A}_{1,2}$, with $\mathbf{A} = \begin{pmatrix} \mathbf{A}_{1,1} & \mathbf{A}_{1,2} \\ \mathbf{A}_{2,1} & \mathbf{A}_{2,2} \end{pmatrix}$
- Threshold pivoting limited to $\mathbf{A}_{1,1}$ (since MUMPS 5.4)
  to reduce factorization time: 60sec $\rightarrow$ 18sec (vibro-acoustic application)

Evolution of Schur feature

- Schur feature, ICNTL(19)=1
  compute $\mathbf{S} = \mathbf{A}_{2,2} - \mathbf{A}_{2,1}\mathbf{A}_{1,1}^{-1}\mathbf{A}_{1,2}$, with $\mathbf{A} = \begin{pmatrix} \mathbf{A}_{1,1} & \mathbf{A}_{1,2} \\ \mathbf{A}_{2,1} & \mathbf{A}_{2,2} \end{pmatrix}$
- Threshold pivoting limited to $\mathbf{A}_{1,1}$ (since MUMPS 5.4)
  to reduce factorization time: 60sec $\rightarrow$ 18sec (vibro-acoustic application)

- Scaling rows and columns of $\mathbf{A}$ $(\mathbf{D_r A D_c})$ was not available with Schur feature
- Scaling can be computed on $\mathbf{A}$ and be restricted to $\mathbf{A}_{1,1}$ (since MUMPS 5.5)

Complex Symmetric matrix, N=63K, Schur size=163

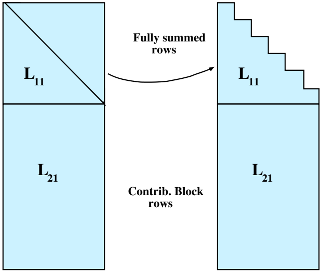
(Olympe Computer on 8 cores, 1 node, 1 MPI)

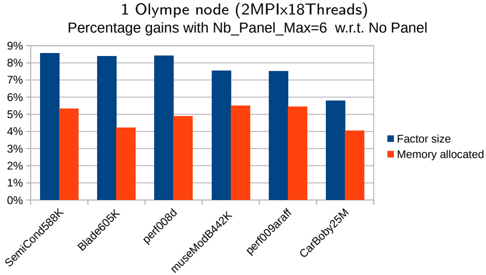
	Flops	Facto. time (sec)	Memory Mbytes
without scaling	1.5D+12	208	4878
with scaling restricted to $\mathbf{A}_{1,1}$	1.9D+09	7	624

Storage of L factors of a symmetric matrix

Storage of $\mathbf{L}$ factors of symmetric matrices can be compacted

(already done for OOC and BLR fronts)





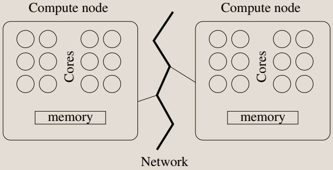
- Memory: reduction of factor size converts into memory gains (with only 6 panels $\rightarrow$ 80% of theoretical gain)
- Time: positive impact on solve phase time

(available since MUMPS 5.5)

Outline

Table of Contents

Managing parallelism: multicores



Many cores sharing memory per compute node

Hybrid parallelization

- Distributed memory parallelism (MPI based)

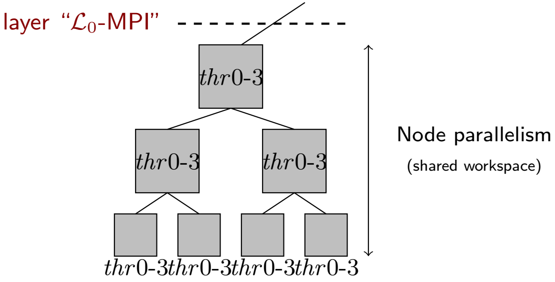
  combined to shared-memory parallelism (multithreading):
  - use of multithreaded BLAS
  - OpenMP directives

Nb of cores per node increases → more multithreading need be exposed

Multithreaded tree parallelism feature

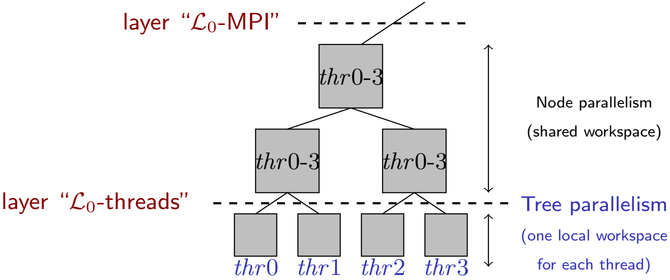
Strategy for hybrid parallelization (case of multiple threads per MPI process):

- parallelism **between nodes** of the elimination tree (MPI only)
- parallelism **within nodes** (MPI and OpenMP)
- under "$\mathcal{L}_0$-MPI": **one** MPI process per subtree (to limit communication)

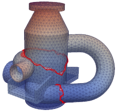


Strategy for hybrid parallelization (case of multiple threads per MPI process):
(work based on W. M. Sid-Lakhdar's PhD thesis, 2014)

- parallelism **between nodes** of the elimination tree (MPI and **OpenMP**)
- parallelism **within nodes** (MPI and OpenMP)
- under "$\mathcal{L}_0$-MPI": **one** MPI process per subtree (to limit communication)
- under "$\mathcal{L}_0$-threads": **one** thread per subtree



$\mathcal{L}_0$-threads feature: impact on BLR and solve phase



Code_Aster RIS pump

Matrix from structural mechanics, real symmetric:
perf009ar from EDF

	Time(sec) on 2 MPI × 18 Threads		Memory allocated
	Factorisation	Solve (1 RHS)	(Gbytes)
$\mathcal{L}_0$-thread OFF	(36 MPI × 1 Thread)		
Full-rank	31.2	0.54	82
BLR ($\varepsilon_{blr} = 10^{-9}$)	**22.4**	**0.41**	**72**
	(2 MPI × 18 Threads)		
Full-rank	45.8	2.70	54
BLR ($\varepsilon_{blr} = 10^{-9}$)	**41.3**	**2.58**	**36**
$\mathcal{L}_0$-thread ON	(2 MPI × 18 Threads)		
Full-rank	28.0	0.61	55
BLR ($\varepsilon_{blr} = 10^{-9}$)	**18.7**	**0.50**	**39**

$\mathcal{L}_0$-thread feature: few take away messages

- Hybrid parallelism versus full MPI
  - To reduce memory footprint: increase the number of threads/MPI
  - Performance of factorization often comparable even if solve benefits more from MPI
  - Preprocessing during numerical phase need be multithreaded

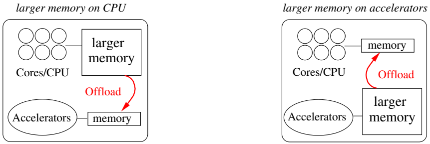
- With $\mathcal{L}_0$-thread feature:
  - factorization time reduction higher in BLR
  - solve time reduction higher than factorization
  - memory footprint slightly higher than without $\mathcal{L}_0$-thread feature

$\rightarrow$ *$\mathcal{L}_0$-thread feature scheduled to be available in MUMPS 5.7 (April 2024)*

Table of Contents

Exploit accelerators

Types of compute nodes with accelerators



Collaborations

- Larger memory on CPU: offload from CPU to GPU (support of Altair), use runtime libraries for BLAS on GPU:
  - cublasXt: provided by Nvidia
  - `XKBlas`: collaboration with Inria-ENS Lyon, also supports AMD GPU
- Larger memory on accelerator
  - NEC SX-Aurora vector processor (collaboration with NEC): offload scalar parts from vector engine to CPU
  - OpenMP 5 approach: target new supercomputer nodes from French national center[7]: 512 GB on four AMD MI250X GPU, 256 GB on CPU

[7] collaboration with GENCI, CINES, HPE and AMD

Offload from CPU to GPU (collaboration with Altair)

External libraries can take care of tiling, allocation/memory management on GPU, CPU $\leftrightarrow$ GPU data transfers

- cublasXt: provided by NVIDIA
- XKBlas: collaboration with T. Gautier[8] (LIP laboratory, ENS Lyon)

Both cublasXt and XKBlas rely on cublas.

Offload approach (will be available in MUMPS 5.7, April 2024)

if Arithmetic Intensity of frontal matrix "large enough" (AI_Threshold) **then**
  Adjust blocking; asynchronous memory pinning
  Wrap `GEMM/TRSM` to call cublasXt or XKBlas
else
  Standard multicore processing of frontal matrix
end if

AI_Threshold depends on cublasXt or XKBlas, GPU type, CPU cores

[8] Gautier et al., A Runtime System for [...] on Heterogeneous Architectures [. . .], IPDPS 2013

Portability of offloading from CPU to GPU approach

XKBlas evolutions (T. Gautier, Inria LIP-ENS Lyon)

- Improve performance and robustness, reduce memory transfers
  - chain GPU operations,
  - new kernels (GEMMT, copy-scale algorithm under discussion)
- Portability on AMD GPUs: XKBlas for AMD GPUs

  (available on the GitLab of XKBlas:
  https://gitlab.inria.fr/xkblas/versions/-/blob/master/xkblas-v0.4-rc7-0-g513c021b.tgz)

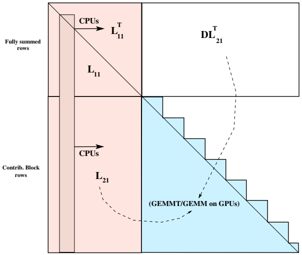
  Preliminary results on MUMPS from T. Gautier:

	LDLT MechaStruct8M		LU 3D Laplacian	
	AMD MI50	V100	AMD MI50	V100
(*)0 GPU	796s	780s	719	749
1 GPU	367s	358s	298	371
2 GPU	292s	295s	240	266

(*) 36 cores, AMD or Intel; (**) benefits from NVLINK

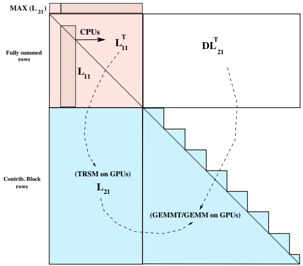
- Ongoing: test and tune XKBlas on recent AMD GPU (8 MI100, 4 MI250X) in the context of GENCI-CINES-HPE-AMD collaboration

Symmetric indefinite matrices on GPU: influence of numerical pivoting



- Numerical pivoting CNTL(1)=0.01
  - Less BLAS-3 kernels than with CNTL(1)=0.0
  - BLAS-2 update of $\mathbf{L}_{21}$:
    - performance issue on GPUs but also on multicores

CALMIP Olympe computer, time for factorization in seconds 3D Wave equations, 18 cores, XKBlas	
	Factorization time (sec)
Numerical pivoting OFF (CNTL(1)=0)	
18 cores	382
18 cores + 1 GPU	**201**
Numerical pivoting ON (CNTL(1)=0.01)	
18 cores	432
18 cores + 1 GPU	**352**



- Numerical pivoting CNTL(1)=0.01
  - Less BLAS-3 kernels than with CNTL(1)=0.0
  - BLAS-2 update of $\mathbf{L}_{21}$:
    - performance issue on GPUs but also on multicores
- Relaxed pivoting with CNTL(1)=0.01
  - Vector of column norms used to represent block $\mathbf{L}_{21}$ see Duff, Pralet SIAM SISC (2007)
  - Enable BLAS-3 update of $\mathbf{L}_{21}$:

CALMIP Olympe computer, time for factorization in seconds 3D Wave equations, 18 cores, XKBlas	
	Factorization time (sec)
Numerical pivoting OFF (CNTL(1)=0)	
18 cores	382
18 cores + 1 GPU	**201**
Numerical pivoting ON (CNTL(1)=0.01)	+ Relaxed pivoting
18 cores	~~432~~ $\rightarrow$ 379
18 cores + 1 GPU	~~352~~ $\rightarrow$ **216**

More during the workshop related to computer driven activities

- Robert Lucas, *A Changing Landscape*
- Thierry Gautier, *XKBlas: under the hood*
- Julien Minière, *Porting MUMPS on CINES-ADASTRA with OpenMP 5*

Enjoy those two days!

MUMPS User Days

June 22-23, 2023

Sorbonne University, Paris, France

Thursday June 22nd

08.30 – 08.45	Registration and welcome coffee
08.45 – 09.00	Fabienne Jézéquel (LIP6) Welcome and presentation of the two day meeting
09.00 – 09.45	MUMPS Group MUMPS overview and recent features
09.45 – 10.15	*Eric Lequiniou (Altair, France)* Leveraging MUMPS to enhance performance of Altair Solvers
10.00 – 10.30	François-Henry Rouet (Ansys, USA) MUMPS-BLR inside a preconditioned eigensolver
10.35 – 11.05	**Coffee Break**
11.05 – 11.25	Olivier Boiteau (EDF Lab Paris-Saclay, France) Feedback in the use of MUMPS in EDF codes
11.25 – 11.45	Ramzi Messahel (Safran Tech, France) Feedback on the use of MUMPS in Safran Tech's applications: a multithread performance evaluation of MUMPS
11.45 – 12.05	Jason Pavlidis (Moduleering Company, Greece) Scalability of normal modes computation in Sunshine using MUMPS solver
12.05 – 12.35	Robert Lucas (Ansys, USA) A Changing Landscape
	Lunch
14.20 – 14.50	Bastien Vieublé (University of Manchester, United Kingdom) A new backward error analysis framework for GMRES and its application to GMRES preconditioned with MUMPS in mixed precision
14.50 – 15.10	Matthieu Gerest (EDF R&D, LIP6-Sorbonne University, France) Solving linear systems efficiently using Block Low-Rank compression in mixed precision
15.10 – 15.30	Roméo Molina (LIP6-Sorbonne University, France) Adaptive precision algorithms for SpMV and iterative solvers
15.30 – 16.00	**Break**
16.00 - 16.30	Hélène Barucq (Inria, France) Performance of time-harmonic wave modeling and inversion using Hybridizable Discontinuous Galerkin discretization and the MUMPS solver
16.30 -17.00	Stéphane Operto (CNRS, France) High-resolution seismic imaging of the subsurface with MUMPS: Solving the time-harmonic wave equation with multiple sparse right-hand sides in large computational meshes
17.00 – 17.30	Chia Wei Hsu (University of Southern California, USA) Augmented partial factorization: efficient computation of the generalized scattering matrix
17.30 – 18.10	MUMPS Group MUMPS on-going work and perspectives. Exchanges and discussions
19.15 – 22.00	**Banquet**

Friday June 23rd

08.30 – 09.00	Welcome coffee
09.00 – 09.30	François Pellegrini (Bordeaux University & Inria, France) What to expect of a 30 y.o. Scotch
09.30 – 10.00	Thierry Gautier (Inria, France) XKBlas: under the hood
10.00 – 10.20	Julien Minière (AS+/Eolen, France) Porting MUMPS on CINES-ADASTRA with OpenMP 5
10.20 – 10.50	**Coffee Break**
10.50 – 11.10	Pierre Jolivet (CNRS, France) MUMPS in PETSc and HPDDM
11.10 – 11.30	Henning Leemhuis (University of Wuppertal, Germany) Direct solves on a multigrid solver in lattice quantum chromodynamics (QCD)
11.30 – 12.00	Cédric Content & Emeric Martin (ONERA, France) High-fidelity simulations of turbulent compressible flows in ærodynamics: some typical applications with ONERA CFD codes
12.00 – 12.20	Yuri Feldman (Ben Gurion University, Israël) Block-Gauss-Seidel Immersed Boundary Method Accelerated by MUMPS
12.20 – 12.30	**Closing session**
12.30 – 14.00	**Lunch**
14:00 – 15.00	**Informal technical discussion between MUMPS group and interested participants**

Credits

This event is supported by

Mumps Technologies · SORBONNE UNIVERSITÉ · LIP · cnrs · EDF

MUMPS ongoing work and perspectives for future releases. Exchanges and discussions

Keywords:
New features and ongoing work; OpenMP 5 to target heterogeneous nodes, Data sparsity and mixed precision

(MUMPS a free software supported by industrials and public research)

Outline

Solve: current features for fwd ($Ly = b$) and bwd ($Ux = y$)

$Ly = b$ (fwd) and format for b

- Dense RHS(1:N,NRHS)
- Sparse RHS_SPARSE,...: exploit sparsity in fwd
- Distributed [I]RHS_loc: sum duplicates, non available rows considered 0, but sparsity not exploited

$Ux = y$ (bwd) and format for x

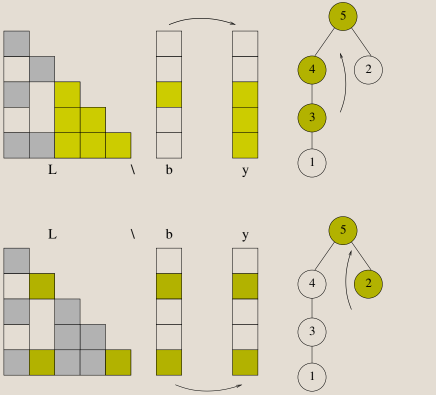
- Dense: overwrite RHS(1:N,NRHS)
- Distributed and dense: [I]SOL_loc, distribution imposed by MUMPS (no sparsity)
- A^{-1} entries ([I]RHS_SPARSE,IRHS_PTR), exploit sparsity in fwd and bwd

Solve evolutions: some expectations from users

- Distributed RHS with empty rows (structured sparse): exploit sparsity to reduce computations
- Format of solution: compute only a subset of solution
- Schur computed by blocks at solve (e.g. when too big to be done at facto): solve features to reduce memory/computations
- Separate forward and backward substitution phases
  (+ diagonal system or access to diagonal matrix D in case of LDL^T factorization)
- User mapping of solution ISOL_loc (entire solution, subset of solution)

Exploiting sparsity during forward substitution

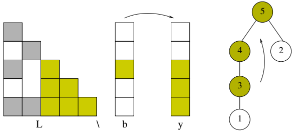
solve $Ly = b$

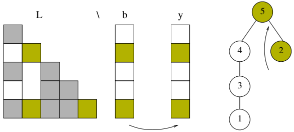


- Tree pruning: process only nodes on paths from b entries up to root [Gilbert-Liu 93]
- Combinatorial problems for multiple RHS (RHS_SPARSE, entries of A^{-1}):
  - see PhD theses of M. Slavova (2009), F.H. Rouet (2012) and G. Moreau (2018)
- Backward substitution: similar tree pruning, keep paths from root to selected entries

Distributed RHS/solution: current API and possible evolution

- **Distributed RHS (ICNTL(20)=10, 11):**
  - user-defined distribution IRHS_loc(1:Nloc_RHS) (local lists of global indices)
  - return distributed RHS_loc(1:LRHS_loc×NRHS)
  - **Possible evolution:** exploit sparsity to reduce computations and internal storage when some rows do not appear in IRHS_loc
    - IRHS_loc=[3] → process only nodes 3, 4, 5
    - IRHS_loc=[2 5] → process only nodes 2, 5



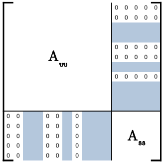


- **Distributed solution (ICNTL(21)=1,JOB=3)**
  - MUMPS returns on each MPI process global solution indices ISOL_loc(1:INFO(23)) + corresponding distributed solution SOL_loc(1:LSOL_loc×NRHS)
  - LSOL_loc≥INFO(23): leading dimension of SOL_loc
  - **Possible evolution:** ISOL_loc defined by user
  - Case of empty rows: exploit sparsity in backward substitution→ large memory and computational gains

Application: another way to compute Schur complement

Schur of $A = \begin{bmatrix} A_{vv} & A_{sv}^T \\ A_{sv} & A_{ss} \end{bmatrix} \Rightarrow S = A_{ss} - A_{sv}Z$, where $Z = (L_{vv}L_{vv}^T)^{-1}A_{sv}^T$

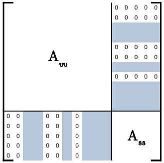
$\begin{bmatrix} A_{vv} & A_{sv}^T \\ A_{sv} & A_{ss} \end{bmatrix}$ has the following structure:



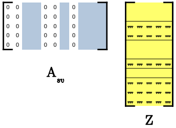
Application: another way to compute Schur complement

Schur of $A = \begin{bmatrix} A_{vv} & A_{sv}^T \\ A_{sv} & A_{ss} \end{bmatrix} \Rightarrow S = A_{ss} - A_{sv}Z$, where $Z = (L_{vv}L_{vv}^T)^{-1}A_{sv}^T$

$\begin{bmatrix} A_{vv} & A_{sv}^T \\ A_{sv} & A_{ss} \end{bmatrix}$ has the following structure:



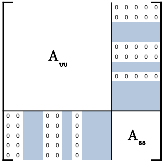
Z is a dense matrix and $A_{sv}Z$ has the following structure:



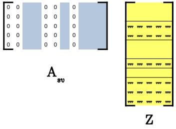
Application: another way to compute Schur complement

Schur of $A = \begin{bmatrix} A_{vv} & A_{sv}^T \\ A_{sv} & A_{ss} \end{bmatrix} \Rightarrow S = A_{ss} - A_{sv}Z$, where $Z = (L_{vv}L_{vv}^T)^{-1}A_{sv}^T$

$\begin{bmatrix} A_{vv} & A_{sv}^T \\ A_{sv} & A_{ss} \end{bmatrix}$ has the following structure:

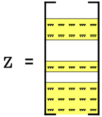


Z is a dense matrix and $A_{sv}Z$ has the following structure:



Only rows in Z corresponding to non-zero columns in A_{sv} are needed

(i.e., only unknowns of the volume in contact with surface)



Z computed with exploit sparsity.

Outline

Null space detection

Null pivot row detection feature (ICNTL(24)=1)

- A pivot row is considered as null if its norm is smaller than a given threshold controlled by the user and defined by CNTL(3)
- The choice of CNTL(3) can be delicate and matrix dependent
  $\rightarrow$ the approach is not always numerically robust.

Null space detection

Null pivot row detection feature (ICNTL(24)=1)

- A pivot row is considered as null if its norm is smaller than a given threshold controlled by the user and defined by CNTL(3)
- The choice of CNTL(3) can be delicate and matrix dependent
  $\rightarrow$ the approach is not always numerically robust.

Rank revealing algorithm (collaboration with SAFRAN) (will be available in MUMPS 5.7)

- Introduce notion of pseudo singularities to pospone variables up to the root node
- Singular value decomposition (SVD) is applied on the root node (with all pseudo singularities) to determine real singularities (the approach is compatible with ICNTL(24))
- CNTL(3) is the unique threshold parameter defined by the user

  Default setting CNTL(3)=0 was shown to be robust on (symmetric matrices from floating subdomain (Neumann problem) and unsymmetric matrices from advection diffusion problem)

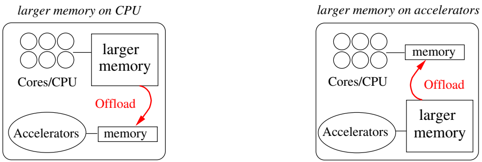
Outline

Diversity of high performance computers and programming issues

- *Manycore processors with vector extension instructions*:
  Classical shared/distributed parallel programming models

- *Manycore processors with accelerators*
  - Constructor specific languages or OpenMP 5 (target ...)

Types of compute nodes with accelerators



  - Larger memory on CPU: offload from CPU to accelerators using libraries designed for accelerators
  - Larger memory on accelerator: vector data on GPU and scalar data processed on CPU possibly offloaded from accelerators to CPU

Table of Contents

Porting MUMPS with OpenMP 5 on ADASTRA-CINES

collaboration ("Contrat de progrès") between CINES, GENCI, and HPE, EOLEN, AMD and Mumps Technologies

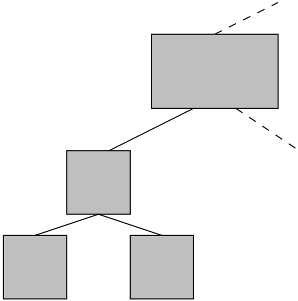
Portable approach based on OpenMP 5

- Data location:
  - Large real arrays in GPU memory
    $\rightarrow$ all computations on large real arrays done on GPU
  - Integer data, symbolic information and flow control remains on CPU
- MPI communications: enable both CPU$\leftrightarrow$CPU and GPU$\leftrightarrow$GPU
- Methodology for porting: bottom-up approach with code functional during development

(Julien Minière, *Porting MUMPS on CINES-ADASTRA with OpenMP 5*)

Algorithmic issues

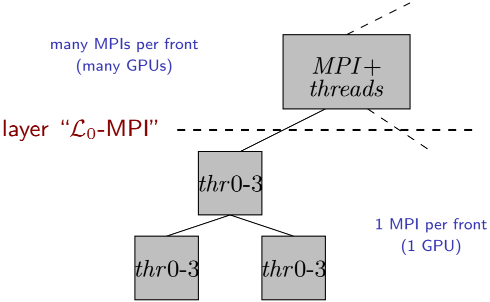
Mapping of the dependency tree



At each node of the dependency tree: partial factorization of a front (dense frontal matrix)

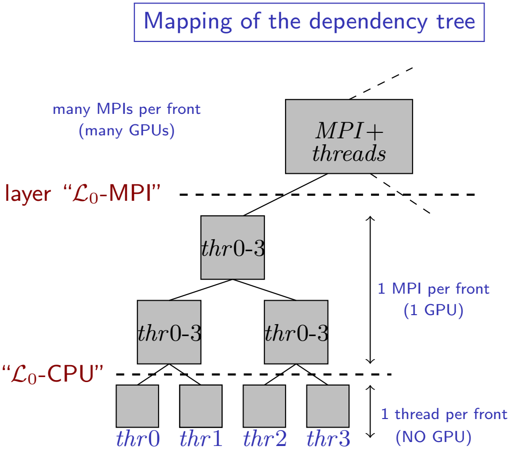
Algorithmic issues

Mapping of the dependency tree



- One GPU per MPI process

Algorithmic issues

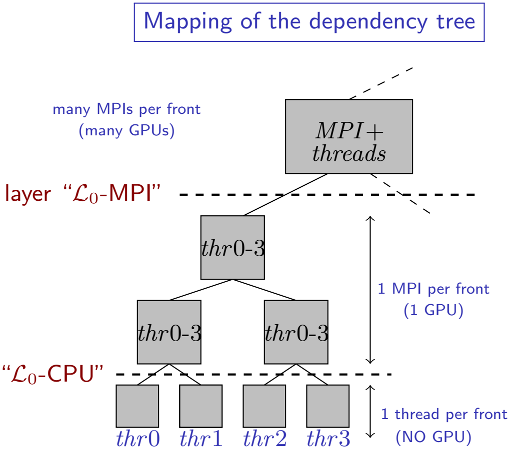


- One GPU per MPI process
- Computation on GPU:

  Limit nb of kernel calls on small data
  - compute on CPU under "$\mathcal{L}_0$-CPU"
    → small matrices processed on CPU

    → boolean `CompOnGPU`
    to indicate if data is mapped on GPU

Algorithmic issues



- One GPU per MPI process
- Computation on GPU:

  Limit nb of kernel calls on small data
  - compute on CPU under "$\mathcal{L}_0$-CPU"
    $\rightarrow$ small matrices processed on CPU

    $\rightarrow$ boolean `CompOnGPU`
    to indicate if data is mapped on GPU
- MPI communications:
  - For portability, enable both
    CPU $\leftrightarrow$ CPU and GPU $\leftrightarrow$ GPU

    $\rightarrow$ boolean `CommOnGPU`
    to exploit GPU $\leftrightarrow$ GPU communication

`CompOnGPU`: *automatically set depending on front position in the dependency tree*

`CommOnGPU`: *to be set if MPI library supports GPU $\leftrightarrow$ GPU communications*

Porting on GPU with OpenMP 5 - ongoing work

- **Ongoing**: LU factorization phase

  Performance analysis and performance comparison with offload approach based on XKBlas

- **Next steps:**
  - Performance profiling and tuning
    - low arithmetic intensity kernels: pivot search, idamax calls, swaps, in-place copies, ...
    - multi MPI/multi GPUs (with direct MPI communication between GPUs)
  - Still to be ported: LDL^T factorization, solution phase (triangular solves), advanced pivot strategies, block low rank (BLR), single+complex arithmetic

Promising approach for efficiency and portability ... still much work to be done

Table of Contents

Tuning MUMPS for NEC vector engine

Experimental environment

- NEC SX-Aurora 10B, 8 cores, 2.15 Tflops/s peak, 48 GBytes
- 3D frequency-domain FWI, 27-point stencil (Geoazur – S. Operto)

Performance of tuned version for VE (since MUMPS 5.5)

Matrix	N (1E6)	complex flops	factors GBytes	1MPIx8threads Gflops/s	1MPIx8threads LU factorization time
$\text{Geo}115^3$	1.52	3.1E13	32.2	717(*)	44 s

(*)corresponds to **2.8 Tflops/s** in single precision, real arithmetic (peak = 4.3 Tflops/s).

Tuning MUMPS for NEC vector engine

Experimental environment

- NEC SX-Aurora 10B, 8 cores, 2.15 Tflops/s peak, 48 GBytes
- 3D frequency-domain FWI, 27-point stencil (Geoazur – S. Operto)

Performance of tuned version for VE (since MUMPS 5.5)

Matrix	N (1E6)	complex flops	factors GBytes	1MPIx8threads Gflops/s	1MPIx8threads LU factorization time
Geo115^3	1.52	3.1E13	32.2	717(*)	44 s

(*)corresponds to **2.8 Tflops/s** in single precision, real arithmetic (peak = 4.3 Tflops/s).

Block Low-Rank compression ($\epsilon = 10^{-4}$) on NEC VE (since MUMPS 5.6)

NEC SX-Aurora VE	Memory used (GBytes)		LU factorization time	
1MPIx8threads	Full-Rank	Block Low-Rank	Full-Rank	Block Low-Rank
10B 1.4 GHz	42	38	44 s	39 s
20B 1.6 GHz	42	38	38 s	**27 s**

Outline

Closing Session

Concluding remarks

About the workshop

- 58 participants
  from Austria, Belgium, France, Germany, Greece, Israel, Luxembourg, UK, USA,
  - 35 industrials
  - 23 academics

  37 participated to the banquet
- 21 talks
  - 2 talks from MUMPS related activity PhD (or recent) students
  - 2 talks from MUMPS group
  - 8 talks from industrials
  - 9 talks from public researchers

Merci à Sorbonne Université pour son accueil

Merci aux organisateurs:
Fabienne Jezequel, Théo May, Chiara Puglisi

The event was supported by:

Experimental environment

- CALMIP center of Toulouse (grant number P0989):

  Olympe nodes

  - CPU node: Two Intel 18-cores Skylake 6140 @2.3 GHz (Peak/core=73.6 GF/s, Peak/node=2.6 TFlops/s DP), 192 GB memory per node
  - GPU node: Two Intel 18-cores Skylake 6140 @2.3 GHz (Peak/core=73.6 GF/s, Peak/node=2.6 TFlops/s DP), 384 GB memory per node,
    4 GP-GPU Nvidia Volta (V100 - 7.8 TFlops/s DP)

  TURPAN from MésoNET project, experimental computer: 15 nodes with

  - Ampere Altra Max Q80-30 (ARM version 8.2) 80 cores @3 GHz (peak 24Gflops/s/core), peak 1.9 Tflops/s DP, 512 GB memory
  - 2 GPU Nvidia A100-80, 19.5 Tflops/s DP per GPU, 2x80 GB memory

- GENCI-CINES, ADASTRA supercomputer: HPE Cray EX235a
  - 61.6 PFlops/s peak, 46 PFlops/s (Linpack); 50 GFlops/Watt
  - accelerated nodes based on AMD Optimized 3rd Generation EPYC 64C 2.4 GHz,
    512 GB on four AMD Instinct MI250X GPU, 256 GB on CPU

- NEC SX-Aurora Type 10B with 8 cores, 1.4GHz, 2.15 TFlops/s peak, 48 GB and NEC SX-Aurora Type 20B, 1.6GHz