Inria Skoltech Moliere Associated Team

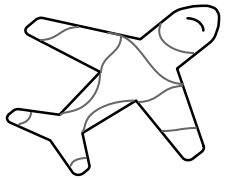
Exploiting Generic Tiled Algorithms Toward Scalable H-Matrices Factorizations on Top of Runtime Systems

6th July, 2021

Rocío Carratalá-Sáez	*Universitat Jaume I (Spain)*
Mathieu Faverge	*Bordeaux INP, Inria, LaBRI, (France)*
Grégoire Pichon	*Univ Lyon, EnsL, UCBL, CNRS, Inria, LIP (France)*
Enrique S. Quintana-Ortí	*Universitat Politècnica de València (Spain)*
Guillaume Sylvand	*Airbus Group Innovations (France)*

UNIVERSITAT JAUME I

Boundary element methods

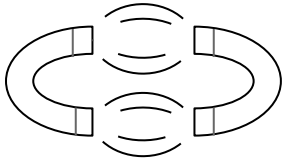


Circuit Physics

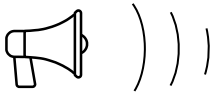
Large-scale systems of linear equations

Geodesy

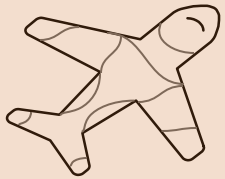
Electromagnetism



Acoustic scattering



Boundary element methods



Hierarchical collections of dense and low-rank blocks matrices

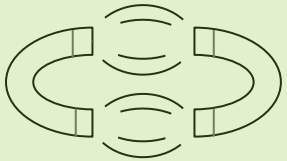
Sparse matrices

Circuit Physics

Large-scale systems of linear equations

Geodesy

Electromagnetism



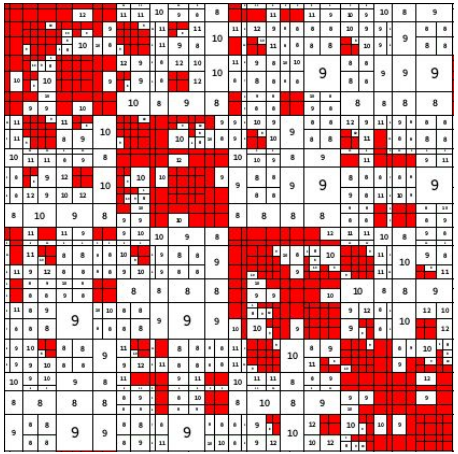
Dense matrices

Acoustic scattering

Overview

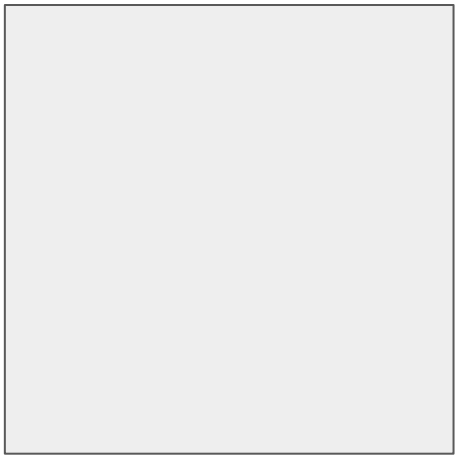
- **Brief introduction to H-Matrices & H-LU**
  - Previous work & alternative approaches
  - Our approach: H-Chameleon
  - Results & Conclusions
  - Future Work

Brief introduction to Hierarchical Matrices



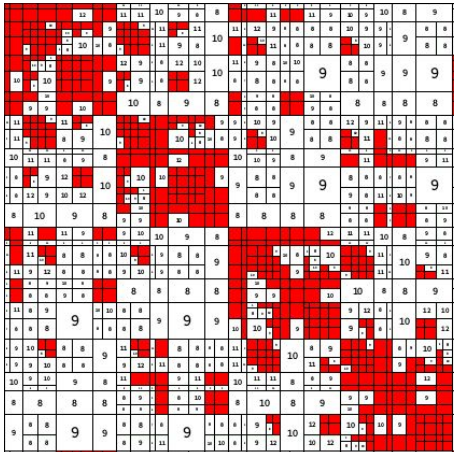
Brief introduction to Hierarchical Matrices

Dense Matrix arising from Boundary Element Methods



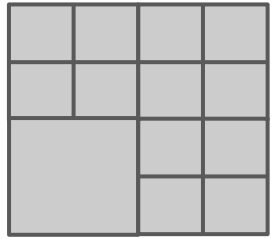
?

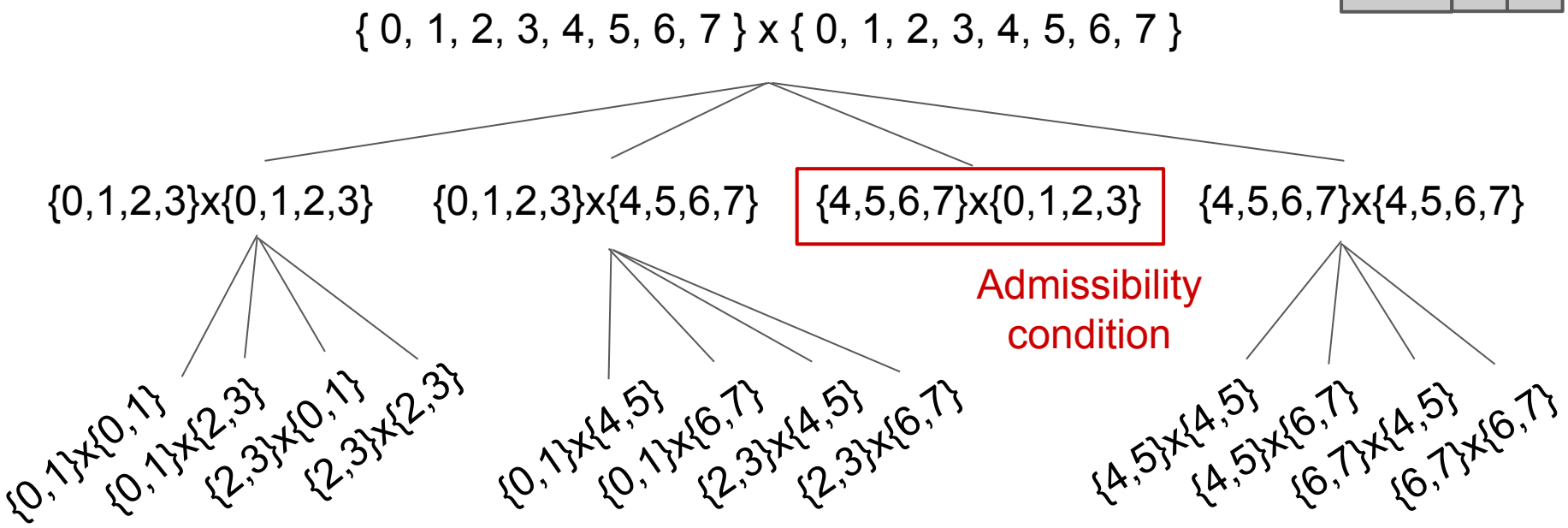
Hierarchical Matrix (H-Matrix)



Brief introduction to Hierarchical Matrices

1) Cluster Trees of the matrix indices (rows + columns)
2) Block cluster tree → defines a partition
3) Apply the partition





Brief introduction to Hierarchical Matrices

- Cluster Trees of the matrix indices (rows + columns)
- Block cluster tree to define a partition
  - Admissibility condition
  - Partitioning criteria
  - Minimum block size

low-rank blocks & dense blocks

✓ **log order in STORAGE [1]**
✓ **log order in COMPUTATIONS [1]**

- Apply the partition

[1] W. Hackbusch. *A sparse matrix arithmetic based on H-matrices*. Computing (1999).

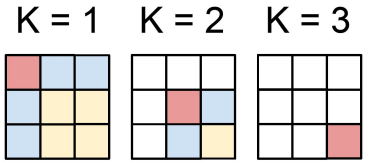
Overview

UNIVERSITAT JAUME I

- Brief introduction to H-Matrices & H-LU
- **Previous work & alternative approaches**
  - Our approach: H-Chameleon
  - Results & Conclusions
  - Future Work

LU Factorization of H-Matrices

$$\begin{pmatrix} A_{11} & A_{12} & A_{13} \\ A_{21} & A_{22} & A_{23} \\ A_{31} & A_{32} & A_{33} \end{pmatrix} = \begin{pmatrix} L_{11} & & \\ L_{21} & L_{22} & \\ L_{31} & L_{32} & L_{33} \end{pmatrix} \begin{pmatrix} U_{11} & U_{12} & U_{13} \\ & U_{22} & U_{23} \\ & & U_{33} \end{pmatrix}$$

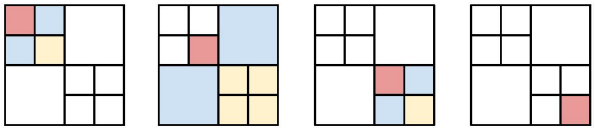


Algorithm 1 Blocked RL algorithm for the LU factorization.

Require: $A \in \mathbb{R}^{n \times n}$

1: **for** $k = 1, 2, \ldots, n_t$ **do**
2: $A_{kk} = L_{kk} U_{kk}$ — **LU**
3: **for** $j = k+1, k+2, \ldots, n_t$ **do**
4: $U_{kj} := L_{kk}^{-1} A_{kj}$
5: **end for** — **TRSM**
6: **for** $i = k+1, k+2, \ldots, n_t$ **do**
7: $L_{ik} := A_{ik} U_{kk}^{-1}$
8: **end for** — **TRSM**
9: **for** $i = k+1, k+2, \ldots, n_t$ **do**
10: **for** $j = k+1, k+2, \ldots, n_t$ **do**
11: $A_{ij} := A_{ij} - L_{ik} \cdot U_{kj}$
12: **end for**
13: **end for** — **GEMM**
14: **end for**

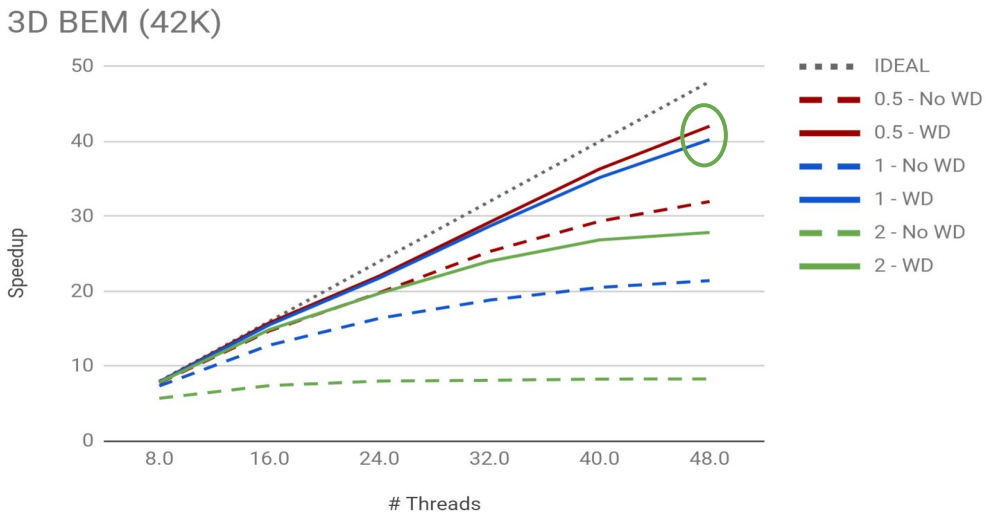
$\mathcal{H}$-Matrices [2]



[2] R. Kriemann. *H-LU factorization on many-core systems.* Computing and Visualization in Science (2013). DOI 10.1007/s00791-014-0226-7

Previous work

- Goal: test task-based parallel programming models and runtimes in the context of H-Algebra [3, 4].
  - Multicore architectures



Up to 42x speedup with 48 cores

[3] R. Carratalá-Sáez et al. *Task-parallel LU factorization of hierarchical matrices using OmpSs.* Proceedings of the 19th IEEE Workshop on Parallel and Distributed Scientific and Engineering Computing, PDSEC 2017, 2017. DOI 10.1109/IPDPSW.2017.124

[4] R. Carratalá-Sáez et al. *Exploiting Nested Task-Parallelism in the H-LU Factorization.* Journal of Computational Science (Elsevier), February 2018. DOI 10.1016/j.jocs.2019.02.004

Previous work

- ➢ Goal: test task-based parallel programming models and runtimes in the context of H-Algebra [3, 4].
  - Multicore architectures
- ➢ Difficulties:
  - Data location (not continuous in memory)
  - Recursive algorithm (nested dependencies)
- OmpSs-2 (*) & Nanos6: new features [5]
  - Nested data regions dependencies
  - Early release
  - Weak dependencies

(*) https://pm.bsc.es/ompss

[5] J.M. Pérez et. al. *Improving the integration of task nesting and dependencies in OpenMP* (2017),

Alternative approaches (runtime)

- Implementation of the Hmat-oss library on top of the STARPU [6]
  - Large number of dependencies
- Semi-automatic DAG generation [7]
  - Trim the DAG from unnecessary edges
- Fake dependencies [8]
  - Recursively introduce the missing dependencies

[6] B. Lizé, Résolution directe rapide pour les éléments finis de frontière en électromagnétisme et acoustique: H-Matrices. Parallélisme et applications industrielles. École doctorale Galilée, Université Sorbonne Paris Nord and Airbus: Ph.D. Dissertation, 2014.

[7] S. Boerm, S. Christophersen, and R. Kriemann, “Semiautomatic task graph construction for h-matrix arithmetic,” https://arxiv.org/abs/1911.07531, 2019.

[8] C. Augonnet, D. Goudin, M. Kuhn, X. Lacoste, R. Namyst, and P. Ramet, “A hierarchical fast direct solver for distributed memory machines with manycore nodes,” CEA/DAM ; Total E&P ; Université de Bordeaux, Research Report, Oct. 2019. [Online]. Available: https://hal-cea.archives-ouvertes.fr/cea-02304706

Alternative approaches (compression)

- Block Low-Rank (BLR) format [9] // Block Separable (BS) format [10]
  - Regular tiling, strong admissibility // weak admissibility
  - High concurrency degree, but not as efficient as H-Matrices

- Lattice H-Matrices [11]
  - Regular tiling + H-Matrices

[9] P. R. Amestoy, C. Ashcraft, O. Boiteau, A. Buttari, J. Y. LExcellent, and C. Weisbecker, “Improving multifrontal methods by means of block lowrank representations,” SIAM Journal on Scientific Computing, vol. 37, no. 3, pp. A1451–A1474, 2015.
[10] H. Cheng, Z. Gimbutas, P. G. Martinsson, and V. Rokhlin, “On the compression of low rank matrices,” SIAM Journal on Scientific Computing, vol. 26, no. 4, pp. 1389–1404, 2005.
[11] A. Ida, “Lattice h-matrices on distributed-memory systems,” in Proceedings of the International Parallel and Distributed Processing Symposium, Rio de Janeiro, Brasil, May 2018, pp. 389–398.

Overview

UNIVERSITAT JAUME I

- Brief introduction to H-Matrices & H-LU
- Previous work & alternative approaches
- **Our approach: H-Chameleon**
  - Results & Conclusions
  - Future Work

Our approach

- Similar to Lattice H-Matrices
- Existing libraries: Chameleon, Hmat-oss
- StarPU runtime to tackle multicore architectures

Our approach

- Similar to Lattice H-Matrices
- Existing libraries: Chameleon, Hmat-oss
- StarPU runtime to tackle multicore architectures
- Very few implementations needed
  - CHAMELEON: slightly modify descriptor, create new tile struct

```
struct chameleon_desc_s {
    ...
    blktile_fct_t  get_blktile;
    CHAM_tile_t   *tiles;
    ...
}
```

```
typedef struct chameleon_tile_s {
    int8_t format;
    int    m, n, ld;
    void  *mat;
} CHAM_tile_t;
```

Our approach

- Similar to Lattice H-Matrices
- Existing libraries: Chameleon, Hmat-oss
- StarPU runtime to tackle multicore architectures
- Very few implementations needed
  - CHAMELEON: slightly modify descriptor, create new tile struct
  - Create our own descriptor

```
struct HCHAM_desc_s {
    CHAM_desc_t              *super;
    hmat_cluster_tree_t     **clusters;
    hmat_admissibility_t     *admissibilityCondition;
    hmat_interface_t         *hi;
    bool                      multi_cluster;
    hmat_matrix_t            *hmatrix;
    int                      *perm;
};
```

Our approach

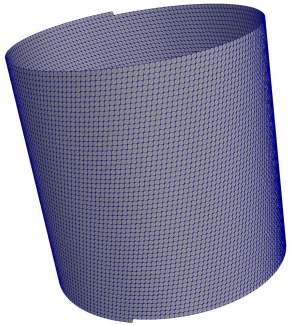
- Similar to Lattice H-Matrices
- Existing libraries: Chameleon, Hmat-oss
- StarPU runtime to tackle multicore architectures
- Very few implementations needed
  - CHAMELEON: slightly modify descriptor, create new tile struct
  - Create our own descriptor
  - HMAT-OSS: some C++ to C wrappers, new clustering algorithm

Overview

UNIVERSITAT JAUME I

- Brief introduction to H-Matrices & H-LU
- Previous work & alternative approaches
- Our approach: H-Chameleon
- **Results & Conclusions**
  - Future Work

Test case: test_FEMBEM (*)



Similar to real industrial applications

Cloud of points on the surface:

$$(x_i)_{1 \leq i \leq n}$$

Interaction kernel between 2 points x_i and x_j separated by a distance

$$d = |x_i - x_j|$$

as:

$K(d) = \exp(ikd)/d$ in the complex case (k ~ wave number)

$K(d) = 1/d$ in the real case

(*) Available: https://gitlab.inria.fr/hiepacs/papers/hmat/hmat-comparison/

Test platform and configurations

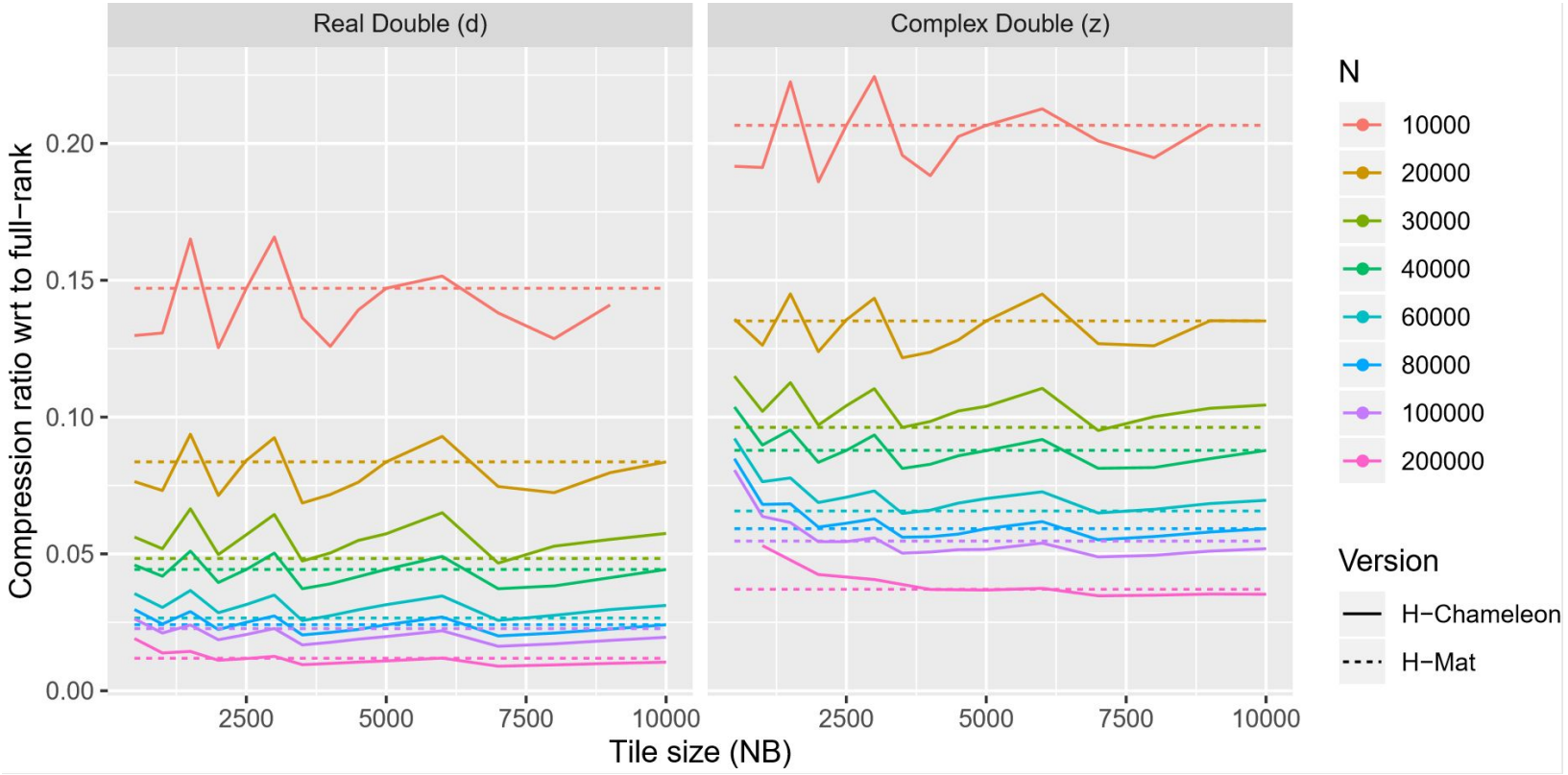
PlaFRIM - bora cluster, each node equipped with 1 Intel Xeon Skylake Gold 6240, 18 cores available running at 2.60 GHz, and 192 GB of memory.

ntel MKL 2010 is used for BLAS kernels.

Real double (d) and double complex (z) precision # Threads 1 to 36 (35)
Different dimensions (best Tile Size chosen):

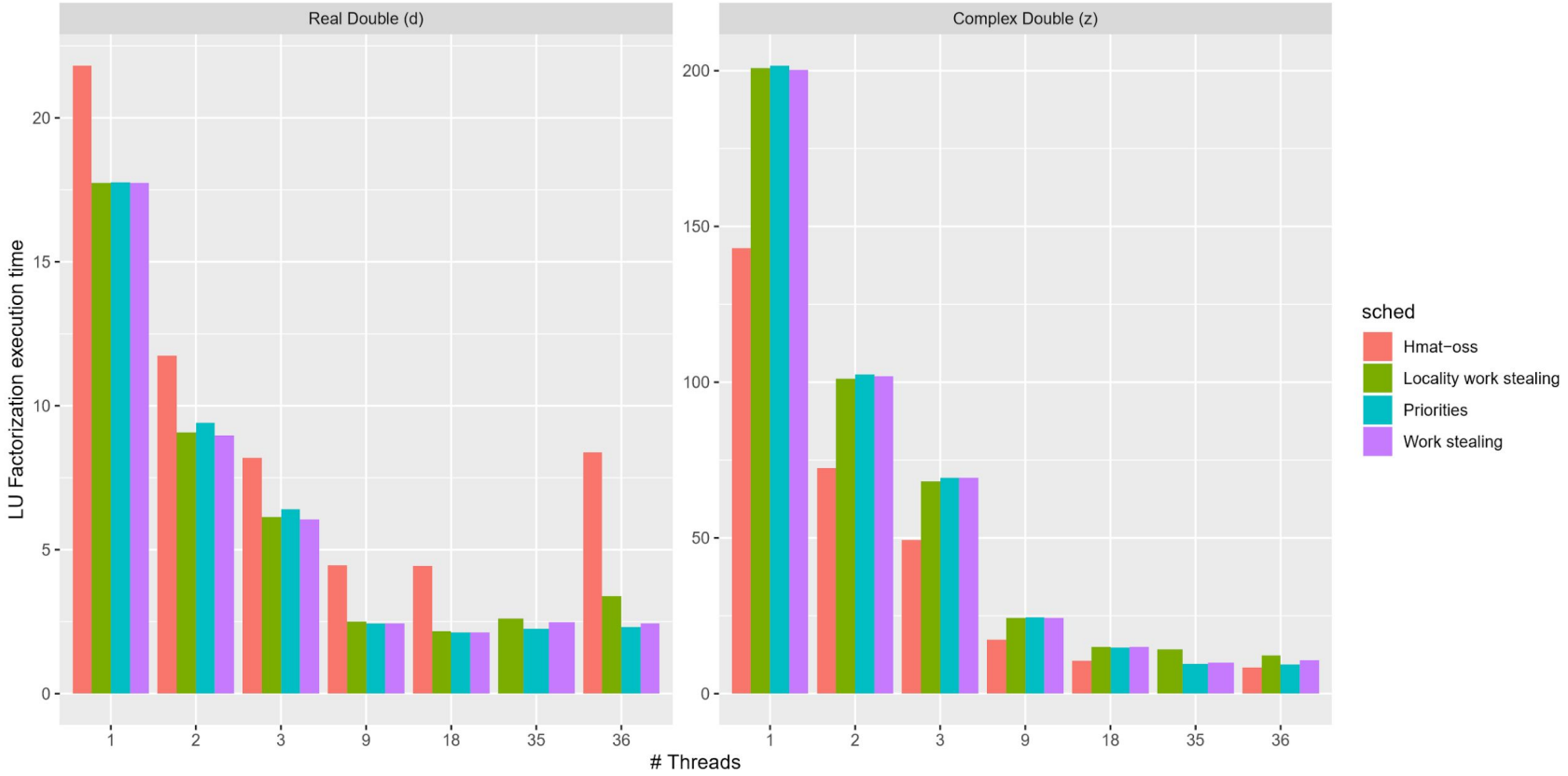
- 10K d → TS 250 10K z → TS 500
- 20K d → TS 500 20K z → TS 500
- 40K d → TS 1000 40K z → TS 1000
- 60K d → TS 1000 60K z → TS 1000
- 80K d → TS 1000 80K z → TS 2000
- 100K d → TS 1000 100K z → TS 2000
- 200K d → TS 2000 200K z → TS 4000

Results - Compression ratio

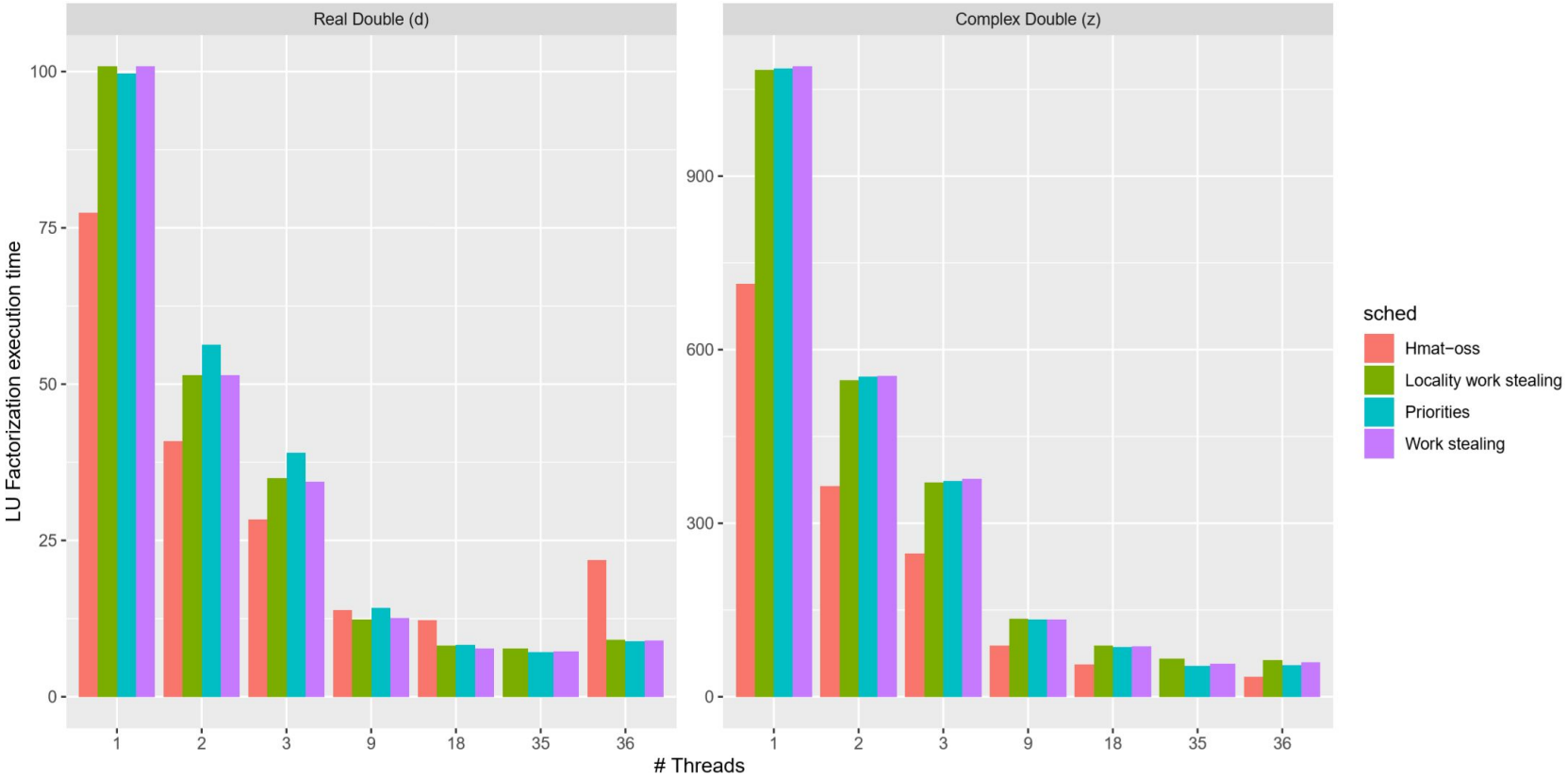


Results - Precision (forward error)

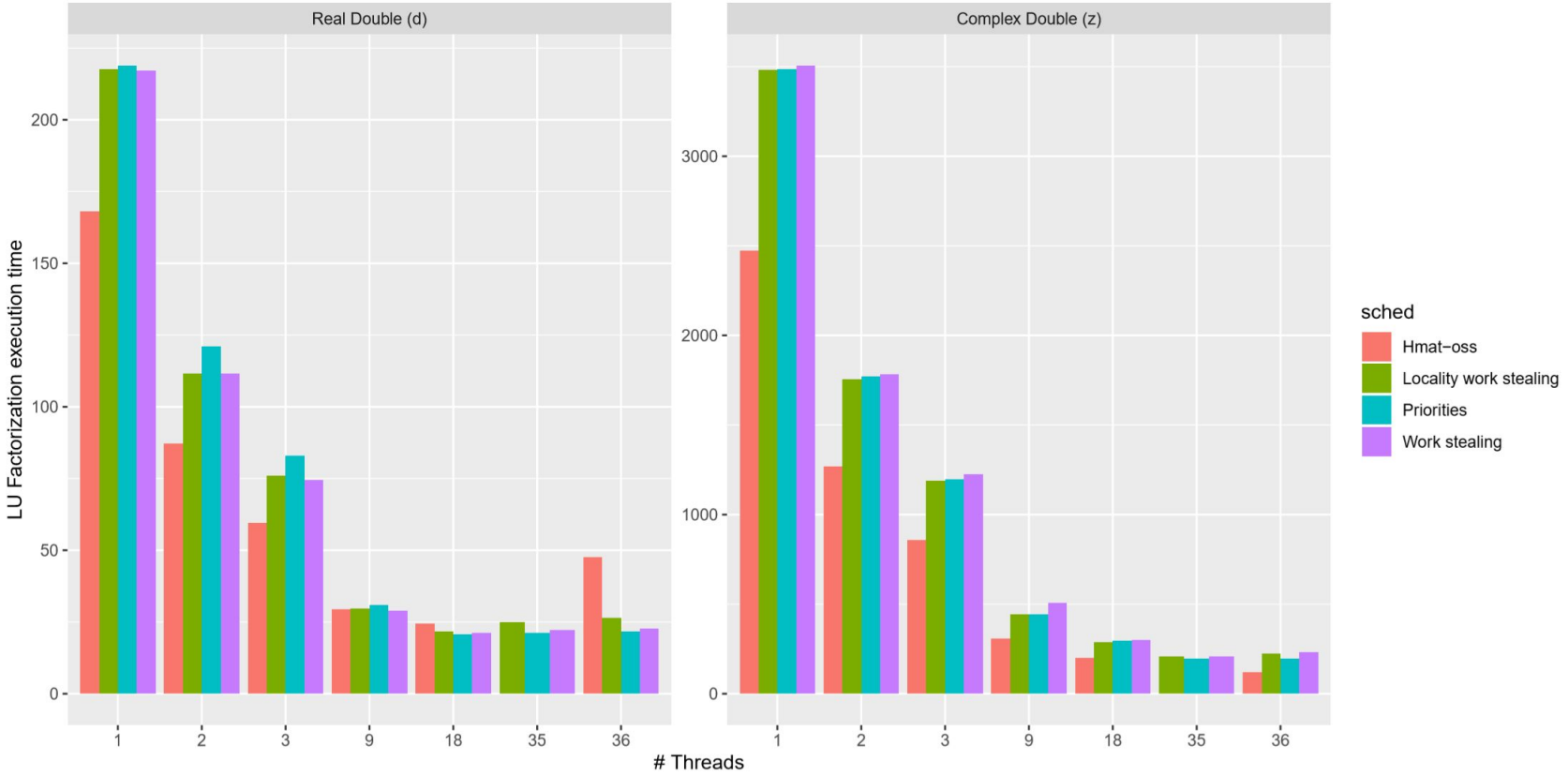
Results - Execution time (dimension 40K)



Results - Execution time (dimension 100K)



Results - Execution time (dimension 200K)



Conclusions

- Good compression ratio wrt Hmat-oss
- Good precision (forward error) wrt Hmat-oss
- Good parallel performance
  - D precision: generally better than Hmat-oss
  - Z precision:
    - Better/same in small matrices
    - Slightly worse in big matrices

Conclusions

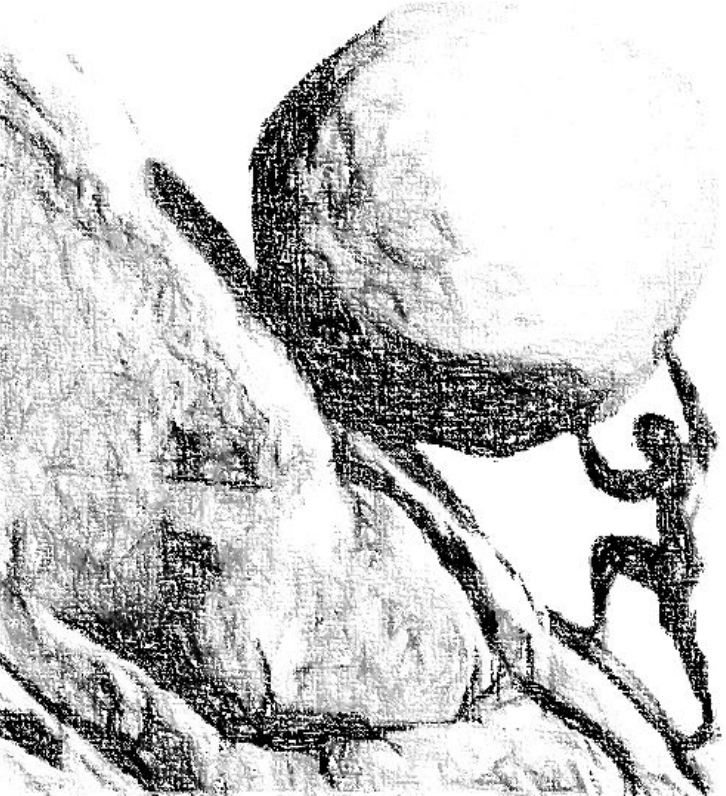
- Good compression ratio wrt Hmat-oss
- Good precision (forward error) wrt Hmat-oss
- Good parallel performance
  - D precision: generally better than Hmat-oss
  - Z precision:
    - Better/same in small matrices
    - Slightly worse in big matrices
- H-Tiling matrices behave well on multicore architectures!

Overview

UNIVERSITAT JAUME I

- Brief introduction to H-Matrices & H-LU
- Previous work & alternative approaches
- Our approach: H-Chameleon
- Results, Conclusions and References
- **Future Work**

Future work



- Performance comparisons
- More test cases (different problems)
- Distributed memory
- … (end of the PhD!)
  - => PhD defended in last February

Exploiting Generic Tiled Algorithms Toward Scalable H-Matrices Factorizations on Top of Runtime Systems

6th July, 2021

Rocío Carratalá-Sáez	rcarrata@uji.es
Mathieu Faverge	mathieu.faverge@inria.fr
Grégoire Pichon	gregoire.pichon@inria.fr
Guillaume Sylvand	guillaume.sylvand@inria.fr
Enrique S. Quintana-Ortí	quintana@disca.upv.es