Utiliser Chameleon pour factoriser des matrices hiérarchiques

February 5, 2020 - Kickoff Solharis

Mathieu Faverge

Introduction

1. Descripteurs récursifs
2. Algorithmes récursifs
3. (Lattice) $\mathcal{H}$-Matrix factorisation
4. Conclusion

1

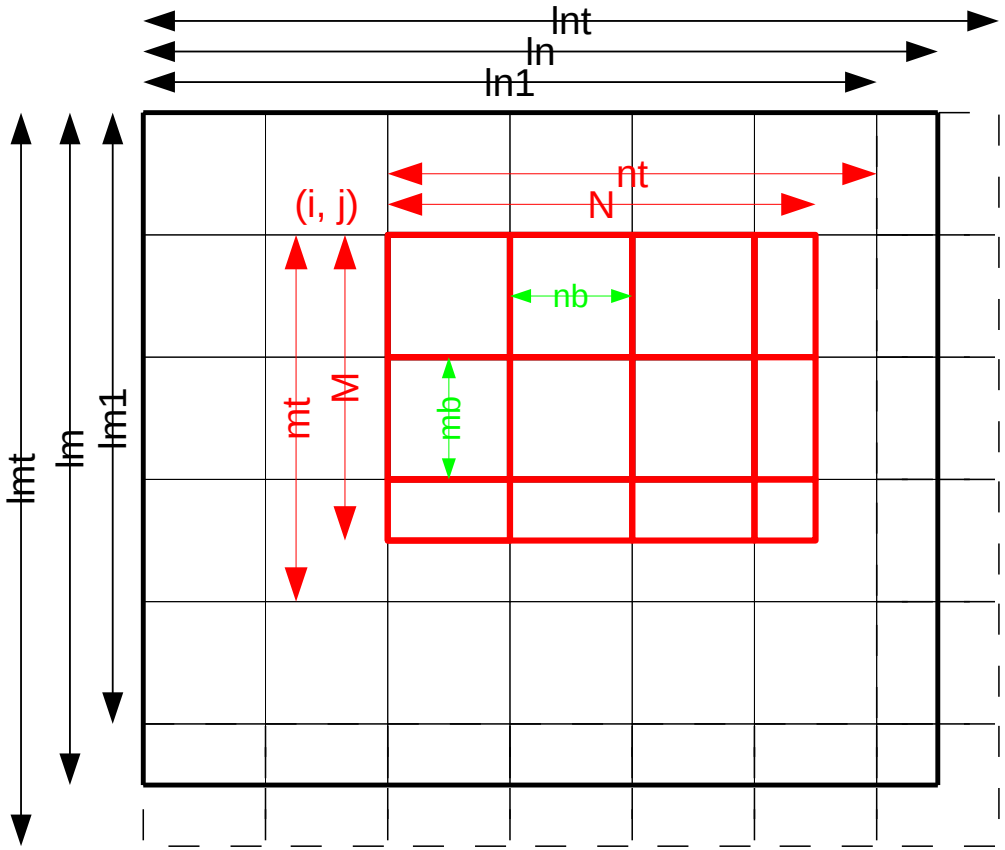
Descripteurs récursifs

Motivations

Why changing the data structures?

- Simplify the algorithms
- Use Chameleon as a sandbox for the recursive algorithms
  Gwenolé Lucas
- Handle fll-rank as well as low-rank matrices
  Rocio Carratalá-Sáez

Version originale

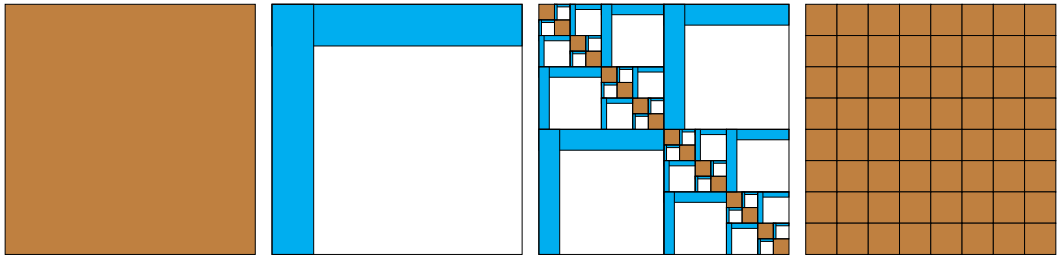


- A long complex structure to describe the tiles distribution
- Pointer to tiles are computed on the fly when submiting the tasks
- An array of handlers keeps track of the tiles for the runtimes

Nouvelle version

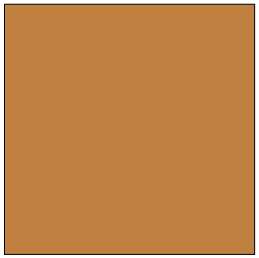
Nouveaux descripteurs

- Une collection de données (tuile).
- Une tuile peut-être n'importe quoi: une matrice dense, un bloc low-rank, une $\mathcal{H}$-Matrix, un descripteur ...

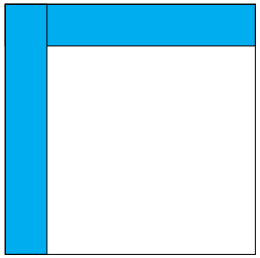


```
typedef struct chameleon_tile_s {
    int8_t format;
    int    m, n, ld;
    void  *mat;
} CHAM_tile_t;
```

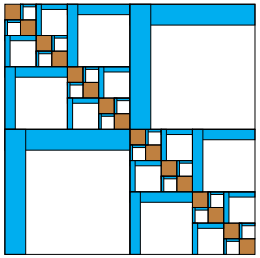
Nouvelle version



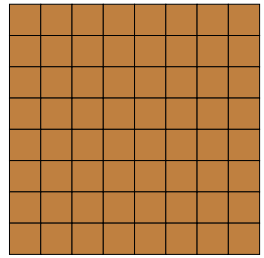
- Classic Chameleon
- OpenMP, PaRSEC, Quark, StarPU



- Block low-rank
- Kernels from H-Mat OSS
- or Kernels from PaStiX?
- Quark, StarPU



- Lattice $\mathcal{H}$-Matrix
- Kernels from H-Mat OSS
- Similar to Lattice $\mathcal{H}$-Matrix from Ida and Ichitaro
- Quark, StarPU



- Recursive description
- Kernels from Chameleon itself
- StarPU

2

Algorithmes récursifs

Intérêts

- Être capable de développer des algorithmes récursifs avec le modèle STF

Intérêts

- Être capable de développer des algorithmes récursifs avec le modèle STF
- Réduction du nombre de tâches dans le système
- Soumission parallèle des tâches tout en conservant le modèle initial

Intérêts

- Être capable de développer des algorithmes récursifs avec le modèle STF
- Réduction du nombre de tâches dans le système
- Soumission parallèle des tâches tout en conservant le modèle initial
- Adaptation automatique de la granularité pour les noyaux CPU/GPU

Intérêts

- Être capable de développer des algorithmes récursifs avec le modèle STF
- Réduction du nombre de tâches dans le système
- Soumission parallèle des tâches tout en conservant le modèle initial
- Adaptation automatique de la granularité pour les noyaux CPU/GPU
- Full H-Matrix solver ?

Descripteurs récursifs

```
int _desc_recursive( CHAM_desc_t **descptr, void *mat, cham_flttype_t dtyp,
                     int level, int *mb, int *nb, int lm, int ln, int m, int n,
                     blkaddr_fct_t   get_blkaddr, blkldd_fct_t get_blkldd,
                     blkrankof_fct_t get_rankof )
{
    /* Creation du descripteur local */
    chameleon_desc_init( desc, mat, dtyp, mb[0], nb[0], mb[0] * nb[0],
                         lm, ln, 0, 0, m, n, p, q,
                         get_blkaddr, get_blkldd, get_rankof );

    /* Creation des sous-descripteurs */
    if ( level > 1 ) {
        for ( n=0; n<desc->nt; n++ ) {
            for ( m=0; m<desc->mt; m++ ) {

                tile = desc->get_blktile( desc, m, n );
                tempmm = m == desc->mt-1 ? desc->m - m * desc->mb : desc->mb;
                tempnn = n == desc->nt-1 ? desc->n - n * desc->nb : desc->nb;
                _chameleon_desc_create_recursive( &tiledesc, tile->mat,
                                                  desc->dtyp, level-1, mb+1, nb+1,
                                                  tile->ld, tempnn, tempmm, tempnn,
                                                  chameleon_getaddr_cm,
                                                  chameleon_getblkldd_cm, NULL );

                tile->format = CHAMELEON_TILE_DESC;
                tile->mat = tiledesc;
            }
        }
    }
}
```

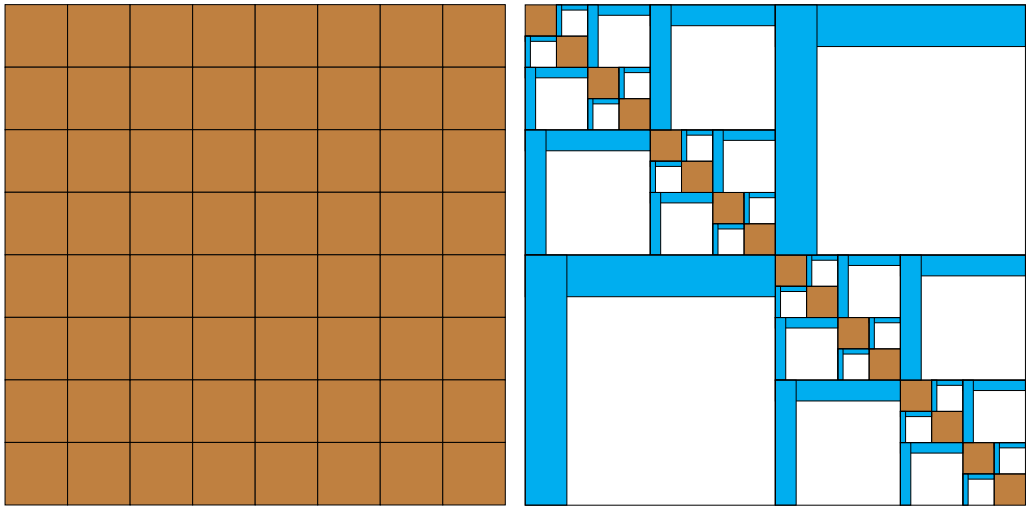
Generated DAG

TODO

3

(Lattice) $\mathcal{H}$-Matrix factorisation

Comment passer de Chamelon à $\mathcal{H}$-Chameleon



Solutions existantes

HACApK

- *Distributed-memory latticeH-matrix factorization*, I. Yamazaki, A. Ida, R. Yokota and J. Dongarra
- OpenMPI + MPI + tile algorithm
- Lattice $\mathcal{H}$-Matricesavec des tailles variables
- Supporte aussi BLR et $\mathcal{H}$-Matrix

CEA

- *A hierarchical fast direct solver for distributed memory machines with manycore nodes*, C. Augonnet, D. Goudin, M. Kuhn, X. Lacoste, R. Namyst, and P. Ramet
- Ordonnanceur maison qui explore la structure hiérachique
- Not open source :(

H-Mat (Airbus)

- *Résolution directe rapide pour les éléments finis de frontière en électromagnétisme et acoustique : $\mathcal{H}$-Matrices. Parallélisme et applications industrielles*, B. Lizé
- StarPU avec enumeration des dépendances

Solutions proposée

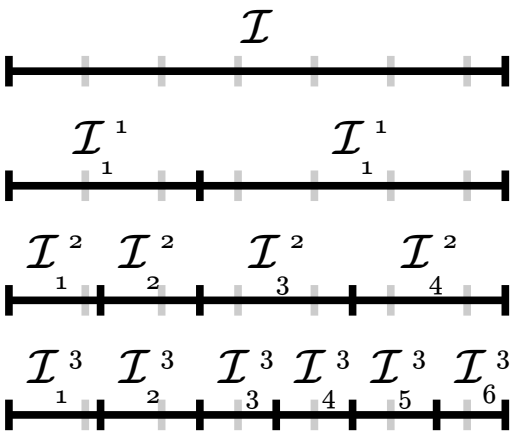
Chameleon

- Large spectres d'algorithmes à base de tâches
- Supporte plusieurs runtimes
- Inconvénient: Travaille avec des tuiles de tailles fixes

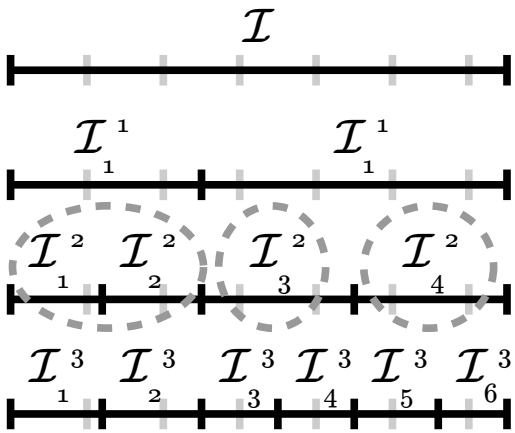
H-Mat OSS

- Version OpenSource de HMat
- Version séquentielle → Parafait our les noyaux de Chameleon
- Inconvénient: Comment s'adapter au découpage régulier de Chameleon?

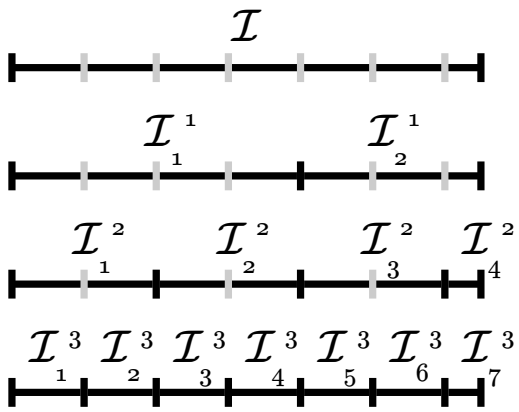
Partitionnement



Format $\mathcal{H}$-Matrix

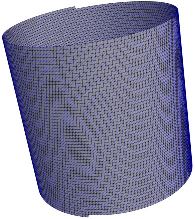


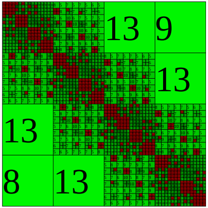
Format Lattice $\mathcal{H}$-Matrix

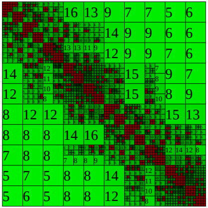


Format $\mathcal{H}$-Chameleon

Test case study

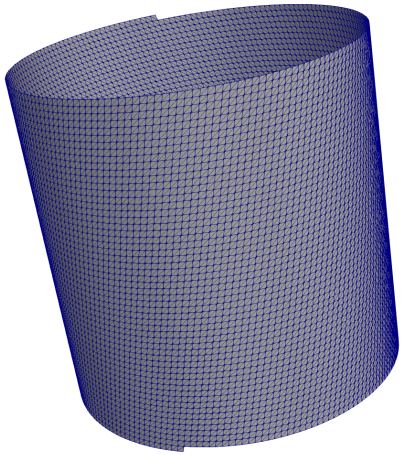


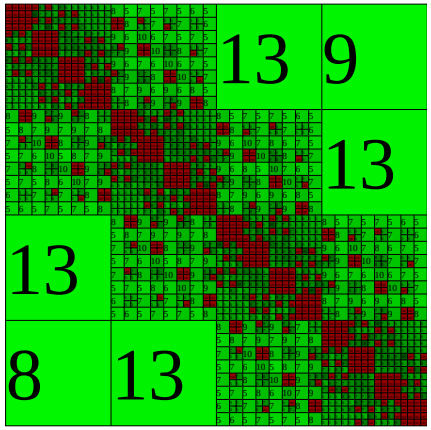


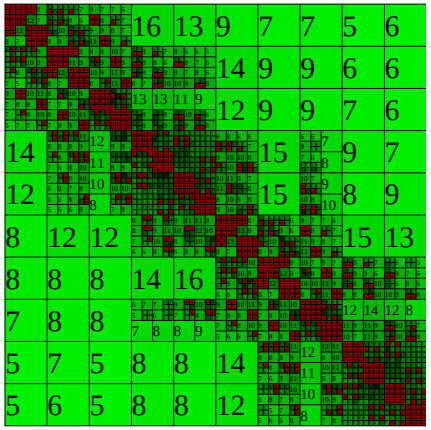


- Tests are done with BEM matrices
- Cylinder with radius of 2 and length of 4 (or 16)
- Example here made with:
  $N = 10000, nb = 1000, l = 16$

Test case study

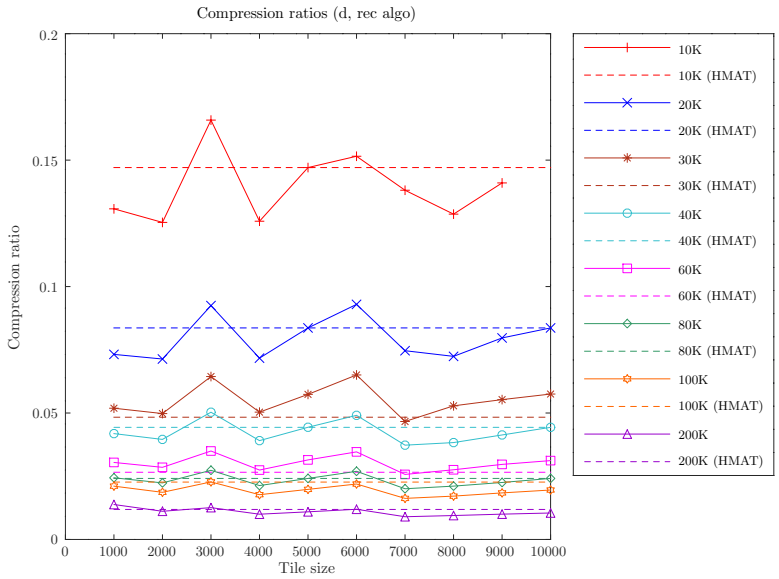


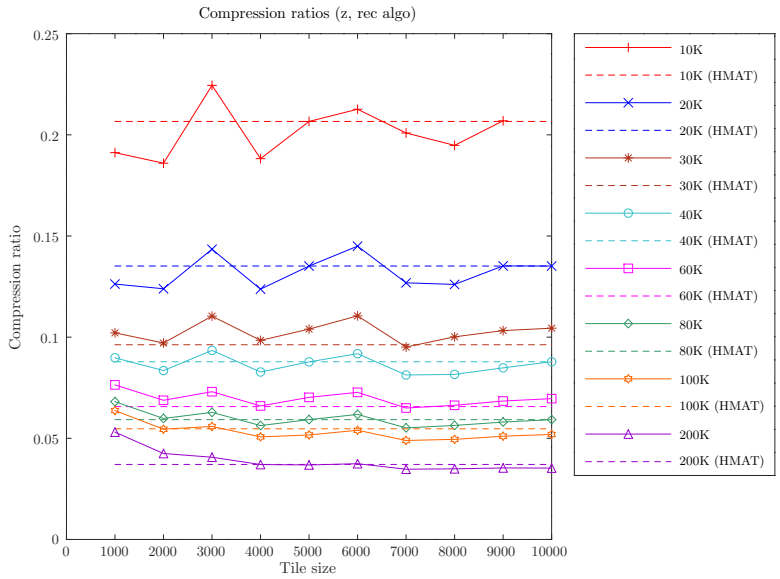




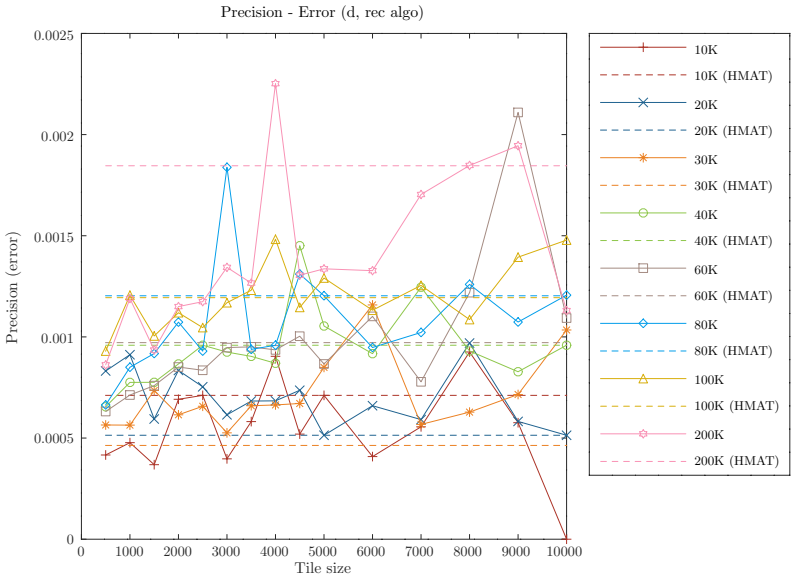
- Tests are done with BEM matrices
- Cylinder with radius of 2 and length of 4 (or 16)
- Example here made with:
  $N = 10000, nb = 1000, l = 16$
- Does it affect the compression ?
- Does it affect the accuracy of the solver ?

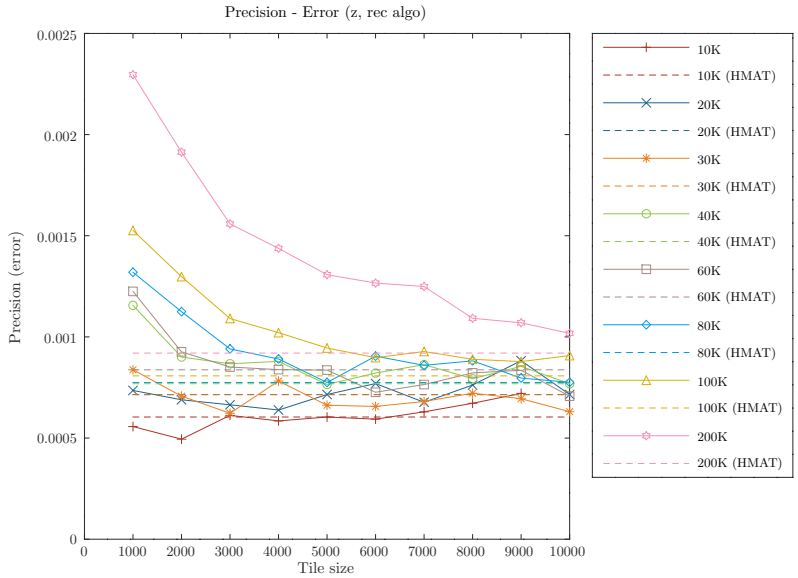
Compression ratio



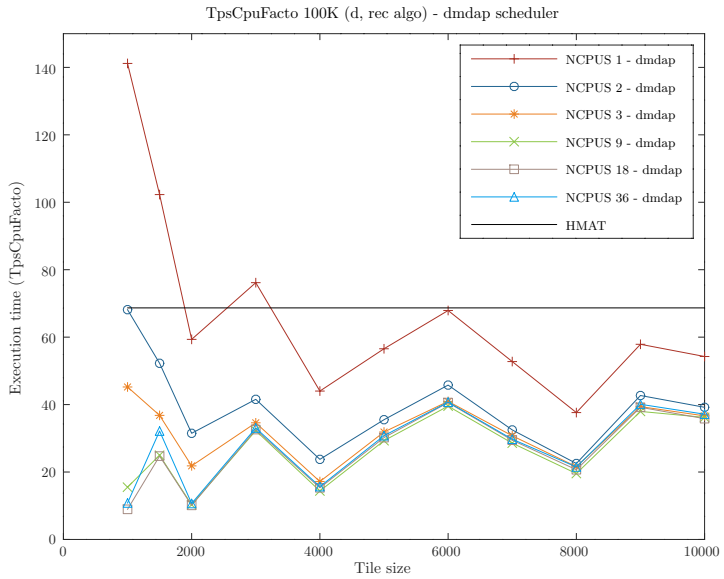


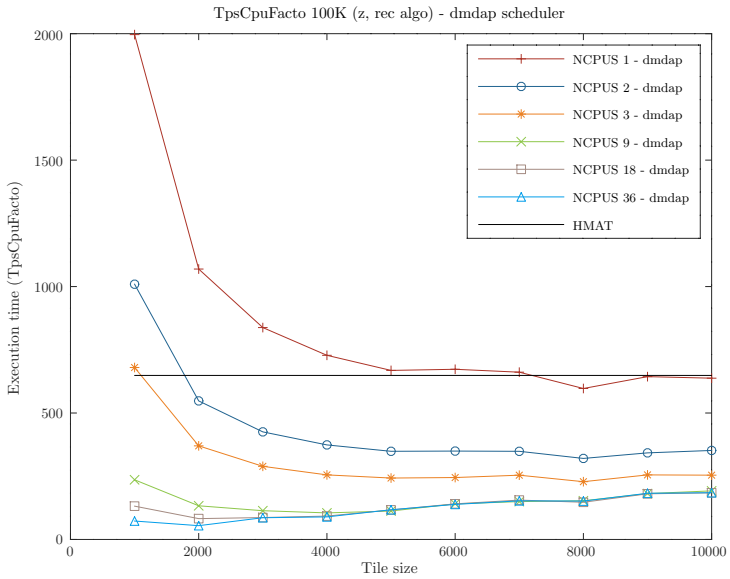
Forward error



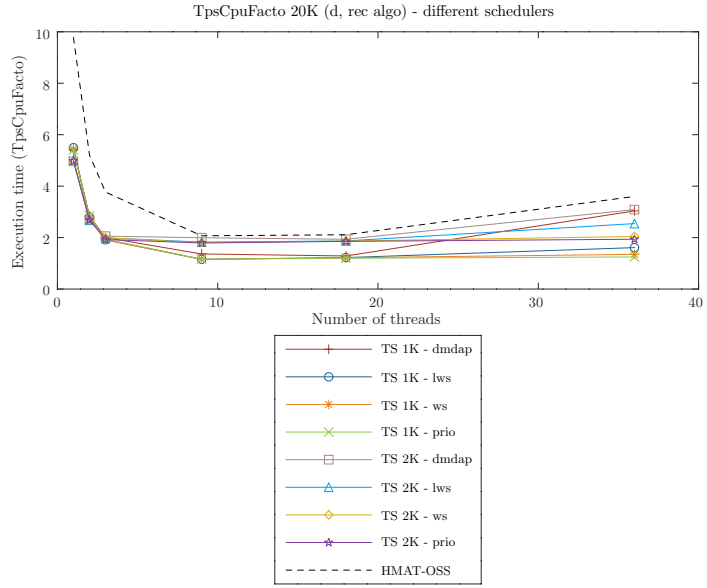


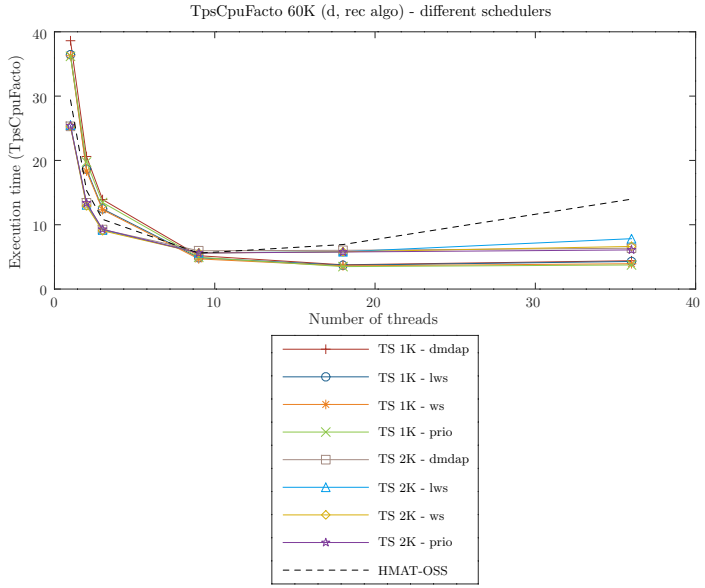
Performance on 100K matrices (Bora)

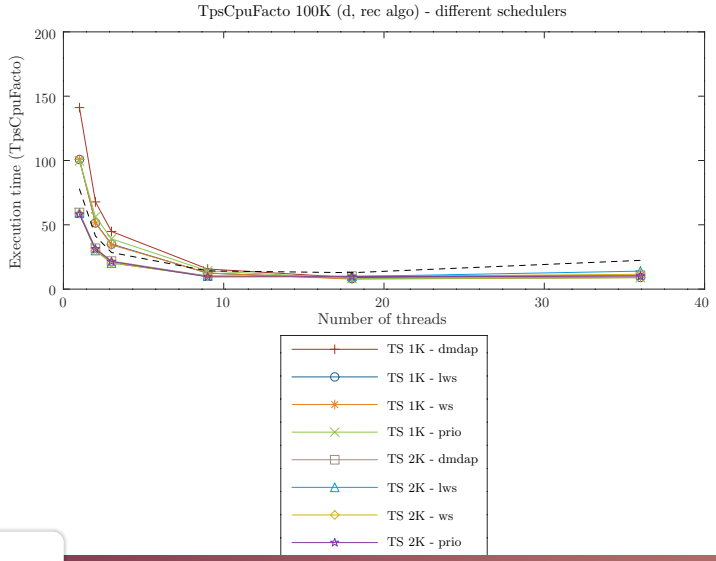


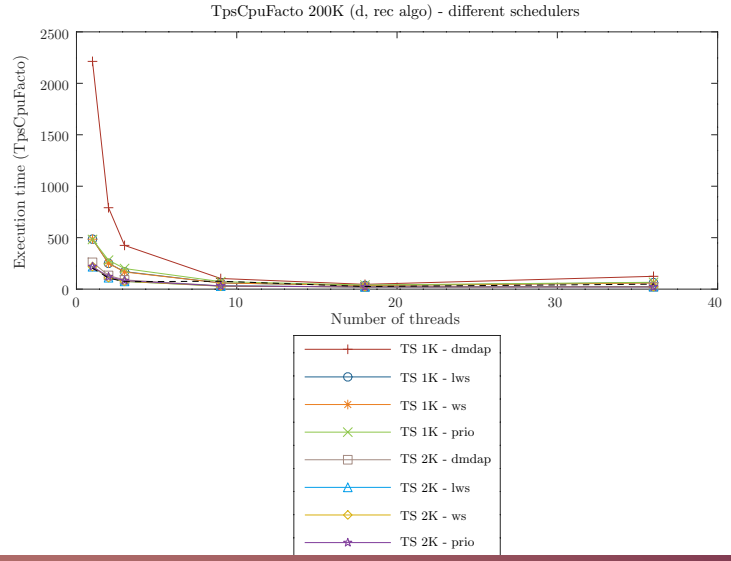


Performance with respect to HMat (Bora - D)

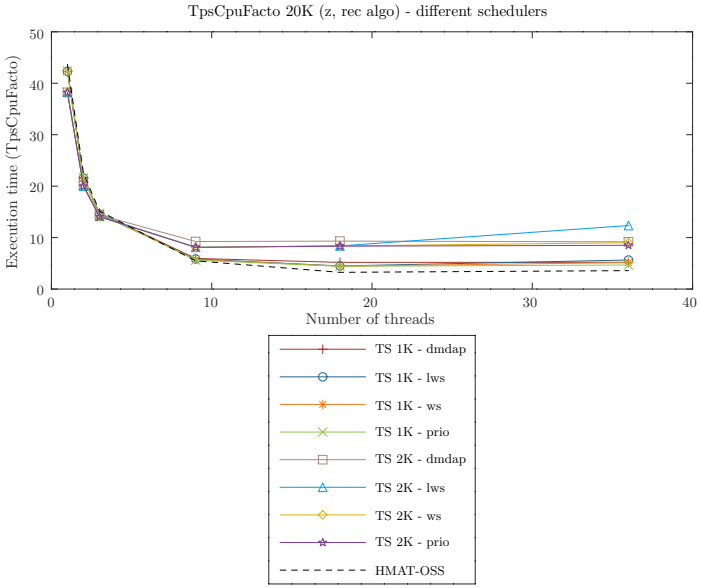


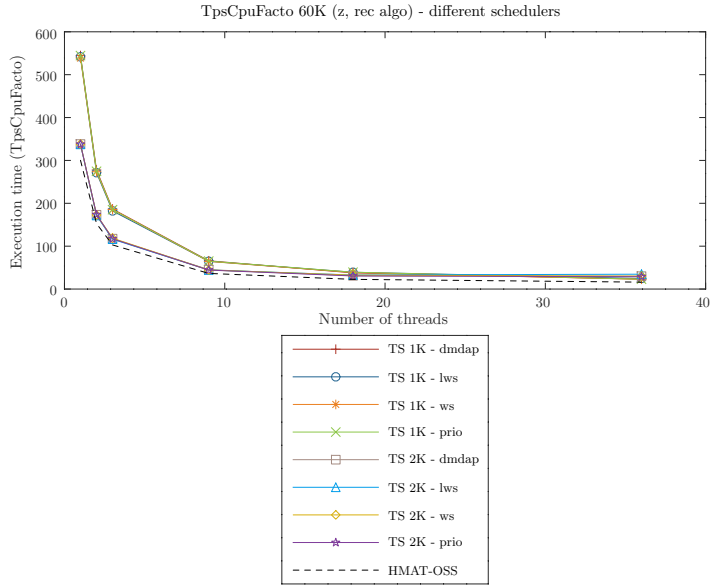


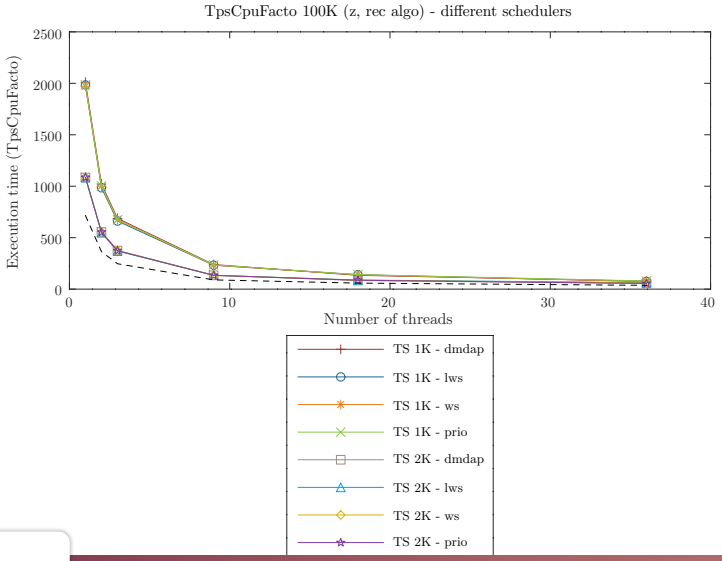


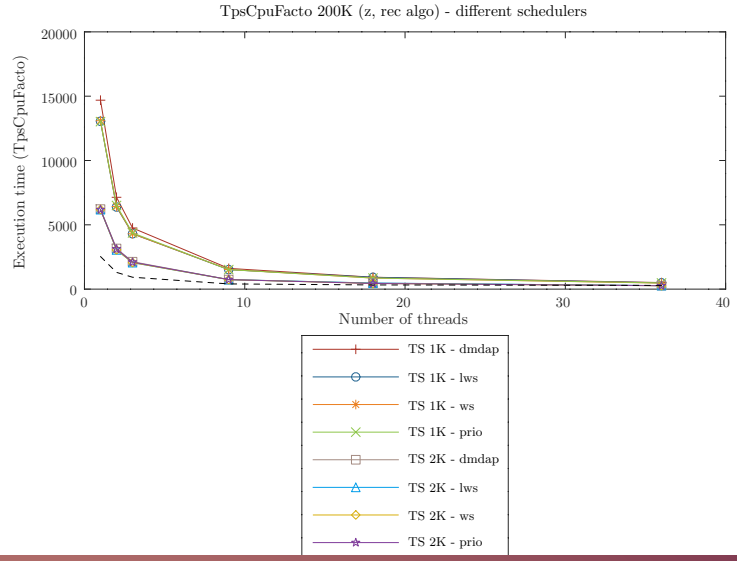


Performance with respect to HMat (Bora - Z)









4

Conclusion

Conclusion

New descriptors

- New capabilities for the Chameleon library
- Recursive tile algorithms (Gwenolé)
- Adaptive granularity
- Test Compression in all algorithms (QR?)
- Available in the release 1.0 (Coming soon !!!)

$\mathcal{H}$-Chameleon

- Compression ration comparable to classic $\mathcal{H}$-Matrix
- Simplest parallelization
- Need to work on the strong scaling capabilities
- Still irregular load with large critical path wrt to the full computation
- Test recursive algorihtms on the diagonal blocks ?

Thanks for your attention!

http://gitlab.inria.fr/solverstack/chameleon