Low rank matrix computing : performance, algorithms and tools

GT TOPAL 11/23

22 juin 2023

Abel Calluaud [1,2] Mathieu Faverge [2] Pierre Ramet [2]

[1] CEA
[2] Univ. Bordeaux, CNRS, Bordeaux INP, INRIA, LaBRI, UMR 5800

Table of content

Problem statement

Objective

Design scalable high-performant portable direct solver.

... but dense direct solver are costly.

> $O(n^3)$ operations
> $O(n^2)$ memory

→ **Parallel computing**

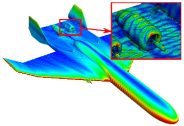
→ **Low rank compression**

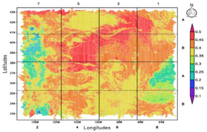
Target applications

Electromagnetic scaterring

Climate modeling

Earthquake simulation





Target architectures

Modern supercomputers featuring multicore/manycore CPUs and GPUs.

State-of-the-art dense direct solver

SLATE : fork-join

DPLASMA : fine deps

CHAMELEON : fine deps GPU

Panel vs tile algorithms

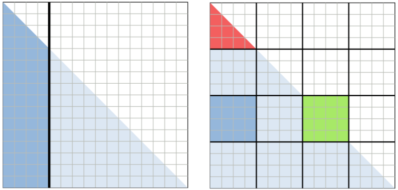


Figure 1: Panel vs tile algorithms

Tile algorithm and task paradigm allow:

> unleash **fine** task parallelism

> use **highly-optimized** linear algebra libraries on **local** tile data.

> leverage **runtime optimizations**

State-of-the-art low rank solver

HICMA : BLR fine deps

LORAPO : BLR fine deps

H2lib : $\mathcal{H}$ sequential

HMAT-OSS : $\mathcal{H}$ sequential

hlib : $\mathcal{H}$ parallel proprietary

STRUMPACK : $\mathcal{H}$SS open-source fork-join distributed

HATRIX-DTD : $\mathcal{H}$SS fine deps distributed

Arlène : Tile-$\mathcal{H}$ proprietary distributed

$\mathcal{H}$-CHAMELEON : open-source distributed Tile-$\mathcal{H}$ coarse deps

$\rightarrow$ **Next $\mathcal{H}$-Chameleon** ? open-source distributed Tile-$\mathcal{H}$ fine deps

Design a scalable direct solver

Objective

Design a scalable direct solver for dense linear algebra with low rank compression.

Building blocks

> Scalable asynchronous tasking engine
> Fine-grain computation decomposition
> Tile Algorithm
> Low rank kernels

Design a scalable direct solver

Objective

Design a scalable direct solver for dense linear algebra with low rank compression.

Building blocks

> Scalable asynchronous tasking engine : StarPU
> Fine-grain computation decomposition : StarPU's hierarchical tasks
> Tile Algorithm : CHAMELEON
> Low Rank kernels : PasTiX's kernels

Roadmap

Objective

Design a scalable direct solver for dense linear algebra with low rank compression.

Roadmap

- ☑ Design Low Rank Algebra kernels : extract kernel from pastix and expose them as a BLAS-like library.
- ☑ Leverage low rank algebra kernels in PaStiX sparse direct solver.
- ☐ Add support to Chameleon for RAPACK block tiles.
- ☐ Add suport to Chameleon for hierarchical tiles.
- ☐ Leverage fine grain dependencies with StarPU's hierarchical tasks.

Low rank approximation

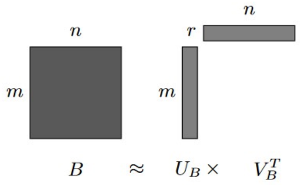


Figure 2: Low rank approximation

- Representation of a matrix B with a lower rank matrix.
- Storage as a outer product $U_B \times V_B^T$.
- Decomposition can be obtained via SVD, QR variants or Adaptive Cross Approximation (ACA).

$\Rightarrow$ **Reduce storage and computation cost**

RAPACK : Low **R**ank **A**lgebra **Pack**age

RAPACK: a low rank linear algebra package.

Objective

Expose low rank linear algebra routines.

Strategy

> Leverage existing linear algebra kernels from BLAS / LAPACK libraries, and PAStiX.
> Expose sequential low rank algebra kernels with a C BLAS-like API.
> A basic interface and an advanced interface allowing to configure compression algorithm, synchronization hooks and memory allocation.

Case study : Low Rank Matrix Multiplication (LRMM)

$$C \leftarrow C + A \times B$$

where A, B, and C can either be dense or low rank matrices.

Difficulties

- 2^3 cases to handle
- Acquiring the data on C may be postponed until the end of the $A \times B$ computation.

Design choices

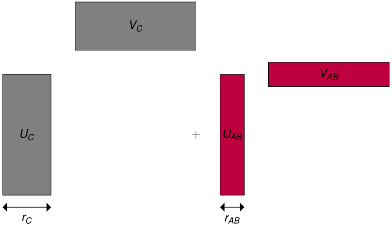
- Provide library hooks allowing users to attach synchronization routines when acquiring and releasing data.
- This is part of the advanced interface available via rapack context structure.

Recompresssion kernel

A low rank matrix $U_C V_C^t$ receive a low rank contribution $U_{AB} V_{AB}^t$

Recompression algorithm

$$U_C V_C^t + U_{AB} V_{AB}^t = ([U_C, U_{AB}]) \times ([V_C, V_{AB}])^t$$



Recompression kernels available in RAPACK :
SVD, QRCP, RQRCP, TQRCP, RQRRT

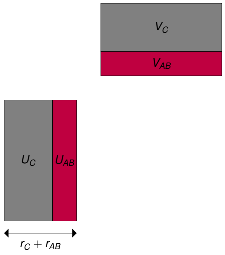
Recompresssion kernel

A low rank matrix $U_C V_C^t$ receive a low rank contribution $U_{AB} V_{AB}^t$

Recompression algorithm

$$U_C V_C^t + U_{AB} V_{AB}^t = ([U_C, U_{AB}]) \times ([V_C, V_{AB}])^t$$

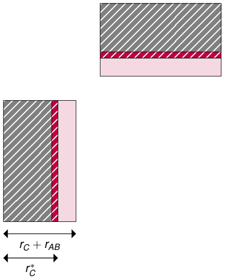
Recompression kernels available in RAPACK :
SVD, QRCP, RQRCP, TQRCP, RQRRT



Recompresssion kernel

A low rank matrix $U_C V_C^t$ receive a low rank contribution $U_{AB} V_{AB}^t$

Recompression algorithm



$$U_C V_C^t + U_{AB} V_{AB}^t = ([U_C, U_{AB}]) \times ([V_C, V_{AB}])^t$$

Recompression kernels available in RAPACK :
SVD, QRCP, RQRCP, TQRCP, RQRRT

Conclusion

- State-of-the-art of low rank solver design
- RAPACK : a Low Rank algebra library
- Use in the PaStiX sparse direct solver

Future work :

- Use in the Chameleon dense direct solver with low rank tiles.