Combined runtime system and compiler techniques for direct hierarchical solver

Abel Calluaud

Mathieu Faverge
Pierre Ramet

David Lugato

Table of contents

Industrial problem

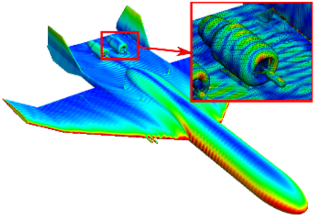
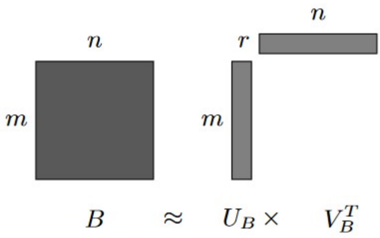


Figure: Electric currents at the surface of an UAV at 2.5 GHz (AGK+19)

- linear system $Ax = b$ arising from Maxwell equations.
- Industrial cases can feature millions of unknowns and thousands of right-hand sides.
- Direct dense factorization has $O(n^3)$ complexity and $\mathcal{O}(n^2)$ storage cost.

$\rightarrow$ **Compression techniques for addressing theses cases**

Low-rank approximations



- A low-rank approximation consist representing a matrix $A_{m \times n}$ by a lower-rank one.
- Can be stored in outer-product form $U_{n,r} \times V^t_{r,m}$.
- low-rank approximation can be calculated with SVD or QR variants or ACA.

$\rightarrow$ **memory and compute cost of operation can be reduced**

Block Low Rank format

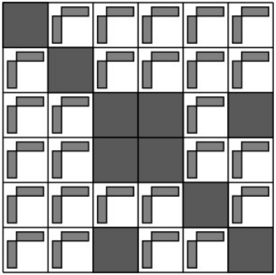


Figure: Block Low Rank Format

- flat partition

Hierarchical format

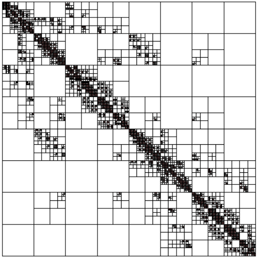


Figure: Hierarchical Format

- hierarchical partition

Implementation of a H-matrix solver

The current high-performance implementation is mainly based on two building blocks:

- High performance low-rank kernels leveraging BLAS routines.
- *libtask*, a dedicated task-based runtime system for communication and distributed memory parallelization.

Libtask

A dedicated task-based runtime system based on STF model.

- Take avantage of hierarchical dependencies to unleash the maximum parallelism.

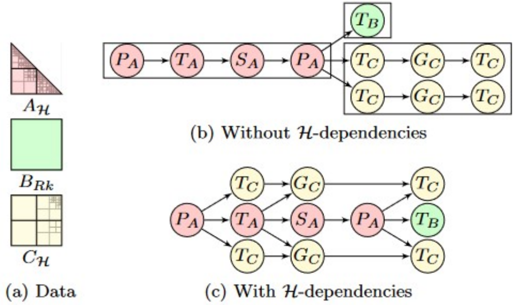


Figure: Panel update where AH and CH are H-matrices and BRk is a Rk matrix (H-POTRF(AH); H-TRSM(AH, BRk); H-TRSM(AH, CH);)

Strong scalability

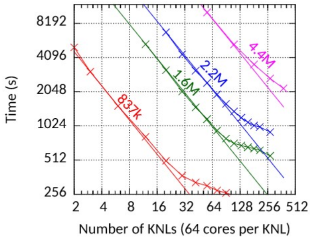


Figure: Strong scalability for sphere geometries up to 4.4 million unknowns over KNLs (TERA1000-2)

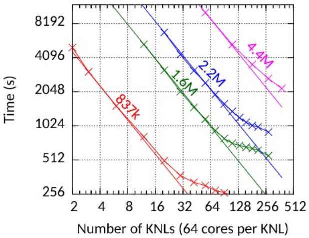


Figure: Strong scalability for sphere geometries up to 1.6 million unknowns over Haswell processors (TERA1000-1)

Challenges for an efficient implementation

Several HPC challenges arise from the hierarchical nature of the data structure :

- Load balancing
- Data locality
- Task overhead

Difficult to leverage GPU architectures due to the irregular and sparse data structure.

Compromise on task-granularity

task granularity	load balancing	task overhead
fine-grains task	✓	X
good-grain task ?	✓	✓
coarse-grain tasks	X	✓

Task

overhead includes task creation and management, scheduling, communications, synchronizations.

→ **A key issue for porting the H-matrix solver on the GPUs ? A room of improvement for CPUs ?**

Optimizing the task graph

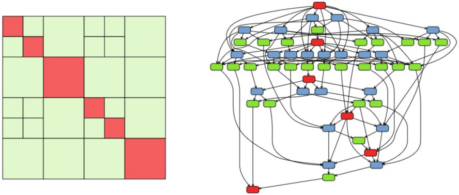


Figure: $\mathcal{H}$-matrix and corresponding DAG for $\mathcal{H}$-matrix factorization

Problems:

- The task graph is huge and does not fit in memory
- The matrix ranks evolve : it is not possible to build the task graph before.

Combined runtime and compiler techniques

- **Inspector-Executor.** Inspection of the data structure in order to collect information for improving execution performances.
- **Multiversioning.** Generate multiple version of a kernel at compile-time. The decision to chose the actual version to run is done at run-time.
- **Specialization.** Generate a specialized kernel version for a given parameter.
- **Autotuning.** Exploring a search space of kernels to find the better performing one.

References I

[AGK+19] Cédric Augonnet, David Goudin, Matthieu Kuhn, Xavier Lacoste, Raymond Namyst, and Pierre Ramet, *A hierarchical fast direct solver for distributed memory machines with manycore nodes*, Research report, CEA/DAM ; Total E&P ; Université de Bordeaux, October 2019.