New parallel features in the sparse solver PaStiX

M. Faverge, **A. Lisito**, P. Ramet

PaStiX = Parallel Sparse Linear Algebra Solver

- Sparse Linear Algebra Solver
  - > Solves $Ax = b$
  - > A matrix with a lot of zeros
- Many variants to support multi-core systems
  - > POSIX Threads:
    Single-thread, Multi-thread with static or dynamic scheduling
  - > Use of external runtime systems:
    StarPU, PaRSEC
- Support of distributed architectures with MPI
- Numerical features
  - > Low / Full rank
  - > Mixed precision
  - > Multi-DOF support (constant and variadic)

How does PaStiX work ?

4 steps:

1. Analyze (ordering, mapping, symbolic fact)
2. **Factorization** (A permutation, Cholesky, LU)
3. **Solve** (vectors permutation, solve)
4. Refinement

P❻ New to PaStiX 6
P❺✚ Improved from PaStiX 5
P❺ From PaStiX 5

Goal: fully support distributed architecture

- Distributed permutation

  P❻ Multiple Degree Of Freedom
  P❺✚ A (Factorization), b and Pb (solve) permutation

- Distributed trsm (solve)

  P❺ Multi-threaded static and dynamic

Current work: factorization with StarPU

P❻ Tasks level and left / right looking algorithm

01

Distributed support in PaStiX

Goal of the distributed work:

P❺✚ Matrix permutation (factorization)

P❺✚ Vectors permutation (solve)

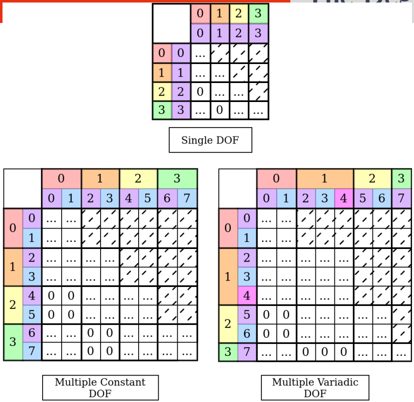
P❺ Solve with the single-threaded solve implementation

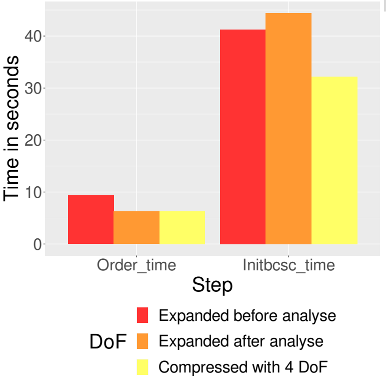
P❺ Solve with the multi-threaded schedulers

P❻ Support for the constant M-DOF

- Reduce analyze cost
- Better block size for the factorization
- Requested by users

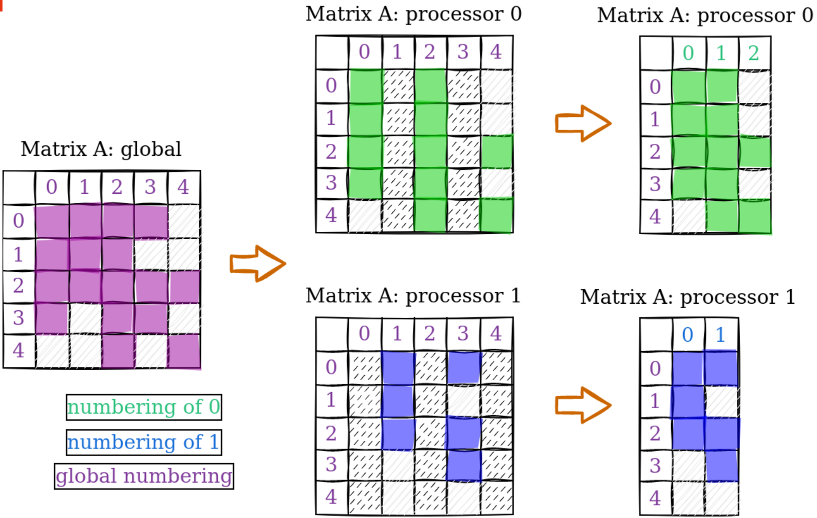
The Degree Of Freedom



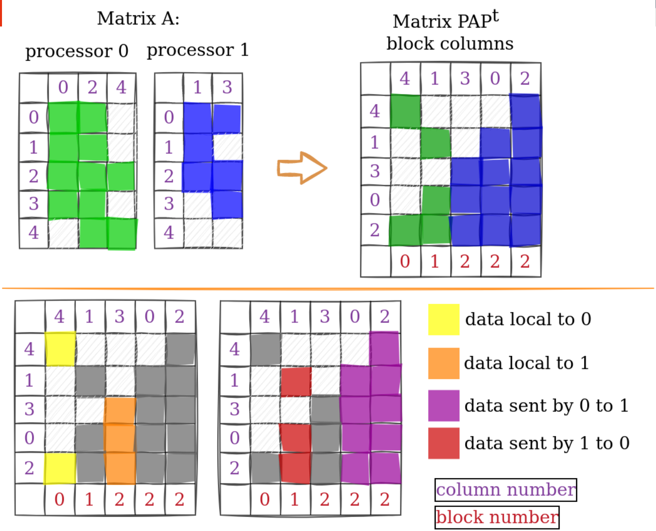


- Ordering faster with compressed A
- A permutation faster when compressed

Matrix distributed



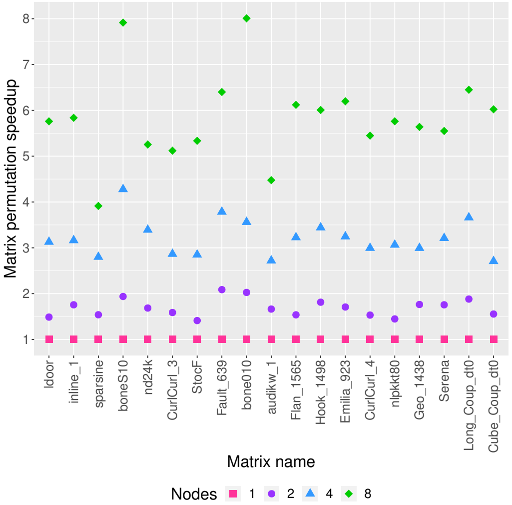
Matrix permuted and block partitionned



Benchmark

- The matrices:
  - Taken from the *SuiteSparse Matrix Collection*
  - Size: from 160 *million* to 7 *billion* non zero elements
  - Reals and symmetrics
- The machines:
  - Inria HPC plateform Plafrim
  - Bora: 2 CPU with 18 cores Intel CascadeLake
  - 1 MPI process per node and 1 thread per MPI process
- The tools version:
  - gcc 11.2
  - hwloc 2.7.0
  - openmpi 4.0.3
  - scotch 6.1.1

MPI matrix permutation



Speedup factor:

$\frac{time(n\_proc)}{time(1\_proc)}$ for n nodes

Average speedup

- 1.2 on 2 nodes
- 3.2 on 4 nodes
- 5.9 on 8 nodes

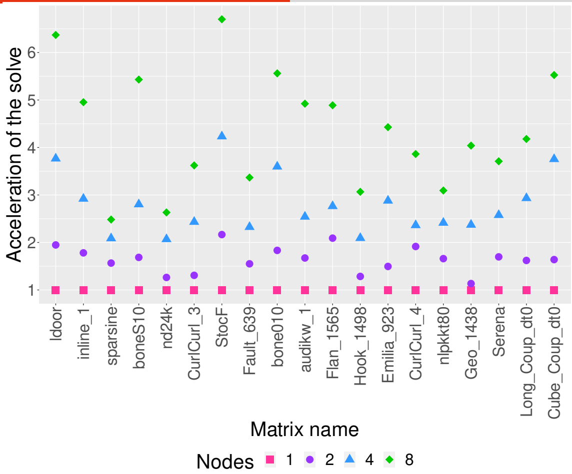
P❺ Single threaded

- With MPI on distributed architecture
- Multiple RHS

P❺ Multi threaded

- With MPI on distributed architecture
- Posix multi-thread internal
- Multiple RHS
- Static:
  - Each thread has a list of column blocks
  - Each thread executes operations on its blocks
- Dynamic:
  - Same as static with work stealing

MPI Single-thread solve

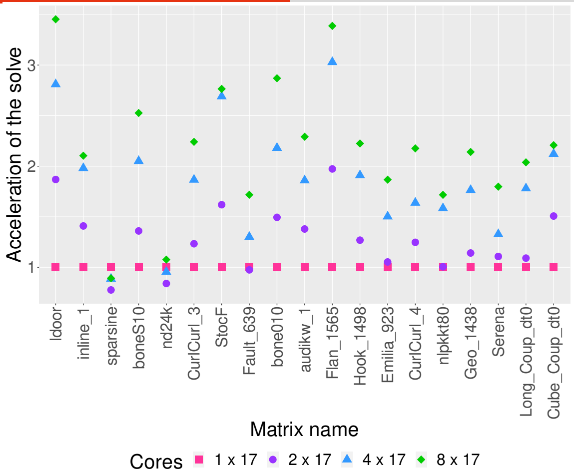


Average speedup

- 1.6 on 2 nodes
- 2.8 on 4 nodes
- 4.4 on 8 nodes

- The matrices:
  - Taken from the *SuiteSparse Matrix Collection*
  - Size: from 160 *million* to 7 *billion* non zero elements
  - Reals and symmetrics
- The machines:
  - Inria HPC plateform Plafrim
  - Bora: 2 CPU with 18 cores Intel CascadeLake
  - **2 MPI process per node and 17 threads per MPI process**
- The tools version:
  - gcc 11.2
  - hwloc 2.7.0
  - openmpi 4.0.3
  - scotch 6.1.1

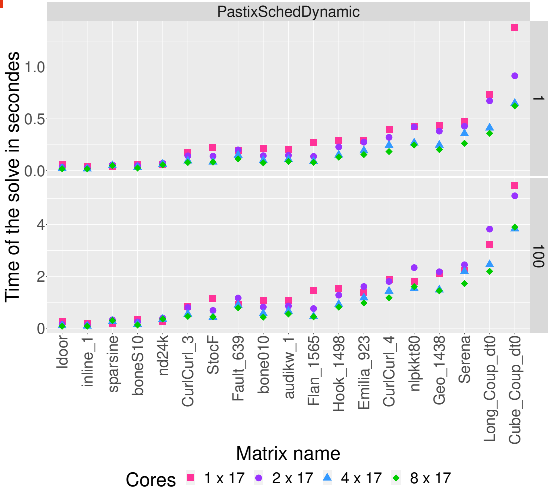
MPI Dynamic Multi-thread solve



Average speedup

- 1.3 on 2 nodes
- 1.9 on 4 nodes
- 2.2 on 8 nodes

MPI Dynamic Multi-thread solve



- Thanks to blas3: not 100 times slower with 100 rhs
- 14 times faster than single threaded solve

Contributions:

- **P❻** Constant M-DOF added in PaStiX 6
- **P❺✚** Matrix and vector permutations
- **P❺** MPI single-thread solve reintroduced from PaStiX 5
- **P❺** MPI Multi-threads Static and Dynamic Solve reintroduced from PaStiX 5

Future work:

- **P❻** Add Variadic M-DOF support for every step

02

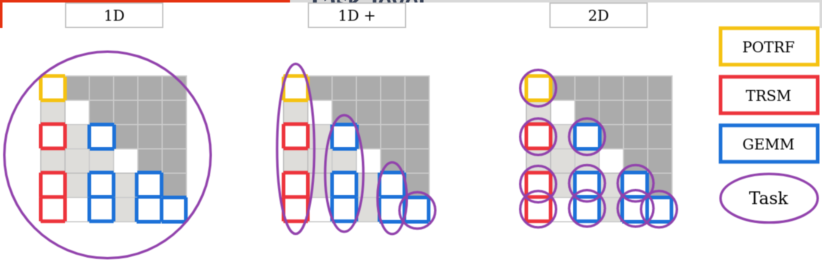
Current work: StarPU Factorization

The goal:

- Use of hierarchical tasks (bubbles) with StarPU, developped by Gwenolé Lucas in his PhD

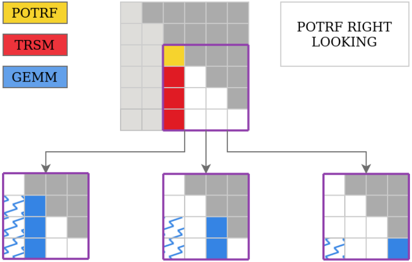
The state of PaStiX 6:

- Right and Left looking algorithm of Cholesky and LU factorization
- *1D+* and *2D* tasks levels



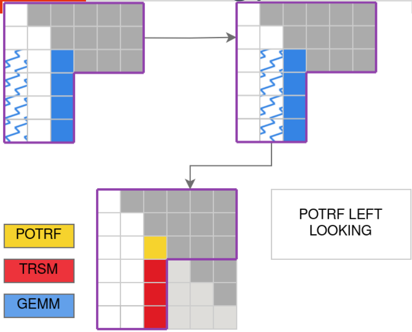
- Tasks submission instead of executing the operations right away
- The *2D* algorithm submit:
  - > *1D+* tasks if the block size is smaller than 2*d_block_size*
  - > *2D* tasks if the block size is greater than 2*d_block_size*
- Hierarchical tasks: mixte of *1D+* and *2D* with *2D* tasks submission chosen dynamically

Right looking *potrf*



- More parallelism
- Early update
- Submit ready tasks (no overload)
- Less fit for the hierarchical tasks

Left looking *potrf*

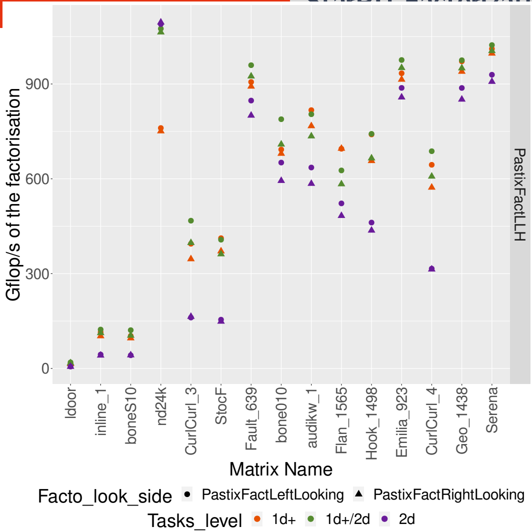


- Less parallelism
- Late update
- Submit not ready tasks
- More fit for the hierarchical tasks

Benchmark

- The matrices:
  - Taken from the *SuiteSparse Matrix Collection*
  - Size: from 160 *million* to 7 *billion* non zero elements
  - Reals, symmetrics and **Positives definites**
- The machines:
  - Inria HPC plateform Plafrim
  - Bora: 2 CPU with 18 cores Intel CascadeLake
  - **1 node and 36 threads StarPU**
- The tools version:
  - gcc 11.2
  - hwloc 2.9.0
  - scotch 6.1.1
  - **starpu 1.4.1**

StarPU Factorization



- *2D* slowest
- *1D+* and *2D* mixed better than just *1D+*
- Left looking slightly better than right looking

The goal:

P❻ Use of hierarchical tasks (bubbles) with StarPU, developped by Gwenolé Lucas in his PhD

Contributions:

P❻ Left looking algorithm of Cholesky and LU decomposition added for the StarPU scheduler with *1D+* and *2D* tasks level

Future study:

- Impact of the blocks size on the tasks levels and left / right looking algorithms
- Impact of left / right looking algorithm on GPU

Thank you for your attention!