



**HAL**  
open science

## New parallel features in the sparse solver PaStiX

Alycia Lisito, Mathieu Faverge, Pierre Ramet

► **To cite this version:**

Alycia Lisito, Mathieu Faverge, Pierre Ramet. New parallel features in the sparse solver PaStiX. Sparse Days 2023, Jun 2023, Toulouse, France. hal-04589448

**HAL Id: hal-04589448**

**<https://inria.hal.science/hal-04589448v1>**

Submitted on 27 May 2024

**HAL** is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.



Distributed under a Creative Commons Attribution 4.0 International License

The Inria logo is written in a white, elegant cursive script on a solid red rectangular background.

# 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

**P6** New to PaStiX 6

**P5+** Improved from PaStiX 5

**P5** From PaStiX 5

### Goal: fully support distributed architecture

- Distributed permutation
  - P6** Multiple Degree Of Freedom
  - P5+**  $A$  (Factorization), vectors (solve) permutation
- Distributed trsm (solve)
  - P5** Multi-threaded static and dynamic

### Current work: factorization with StarPU

**P6** Tasks level and left / right looking algorithm

# 01

## Distributed support in PaStiX

### Goal of the distributed work:

**P5+** Matrix permutation (factorization)

**P5+** Vectors permutation (solve)

**P5** Solve with the single-threaded solve implementation

**P5** Solve with the multi-threaded schedulers

### **P6** Support for the constant M-DOF

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

# The Degree Of Freedom

|   |   |     |     |     |     |
|---|---|-----|-----|-----|-----|
|   |   | 0   | 1   | 2   | 3   |
|   |   | 0   | 1   | 2   | 3   |
| 0 | 0 | ... | ... | ... | ... |
| 1 | 1 | ... | ... | ... | ... |
| 2 | 2 | 0   | ... | ... | ... |
| 3 | 3 | ... | 0   | ... | ... |

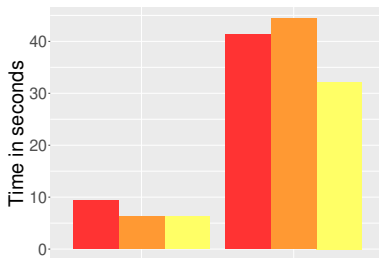
Single DOF

|   |   |     |     |     |     |     |     |     |     |
|---|---|-----|-----|-----|-----|-----|-----|-----|-----|
|   |   | 0   | 1   | 2   | 3   |     |     |     |     |
|   |   | 0   | 1   | 2   | 3   | 4   | 5   | 6   | 7   |
| 0 | 0 | ... | ... | ... | ... | ... | ... | ... | ... |
|   | 1 | ... | ... | ... | ... | ... | ... | ... | ... |
| 1 | 2 | ... | ... | ... | ... | ... | ... | ... | ... |
|   | 3 | ... | ... | ... | ... | ... | ... | ... | ... |
| 2 | 4 | 0   | 0   | ... | ... | ... | ... | ... | ... |
|   | 5 | 0   | 0   | ... | ... | ... | ... | ... | ... |
| 3 | 6 | ... | ... | 0   | 0   | ... | ... | ... | ... |
|   | 7 | ... | ... | 0   | 0   | ... | ... | ... | ... |

Multiple Constant  
DOF

|   |   |     |     |     |     |     |     |     |     |
|---|---|-----|-----|-----|-----|-----|-----|-----|-----|
|   |   | 0   | 1   | 2   | 3   |     |     |     |     |
|   |   | 0   | 1   | 2   | 3   | 4   | 5   | 6   | 7   |
| 0 | 0 | ... | ... | ... | ... | ... | ... | ... | ... |
|   | 1 | ... | ... | ... | ... | ... | ... | ... | ... |
| 1 | 2 | ... | ... | ... | ... | ... | ... | ... | ... |
|   | 3 | ... | ... | ... | ... | ... | ... | ... | ... |
| 2 | 4 | ... | ... | ... | ... | ... | ... | ... | ... |
|   | 5 | 0   | 0   | ... | ... | ... | ... | ... | ... |
| 3 | 6 | 0   | 0   | ... | ... | ... | ... | ... | ... |
|   | 7 | ... | ... | 0   | 0   | 0   | ... | ... | ... |

Multiple Variadic  
DOF



Step

- Expanded before analyse
- Expanded after analyse
- Compressed with 4 DoF

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

# Matrix distributed

Matrix A: global

|   | 0 | 1 | 2 | 3 | 4 |
|---|---|---|---|---|---|
| 0 | █ | █ | █ | █ | ▨ |
| 1 | █ | █ | █ | ▨ | ▨ |
| 2 | █ | █ | █ | █ | █ |
| 3 | █ | ▨ | █ | █ | ▨ |
| 4 | ▨ | ▨ | █ | ▨ | █ |

numbering of 0

numbering of 1

global numbering



Matrix A: processor 0

|   | 0 | 1 | 2 | 3 | 4 |
|---|---|---|---|---|---|
| 0 | █ | ▨ | █ | ▨ | ▨ |
| 1 | █ | ▨ | █ | ▨ | ▨ |
| 2 | █ | ▨ | █ | ▨ | █ |
| 3 | █ | ▨ | █ | ▨ | ▨ |
| 4 | ▨ | ▨ | █ | ▨ | █ |



Matrix A: processor 0

|   | 0 | 1 | 2 |
|---|---|---|---|
| 0 | █ | █ | ▨ |
| 1 | █ | █ | ▨ |
| 2 | █ | █ | █ |
| 3 | █ | ▨ | █ |
| 4 | ▨ | █ | █ |

Matrix A: processor 1

|   | 0 | 1 | 2 | 3 | 4 |
|---|---|---|---|---|---|
| 0 | ▨ | █ | ▨ | █ | ▨ |
| 1 | ▨ | █ | ▨ | ▨ | ▨ |
| 2 | ▨ | █ | ▨ | █ | ▨ |
| 3 | ▨ | ▨ | ▨ | █ | ▨ |
| 4 | ▨ | ▨ | ▨ | ▨ | █ |

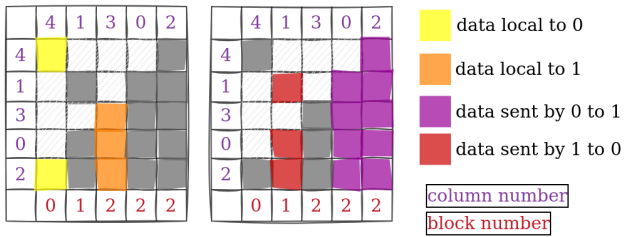
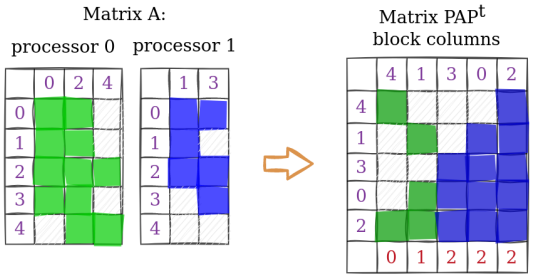


Matrix A: processor 1

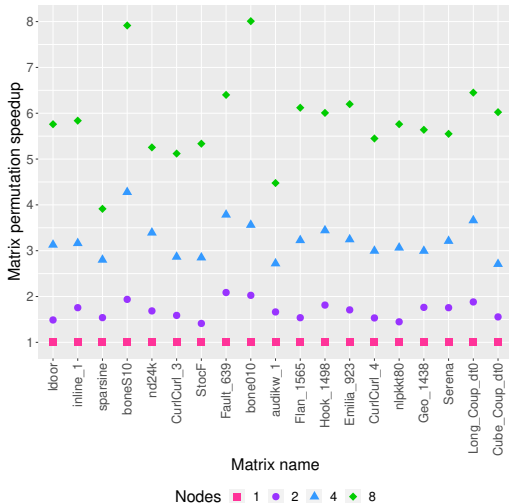
|   | 0 | 1 |
|---|---|---|
| 0 | █ | █ |
| 1 | █ | ▨ |
| 2 | █ | █ |
| 3 | ▨ | █ |
| 4 | ▨ | ▨ |



# Matrix permuted and block partitioned



- The matrices:
  - > Taken from the *SuiteSparse Matrix Collection*
  - > Size: from 160 *million* to 7 *billion* non zero elements
  - > Reals and symmetric
- The machines:
  - > Inria HPC platform 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



Speedup factor:

$$\frac{\text{time}(1\text{-proc})}{\text{time}(n\text{-proc})} \text{ for } n \text{ nodes}$$

Average speedup

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



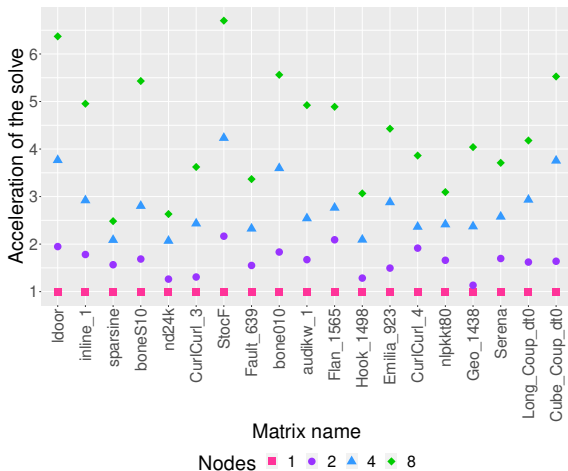
### P5 Single threaded

- With MPI on distributed architecture
- Multiple RHS

### P5 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

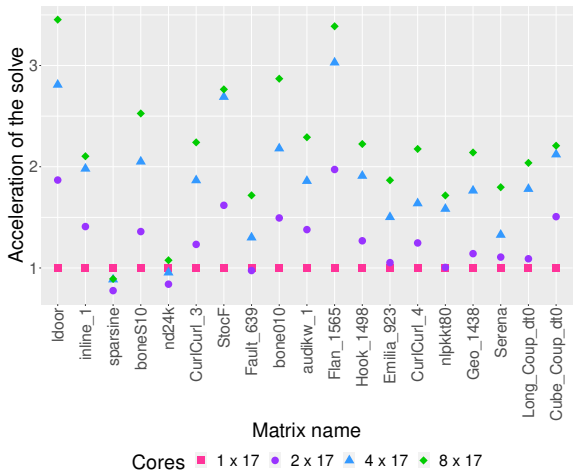
# MPI Single-thread solve



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 symmetric
- The machines:
  - > Inria HPC platform 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

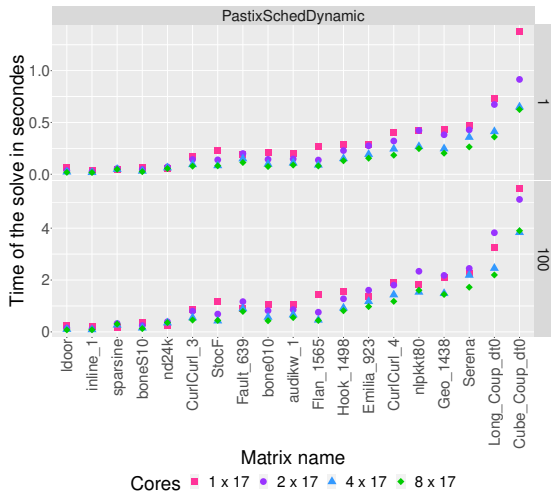


Average speedup

- 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:

- P6** Constant M-DOF added in PaStiX 6
- P5+** Matrix and vector permutations
- P5** MPI single-thread solve reintroduced from PaStiX 5
- P5** MPI Multi-threads Static and Dynamic Solve reintroduced from PaStiX 5

### Future work:

- P6** Add Variadic M-DOF support for every step

# 02

Current work:  
StarPU Factorization

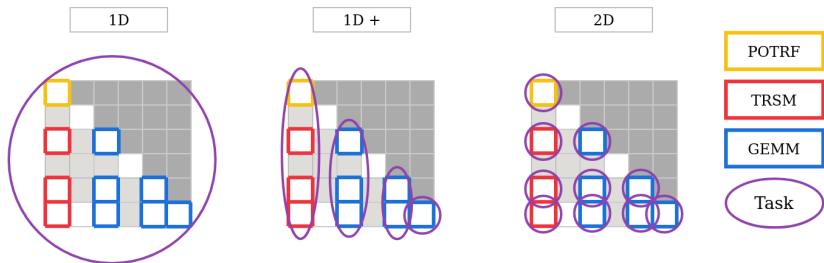
### The goal:

**P⑥** Use of hierarchical tasks (bubbles) with StarPU, developed by Gwenolé Lucas in his PhD

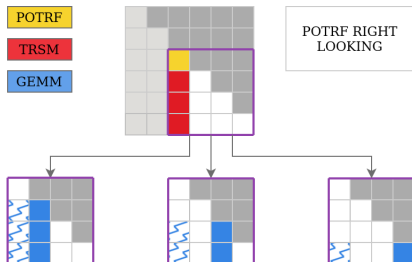
### The state of PaStiX 6:

**P⑥** Right and Left looking algorithm of Cholesky and LU factorization

**P⑥**  $1D+$  and  $2D$  tasks levels

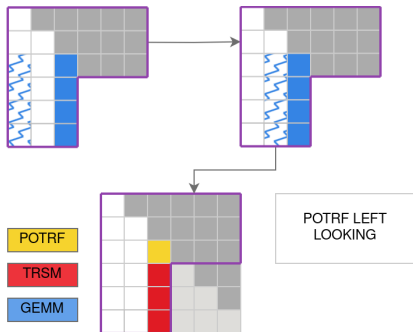


- Tasks submission instead of executing the operations right away
- The  $2D$  algorithm submit:
  - >  $1D+$  tasks if the block size is smaller than  $2d\_block\_size$
  - >  $2D$  tasks if the block size is greater than  $2d\_block\_size$
- Hierarchical tasks: mixte of  $1D+$  and  $2D$  with  $2D$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



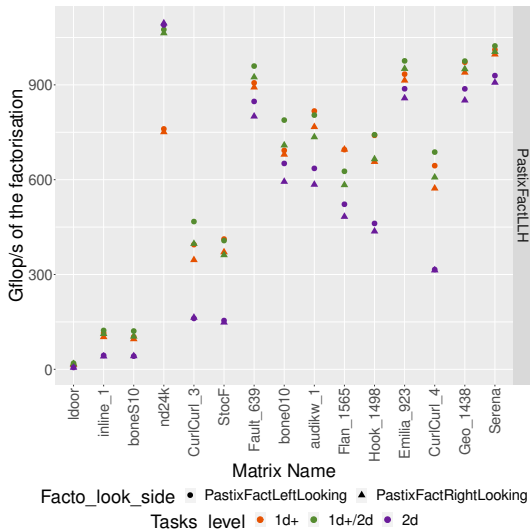
- More parallelism
- Early update
- Submit ready tasks (no overload)
- Less fit for the hierarchical tasks

## Left looking *potrf*



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

- The matrices:
  - > Taken from the *SuiteSparse Matrix Collection*
  - > Size: from 160 *million* to 7 *billion* non zero elements
  - > Reals, symmetric and **Positives definites**
- The machines:
  - > Inria HPC platform 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**



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



### The goal:

**P6** Use of hierarchical tasks (bubbles) with StarPU, developed by Gwenolé Lucas in his PhD

### Contributions:

**P6** 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!