New parallel features in the sparse solver PaStiX

M. Faverge, A. Lisito, P. Ramet
What is PaStiX?
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

Goal: fully support distributed architecture
- Distributed permutation
  - New to PaStiX 6
  - Multiple Degree Of Freedom
  - Improved from PaStiX 5
  - From PaStiX 5
- Distributed trsm (solve)
  - Multi-threaded static and dynamic

Current work: factorization with StarPU
- Tasks level and left / right looking algorithm

3 / 25 -New parallel features in the sparse solver PaStiX- Alycia Lisito
Distributed support in PaStiX
## New distributed MPI and M-DOF features

### Goal of the distributed work:

<table>
<thead>
<tr>
<th>P5+</th>
<th>Matrix permutation (factorization)</th>
</tr>
</thead>
<tbody>
<tr>
<td>P5+</td>
<td>Vectors permutation (solve)</td>
</tr>
</tbody>
</table>

- Solve with the single-threaded solve implementation
- 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
• Ordering faster with compressed $A$
• $A$ permutation faster when compressed
Matrix distributed

Matrix A: processor 0

Matrix A: processor 0

Matrix A: global

numbering of 0

numbering of 1

global numbering

Matrix A: processor 1

Matrix A: processor 1

7 / 25 - New parallel features in the sparse solver PaStiX - Alycia Lisito
Matrix permuted and block partitioned

Matrix A:

<table>
<thead>
<tr>
<th>processor 0</th>
<th>processor 1</th>
</tr>
</thead>
<tbody>
<tr>
<td>0  2  4</td>
<td>1  3</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>4</td>
<td>4</td>
</tr>
</tbody>
</table>

Matrix $P_A^t$ block columns

<table>
<thead>
<tr>
<th>4  1  3  0  2</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
</tr>
<tr>
<td>1</td>
</tr>
<tr>
<td>3</td>
</tr>
<tr>
<td>0</td>
</tr>
<tr>
<td>2</td>
</tr>
</tbody>
</table>

Colors:
- Yellow: data local to 0
- Orange: data local to 1
- Purple: data sent by 0 to 1
- Red: data sent by 1 to 0

Column number

Block number

8 / 25 - New parallel features in the sparse solver PaStiX - Alycia Lisito
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

- New parallel features in the sparse solver PaStiX - Alycia Lisito
MPI matrix permutation

Speedup factor:
\[ \frac{\text{time}(n\text{-proc})}{\text{time}(1\text{-proc})} \quad \text{for } n \text{ nodes} \]

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

Matrix name

Nodes

10 / 25 - New parallel features in the sparse solver PaStiX - Alycia Lisito
### Distributed substask solve

#### Single threaded
- With MPI on distributed architecture
- Multiple RHS

#### 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

Matrix name

Nodes 1 2 4 8

Average speedup

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

13 / 25 - New parallel features in the sparse solver PaStiX - Alycia Lisito
MPI Dynamic Multi-thread solve

Matrix name

Cores

Acceleration of the solve

Average speedup

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

14 / 25 -New parallel features in the sparse solver PaStiX- Alycia Lisito
<table>
<thead>
<tr>
<th>Matrix Name</th>
<th>Time of the solve in seconds</th>
</tr>
</thead>
<tbody>
<tr>
<td>PastixSchedDynamic</td>
<td>100</td>
</tr>
<tr>
<td>ldoor</td>
<td>inline_1</td>
</tr>
<tr>
<td>sparsine</td>
<td>nd24k</td>
</tr>
<tr>
<td>boneS10</td>
<td>CurlCurl_3</td>
</tr>
<tr>
<td>StocF</td>
<td>Fault_639</td>
</tr>
<tr>
<td>bone010</td>
<td>audikw_1</td>
</tr>
<tr>
<td>Flan_1565</td>
<td>Hook_1498</td>
</tr>
<tr>
<td>Emilia_923</td>
<td>CurlCurl_4</td>
</tr>
<tr>
<td>nlpkt80</td>
<td>Geo_1438</td>
</tr>
<tr>
<td>Serena</td>
<td>Long_Coup_dt0</td>
</tr>
<tr>
<td>Cube_Coup_dt0</td>
<td>1 x 17</td>
</tr>
<tr>
<td></td>
<td>2 x 17</td>
</tr>
<tr>
<td></td>
<td>4 x 17</td>
</tr>
<tr>
<td></td>
<td>8 x 17</td>
</tr>
</tbody>
</table>

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

Contributions:

P6 Constant M-DOF added in PaStiX 6
P6+ 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
Current work: StarPU Factorization
The goal:

- Use of hierarchical tasks (bubbles) with StarPU, developed 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 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
Right looking *potrf*

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

*New parallel features in the sparse solver PaStiX* - Alycia Lisito
- 2D slowest
- 1D+ and 2D mixed better than just 1D+
- Left looking slightly better than right looking
The goal:

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

Contributions:

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
The end

Thank you for your attention!