A Selective Nesting Approach for Task-Based Sparse
Cholesky Factorization

Valentin Le Fèvre*, Tetsuzo Usui†, Marc Casas*

*Barcelona Supercomputing Center, Spain
†Next Generation Technical Computing Unit, Fujitsu Limited, Japan

December 16, 2021
Outline

1. Introduction/Context

2. OpenMP parallelization of CHOLMOD and selective nesting

3. Results and perspectives
A: symmetric definite positive matrix

$$A = LL^T$$

$$Ax = b \Rightarrow LL^T x = b \Rightarrow x = (L^T)^{-1}L^{-1}b$$

Several solvers for sparse matrices: CHOLMOD, PARDISO, MUMPS, . . .
A: symmetric definite positive matrix

$$A = LL^T$$

$$Ax = b \Rightarrow LL^Tx = b \Rightarrow x = (L^T)^{-1}L^{-1}b$$

Several solvers for sparse matrices: **CHOLMOD**, PARDISO, MUMPS, ... CHOLMOD is originally a sequential code.
Structure of a solver

Usually 3 phases:
- Analyze
- Factorize
- Solve
Structure of a solver

Usually 3 phases:

- Analyze $\Rightarrow$ perform some optimizations (reduce fill-in), build the elimination tree, aggregate nodes...
- Factorize
- Solve
Structure of a solver

Usually 3 phases:

- **Analyze** ⇒ perform some optimizations (reduce fill-in), build the elimination tree, aggregate nodes...
- **Factorize** ⇒ main part of the solver, depends on the optimizations done in the previous phase.
- **Solve**
Structure of a solver

Usually 3 phases:

- **Analyze** ⇒ perform some optimizations (reduce fill-in), build the elimination tree, aggregate nodes...
- **Factorize** ⇒ main part of the solver, depends on the optimizations done in the previous phase.
- **Solve** ⇒ simple: triangular solve.
Elimination tree

- Describes the structure of the sparse matrix
- One node = one column
Elimination tree

- Describes the structure of the sparse matrix
- One node $\equiv$ several contiguous columns

Node amalgamation: *supernodes*
Each supernode can be transformed with BLAS calls.
Elimination tree

- Describes the structure of the sparse matrix
- One node = several contiguous columns

Node amalgamation: *supernodes*
Each supernode can be transformed with BLAS calls.
1. Introduction/Context

2. OpenMP parallelization of CHOLMOD and selective nesting

3. Results and perspectives
Task-based approach

```
for (s = 0 ; s < nsuper ; s++)
{
    #pragma omp task in(*dep_in[s][ii]), ii=0;num_in[s]) out(*dep_out[s]) \n    default(none) shared(...) firstprivate(...) private(...) label(outer)
{
    // Construction of the supernode
    for (idxS = STp[s] ; idxS < STp[s+1] ; idxS++) {
        d = STi[idxS] ;
        if (d==s) continue ;
        #pragma omp task default(none) shared(...) \n        private(...) firstprivate(...) private(...) label(inner)
        {
            // SYRK and GEMM
            omp_set_lock(&omp_lock);
            // Assembly of supernode
            omp_unset_lock(&omp_lock);
        }
    }
    #pragma omp taskwait
    // POTRF and TRSM
}
```
Selective Nesting

bone010

(a) Without nested tasks (NON-NESTED)

(b) With nested tasks (NESTED)
Selective Nesting

inline_1

(c) Without nested tasks (NON-NESTED)

(d) With nested tasks (NESTED)
Selective Nesting

- Trade-off between parallelism and overhead of task creation
- If no task is nested: lack of parallelism, especially when reaching the root of the elimination tree
Selective Nesting

- **Trade-off between parallelism and overhead of task creation**
- If no task is nested: lack of parallelism, especially when reaching the root of the elimination tree
- If every task is nested: too many tasks, overhead of task creation and destruction becomes a bottleneck
Determining threshold $D$

![Graph showing the relationship between the number of inner tasks and the number of outer tasks.](image)

![Graph showing the relationship between the threshold $D$ and the number of tasks and time.](image)

bone010
Analysis on a few matrices:

- Outer tasks with most *inner* tasks should be nested
- Optimal $D$ always less than 30% of max number of *inner* tasks
- Ratio between number of columns and number of tasks tend to be close (at optimal $D$)
- $\Rightarrow$ Algorithm $\text{OPT-D}$
Determining threshold $D$

Analysis on a few matrices:

- Outer tasks with most inner tasks should be nested
- Optimal $D$ always less than 30% of max number of inner tasks
- Ratio between number of columns and number of tasks tend to be close (at optimal $D$)

$\Rightarrow$ Algorithm $\text{OPT-D}$

If an inner task is small: keep it inside the outer loop $\Rightarrow$ Algorithm $\text{OPT-D-COST}$
Determining threshold $D$

Analysis on a few matrices:

- Outer tasks with most inner tasks should be nested
- Optimal $D$ always less than 30% of max number of inner tasks
- Ratio between number of columns and number of tasks tend to be close (at optimal $D$)
- $\Rightarrow$ Algorithm OPT-D

If an inner task is small: keep it inside the outer loop $\Rightarrow$ Algorithm OPT-D-COST

We will compare to mt-BLAS: sequential cholmod with multi-threaded BLAS.
Outline

1. Introduction/Context

2. OpenMP parallelization of CHOLMOD and selective nesting

3. Results and perspectives
Matrices with $10,000 < \#nnz < 50,000$
Matrices with $3,000,000 < \#nnz < 6,000,000$
Task-graph should depend on the data

Finding a good granularity can drastically improve the performance

Algorithm OPT-D-COST suitable for a large variety of matrices

without trying lots of configurations each time

mt-BLAS can be a good alternative for several matrices that can we can detect
Limits:

- Algorithm tuned from experiments on a given platform (here A64FX processors)
- For a few matrices in the evaluation, the performance is degraded

Questions/Collaborations:

- Is there a way to find a good model, used to determine $D$ afterwards? Hopefully, not platform-dependent
- What makes some matrices very different?
- Implementations of other algorithms to try this strategy