TASK-BASED SPARSE CHOLESKY SOLVER ON TOP OF RUNTIME SYSTEM

Iain S. Duff, Jonathan D. Hogg and **Florent Lopez**

Sparse Days, 2016

Rutherford Appleton Laboratory
NLAFET Project

Objective

Solve $Ax = b$, where A is large and sparse, on modern architectures.

Exploiting modern platforms is challenging:

- Multicore processors and deep memory hierarchy.
- Heterogeneous e.g. CPU & GPU or Xeon Phi.
- Distributed-memory systems.

Use Direct Method: Sparse Cholesky factorization $A = LL^T$

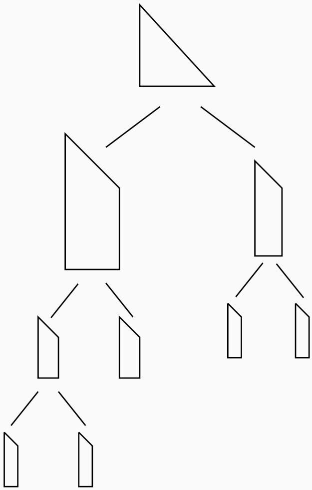
▲ Numerically robust and general purpose

▼ High memory usage and computational cost

The numerical factorization of A rely on an *elimination tree* expressing data dependencies in the factor L. Each node, referred to as *supernode*, is a dense lower trapezoidal submatrix of L.

The tree is traversed in a topological order, and each node is factorised using dense Cholesky algorithm.

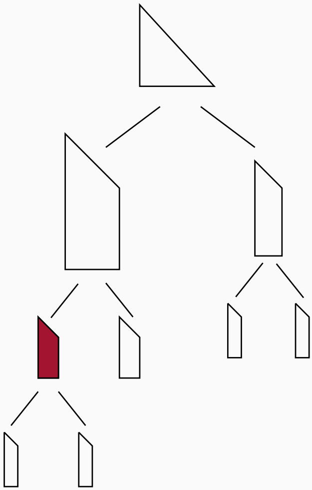
Updates between node are handled using a supernodal scheme i.e. updates are applied directly to the target supernode.



The numerical factorization of A rely on an *elimination tree* expressing data dependencies in the factor L. Each node, referred to as *supernode*, is a dense lower trapezoidal submatrix of L.

The tree is traversed in a topological order, and each node is factorised using dense Cholesky algorithm.

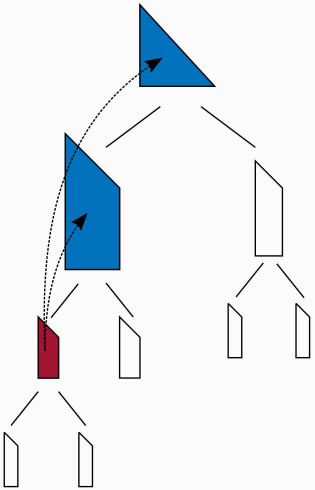
Updates between node are handled using a supernodal scheme i.e. updates are applied directly to the target supernode.



The numerical factorization of A rely on an *elimination tree* expressing data dependencies in the factor L. Each node, referred to as *supernode*, is a dense lower trapezoidal submatrix of L.

The tree is traversed in a topological order, and each node is factorised using dense Cholesky algorithm.

Updates between node are handled using a supernodal scheme i.e. updates are applied directly to the target supernode.



Sequential Task Flow (STF) programming model:

- Tasks are submitted to the runtime system following the sequential algorithm.
- The runtime analyses manipulated data and infers task dependencies in order to ensure the sequential consistency of the parallel code.
- The DAG is executed via a dynamic scheduling of the (ready) tasks on the architectures.
- The runtime may be capable of automatically handling the data transfer across the architecture.
- Superscalar analysis in processors: dependency detection between instructions in order to issue them in parallel.

OpenMP 4.0

- `task` construct and `depend` clause (`in`, `out`, `inout`).
- No control on the scheduling strategy.
- Shared-memory system only.

StarPU

- `starpu_insert_task`, data *handle* with access mode (`R`, `W`, `RW`).
- Full control on schduling policy with possibility to implement new one.
- API for distributed-memory systems.

The STF Sparse Cholesky Factorization

```
forall nodes snode in post-order
   call alloc(snode) ! allocate data structures

   call submit(init, snode:W) ! initianlize node structure
end do

forall nodes snode in post-order
  ! factorize node
  do k=1..n in snode
    call submit(factorize, snode:R, blk(k,k):RW) ! factorize block

    do i=k+1..m in snode
        call submit(solve, blk(k,k):R, blk(i,k):RW) ! perform solve
    end do

    do j=k+1..n in snode
       do i=k+1..m in snode
          call submit(update, blk(j,k):R, blk(i,k):R, blk(i,j):RW)
       end do
    end do

    forall ancestors(snode) anode
      do j=k+1..p(anode) in snode
         do i=k+1..m in snode
            call submit(update_btw, blk(j,k):R, blk(i,k):R, a_blk(rmap(i), cmap(j)):RW)
         end do
      end do
    end do

  end do
end do
```

Experiments

#	Matrix	Flops (10^9)	Application/description
1	Schmid/thermal2	18.6	Unstructured thermal FEM
2	Rothberg/gearbox	22.8	Aircraft flap actuator
3	DNVS/m_t1	23.4	Tubular joint
4	DNVS/thread	35.7	Threaded connector
5	DNVS/shipsec1	40.5	Ship section
6	GHS_psdef/crankseg_2	48.8	Linear static analysis
7	AMD/G3_circuit	67.3	Circuit simulation
8	Koutsovasilis/F1	228	AUDI engine crankshaft
9	Oberwolfach/boneS10	297	Bone micro-FEM
10	ND/nd12k	514	3D mesh problem
11	JGD Trefethen/Trefethen_20000	669	Integer matrix
12	ND/nd24k	2080	3D mesh problem
13	Oberwolfach/bone010	3910	Bone micro-FEM
14	GHS_psdef/audikw_1	5840	Automotive crankshaft

- Symmetric positive-definite matrices
- Metis nested disection ordering
- Machine: 2 x 14 cores E5-2695 v3 (Haswell) @ 2.30GHz

Parametrized Task Graph (PTG) programming model:

- Uses a compact representation of the DAG which is problem size independent.
- The dataflow between tasks is explicitly encoded (i.e. task dependencies are explicitly given).
- The runtime handles the communications implicitly using the dataflow representation.
- Under some hypothesis, the dataflow information can be automatically extracted from the sequential code using a dedicated compiler: not in our case unfortunately.