Accelerating generalized Cholesky decomposition using multiple processors

C.C.Tscherning & M.Veicherts, University of Copenhagen, Jan. 2008

Application in Least-Squares Collocation

$$y_i = L_i(T) + e_i + A_i^T \bullet X$$

$$\tilde{T}(P) = \{C_{Pi}\}^T \bar{C}^{-1}(y - A^T X)$$

$$\tilde{X} = (A^T \bar{C}^{-1} A + W)^{-1}(A^T \bar{C}^{-1} y)$$

Error-covariance estimation

$$H = \{COV(L_k, L_i\}^T \overline{C}^{-1}, \text{ MxN matrix}$$

$$m_X^2 = (A^T \overline{C}^{-1} A + W)^{-1}$$

$$\{ec_{kl}\} = \{\sigma_{kl}\} - H\{\text{cov}(L_j, L_l)\} + HAm_X^2 (HA)^T$$

Cholesky Factorization

- *L*: lower triangular matrix

$$C x = L L^T x = y$$

$$L^{-1}(L L^T)x = L^{-1} y$$

$$L^T x = L^{-1} y, \text{ now upper - triangular system.}$$

$$L_{ij} = (C_{ij} - \sum_{m=1}^{j-1} L_{im} L_{jm}) / L_{jj}$$

$$L_{jj} = (C_{jj} - \sum_{m=1}^{j-1} L_{jm}^2)^{1/2}$$

Generalized Cholesky

$$\begin{Bmatrix} \{\overline{C}_{ij}\}_{nn} & \{A_{jl}\}_{mn} \\ \{A_{ik}\}^{T}_{nm} & \{W_{kl}\}_{mm} \end{Bmatrix} \begin{Bmatrix} x \\ X \end{Bmatrix} = \begin{Bmatrix} y \\ p \end{Bmatrix},$$

Can be Cholesky - factorized using negative accumulation when subscript $\geq$ n.

More Generalized Cholesky

$$\begin{Bmatrix} \{\overline{C}_{ij}\}_{pp} & \{A_{jl}\}_{pq} & \{C_{jm}\}_{pr} \\ \{A_{ik}\}^{T}_{qp} & \{W_{kl}\}_{qq} & \{A_{jk}\}_{qr} \\ \{C_{in}\}^{T}_{rp} & \{A_{ni}\}^{T}_{rq} & \{\sigma_{nm}\}_{rr} \end{Bmatrix}$$

when factorized with negative accumulation for subscript larger than p and no accumulation for subscript large than $p+q$ it returns error - covariances.

Parallization

- When diagonal element has been computed may each element in the row be reduced separately:

$$L_{ij} \cdot L_{jj} = (C_{ij} - \sum_{m=1}^{j-1} L_{im} L_{jm}),$$

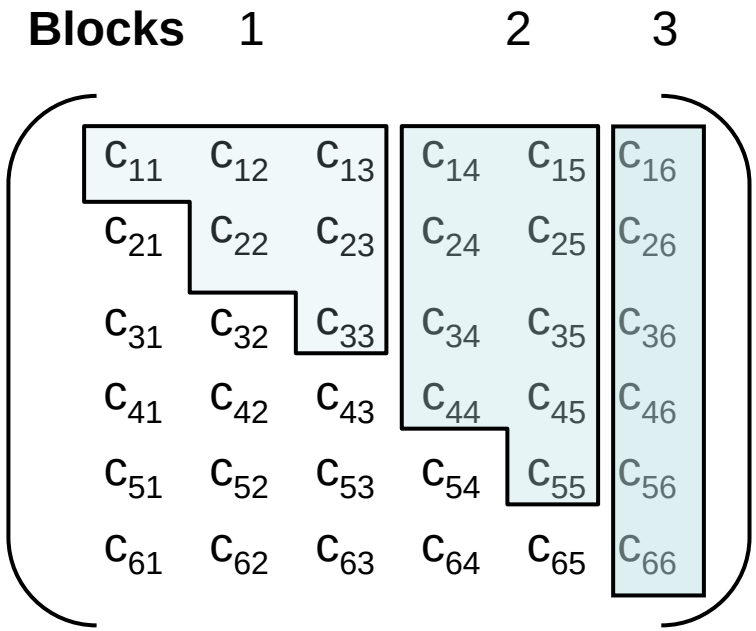
$$\text{involves only row - elements in same column } j, i \geq j.$$

- Hence each processor may take care of one column.

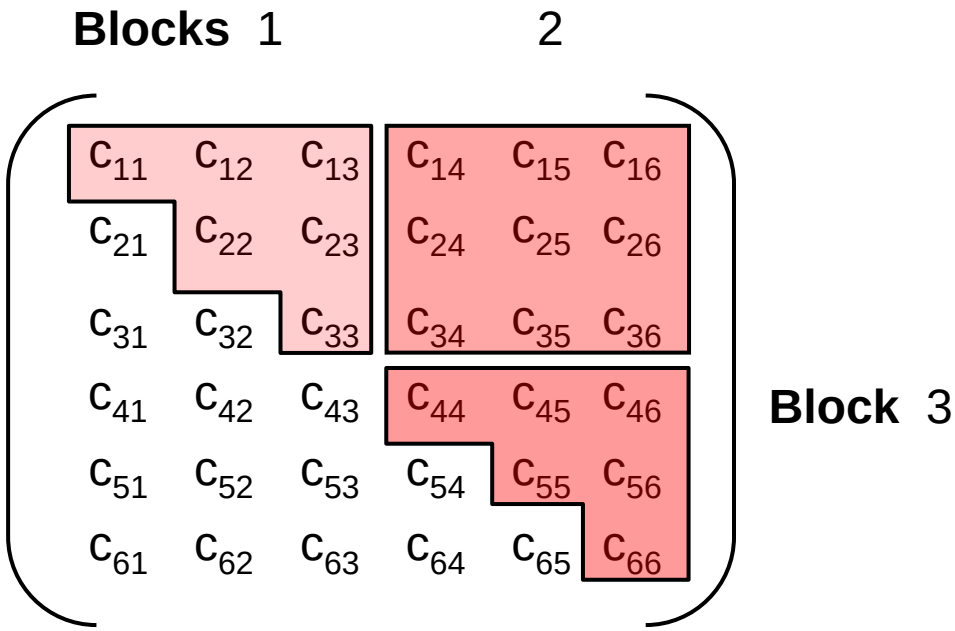
Blockwise factorization

- Should one row be factorized at at time ?
- Or should we make the factorization of blocks of elements ?
- Out-of-core factorization needed for large matrices, so let the processors work on blocked matrices.

Block division
Column-wise and rectangular



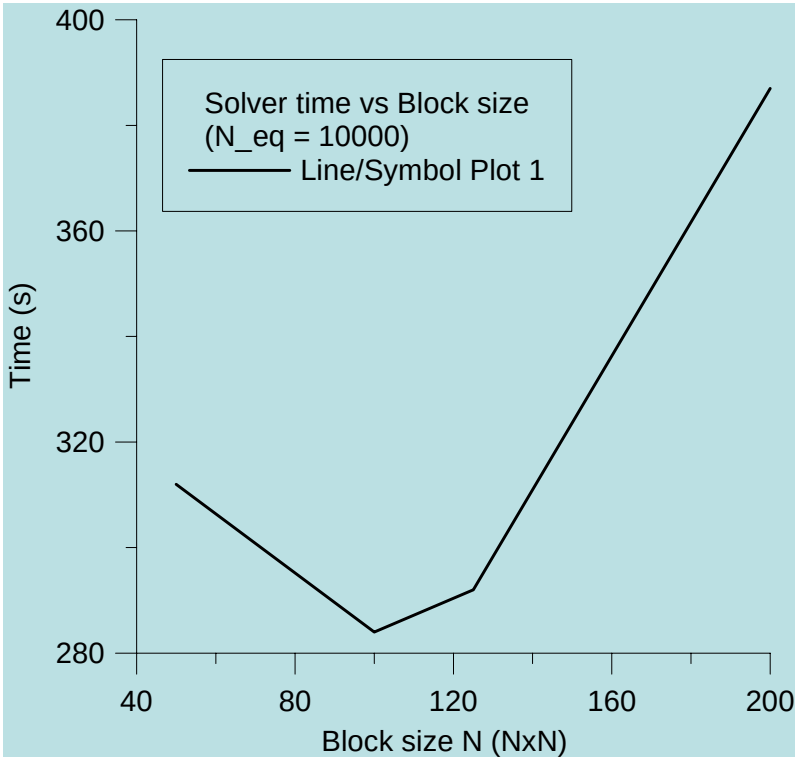
3 blocks ‘Column-wise’ 1-dim.
of size 9



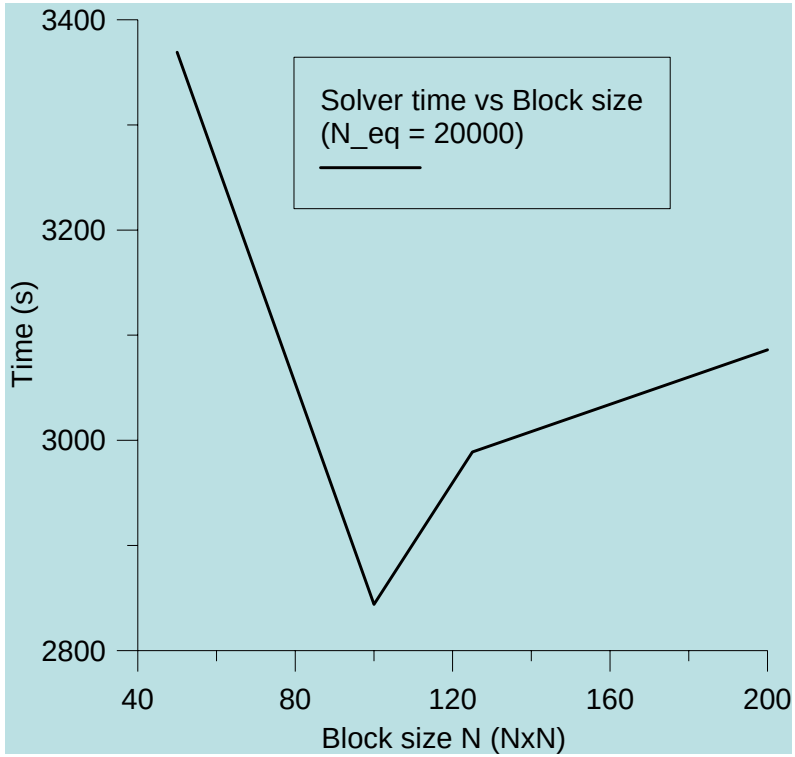
3 blocks rectangular 2-dim.
of size 3*3

Blocksize tests

NEQ = 10000, Nproc = 4

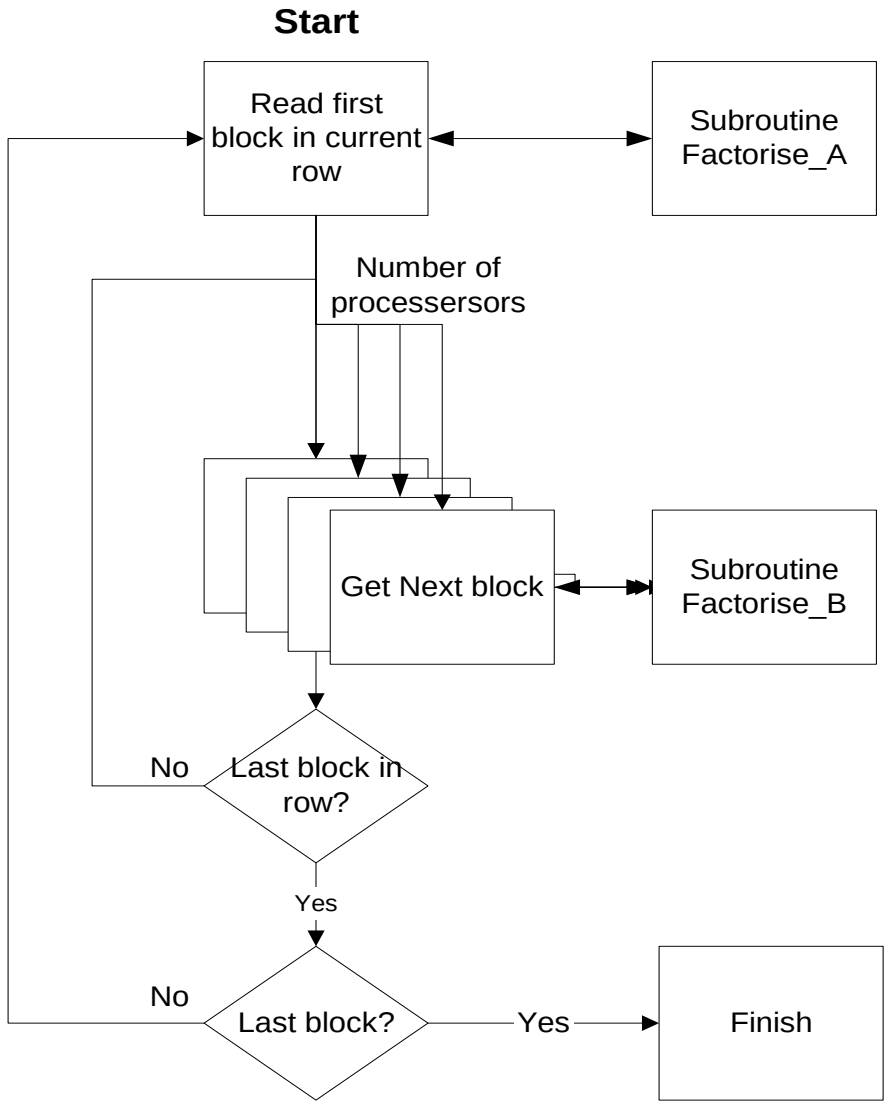


NEQ = 20000, Nproc = 2



Parallelization

Flowchart over the Choleski factorisation with NES_MP and related subroutine(s)



Parallelization Results

Results (Perf. test on two PCs, Compiler PGF90)

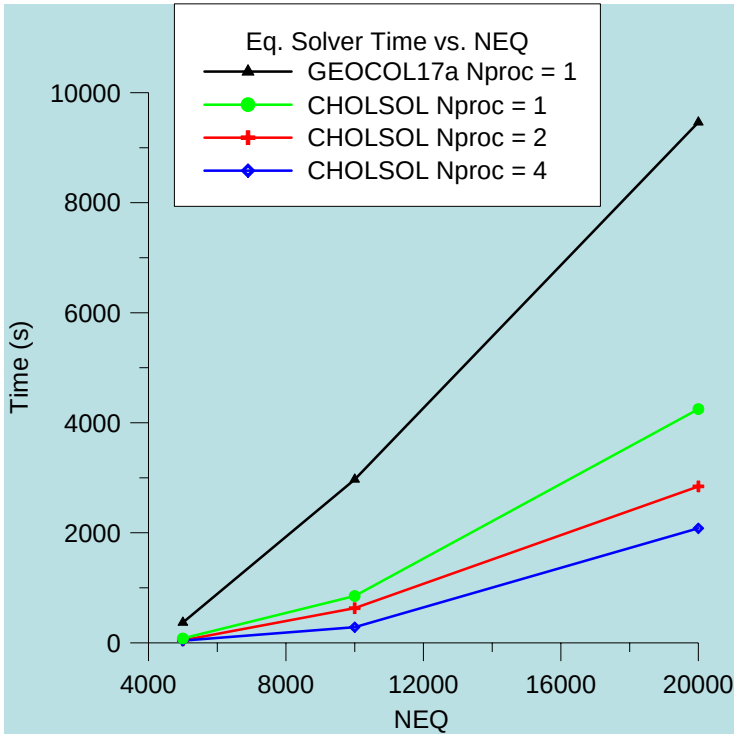
		GOCE (4x3GHz, 2GB)		IKOS (4x2.66GHz, 4GB)	
PROC	NEQ.	NES	NES_MP	NES	NES_MP
1	6400	775	177		
2	6400		130	136	71
4	6400		87		
1	8100	1570	347		
2	8100		228		
4	8100		177		
1	10000	2966	650	586	290
2	10000		446		159
4	10000		369		

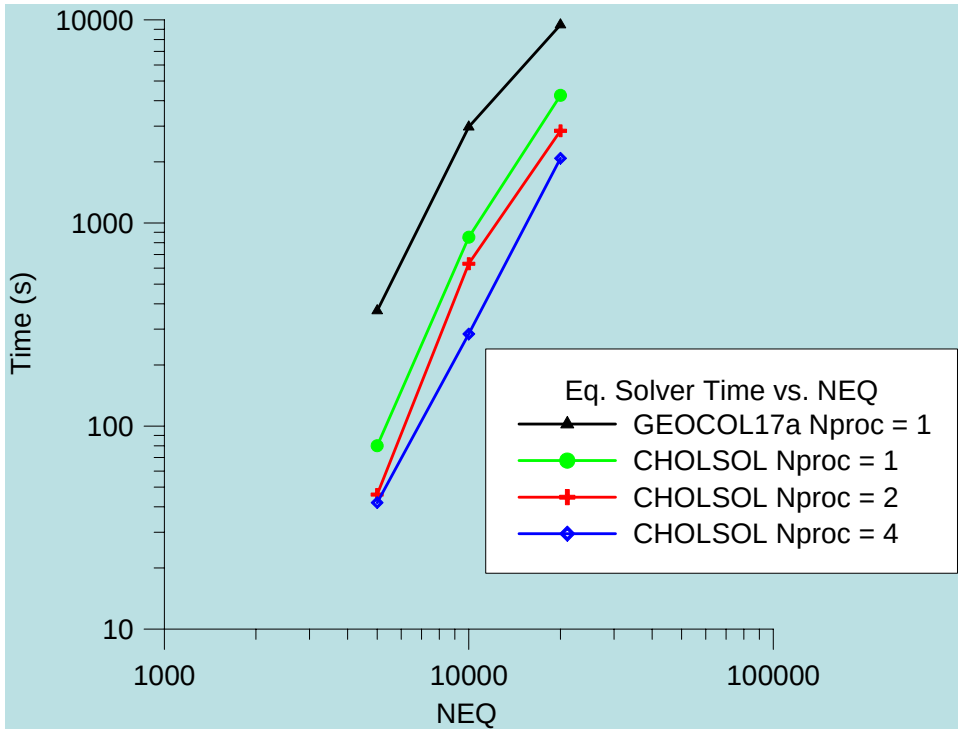
Integration in GEOCOL18

Geocol integration tests:
Timing (in s) for equation solving only.

Server	NEQ	Geocol17a	Geocol18zr Processors		
			1	2	4
GOCE	5000	370	80	46	24
	10000	2971	851	630	354
	20000	9464	4249	2844	2081
IKOS	5000		23		
	10000		330		

Performance Increase





Conclusion

- Generalized Cholesky-factorization enables the use of parallelization for solution and error-covariance computation.
- Time gain using parallelization depends on number of processors, block-size and how busy the computer is doing other things.

Note: further use of multiprocessing

- Evaluation of spherical harmonic series (N.Pavlis et al.).
- Establishing the normal-equation matrix or computing a column of covariances
- Factorisation may start as soon as a row of blocks has been established.
- Gives realistic speeds of LSC applications (minutes instead of days).