Using recursion to improve performance of dense linear algebra software

Erik Elmroth
Dept of Computing Science & HPC2N
Umeå University, Sweden

Joint work with Fred Gustavson, Isak Jonsson & Bo Kågström

PMAA 2006, Rennes,
September 7 – 9, 2006

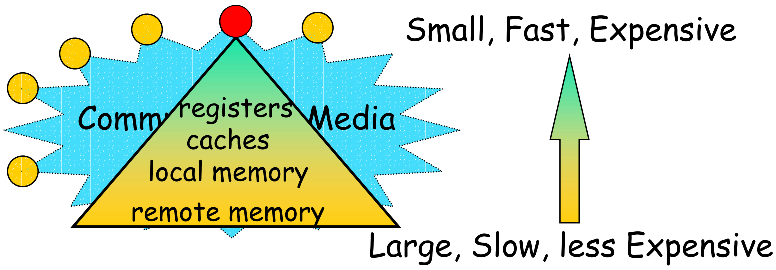
High Performance Computing
Center North (HPC2N)

Matrix Computations

- Fundamental and ubiquitous in computational science and its vast application areas
- Library software – optimized building blocks for fundamental operations
  - BLAS, (Sca)LAPACK, SLICOT (see also NETLIB)
  - ESSL and other vendors
  - Portability and efficiency
- Architecture evolution: HPC systems with multiple SMP nodes, several levels of caches, more functional units per CPU
- Continuing demand for new and improved algorithms and software

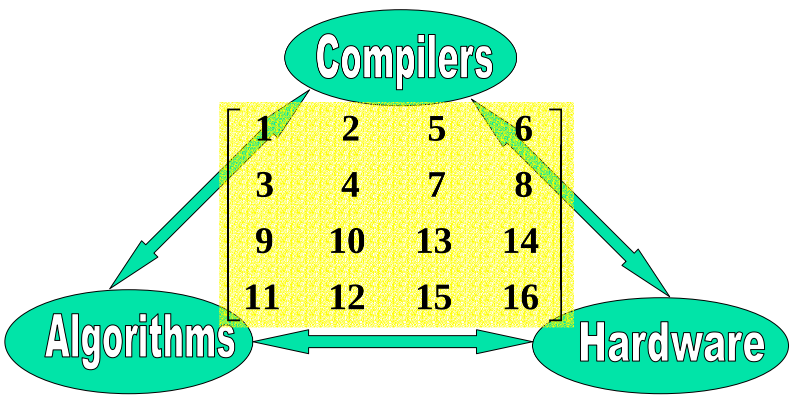
“Data transport” in memory hierarchies

- of today’s computer systems
  - PC - cluster - supercomputer



Key to performance: understand algorithm - architecture interaction
Hierarchical blocking

The fundamental AHC triangle



Outline

- Hierarchical blocking: motivation and implications
- Recursive blocked templates
- (Recursive blocked data structures)
- Case studies:
  1. General matrix multiply and add (GEMM)
  2. QR factorization
  3. Over- and under-determined linear systems
  4. Triangular matrix equations and condition estimation
  5. (Packed Cholesky factorization)
- Concluding remarks

Block algorithms

- Block algorithms instead of point-wise
- Matrix operations instead of scalar ops
  (key to performance: $O(n^3)$ ops on $O(n^2)$ data)
- (Explicit) blocking through multiple levels of nested loops/subroutine calls
- Small fraction level-1 and level-2
- Bulk computations as level-3
- Typically, level-3 fraction increases with matrix size

Recursive Blocked Algorithms

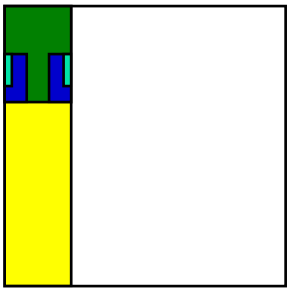
- Automatic variable blocking
- Replaces level-1 & -2 ops by level-3
  - further improves performance
  - reduces the amount of code needed (level-2 routines)
- Improve on the temporal locality

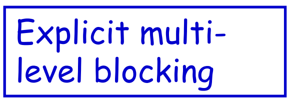
Further performance improvements

- Match data structure with the algorithm
- Recursive blocked data structures improve on the spatial locality

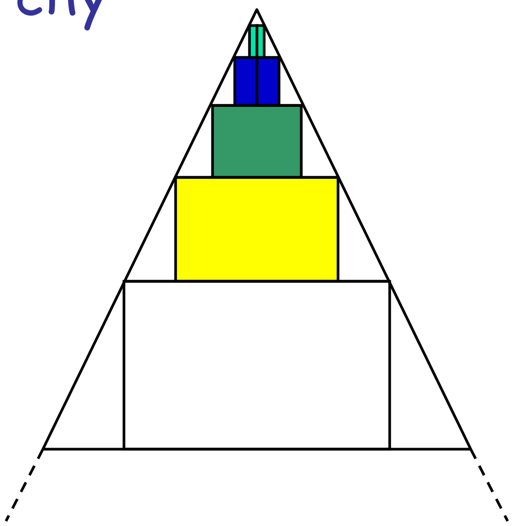
Some illustrations

Traditional blocking for a memory hierarchy

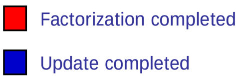




Explicit multi-level blocking

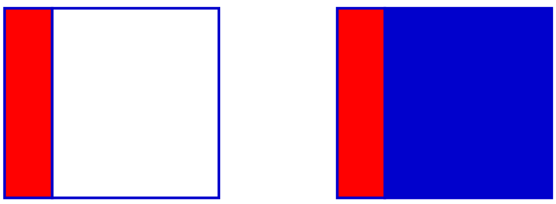


Standard (LAPACK-style) factorization block algorithms



Factor fixed size block column

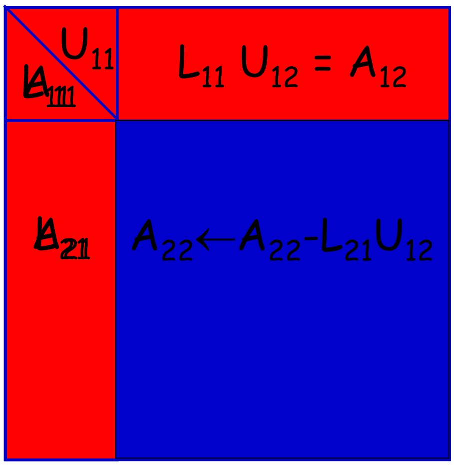
Update remaining matrix



Repeat for updated matrix

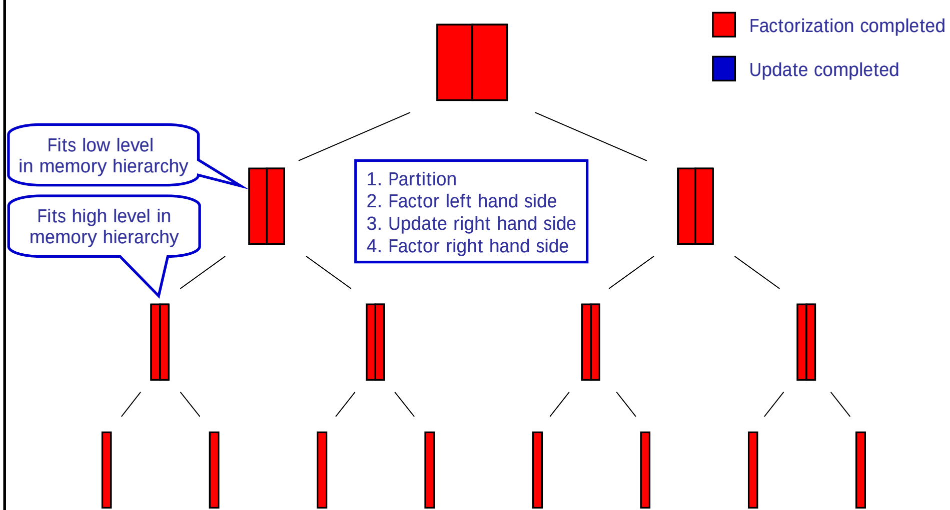
LAPACK-style LU factorization

Factor fixed size block column

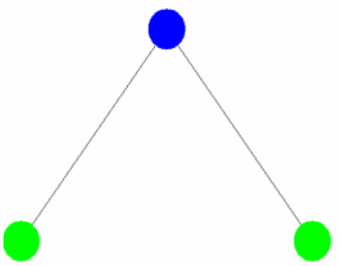


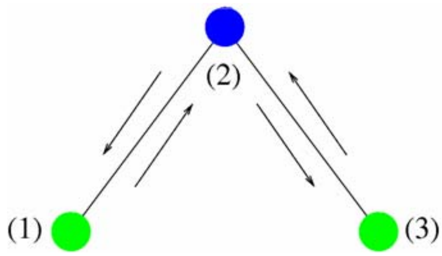
Repeat for updated matrix

Recursion template for one-sided matrix factorization



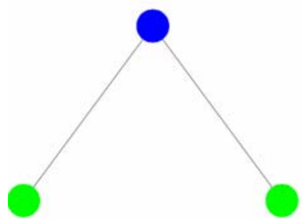
Splittings defining independent and dependent tasks





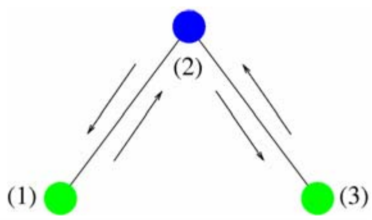
Critical path of subtasks:
(1), (2), (3)

TRSM Operation: AX = C, A mxm upper triangular, C/X mxn



$$A \begin{bmatrix} X_1 & X_2 \end{bmatrix} = \begin{bmatrix} C_1 & C_2 \end{bmatrix}$$

$$AX_1 = C_1,$$
$$AX_2 = C_2.$$



$$\begin{bmatrix} A_{11} & A_{12} \\ & A_{22} \end{bmatrix} \begin{bmatrix} X_1 \\ X_2 \end{bmatrix} = \begin{bmatrix} C_1 \\ C_2 \end{bmatrix}$$

$$A_{11}X_1 = C_1 - A_{12}X_2,$$
$$A_{22}X_2 = C_2.$$

Case Study 1

General matrix multiply and add (GEMM)

Recursive splittings for GEMM:

$$C \leftarrow \beta \mathrm{op}(C) + \alpha \mathrm{op}(A)\mathrm{op}(B)$$

<u>Split</u> m × n m × k k × n

m, n, k

$$\begin{bmatrix} C_{11} & C_{12} \\ C_{21} & C_{22} \end{bmatrix} + \begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix} \begin{bmatrix} B_{11} & B_{12} \\ B_{21} & B_{22} \end{bmatrix} =$$

m

$$= \begin{bmatrix} \begin{bmatrix} C_{11} & C_{12} \end{bmatrix} + \begin{bmatrix} A_{11} & A_{12} \end{bmatrix} \begin{bmatrix} B_{11} & B_{12} \\ B_{21} & B_{22} \end{bmatrix} \\ \begin{bmatrix} C_{21} & C_{22} \end{bmatrix} + \begin{bmatrix} A_{21} & A_{22} \end{bmatrix} \begin{bmatrix} B_{11} & B_{12} \\ B_{21} & B_{22} \end{bmatrix} \end{bmatrix} =$$

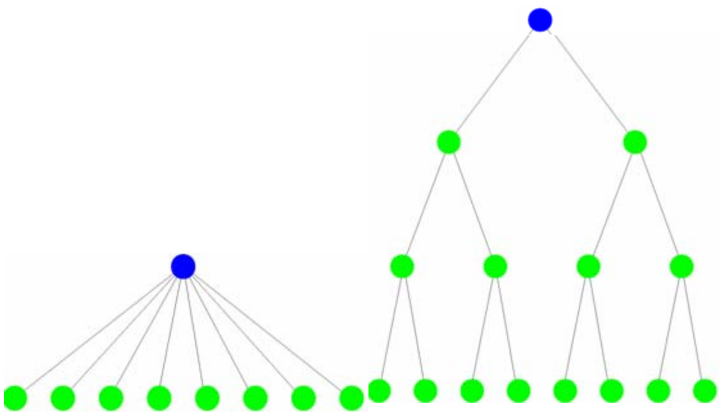
n

$$= \begin{bmatrix} \begin{bmatrix} C_{11} \\ C_{21} \end{bmatrix} + \begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix} \begin{bmatrix} B_{11} \\ B_{21} \end{bmatrix}, & \begin{bmatrix} C_{12} \\ C_{22} \end{bmatrix} + \begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix} \begin{bmatrix} B_{12} \\ B_{22} \end{bmatrix} \end{bmatrix} =$$

k

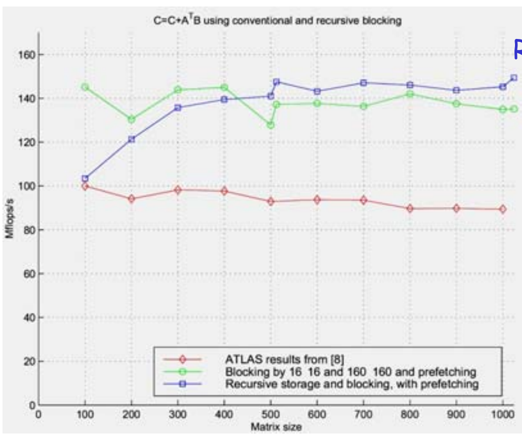
$$= \begin{bmatrix} C_{11} & C_{12} \\ C_{21} & C_{22} \end{bmatrix} + \begin{bmatrix} A_{11} \\ A_{21} \end{bmatrix} \begin{bmatrix} B_{11} & B_{12} \end{bmatrix} + \begin{bmatrix} A_{12} \\ A_{22} \end{bmatrix} \begin{bmatrix} B_{21} & B_{22} \end{bmatrix}$$

Recursive splitting - by breadth or by depth



Recursive GEMM: multi-level vs. recursive blocking

IBM PPC604, 112 MHz



RGEMM

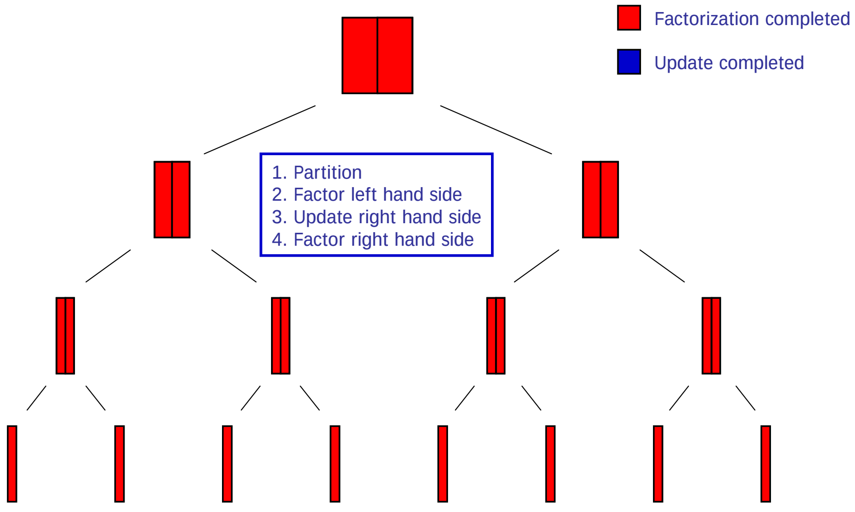
Explicit blocking

ATLAS

Case Study 2

QR factorization

Recursion template for one-sided matrix factorization



Recursive blocked QR factorization

$$\begin{pmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{pmatrix} = Q \begin{pmatrix} R_{11} & R_{12} \\ 0 & R_{22} \end{pmatrix}$$

1. Divide *A mxn* in two parts (left & right)

Stopping criteria: if n ≤ 4 use standard algorithm

2. Factorize left hand side by a *recursive* call $Q_1 \begin{pmatrix} R_{11} \\ 0 \end{pmatrix} = \begin{pmatrix} A_{11} \\ A_{21} \end{pmatrix}$

3. Update right hand side $\begin{pmatrix} R_{12} \\ \tilde{A}_{22} \end{pmatrix} \longleftarrow Q_1^T \begin{pmatrix} A_{12} \\ A_{22} \end{pmatrix}$

4. Factorize by a *recursive* call $Q_2 R_{22} = \tilde{A}_{22}$

Need to combine $Q_1 = I - Y_1 T_1 Y_1^T$ & $Q_2 = I - Y_2 T_2 Y_2^T$

Combining $Q_1 = I - Y_1 T_1 Y_1^T$ & $Q_2 = I - Y_2 T_2 Y_2^T$

Given $Q_1 = I - \tau_1 v_1 v_1^T$ and $Q_2 = I - \tau_2 v_2 v_2^T$, then

$$T = \begin{pmatrix} \tau_1 & -\tau_1 v_1^T v_2 \tau_2 \\ 0 & \tau_2 \end{pmatrix} \text{ and } Y = (v_1 \quad v_2)$$

Two elementary transformations

Given $Q_1 = I - Y_1 T_1 Y_1^T$ and $Q_2 = I - \tau_2 v_2 v_2^T$, then

$$T = \begin{pmatrix} T_1 & -T_1 Y_1^T v_2 \tau_2 \\ 0 & \tau_2 \end{pmatrix} \text{ and } Y = (Y_1 \quad v_2)$$

One block and one elementary transformation

Column by column using Level 2 operations

Given $Q_1 = I - Y_1 T_1 Y_1^T$ and $Q_2 = I - Y_2 T_2 Y_2^T$, then

$$T = \begin{pmatrix} T_1 & -T_1 Y_1^T Y_2 T_2 \\ 0 & T_2 \end{pmatrix} \text{ and } Y = (Y_1 \quad Y_2)$$

Two block transformations

Recursively, block by block using Level 3 operations

RGEQR3 - Recursive algorithm for QR factorization

[Y, R, T] = RGEQR3 A(1:m, 1:n)

if (n == 1):

Compute Householder transformation

return (u, x, t)

else

n_1 = min(n/2, nb)

let ~~n_1 = n/2~~ and $j_1 = n_1 + 1$

$[Y_1, R_1, T_1]$ = RGEQR3 A(1:m, 1:n_1) ! Recursively factor first part

A(1:m, j_1:n) ← $(I - Y_1 T Y_1^T)^T$ A(1:m, j_1:n) ! Update second part of A

$[Y_2, R_2, T_2]$ = RGEQR3 A(j_1:m, j_1:n) ! Recursively factor second part of A

$T_3 = -T_1(Y_1^T Y_2) T_2$

Let R_3 = A(1:n_1, j_1:n)

Now, $Y = (Y_1 \quad Y_2)$, $R = \begin{pmatrix} R_1 & R_3 \\ 0 & R_2 \end{pmatrix}$ and $T = \begin{pmatrix} T_1 & T_3 \\ 0 & T_2 \end{pmatrix}$

return [Y, R, T]

end

In practice, Y and R overwrite A

#flops grows cubically with # Householder transformations being aggregated (compact WY)!

Recursive blocked QR highlights

- Recursive splitting controlled by nb (splitting point = min(nb, n/2), nb = 32-64)
- Level 3 algorithm for generating $Q = I - YTY^T$ (compact WY) within the recursive blocked algorithm (T triangular of size <= nb)
- Replaces LAPACK level 2 and 3 algorithms

Recursive QR vs. LAPACK

m = n　　　　m >> n

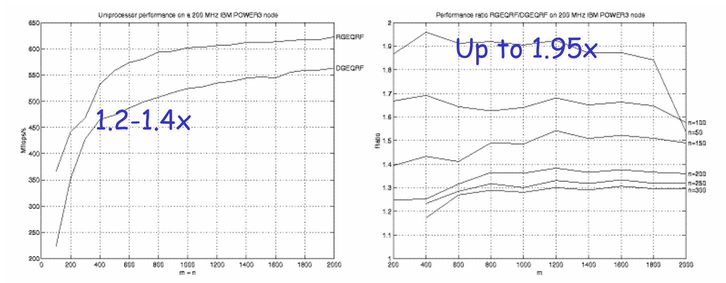
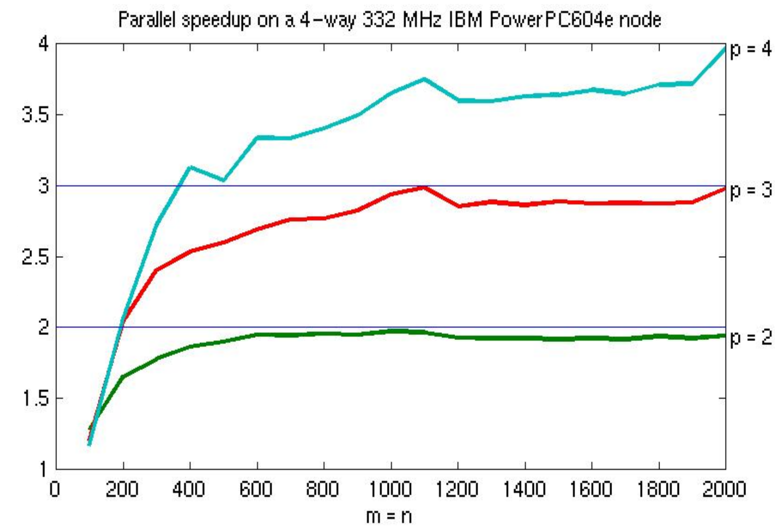


Fig. 4.1 *Performance results in Mflops/s for square matrices (left) and performance ratio for tall, thin matrices (right) for the recursive algorithm RGEQRF and DGEQRF of LAPACK on the 200 MHz IBM Power3.*

Parallel speedup - 4 processor PPC604e

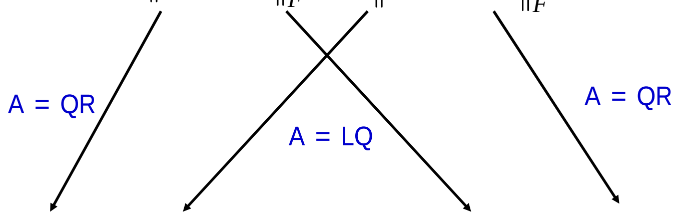


Case Study 3

Over- and under-determined linear systems

LAPACK DGELS

$$\text{Solve } \|AX - B\|_F \text{ or } \|A^T X - B\|_F$$



Least squares solution
(over-determined systems)

Minimum norm solution
(under-determined systems)

Rough outline of basic algorithms

- Factor A into QR (or LQ)
- Least squares: Apply Q^T (or Q) to B, solve triangular system
- Min. norm. soln.: Solve triangular system, apply Q (or Q^T) to solution

Least squares recursive algorithm

$X = \mathrm{RGELS}(A, B, nb)$ GEMM + TRMM + TRSM

If $n \le nb$

1. Factor $A = Q \begin{bmatrix} R \\ 0 \end{bmatrix}$; $\tilde{B} \leftarrow Q^T B$; solve $RX = \tilde{B}(1:n,:)$

else

2. Let $A = [\, A_1 \quad A_2 \,]$; $B = \begin{bmatrix} B_1 \\ B_2 \end{bmatrix}$ with nb cols in A_1, nb rows in B_1
3. Factor $A_1 = Q_1 \begin{bmatrix} R_{11} \\ 0 \end{bmatrix}$
4. Set $\begin{bmatrix} R_{12} & \tilde{B}_1 \\ A_{22} & \tilde{B}_2 \end{bmatrix} \leftarrow Q_1^T [\, A_2 \quad B \,]$ GEMM + TRMM
5. $X_2 = \mathrm{RGELS}(A_{22}, \tilde{B}_2, nb)$
6. Solve $R_{11} X_1 = \tilde{B}_1 - R_{12} X_2$; return $X = \begin{bmatrix} X_1 \\ X_2 \end{bmatrix}$ GEMM + TRSM

endif

Fig. 4.2 *Recursive least squares RGELS algorithm for computing the solution to $AX = B$, where A is $m \times n$ ($m \ge n$).*

Factorization, update and triangular solve are interleaved for each block => data reuse

Minimum norm solution $\left\| A^T X - B \right\|_F$

- Similar-style algorithm
- Basic steps (preformed recursively):
  - QR factorization of block columns of A
  - Solve triangular systems
  - Apply Q to solution

RGELS - Remaining Cases

- In LAPACK DGELS, A = LQ is computed for
  - least squares solution - transposed case
  - minimum norm solution - non-transposed case
- Each Householder transformation is computed on a row of A, i.e., working on elements stored with stride = LDA
- RGELS performs explicit transposition $C = A^T$ and solves $||CX - B||$ or $||C^TX - B||$ using one of the two algorithms already presented

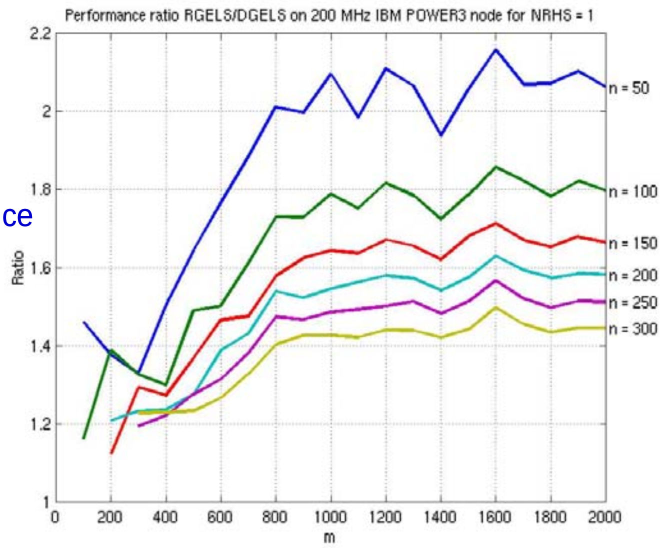
— Transposition requires additional storage to be allocated and extra operations

+ Additional performance improvements by roughly a factor of 2 AND

+ Reduces amount of code by roughly a factor of 2

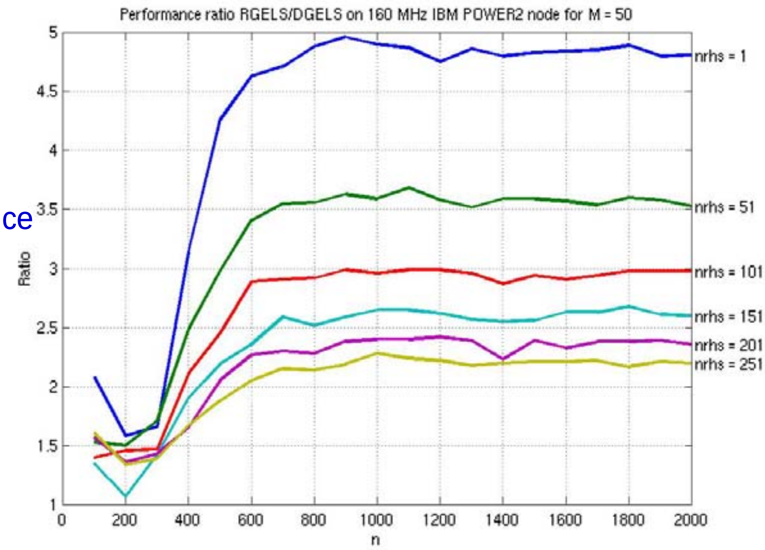
RGELS - LSQ: NRHS = 1 $\|AX - B\|_F$

Performance ratio



RGELS - LSQ: transposed, M = 50 $\|A^T X - B\|_F$

Performance ratio



Case Study 4

Triangular matrix equations and condition estimation

Matrix equations

Name	Matrix equation	Acronym
Standard Sylvester (CT)	$AX - XB = C$	SYCT
Standard Lyapunov (CT)	$AX + XA^T = C$	LYCT
Generalized coupled Sylvester	$(AX - YB, DX - YE) = (C, F)$	GCSY
Standard Sylvester (DT)	$AXB^T - X = C$	SYDT
Standard Lyapunov (DT)	$AXA^T - X = C$	LYDT
Generalized Sylvester	$AXB^T - CXD^T = E$	GSYL
Generalized Lyapunov (CT)	$AXE^T + EXA^T = C$	GLYCT
Generalized Lyapunov (DT)	$AXA^T - EXE^T = C$	GLYDT

One-sided (top) and two-sided (bottom)

Recursive blocked SYCT template

Case 1: 1 <= n <= m/2

A (m × m), B (n × n) upper tri.

$$\left[\begin{array}{c|c} A_{11} & A_{12} \\ \hline & A_{22} \end{array}\right] \left[\begin{array}{cc} X_{11} & X_{12} \\ \hline X_{21} & X_{22} \end{array}\right] - \left[\begin{array}{cc} X_{11} & X_{12} \\ \hline X_{21} & X_{22} \end{array}\right] \left[\begin{array}{cc} B_{11} & B_{12} \\ & B_{22} \end{array}\right] = \left[\begin{array}{cc} C_{11} & C_{12} \\ \hline C_{21} & C_{22} \end{array}\right]$$

Case 2: 1 <= m <= n/2

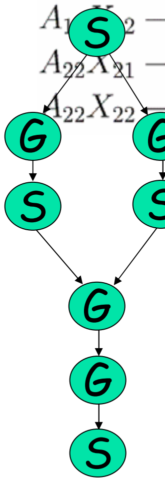
$$\left[\begin{array}{cc} A_{11} & A_{12} \\ & A_{22} \end{array}\right] \left[\begin{array}{c|c} X_{11} & X_{12} \\ X_{21} & X_{22} \end{array}\right] - \left[\begin{array}{c|c} X_{11} & X_{12} \\ X_{21} & X_{22} \end{array}\right] \left[\begin{array}{c|c} B_{11} & B_{12} \\ & B_{22} \end{array}\right] = \left[\begin{array}{c|c} C_{11} & C_{12} \\ C_{21} & C_{22} \end{array}\right]$$

Case 3: n/2 < m < 2n

$$\left[\begin{array}{c|c} A_{11} & A_{12} \\ \hline & A_{22} \end{array}\right] \left[\begin{array}{c|c} X_{11} & X_{12} \\ \hline X_{21} & X_{22} \end{array}\right] - \left[\begin{array}{c|c} X_{11} & X_{12} \\ \hline X_{21} & X_{22} \end{array}\right] \left[\begin{array}{c|c} B_{11} & B_{12} \\ \hline & B_{22} \end{array}\right] = \left[\begin{array}{c|c} C_{11} & C_{12} \\ \hline C_{21} & C_{22} \end{array}\right]$$

Recursive SYCT - Case 3

$$\begin{aligned} A_{11}X_{11} - X_{11}B_{11} &= C_{11} - A_{12}X_{21} \\ A_{11}X_{12} - X_{12}B_{22} &= C_{12} - A_{12}X_{22} + X_{11}B_{12} \\ A_{22}X_{21} - X_{21}B_{11} &= C_{21} \\ A_{22}X_{22} - X_{22}B_{22} &= C_{22} + X_{21}B_{12} \end{aligned}$$



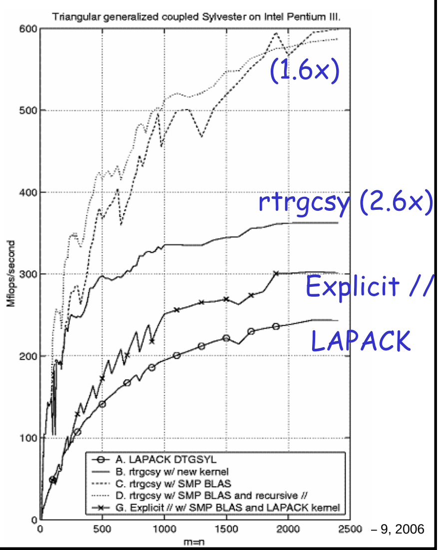
1. SYLV('N', 'N', A_{22}, B_{11}, C_{21})
2a. GEMM('N', 'N', $\alpha = +1$, C_{21}, B_{12}, C_{22})
2b. GEMM('N', 'N', $\alpha = -1$, A_{12}, C_{21}, C_{11})
3a. SYLV('N', 'N', A_{22}, B_{22}, C_{22})
3b. SYLV('N', 'N', A_{11}, B_{11}, C_{11})
4. GEMM('N', 'N', $\alpha = -1$, A_{12}, C_{22}, C_{12})
5. GEMM('N', 'N', $\alpha = +1$, C_{11}, B_{12}, C_{12})
6. SYLV('N', 'N', A_{11}, B_{22}, C_{12})

Triangular generalized coupled Sylvester equation - GCSY

AX - YB = C
DX - YE = F

(A, D) and (B, E) in generalized Schur form

Solution (X, Y) over-writes r.h.s. (C, F)



RECSY library

- Recursive blocked algorithms for solving reduced matrix equations
- Recursion implemented in F90
- SMP versions using OpenMP
- F77 wrappers for LAPACK and SLICOT routines
- www.cs.umu.se/research/parallel/recsy/

Recursive blocking ...

- creates new algorithms for linear algebra software
- expresses dense linear algebra algorithms entirely in terms of level-3 BLAS like matrix-matrix operations
- introduces an automatic variable blocking that targets multiple levels of a deep memory hierarchy
- can also be used to define hybrid data formats for storing block-partitioned matrices (general, triangular, symmetric, packed) - L1, L2 and TLB misses are minimized for certain block sizes (Park-Hong-Prasanna´03)

High-performance software

- Make use of data locality and superscalar optimization techniques
  - Recursive blocked algorithms improve on the temporal data locality
  - Hybrid data formats improve on the spatial data locality
  - Portable and generic superscalar kernels ensure that all functional units on the processor(s) are used efficiently

Acknowledgements

- Fred Gustavson, Isak Jonsson, Bo Kågström (co-authors and co-workers)
- André Henriksson, Olov Gustavsson and Andreas Lindkvist (earlier MSc students)
- Bjarne Andersén, Jerzy Wasniewski (e.g., packed Cholesky)
- Robert Granat (PhD student)
- HPC and LA team at Umeå University
- Community that do related and complementary work!