Management of Deep Memory Hierarchies – Recursive Blocking and Hybrid Data Structures for Dense Matrix Computations

Bo Kågström
Dept of Computing Science & HPC2N
Umeå University, Sweden

5th Workshop on Linux Clusters for Supercomputing, October18-19, 2004, Linköping University

High Performance Computing Center North (HPC2N)

VETENSKAPSRÅDET
THE SWEDISH RESEARCH COUNCIL

Swedish Foundation for Strategic Research

Bo Kågström 2004

HPC2N – "HPC to North"

- National center for Scientific and Parallel Computing

Sarek- new super cluster (HP, installed 2004-06-07):
- 384 proc (64 bit, AMD Opteron 2.2 GHz)
- 1.5 TB memory (8 GB per node)
- Myrinet-2000 (HPI with 250 MB/s)
- > 1.3 Tflops/s HP-Linpack (~79% of peak)
- Most powerful computer in Sweden
- Funded by the Wallenberg Foundation (KAW)

- Funded by the Swedish Research Council and its meta-center SNIC

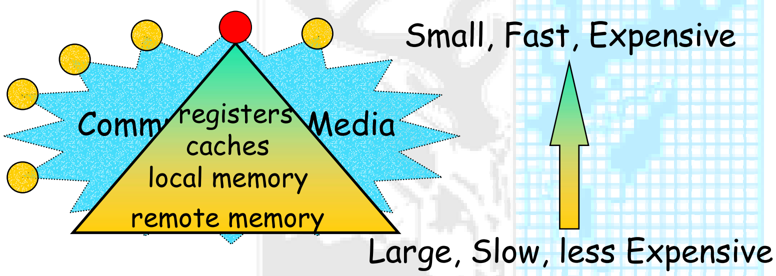
Bo Kågström 2004

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
- Continuing demand for new and improved algorithms and software along with the computer evolution

"Data transport" in memory hierarchies

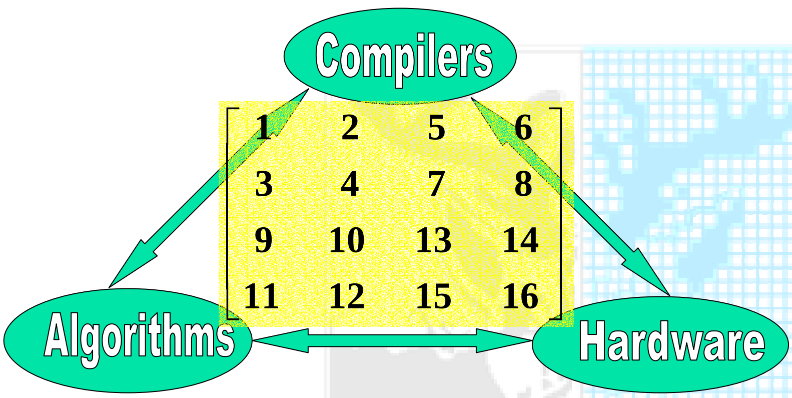
- of today's computer systems
  - PC - cluster - supercomputer



Management of deep memory hierarchies

- Architecture evolution: HPC systems with multiple SMP nodes, several levels of caches, more functional units per CPU
- Key to performance: understand the algorithm and architecture interaction
- Hierarchical blocking

The fundamental AHC triangle



Outline

- Hierarchical blocking: motivation and implications
- Recursive blocked templates and algorithms
- Recursive blocked data structures
- Case studies:
  - General matrix multiply and add (GEMM)
  - Packed Cholesky factorization
  - QR factorization and linear systems
  - Triangular matrix equations and condition estimation
- Some related and complementary work
- Work in progress: periodic matrix equations
- Concluding remarks

SIAM REVIEW
Vol. 46, No. 1, pp. 3–45

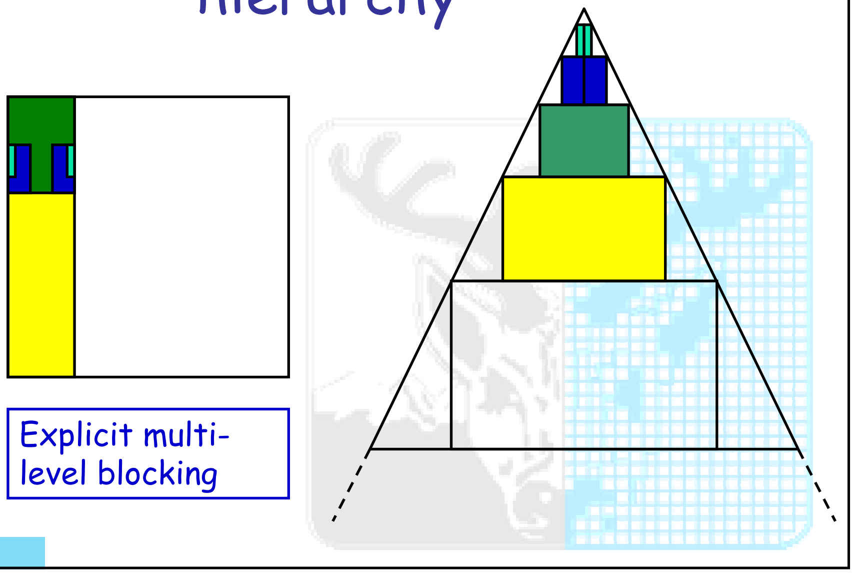
Recursive Blocked Algorithms and Hybrid Data Structures for Dense Matrix Library Software*

Joint work with:

Erik Elmroth†
Fred Gustavson‡
Isak Jonsson†
Bo Kågström†

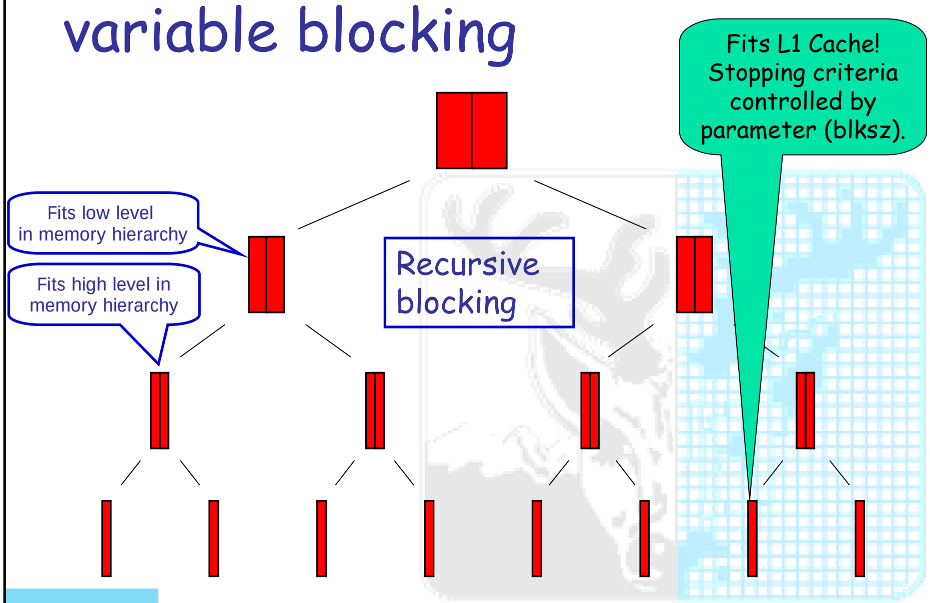
Abstract. Matrix computations are both fundamental and ubiquitous in computational science and its vast application areas. Along with the development of more advanced computer systems with complex memory hierarchies, there is a continuing demand for new algorithms and library software that efficiently utilize and adapt to new architecture features. This article reviews and details some of the recent advances made by applying the paradigm of recursion to dense matrix computations on today's memory-tiered computer systems. Recursion

Blocking for a memory hierarchy

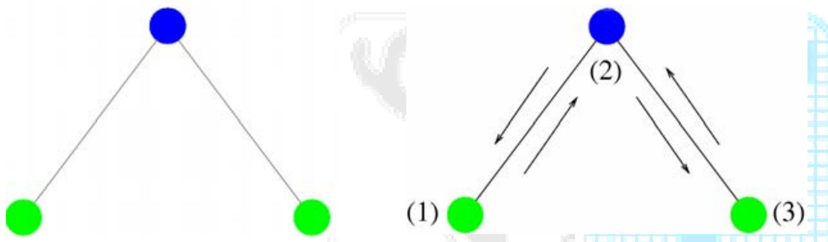


Explicit multi-level blocking

Recursion leads to automatic variable blocking

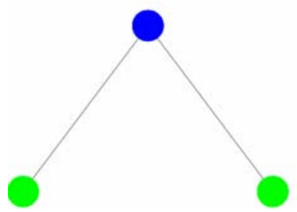


Splittings defining independent and dependent tasks



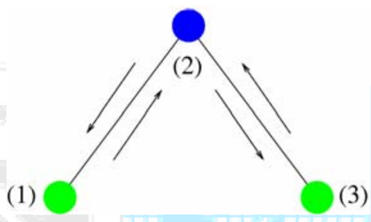
Critical path of subtasks:
(1), (2), (3)

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

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

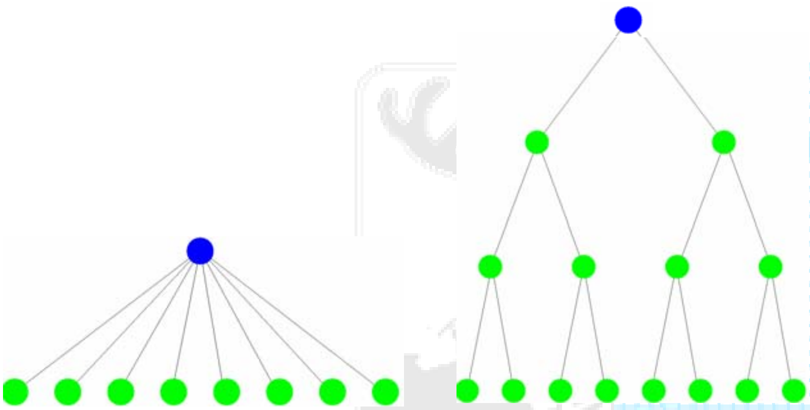
n

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

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



GEMM recursive blocked template - splitting by depth

When to end the recursive splitting?

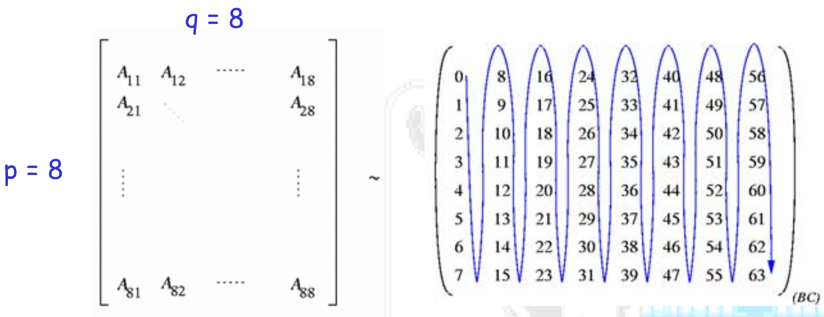
```
C = rgemm( A, B, C, blksz)
If m, n, and k <= blksz
   C = opt_gemm( A, B, C) % optimized GEMM kernel!

elseif m = max(m, n, k) % split m: m2 = m/2
   C(1:m2, :)     = rgemm( A(1:m2, :), B, C(1:m2, :), blksz)
   C(m2+1:m, :) = rgemm( A(m2+1:m, :), B, C(m2+1:m, :), blksz)

elseif n = max(n,k)      % split n: n2 = n/2, k
   C(:,1:n2)       = rgemm( A, B(:,1:n2), C(:,1:n2), blksz)
   C(:,n2+1:n)    = rgemm( A, B(:,n2+1:n), C(:,n2+1:n), blksz)

else                              % split k: k2 = k/2,
   C  = rgemm(A(:,1:n2), B(1:m2, :), C, blksz)
   C  = rgemm(A(:,n2+1:n), B(m2+1:m, :), C, blksz)
```

Locality of reference

- Recursive blocked algorithms mainly improve on the temporal locality
- Further performance improvements by matching the data structure with the algorithm (and vice versa)
- Recursive blocked data structures improve on the spatial locality

Bo Kågström 2004

Blocked data formats



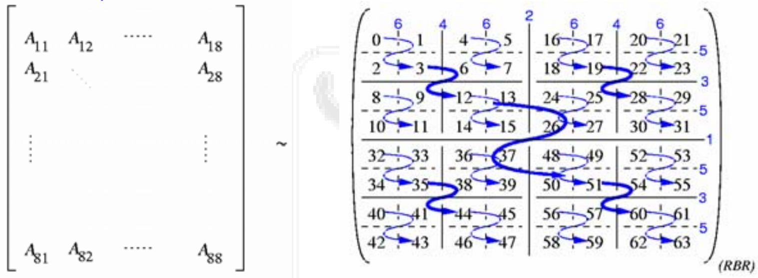
Blocks A_{ij} of size mb x nb can be ordered in (pq)! different ways

Bo Kågström 2004

Recursive blocked row format

q = 8

p = 8



Recursive ordering: a 1-dim tour through a 2-dim object (Hilbert space filling heuristics)

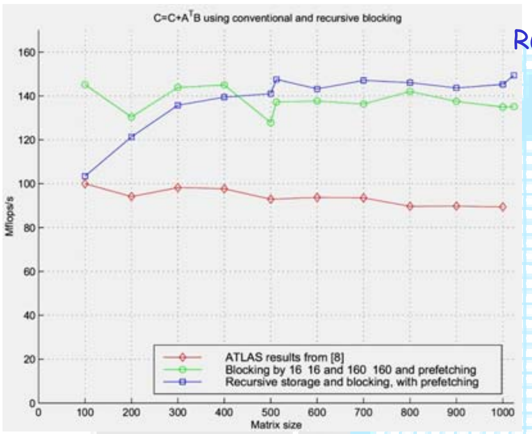
RBR <-> Z-Morton ordering

Recursive GEMM: multi-level vs. recursive blocking

IBM PPC604, 112 MHz

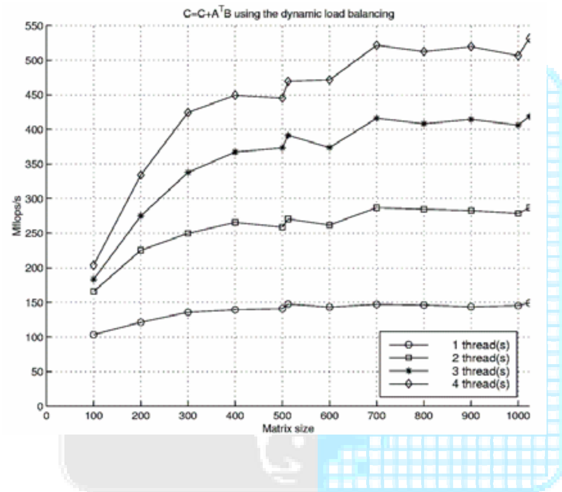
RGEMM

ATLAS

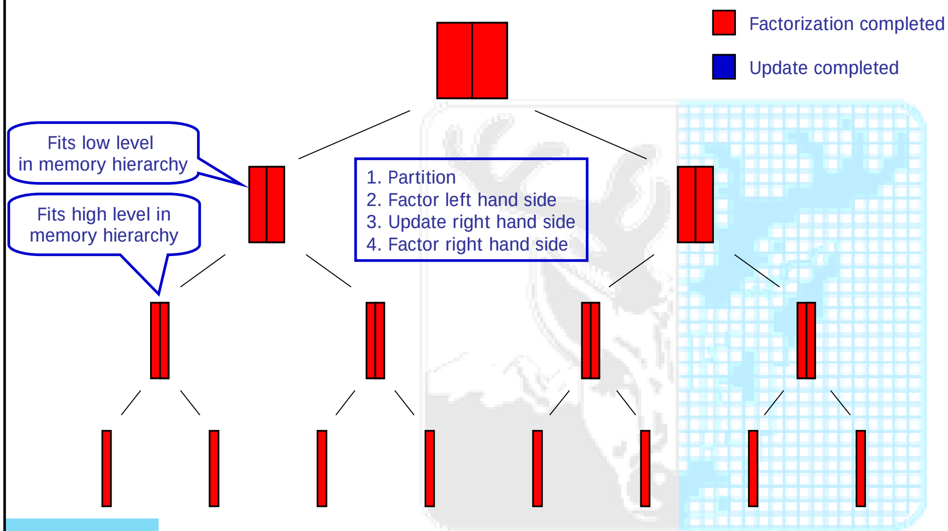


Recursive blocked GEMM and SMP parallelism via threads

IBM PPC604, 4 proc



Recursion template for one-sided matrix factorization



Case Study 2

Cholesky factorization for matrices in packed format

Packed Cholesky factorization

$$A \equiv \begin{bmatrix} A_{11} & A_{21}^T \\ A_{21} & A_{22} \end{bmatrix} = LL^T \equiv \begin{bmatrix} L_{11} & 0 \\ L_{21} & L_{22} \end{bmatrix} \begin{bmatrix} L_{11}^T & L_{21}^T \\ 0 & L_{22} \end{bmatrix}$$

Standard approach (typified by LAPACK):

- Packed storage -> cannot use standard level 3 BLAS (e.g., DGEMM)
- Possible to produce packed level 3 BLAS routines at a great programming cost
- Run at level 2 performance, i.e., much below full storage routines.
- Minimum storage: 1/2n(n+1) elements

Packed recursive blocked data

1	2	4	7	11	16	22
	3	5	8	12	17	23
		6	9	13	18	24
			10	14	19	25
				15	20	26
					21	27
						28

Packed upper

1	2	3	7	10	13	16
	4	5	8	11	14	17
		6	9	12	15	18
			19	20	22	24
				21	23	25
					26	27
						28

Packed recursive upper

FIG. 3.1. *Memory indices for 7 × 7 upper triangular matrix stored in traditional packed format and recursive packed format.*

- Divide into two isosceles triangles T1, T2 and rectangle R
- Divide triangles recursively down to element level
- Store in order: T1, R, T2
- Rectangles stored in full format ➔
  Possible to use full storage level 3 BLAS

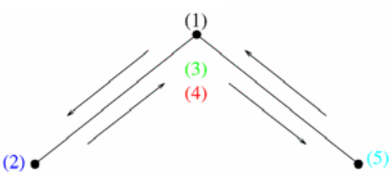
Cholesky recursive blocked template

$$A = \begin{pmatrix} A_{11} & A_{21}^T \\ A_{21} & A_{22} \end{pmatrix} = LL^T = \begin{pmatrix} L_{11} & 0 \\ L_{21} & L_{22} \end{pmatrix} \begin{pmatrix} L_{11}^T & L_{21}^T \\ 0 & L_{22} \end{pmatrix} \quad (1)$$

$$\text{Factor}: \ A_{11} = L_{11}L_{11}^T. \quad (2)$$

$$\text{TRSM}: \ L_{21}L_{11}^T = A_{21}. \quad (3)$$

$$\text{SYRK}: \ \tilde{A}_{22} = A_{22} - L_{21}L_{21}^T \quad (4)$$



$$\text{Factor}: \ \tilde{A}_{22} = L_{22}L_{22}^T \quad (5)$$

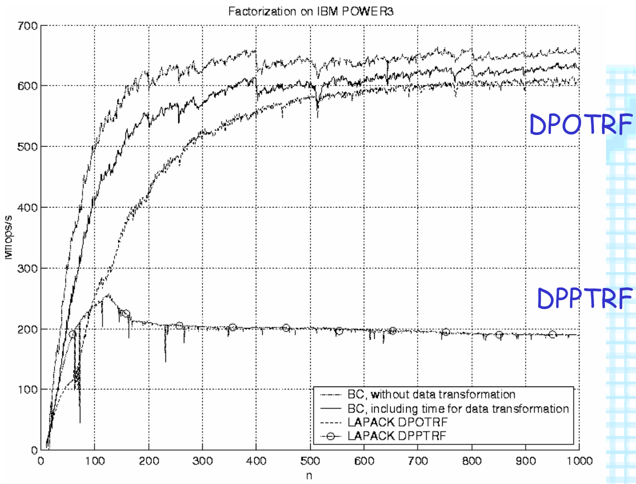
Packed recursive blocked Cholesky highlights

- Recursive blocked algorithm + recursive packed data layout => can make use of high performance level 3 BLAS routines (e.g., DGEMM)
- Use minimal storage for matrix A
- Temporary workspace = $1/8n^2$ elements (~25%)
- Leaf problems (< blksz) are solved using superscalar kernels (Cholesky, TRSM, SYRK)

Bo Kågström 2004

Recursive blocked Cholesky vs. LAPACK – (rec.) packed format

Runs at level 3 performance – at least!



Bo Kågström 2004

Case Study 3

QR factorization and linear systems

Recursive blocked QR factorization

$$\left(\begin{array}{c|c} A_{11} & A_{12} \\ A_{21} & A_{22} \end{array}\right) = Q \begin{pmatrix} R_{11} & R_{12} \\ 0 & R_{22} \end{pmatrix}$$

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

Stopping criteria: if n < 4 use standard algorithm

2. Factorize left hand side by a

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

$$Q_1 \begin{pmatrix} R_{11} \\ 0 \end{pmatrix} = \begin{pmatrix} A_{11} \\ A_{21} \end{pmatrix}$$

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

$$Q_2 R_{22} = \tilde{A}_{22}$$

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

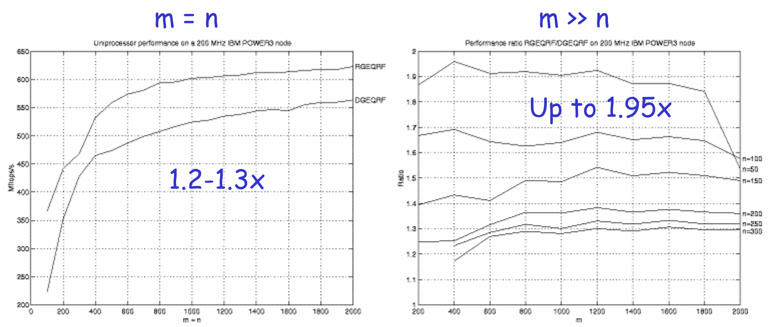


Fig. 4.1 *Performance results in Mflops/s for square matrices (left) and performance ratio for tall, thin matrices (right) for the recursive algorithm RGEQRF and DGEQRF of LAPACK on the 200 MHz IBM Power3.*

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)

Separation of two matrices

$$\text{Sep}[A,B] = \inf_{\|X\|_F=1} \|AX - XB\|_F = \sigma_{\min}(Z),$$
$$\text{where} \quad Z = I_n \otimes A - B^T \otimes I_m.$$

Computing Sep[A,B] costs $O(m^3n^3)$ – impractical!

Reliable Sep-estimates of cost $O(m^2n + mn^2)$:

$$\frac{\|x\|_2}{\|y\|_2} = \frac{\|X\|_F}{\|C\|_F} \le \|Z^{-1}\|_2 = \frac{1}{\sigma_{\min}(Z)} = \text{Sep}^{-1},$$

$$(mn)^{-1/2}\|Z^{-1}\|_1 \le \|Z^{-1}\|_2 \le \sqrt{mn}\|Z^{-1}\|_1.$$

Matrix equation Sep-functions

Z-matrix	Sep-function $= \sigma_{\min}(Z\_)$
$Z_{\text{SYCT}} = I_n \otimes A - B^T \otimes I_m$	$\inf_{\|X\|_F=1} \|AX - XB\|_F$
$Z_{\text{LYCT}} = I_n \otimes A + A \otimes I_n$	$\inf_{\|X\|_F=1} \|AX - X(-A^T)\|_F$
$Z_{\text{GCSY}} = \begin{bmatrix} I_n \otimes A & -B^T \otimes I_m \\ I_n \otimes D & -E^T \otimes I_m \end{bmatrix}$	$\inf_{\|(X,Y)\|_F=1} \|(AX - YB, DX - YE)\|_F$
$Z_{\text{SYDT}} = B \otimes A - I_n \otimes I_m$	$\inf_{\|X\|_F=1} \|AXB^T - X\|_F$
$Z_{\text{LYDT}} = A \otimes A - I_n \otimes I_n$	$\inf_{\|X\|_F=1} \|AXA^T - X\|_F$
$Z_{\text{GSYL}} = B \otimes A - D \otimes C$	$\inf_{\|X\|_F=1} \|AXB^T - CXD^T\|_F$
$Z_{\text{GLYCT}} = E \otimes A + A \otimes E$	$\inf_{\|X\|_F=1} \|AXE^T - EX(-A^T)\|_F$
$Z_{\text{GLYDT}} = A \otimes A - E \otimes E$	$\inf_{\|X\|_F=1} \|AXA^T - EXE^T\|_F$

$Z\,x = b$, Z is a Kronecker product representation

Sep-function = smallest singular value of Z

Recursive SYCT - Case 3

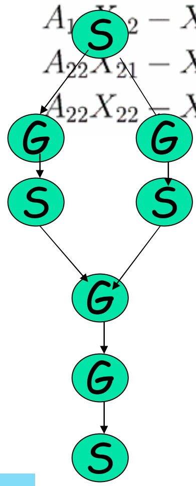
$$\left[\begin{array}{c|c} A_{11} & A_{12} \\ \hline & A_{22} \end{array}\right]\left[\begin{array}{c|c} X_{11} & X_{12} \\ \hline X_{21} & X_{22} \end{array}\right] - \left[\begin{array}{c|c} X_{11} & X_{12} \\ \hline X_{21} & X_{22} \end{array}\right]\left[\begin{array}{c|c} B_{11} & B_{12} \\ \hline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

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

Bo Kågström 2004

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})

Bo Kågström 2004

SYCT and matrix functions

- A triangular => $F := f(A)$ triangular
- f analytic => exists series expansion => $A F - F A = 0$
- recursive template:

$$\begin{bmatrix} A_{11} & A_{12} \\ & A_{22} \end{bmatrix} \begin{bmatrix} F_{11} & F_{12} \\ & F_{22} \end{bmatrix} - \begin{bmatrix} F_{11} & F_{12} \\ & F_{22} \end{bmatrix} \begin{bmatrix} A_{11} & A_{12} \\ & A_{22} \end{bmatrix} = \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix}$$

$$\Longrightarrow$$

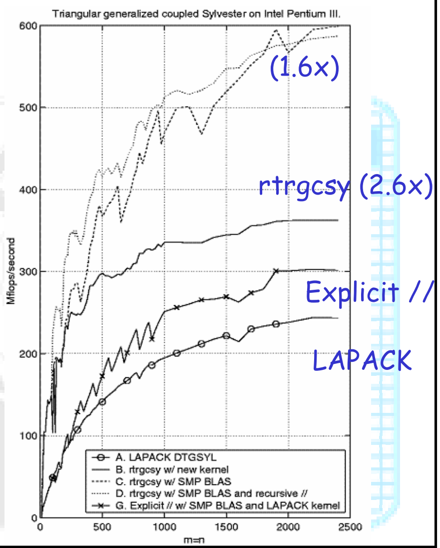
$$\begin{aligned} A_{11}F_{11} - F_{11}A_{11} &= 0 \\ A_{11}F_{12} - F_{12}A_{22} &= F_{11}A_{12} - A_{12}F_{22}, \\ A_{22}F_{22} - F_{22}A_{22} &= 0 \end{aligned}$$

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)



Two-sided matrix equation: GLYDT

- $AXA^T - EXE^T = C$
- $C = C^T$ nxn; (A, E) n x n in gen. Schur form
- Unique sol'n $X = X^T$ ⇔
  λ_i of $A - \lambda E$ satisfy $\lambda_i \lambda_j \neq 1$
- Recursive splitting:

$$\begin{bmatrix} A_{11} & A_{12} \\ & A_{22} \end{bmatrix} \begin{bmatrix} X_{11} & X_{12} \\ X_{21} & X_{22} \end{bmatrix} \begin{bmatrix} A_{11}^T & \\ A_{12}^T & A_{22}^T \end{bmatrix} -$$

$$\begin{bmatrix} E_{11} & E_{12} \\ & E_{22} \end{bmatrix} \begin{bmatrix} X_{11} & X_{12} \\ X_{21} & X_{22} \end{bmatrix} \begin{bmatrix} E_{11}^T & \\ E_{12}^T & E_{22}^T \end{bmatrix} = \begin{bmatrix} C_{11} & C_{12} \\ C_{21} & C_{22} \end{bmatrix}$$

Bo Kågström 2004

Two-sided matrix product

$$C = \beta C + \alpha \mathrm{op}(A) X \mathrm{op}(B)^T$$

- A and/or B can be dense or triangular
- One or several of A, B and C can be symmetric
- Extra workspace – size of r.h.s.

Make use of symmetry, e.g., in GLYDT:

$$C_{11} = C_{11} - A_{12} X_{22} A_{12}^T \quad \text{and} \quad C_{11} = C_{11} + E_{12} X_{22} E_{12}^T$$

Bo Kågström 2004

GLYDT performance with optional condition estimation

Table 5.3 *Timings for solving unreduced two-sided matrix equations (GLYDT) with optional condition estimation. (Job = X, compute solution only; Job = X + Sep, compute solution and Sep-estimation.) Results from 375 MHz IBM Power3.*

(a)

n	SG03AD using SG03AX Total time	Solver part	SG03AD using *rtrglydt* Total time	Solver part	Speedup	Job
50	0.0277	49.9 %	0.0185	20.1 %	1.50	X
100	0.180	51.2 %	0.0967	9.0 %	1.86	X
250	2.89	46.8 %	1.62	4.7 %	1.79	X
500	59.0	42.3 %	34.5	1.5 %	1.71	X
750	303.4	42.0 %	177.5	0.9 %	1.71	X
1000	646.6	44.6 %	361.8	1.0 %	1.79	X
50	0.117	87.6 %	0.0263	45.6 %	4.44	X+Sep
100	0.709	87.3 %	0.152	40.6 %	4.68	X+Sep
250	9.98	84.5 %	2.08	25.4 %	4.81	X+Sep
500	178.6	80.9 %	37.8	9.4 %	4.73	X+Sep
750	924.1	80.9 %	184.4	4.5 %	5.01	X+Sep
1000	2076.6	82.7 %	391.8	8.4 %	5.30	X+Sep

Up to 1.9x

Up to 5.3x

Bo Kågström 2004

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/
- Part of Isak Jonsson's PhD Thesis, Dec 2003

Bo Kågström 2004

SCASY library

- ScaLAPACK-style software package of matrix equation solvers for distributed memory machines.
- Triangular solvers are used in implementing parallel condition estimators for each matrix equation.
- With Robert Granat, PhD student

$op(A)X \pm Xop(B) = C$	SYCT	√
$op(A)X + Xop(A^T) = C$	LYCT	√
$op(A)Xop(B) \pm X = C$	SYDT	√
$op(A)Xop(A^T) - X = C$	LYDT	√
$op(A)X \pm Yop(B) = C,$ $op(D)X \pm Yop(E) = F$	GCSY	
$op(A)Xop(B) \pm op(D)Xop(E) = C$	GSYL	
$op(A)Xop(A^T) - op(E)Xop(E^T) = C$	GLYCT	
$op(A)X(E^T) + op(E)Xop(A^T) = C$	GLYDT	

Discrete-time periodic systems

$$\begin{aligned} x_{k+1} &= A_k x_k + B_k u_k \\ y_k &= C_k x_k + D_k u_k \end{aligned}$$

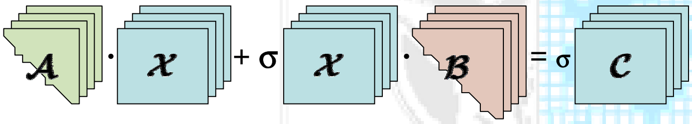
$$\left.\begin{array}{ll} A_k \in \mathrm{R}^{n_{k+1} \times n_k}, & B_k \in \mathrm{R}^{n_{k+1} \times m} \\ C_k \in \mathrm{R}^{p \times n_k}, & D_k \in \mathrm{R}^{p \times m} \end{array}\right\} - K\text{-periodic}$$

$$(A_{k+K} = A_k, B_{k+K} = B_k, \ldots)$$

From discretization of continuous-time periodic models: revolving satellite; helicopter in forward flight; multi-rate sampled control systems → time-varying dimensions

Periodic Sylvester equation

- A(:,:,k) X(:,:,k) + X(:,:,k+1) B(:,:,k) = C(:,:, k+1), for k = 1 : K -1
- A(:,:,K) X(:,:,K) + X(:,:,1) B(:,:,K) = C(:,:, 1)



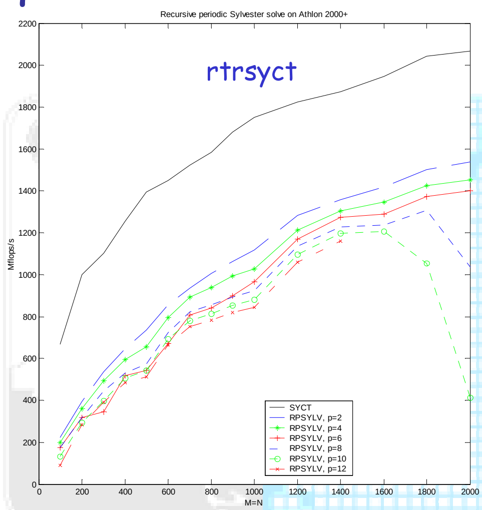
Script notations: X_k K-periodic

$$\mathcal{X}_k := \mathrm{diag}\,(X_k, X_{k+1}, \ldots, X_{k+K-1})$$

$$\sigma\mathcal{X}_k := \mathrm{diag}\,(X_{k+1}, \ldots, X_{k+K-1}, X_k)$$

Recursive blocked periodic Sylvester equation

Some preliminary performance results



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 every level 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-Prosana´03)

High-performance software

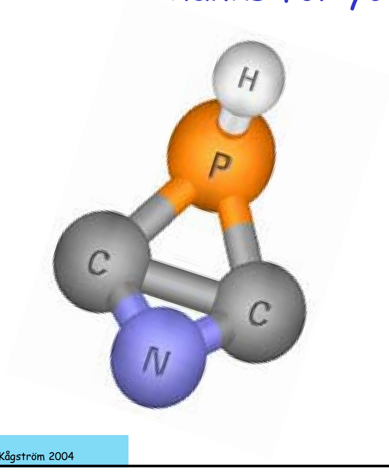
- implementations are based on data locality and superscalar optimization techniques
- recursive blocked algorithms improve on the temporal data locality
- hybrid data formats improve on the spatial data locality
- portable and generic superscalar kernels ensure that all functional units on the processor(s) are used efficiently

Acknowledgements

- Erik Elmroth, Isak Jonsson, Fred Gustavson (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! (see SIAM Rev. 2004)

Bo Kågström 2004

- Thanks for your attention!



Bo Kågström 2004

Some related and complementary work

- Recursive algorithms and hybrid data structures
  - Winograd-Strassen'69: Douglas etal'94, ESSL, Demmel-Higham'92 (stability)
  - Quad- and octtrees: Samet'84, Salman-Warner'94 (N-body, Barnes-Hut'84)
  - Cache oblivious algorithms: Leiserson etal'99 (sorting, FFT, A^T)
  - GEMM: Chatterjee etal'02, Valsalam and Skjellum'02, ATLAS-project
  - LU: Toledo'97(dense), Dongarra, Eijkhout Luszczek'01 (sparse)
  - QR: Rabani and Toledo'01 (out-of-core), Frens and Wise'03 (Givens-based)

Some related and complementary work

- Automated generation of library software and compiler technology
  - Empirical optimization:
    PHiPAC – Bilmes, Demmel etal'97,
    ATLAS – Whaley, Petitet and Dongarra'00,
    Sparse kernels – Vuduc, Demmel et al'03
  - FLAME: Gunnels, Goto, Van de Geijn etal'01, '02
  - Compiler blockability: Wolf and Lam'91 (loop transformations), Carr and Lehoucq'97
  - Automatic generation of recursive codes: Ahmed and Pingali'00 (iterative algorithms -> recursive), Yi, Adve and Kennedy'00 (convert loop nests into recursive form)

Bo Kågström 2004