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

---

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

Small, Fast, Expensive

Large, Slow, less Expensive

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

Erik Elmroth, PMAA 2006, Rennes, September 7 – 9, 2006
LAPACK-style LU factorization

Factor fixed size block column

\[
\begin{align*}
\begin{bmatrix}
L_{11} & U_{12} \\
B_{21} & A_{22}
\end{bmatrix} & = \begin{bmatrix}
A_{11} \\
\end{bmatrix}
\end{align*}
\]

Repeat for updated matrix

Recursion template for one-sided matrix factorization

1. Partition
2. Factor left hand side
3. Update right hand side
4. Factor right hand side

Fits low level in memory hierarchy
Fits high level in memory hierarchy

Erik Elmroth, PMAA 2006, Rennes, September 7 – 9, 2006
Splitting defining independent and dependent tasks

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

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

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

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

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

Erik Elmroth, PMAA 2006, Rennes, September 7 – 9, 2006
Case Study 1

General matrix multiply and add (GEMM)

Recursive splittings for GEMM:

\[ C \leftarrow \beta \text{op}(C) + \alpha \text{op}(A)\text{op}(B) \]

<table>
<thead>
<tr>
<th>Split</th>
<th>m x n</th>
<th>m x k</th>
<th>k x n</th>
</tr>
</thead>
<tbody>
<tr>
<td>m, n, k</td>
<td>[ \begin{bmatrix} C_{11} &amp; C_{12} \ C_{21} &amp; C_{22} \end{bmatrix} + \begin{bmatrix} A_{11} &amp; A_{12} \ A_{21} &amp; A_{22} \end{bmatrix} \begin{bmatrix} B_{11} &amp; B_{12} \ B_{21} &amp; B_{22} \end{bmatrix} = ]</td>
<td>[ \begin{bmatrix} B_{11} &amp; B_{12} \ B_{21} &amp; B_{22} \end{bmatrix} ]</td>
<td>[ \begin{bmatrix} B_{11} &amp; B_{12} \ B_{21} &amp; B_{22} \end{bmatrix} ]</td>
</tr>
</tbody>
</table>
Recursive splitting - by breadth or by depth

Recursive GEMM: multi-level vs. recursive blocking

IBM PPC604, 112 MHz
Case Study 2
QR factorization

Recursion template for one-sided matrix factorization

1. Partition
2. Factor left hand side
3. Update right hand side
4. Factor right hand side

Factorization completed
Update completed
**Recursive blocked QR factorization**

1. Divide $A_{mxn}$ in two parts (left & right)

2. Factorize left hand side by a recursive call

3. Update right hand side

4. Factorize by a recursive call

**Stopping criteria:**
if $n \leq 4$ use standard algorithm

---

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

- Given $Q_1 = I - T_1 Y_1 v_1^T$ and $Q_2 = I - T_2 Y_2 v_2^T$, then
  \[ T = \begin{pmatrix} T_1 & -T_1 Y_1 v_2 \\ 0 & T_2 \end{pmatrix} \text{ and } Y = (v_1, v_2) \]

- Given $Q_1 = I - Y_1 T_1 Y_1^T$ and $Q_2 = I - T_2 Y_2 v_1^T$, then
  \[ T = \begin{pmatrix} T_1 & -T_1 Y_1 v_2 \\ 0 & T_2 \end{pmatrix} \text{ and } Y = (Y_1, v_2) \]

- Given $Q_1 = I - T_1 Y_1 Y_1^T$ and $Q_2 = I - Y_2 T_2 Y_2^T$, then
  \[ T = \begin{pmatrix} T_1 & -T_1 Y_1 Y_2 \\ 0 & T_2 \end{pmatrix} \text{ and } Y = (Y_1, Y_2) \]

---

- Two elementary transformations
- One block and one elementary transformation
- Two block transformations
- Column by column using Level 2 operations
- Recursively, block by block using Level 3 operations

---

Erik Elmroth, PMAA 2006, Rennes, September 7 – 9, 2006
RGEQR3 - Recursive algorithm for QR factorization

\[ \{Y, R, T\} = \text{RGEQR3} \ A(1:m, 1:n) \]

if \( n == 1 \):
    \begin{align*}
    \text{Compute Householder transformation } \ Q = I - t \ u \ u^T, \text{ such that } \ Q^T A = (x, 0)^T \\
    \text{return } (u, x, t)
    \end{align*}
else
    \begin{align*}
    n_1 &= \min(n/2, nb) \\
    & \text{let } n_1 = n/2 \text{ and } j_1 = n_1 + 1 \\
    \{Y_1, R_1, T_1\} &= \text{RGEQR3} \ A(1:m, 1:n_1) \quad \text{! Recursively first part} \\
    A(1:m, j_1:n) &\leftarrow (I - Y_1 T Y_1^T) T A(1:m, j_1:n) \quad \text{! Update second part of A} \\
    \{Y_2, R_2, T_2\} &= \text{RGEQR3} \ A(j_1:m, j_1:n) \quad \text{! Recursively second part of A} \\
    T_3 &= -T_1 Y_1^T Y_2^T \\
    \text{Let } R_3 &= A(1:n_1, j_1:n) \\
    \text{Now, } \quad \begin{bmatrix}
    \cdot & \cdot \\
    \cdot & \cdot
    \end{bmatrix} = \begin{bmatrix}
    Y_1 & Y_2 \\
    \cdot & \cdot
    \end{bmatrix}, \quad R = \begin{bmatrix}
    R_1 & R_3 \\
    0 & R_3
    \end{bmatrix} \quad \text{and } T = \begin{bmatrix}
    T_1 & T_3 \\
    0 & T_2
    \end{bmatrix} \\
    \text{return } \{Y, R, T\}
    \end{align*}

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 \(\leq nb\))
- Replaces LAPACK level 2 and 3 algorithms
Recursive QR vs. LAPACK

\[ m = n \quad \quad m \gg n \]

**Fig. 6.1** Performance results in MFlop/s for square matrices (left) and performance ratios 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

Parallel speedup on a 4-way 332 MHz IBM PowerPC604e node

Erik Elmroth, PMAA 2006, Rennes, September 7 – 9, 2006
Case Study 3

Over- and under-determined linear systems

LAPACK DGELS

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

A = QR  \quad A = LQ  \quad A = QR

Least squares solution (over-determined systems)  \quad 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

\[ X = \text{RGELS}(A, B, nb) \]

\[ \text{If } n \leq nb \]

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

else

2. Let \( A = \begin{bmatrix} A_1 & A_2 \end{bmatrix} \); \( B = \begin{bmatrix} b_1 \\ b_2 \end{bmatrix} \) with \( nb \) cols in \( A_1 \), \( nb \) rows in \( B_1 \)

3. Factor \( A_1 = Q_1 \begin{bmatrix} \tilde{A}_1 \\ \tilde{B}_1 \end{bmatrix} \)

4. Set \( \begin{bmatrix} R_{12} & \tilde{B}_1 \\ A_{22} & \tilde{B}_2 \end{bmatrix} \leftarrow Q_1^T \begin{bmatrix} A_2 & B \end{bmatrix} \)

5. \( X_2 = \text{RGELS}(A_{22}, \tilde{B}_2, nb) \)

6. Solve \( R_{12}X_1 = \tilde{B}_1 - R_{12}X_2 \); return \( X = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \)

\[ \text{GEMM + TRMS + TRSM} \]

\[ \text{GEMM + TRMM} \]

\[ \text{Fig. 4.2 Recursive least squares RGELS algorithm for computing the solution to } AX = B, \text{ where } A \text{ is } m \times n (m \geq n). \]

Factorization, update and triangular solve are interleaved for each block \( \Rightarrow \) data reuse.

Minimum norm solution \( \| A^T X - B \|_F \)

- 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 $||CTX - B||$ 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

$\|AX - B\|_F$
Case Study 4

Triangular matrix equations and condition estimation
Matrix equations

<table>
<thead>
<tr>
<th>Name</th>
<th>Matrix equation</th>
<th>Acronym</th>
</tr>
</thead>
<tbody>
<tr>
<td>Standard Sylvester (CT)</td>
<td>$AX - XB = C$</td>
<td>SYCT</td>
</tr>
<tr>
<td>Standard Lyapunov (CT)</td>
<td>$AX + XA^T = C$</td>
<td>LYCT</td>
</tr>
<tr>
<td>Generalized coupled Sylvester</td>
<td>$(AX - YB, DX - YE) = (C, F)$</td>
<td>GCSY</td>
</tr>
<tr>
<td>Standard Sylvester (DT)</td>
<td>$AXB^T - X = C$</td>
<td>SYDT</td>
</tr>
<tr>
<td>Standard Lyapunov (DT)</td>
<td>$AXA^T - X = C$</td>
<td>LYDT</td>
</tr>
<tr>
<td>Generalized Sylvester</td>
<td>$AXB^T - CXD^T = E$</td>
<td>GSYL</td>
</tr>
<tr>
<td>Generalized Lyapunov (CT)</td>
<td>$AX^T + EXA^T = C$</td>
<td>GLYCT</td>
</tr>
<tr>
<td>Generalized Lyapunov (DT)</td>
<td>$AXA^T - EX^T = C$</td>
<td>GLYDT</td>
</tr>
</tbody>
</table>

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

Recursive blocked SYCT template

**Case 1**: $1 \leq n \leq m/2$

$A (m \times m), B (n \times n)$ upper tri.

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

**Case 2**: $1 \leq m \leq n/2$

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

**Case 3**: $n/2 < m < 2n$

$$
\begin{pmatrix}
A_{11} & A_{12} \\
A_{22}
\end{pmatrix}
\begin{pmatrix}
X_{11} & X_{12} \\
X_{21} & X_{22}
\end{pmatrix}
- \begin{pmatrix}
X_{11} & X_{12} \\
X_{21} & X_{22}
\end{pmatrix}
\begin{pmatrix}
B_{11} & B_{12} \\
B_{21} & B_{22}
\end{pmatrix}
= \begin{pmatrix}
C_{11} & C_{12} \\
C_{21} & C_{22}
\end{pmatrix}
$$
Recursive SYCT - Case 3

\[
\begin{align*}
A_{11}X_{11} - X_{11}B_{11} &= C_{11} - A_{12}X_{21} \\
A_{12}X_{12} - X_{12}B_{22} &= C_{12} - A_{12}X_{22} + X_{11}B_{12} \\
A_{21}X_{21} - X_{21}B_{11} &= C_{21} \\
A_{22}X_{22} - X_{22}B_{22} &= C_{22} + X_{21}B_{12}
\end{align*}
\]

1. SYLV(‘N’, ‘N’, A_{22}, B_{11}, C_{21})
2a. GEMM(‘N’, ‘N’, A_{22}, B_{11}, C_{22})
2b. GEMM(‘N’, ‘N’, A_{12}, B_{21}, C_{12})
3a. SYLV(‘N’, ‘N’, A_{22}, B_{22}, C_{22})
3b. SYLV(‘N’, ‘N’, A_{11}, B_{11}, C_{11})
4. GEMM(‘N’, ‘N’, A_{12}, B_{22}, C_{12})
5. GEMM(‘N’, ‘N’, A_{11}, B_{12}, C_{12})
6. SYLV(‘N’, ‘N’, A_{11}, B_{22}, C_{12})

---

Triangular generalized coupled Sylvester equation - GCSY

\[
\begin{align*}
AX - YB &= C \\
DX - YE &= F
\end{align*}
\]

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

Solution (X, Y) overwrites 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!