Towards the Final Generation of Dense
Linear Algebra Libraries

Robert A. van de Geijn
Department of Computer Sciences
The University of Texas at Austin
rvdg@cs.utexas.edu
http://www.cs.utexas.edu/users/flame/
Current FLAME Team

- UT Austin Department of Computer Sciences
  - Paolo Bientinesi
  - Maggie Myers
  - Ernie Chan
  - Robert van de Geijn
  - Tze Meng Low
  - Field Van Zee
  - Zaiqing Xu

- UT Texas Advanced Computing Center
  - Kazushige Goto
  - Kent Milfelt

- UT Center for Space Research
  - Brian Gunter

- Universidad Jaume I (Spain)
  - Enrique Quintana-Orti
  - Gregorio Quintana-Orti

- Industry
  - Hewlett-Packard
  - National Instruments
  - Intel
  - IBM
  - NEC Solutions (America), Inc
Support

- National Science Foundation
  - Modest Funding through 2006
- Hewlett-Packard
  - Equipment donations
- Unrestricted grants
  - Dr. James Truchard (National Instruments)
  - NEC Solutions (America), Inc
Motivation

- Developing dense linear algebra libraries
  - Traditional approach:
    - Evolve from existing libraries
    - Ask the question: What added functionality is needed?
    - Reactive to current needs
    - Always a “Next Generation” library
  - The Big Question:
    - Can we build a “Final Generation” library?
    - Proactive to (as of yet undefined) future needs
Motivation

Properties of a Final Generation Library

- Forward compatible to
  - New architectures
  - New languages
  - New datastructures
  - New operations
What do we do by hand?

- Mathematical specification of operation
- Derivation of algorithms
- An algorithm
- Manual analysis of algorithm or trial-and-error optimization
- A library routine
- Manual translation to code
The Final Generation

- Mathematical specification of operation
- Mechanical derivation of algorithms
- Family of algorithms
- Mechanical analysis of algorithms
- Library of routines
- Mechanical translation to code
- Rewrite rules for language

Aug. 25, 2005
http://www.cs.utexas.edu/users/flame/
Why a family of algorithms?
Performance on Itanium2 SMP

LAPACK: Rich in TRMM

New Variant: Rich in GEMM

1 CPU
Performance on Itanium2 SMP

New Variant: Rich in GEMM

LAPACK: Rich in TRMM

4 CPUs

Aug. 25, 2005  
http://www.cs.utexas.edu/users/flame/
A Note on Performance

- The FLAME team is recognized for performance:
  - GOTO BLAS by Kazushige Goto:
    - Fastest for essentially all platforms
  - FLAME:
    - Outperforms LAPACK
  - PLAPACK:
    - Outperforms ScaLAPACK for all major operations
- I will show few performance graph
Kazushige Goto’s BLAS

PPC440 FP2 DGEMM

- GOTO
- ESSL

MFlops

Matrix Order
Overview

- New Notation for Expressing Algorithms
- APIs for Representing Algorithms in Code
- Mechanical Derivation of Algorithms
- Mechanical Analysis of Algorithms
- Addressing Future Challenges
- Conclusion
The Final Generation

Mathematical specification of operation

Mechanical derivation of algorithms

Family of algorithms

Mathematical specification of architecture

Mechanical analysis of algorithms

Rewrite rules for language

Mechanical translation to code

Optimized library
Step 1:

Change the Notation for Expressing Algorithms
Example: QR factorization via Householder Transformations

n Blocked Algorithm:
  n Factor current panel
  n Form compact WY transform
  n Update rest of matrix
QR factorization via Householder transformations

- Blocked Algorithm:
  - Factor current panel
  - Form compact WY transform
  - Update rest of matrix
  - Move forward
Capturing movement through matrices
This picture has been around:

- In a typical talk on LAPACK:

  ![Left-looking LU](image1)
  ![Right-looking LU](image2)
  ![Crout LU](image3)

- Pete Stewart’s recent book:
Can the picture be the algorithm?
Algorithm: \( [A,s] := \text{QRBLK}(A) \)

Partition \( A \to \left( \begin{array}{c|c} A_{TL} & A_{TR} \\ \hline A_{BL} & A_{BR} \end{array} \right) \) and \( t \to \left( \begin{array}{c} s_T \\ \hline s_B \end{array} \right) \)

where \( A_{TL} \) is \( 0 \times 0 \) and \( s_T \) has 0 elements

while \( n(A_{BR}) \neq 0 \) do

Determine block size \( k \)

Repartition

\[
\left( \begin{array}{c|c} A_{TL} & A_{TR} \\ \hline A_{BL} & A_{BR} \end{array} \right) \to \left( \begin{array}{c|c|c} A_{00} & A_{01} & A_{02} \\ \hline A_{10} & A_{11} & A_{12} \\ \hline A_{20} & A_{21} & A_{22} \end{array} \right) \quad \text{and} \quad \left( \begin{array}{c} s_T \\ \hline s_B \end{array} \right) \to \left( \begin{array}{c} s_0 \\ \hline s_1 \\ \hline s_2 \end{array} \right)
\]

where \( A_{11} \) is \( k \times k \) and \( s_1 \) has \( k \) elements

\[
\left[ \left( \frac{A_{11}}{A_{21}} \right), s_1 \right] := \left[ \left( \frac{U \setminus R}{U_{21}} \right)_{11}, s_1 \right] = \text{QRUNB} \left( \left( \frac{A_{11}}{A_{21}} \right) \right)
\]

Compute \( S_1 \) from \( \left[ \left( \frac{U_{11}}{U_{21}} \right), s_1 \right] \)

Update

\[
\left( \frac{A_{12}}{A_{22}} \right) := \left( I + \left( \frac{U_{11}}{U_{21}} \right) S_1 \left( \begin{array}{c} U_{11} \\ U_{21} \end{array} \right)^T \right)^T \left( \frac{A_{12}}{A_{22}} \right)
\]

Continue with

\[
\left( \begin{array}{c|c} A_{TL} & A_{TR} \\ \hline A_{BL} & A_{BR} \end{array} \right) \leftarrow \left( \begin{array}{c|c|c} A_{00} & A_{01} & A_{02} \\ \hline A_{10} & A_{11} & A_{12} \\ \hline A_{20} & A_{21} & A_{22} \end{array} \right) \quad \text{and} \quad \left( \begin{array}{c} s_T \\ \hline s_B \end{array} \right) \leftarrow \left( \begin{array}{c} s_0 \\ \hline s_1 \\ \hline s_2 \end{array} \right)
\]

endwhile
Partition $A \rightarrow \begin{pmatrix} A_{TL} & A_{TR} \\ A_{BL} & A_{BR} \end{pmatrix}$

where $A_{TL}$ is $0 \times 0$

while $n(A_{BR}) \neq 0$ do

Determine block size $b$

Repartition

$\begin{pmatrix} A_{TL} & A_{TR} \\ A_{BL} & A_{BR} \end{pmatrix} \rightarrow \begin{pmatrix} A_{00} & A_{01} & A_{02} \\ \hline A_{10} & A_{11} & A_{12} \\ A_{20} & A_{21} & A_{22} \end{pmatrix}$

where $A_{11}$ is $b \times b$

$\begin{bmatrix} \begin{pmatrix} A_{11} \\ A_{21} \end{pmatrix}, s_1 \end{bmatrix} := \begin{bmatrix} \begin{pmatrix} U \setminus R \end{pmatrix}_{11} \\ U_{21} \end{pmatrix}, s_1 \end{bmatrix} = QR \begin{pmatrix} A_{11} \\ A_{21} \end{pmatrix}$

Compute $S_1$ from $\begin{bmatrix} \begin{pmatrix} U_{11} \\ U_{21} \end{pmatrix}, s_1 \end{bmatrix}$

Update

$\begin{pmatrix} A_{12} \\ A_{22} \end{pmatrix} := \left( I + \begin{pmatrix} U_{11} \\ U_{21} \end{pmatrix} S_1 \begin{pmatrix} U_{11} \\ U_{21} \end{pmatrix}^T \right)^T \begin{pmatrix} A_{12} \\ A_{22} \end{pmatrix}$

Continue with

$\begin{pmatrix} A_{TL} & A_{TR} \\ A_{BL} & A_{BR} \end{pmatrix} \leftarrow \begin{pmatrix} A_{00} & A_{01} & A_{02} \\ \hline A_{10} & A_{11} & A_{12} \\ A_{20} & A_{21} & A_{22} \end{pmatrix}$

endwhile
Partition \( A \rightarrow \begin{pmatrix} A_{TL} & A_{TR} \\ A_{BL} & A_{BR} \end{pmatrix} \)
where \( A_{TL} \) is \( 0 \times 0 \)

while \( n(A_{BR}) \neq 0 \) do
  Determine block size \( b \)
  Repartition
  \[
  \begin{pmatrix} A_{TL} & A_{TR} \\ A_{BL} & A_{BR} \end{pmatrix} \rightarrow \begin{pmatrix} A_{00} & A_{01} & A_{02} \\ A_{10} & A_{11} & A_{12} \\ A_{20} & A_{21} & A_{22} \end{pmatrix}
  \]
  where \( A_{11} \) is \( b \times b \)
  
  \[
  \left[ \begin{pmatrix} A_{11} \\ A_{21} \end{pmatrix}, s_1 \right] := \left[ \begin{pmatrix} \{U \setminus R\}_{11} \\ U_{21} \end{pmatrix}, s_1 \right] = QR \begin{pmatrix} A_{11} \\ A_{21} \end{pmatrix}
  \]
  Compute \( S_1 \) from \( \left[ \begin{pmatrix} U_{11} \\ U_{21} \end{pmatrix}, s_1 \right] \)
  Update
  \[
  \begin{pmatrix} A_{12} \\ A_{22} \end{pmatrix} := \left( I + \begin{pmatrix} U_{11} \\ U_{21} \end{pmatrix} S_1 \begin{pmatrix} U_{11} \\ U_{21} \end{pmatrix}^T \right)^T \begin{pmatrix} A_{12} \\ A_{22} \end{pmatrix}
  \]
  Continue with
  \[
  \begin{pmatrix} A_{TL} & A_{TR} \\ A_{BL} & A_{BR} \end{pmatrix} \leftarrow \begin{pmatrix} A_{00} & A_{01} & A_{02} \\ A_{10} & A_{11} & A_{12} \\ A_{20} & A_{21} & A_{22} \end{pmatrix}
  \]
endwhile
Algorithm: $[A,t] := \text{QR}(A)$

$[\text{ATL, ATR, ...}]
\text{ABL, ABR}] = \text{FLA}_\text{Part}_2\times2(A, 0, 0, '\text{FLA}_{\text{TL}}');$

while (size( ATL, 2 ) \neq size( A, 2 ) )

$b = \min(\text{size( ABR, 1 )}, \text{nb}_\text{alg});$

$[\text{A00, A01, A02, ...}]
\text{A10, A11, A12, ...}
\text{A20, A21, A22}] = \text{FLA}_\text{Repart}_2\times2\text{to}_3\times3(\text{ATL, ATR, ...}
\text{ABL, ABR, ...})$

$[\text{UL, s1}] = \text{QR}\_\text{unb\_var1}( [\text{A11, b, b, 'FLA}_{\text{BR}}], \text{A21}]);$

$[\text{A11, ...}]
\text{A21}] = \text{FLA}_\text{Part}_2\times1(\text{UL, b, 'FLA}_{\text{TOP}});$

$S1 = \text{Accum}_S(UL, s1);$  

$\text{U11} = \text{trilu}(\text{A11});$
$\text{U21} = \text{A21};$
$\text{W12} = S1' \ast (\text{U11}' \ast \text{A12} + \text{U21}' \ast \text{A22});$
$\text{A12} = \text{A12} - \text{U11} \ast \text{W12};$
$\text{A22} = \text{A22} - \text{U21} \ast \text{W12};$

$[\text{ATL, ATR, ...}]
\text{ABL, ABR}] = \text{FLA}_\text{Cont\_with}_3\times3\text{to}_2\times2(\text{A00, A01, A02, ...}
\text{A10, A11, A12, ...}
\text{A20, A21, A22, 'FLA}_{\text{TL}});$

end
Step 2:

APIs for Representing Algorithms in Code
The Final Generation

Mathematical specification of operation

Mechanical derivation of algorithms

Rewrite rules for language

Family of algorithms

Mechanical translation to code

Optimized library

Mathematical specification of architecture

Mechanical analysis of algorithms

Aug. 25, 2005

http://www.cs.utexas.edu/users/flame/
LAPACK API
DO 10 I = 1, K - NX, NB
  IB = MIN( K-I-1, NB )

  Compute the QR factorization of the current block
  A(i:m,i:i+ib-1)

  CALL DGEQR2( M-I-1, IB, A( I, I ), LDA, TAU( I ), WORK, IINFO )
  IF( I+IB.LE.N ) THEN

    Form the triangular factor of the block reflector
    H = H(i) H(i+1) . . . H(i+ib-1)

    CALL DLARFT( 'Forward', 'Columnwise', M-I-1, IB, $  
      A( I, I ), LDA, TAU( I ), WORK, LDWORK )

    Apply H' to A(i:m,i+ib:n) from the left

    CALL DLARFB( 'Left', 'Transpose', 'Forward', $  
      'Columnwise', M-I-1, N-I-IB-1, IB, $  
      A( I, I ), LDA, WORK, LDWORK, A( I, I+IB ), $  
      LDA, WORK( IB+1 ), LDWORK )

  END IF
10   CONTINUE
DO 10 I = 1, K - NX, NB
   IB = MIN( K-I-1, NB )
   ** Compute the QR factorization of the current block
   ** A(i:m,i:i+ib-1)
   **
   CALL DGEQR2( M-I-1, IB, A( I, I ), LDA, TAU( I ), WORK, $  
               IIINFO )
   IF( I+IB.LE.N ) THEN
   ** Form the triangular factor of the block reflector
   ** H = H(i) H(i+1) . . . H(i+ib-1)
   **
   CALL DLARFT( 'Forward', 'Columnwise', M-I-1, IB, $  
                  A( I, I ), LDA, TAU( I ), WORK, LDWORK )
   ** Apply H' to A(i:m,i+ib:n) from the left
   **
   CALL DLARFB( 'Left', 'Transpose', 'Forward', $  
                 'Columnwise', M-I-1, N-I-IB-1, IB, $  
                 A( I, I ), LDA, WORK, LDWORK, A( I, I+IB ), $  
                 LDA, WORK( IB+1 ), LDWORK )
   END IF
10    CONTINUE

Warning: I introduced an error!
Step 2:

New APIs that capture the algorithm in code
FLAME@lab

(FLAME/MATLAB API)
[ ATL, ATR, 
  ABL, ABR ] = FLA_Part_2x2( A, 0, 0, 'FLA_TL' );

while ( size( ATL, 2 ) ~= size( A, 2 ) )

  b = min( size( ABR, 1 ), nb_alg );

  [ A00, A01, A02, 
    A10, A11, A12, 
    A20, A21, A22 ] = FLA_Repart_2x2_to_3x3( ATL, ATR, 
                                           ABL, ABR, 
                                           b, b, 'FLA_BR' );

  [ U1, s1 ] = QR_unb_var1([ A11 A21 ], s1 );

  [ A11, 
    A21 ] = FLA_Part_2x1( U1, b, 'FLA_TOP' );

  S1 = Accum_S( U1, s1 );

  U11 = trilu( A11 );
  U21 = A21;
  W12 = S1' * ( U11' * A12 + U21' * A22 )
  A12 = A12 - U11 * W12;
  A22 = A22 - U21 * W12;

[ ATL, ATR, 
  ABL, ABR ] = FLA_Cont_with_3x3_to_2x2( A00, A01, A02, 
                                         A10, A11, A12, 
                                         A20, A21, A22, 'FLA_TL' );

end
[ ATL, ATR, ...  
 ABL, ABR ] = FLA_Part_2x2( A, 0, 0, 'FLA_TL' );  
[ sT, ...  
 sB ] = FLA_Part_2x1( s, 0, 'FLA_TOP' );  
while ( size( ATL, 2 ) ~= size( A, 2 ) )  
 b = min( size( ABR, 1 ), nb_alg );  
 [ A00, A01, A02, ...  
 A10, A11, A12, ...  
 A20, A21, A22 ] = FLA_Repart_2x2_to_3x3( ATL, ATR, ... 
 ABL, ABR, b, b, 'FLA_BR' );  
[ s0, ...  
 s1, ...  
 s2 ] = FLA_Repart_2x1_to_3x1( sT, ... 
 sB, b, 'FLA_BOTTOM' );  
%---------------------------------------------------------------------------%  
[ U1, s1 ] = QR_unb_var1( [ A11  
 A21 ], s1 );  
[ A11, ...  
 A21 ] = FLA_Part_2x1( U1, b, 'FLA_TOP' );  
 s1 = Accum_S( U1, s1 );  
 % update rest of matrix  
 U11 = trilu( A11 );  
 U21 = A21;  
 W12 = s1' * ( U11' * A12 + U21' * A22 );  
 A12 = A12 - U11 * W12;  
 A22 = A22 - U21 * W12;  
%---------------------------------------------------------------------------%  
[ ATL, ATR, ...  
 ABL, ABR ] = FLA_Cont_with_3x3_to_2x2( A00, A01, A02, ...  
 A10, A11, A12, ...  
 A20, A21, A22, 'FLA_TL' );  
[ sT, ...  
 sB ] = FLA_Cont_with_3x1_to_2x1( s0, ... 
 s1, ...  
 s2, 'FLA_TOP' );  
end
\[
\begin{align*}
\text{[ ATL, ATR, } & \ldots \\
\text{ABL, ABR } = \text{FLA}_\text{Part}_2\text{x}2( A, 0, 0, 'FLA_{TL}' );
\end{align*}
\]
\[
\begin{align*}
\text{[ st, } & \ldots \\
\text{sB } = \text{FLA}_\text{Part}_2\text{x}1( s, 0, 'FLA_{TOP}' );
\end{align*}
\]
\[
\text{while ( size( ATL, 2 ) } \sim= \text{size( A, 2 ) )}
\]
\[
\begin{align*}
& b = \min( \text{size( ABR, 1 ), nb_alg } ); \\
& \text{[ A00, A01, A02, } \ldots \\
& \text{A10, A11, A12, } \ldots \\
& \text{A20, A21, A22 ] = FLA}_\text{Repart}_2\text{x}2\text{to}_3\text{x}3( \text{ATL, ATR, } \ldots \\
& \text{ ABL, ABR, } b, b, 'FLA_{BR}' );
\end{align*}
\]
\[
\begin{align*}
& \text{[ s0, } \ldots \\
& \text{s1, } \ldots \\
& \text{s2 ] = FLA}_\text{Repart}_2\text{x}1\text{to}_3\text{x}1( \text{sT, } \ldots \\
& \text{sB, } b, 'FLA_{BOTTOM}' );
\end{align*}
\]
\[
\begin{align*}
% - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - \\
& \text{[ U1, s1 ] = QR}_\text{unb}_\text{var1}( [ \text{A11} \\
& \text{A21 }, s1 );
\end{align*}
\]
\[
\begin{align*}
& \text{[ A11, } \ldots \\
& \text{A21 ] = FLA}_\text{Part}_2\text{x}1( U1, b, 'FLA_{TOP}' ); \\
& \text{S1 } = \text{Accum}_S( U1, s1 ); \\
& \% \text{ Update rest of matrix}
\end{align*}
\]
\[
\begin{align*}
& \text{U11 } = \text{trilu}( A11 ); \\
& \text{U21 } = \text{A21}; \\
& \text{W12 } = \text{S1}' * ( \text{U11}' * \text{A12 } + \text{U21}' * \text{A22 } ); \\
& \text{A12 } = \text{A12 } - \text{U11 } * \text{W12}; \\
& \text{A22 } = \text{A22 } - \text{U21 } * \text{W12};
\end{align*}
\]
\[
\begin{align*}
% - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - \\
& \text{[ ATL, ATR, } \ldots \\
& \text{ABL, ABR ] = FLA}_\text{Cont}_3\text{x}3\text{to}_2\text{x}2( \text{A00, A01, A02, } \ldots \\
& \text{A10, A11, A12, } \ldots \\
& \text{A20, A21, A22, 'FLA_{TL}' });
\end{align*}
\]
\[
\begin{align*}
& \text{[ st, } \ldots \\
& \text{sB ] = FLA}_\text{Cont}_3\text{x}1\text{to}_2\text{x}1( \text{s0, } \ldots \\
& \text{s1, } \ldots \\
& \text{s2, 'FLA_{TOP}' });
\end{align*}
\]
\[
\text{end}
\]
while ( size( ATL, 2 ) ~= size( A, 2 ) )
  b = min( size( ABR, 1 ), nb_alg );
  [ A00, A01, A02, ...
    A10, A11, A12, ...
    A20, A21, A22 ] = FLA_Repart_2x2_to_3x3( ATL, ATR, ...
                                                                 ABL, ABR, b, b, 'FLA_BR' );

%------------------------------------------

[ U1, s1 ] = QR_unb_var1( [ A11
                        A21 ], s1 );

[ A11, ...
  A21 ] = FLA_Part_2x1( U1, b, 'FLA_TOP' );
S1   = Accum_S( U1, s1 );
U11  = trilu( A11 );
U21  = A21;
W12  = S1' * ( U11' * A12 + U21' * A22 );
A12  = A12 - U11 * W12;
A22  = A22 - U21 * W12;
%------------------------------------------

[ ATL, ATR, ...
  ABL, ABR ] = FLA_Cont_with_3x3_to_2x2( A00, A01, A02, ...
                                         A10, A11, A12, ...
                                         A20, A21, A22, 'FLA_TL' );
end
function [ S ] = Accum_S( U, s )

U = trilu( U ); % U = lower unit trapezoidal part of U
s = ones( size( s ) ) ./ s; % Set each element of s to its inverse
S = inv( triu( U' * U, 1 ) + diag( s ));
return


C. Puglisi. Modification of the Householder method based on the compact WY representation. SISC, 18, 723-726, 1992

APIs for C
FLAME/C
QR factorization
FLA_Part_2x2( A, &ATL, &ATR, 
&ABL, &ABR, 0, 0, FLA_TL );

while ( FLA_Obj_width ( ATL ) != FLA_Obj_width ( A ) ){
    b = min( FLA_Obj_width( ABR ), nb_alg );

    FLA_Repart_2x2_to_3x3( ATL, /**/ ATR, &A00, /**/ &A01, &A02,
    /* ********** */ /* ********** */ &A10, /**/ &A11, &A12,
    ABL, /**/ ABR, &A20, /**/ &A21, &A22,
    b, b, FLA_BR );
}

FLA_QR_unb( A11, A21, s1 );

FLA_Accum_S( A11, A21, s1, S1 );

FLA_Apply_blk_transform( FLA_LEFT, FLA_TRANSPOSE, A11, S1, A12, 
A21, A22 );

FLA_Cont_with_3x3_to_2x2( &ATL, /**/ &ATR, A00, A01, /**/ A02, 
A10, A11, /**/ A12,
    /* ********** */ /* ********** */ &ABL, /**/ &ABR, A20, A21, /**/ A22,
    FLA_TL );
}
Spark: A Tool for Generating Representations

Spark

FLAME code skeleton generator

The name tabs on the left will help you generate code for algorithms that resulted from the FLAME approach to deriving linear algebra algorithms.

The first section allows you to:

1. Indicate the name of the function to be generated.

2. Choose a name that is related to the operation. For example, consider the operation

   \[ B = A \times B \]

   which is often referred to as a Triangle Solve with Multiple Right-hand sides (TranM). Here A is an \( n \times n \) matrix and \( B \) is a \( k \times n \) matrix. We would suggest a name like

   `TranM`.

   where,Q local indicates that the matrix \( A \) is

   \[ B \] in the left of matrix \( B \),

   and \( B \) is invalid.

   which are familiar with the level-3 Basic Linear Algebra Subroutines (BLAS).

2. Decide whether the algorithm is unblocked, blocked, or recursive.

3. Indicate the name that is the function name for the given operation, the code that is being coded.

In the second section, you need to indicate the number of operands for the function and choose the distribute of these operands.
The Final Generation

- Family of algorithms
- POOCLAPACK (distributed parallel OOC)
- PLAPACK (distributed parallel)
- OpenFLAME (OpenMP)
- FLASH (Matrices stored hierarchically)
- Mechanical translation to code
- Rewrite rules for language
- FLaTeX (LaTeX)
- FLAME@lab (Matlab)
- FLAME/C
- FLAME/Fortran
- PictureFLAME (LabView)
The Final Generation

Mathematical specification of operation

Mechanical derivation of algorithms

Family of algorithms

Mathematical specification of architecture

Mechanical analysis of algorithms

Rewrite rules for language

Mechanical translation to code

Optimized library
Some Wisdom from the Past

- The only effective way to raise the confidence level of a program significantly is to give a convincing proof of its correctness. But one should not first make the program and then prove its correctness, because then the requirement of providing the proof would only increase the poor programmer's burden. On the contrary: the programmer should let correctness proof and program grow hand in hand. (E.W. Dijkstra: "The Humble Programmer," 1972 Turing Award lecture, in ACM Turing Award Lectures: The First Twenty Years, 1966-1985, ACM Press, New York, 1987.)
What else does the new notation buy us?

State of the matrix at the top of the loop

Right-looking algorithm

Left-looking algorithm

Aug. 25, 2005
Key insight

- Given the state that is to be maintained, an algorithm can be systematically derived.

- The method is sufficiently systematic that it can be made mechanical.
A simpler example: TRSM

\[ B := U^{-1} B \text{ where } U \text{ is upper triangular} \]

\[
\begin{bmatrix}
B_T \\
\hline
B_B
\end{bmatrix}
::=
\begin{bmatrix}
U_{TL}^{-1} (B_T - U_{TR} U_{BR}^{-1} B_B) \\
\hline
U_{BR}^{-1} B_B
\end{bmatrix}
\]
Mathematical specification of operation

Mechanical derivation of algorithms

Family of algorithms
\[ U_{TL}^{-1} (B_T - U_{TR} U_{BR}^{-1} B_B) \]

\[ U_{BR}^{-1} B_B \]

**MATHEMATICA® 5**

Family of algorithms
Mechanical Derivation

Mathematical specification of operation

Mechanical derivation of algorithms

Family of algorithms
Switch to Demo
Notation\[E\left(\frac{A}{B}, \frac{C}{D}, \frac{E}{F}\right) \rightarrow \text{coupledSylv}[A, B, C, D, E, F]\]

PMEs

1x2
2x1
2x2

worksheet[coupledSylv,
\{("A", "UpperTriangular", "TL"),
("E", "UpperTriangular", "BR"),
("C", "Overwrite", "TR"),
("D", "UpperTriangular", "TL"),
("F", "UpperTriangular", "BR"),
("E", "TR", "Overwrite")\}]

2x1

2x2

PME 2x2

Loop Inv 1

\[
\begin{pmatrix}
C_{TL} \\
\Xi \left( \frac{A_{BR}, B_{TL}, C_{BL}}{D_{BR}, E_{TL}, F_{BL}} \right) [1] \\
F_{TL} \\
\Xi \left( \frac{A_{BR}, B_{TL}, C_{BL}}{D_{BR}, E_{TL}, F_{BL}} \right) [2]
\end{pmatrix}
\begin{pmatrix}
C_{TR} \\
C_{BR} \\
F_{TR} \\
F_{BR}
\end{pmatrix}
\]

State at top of loop

Loop Inv 2

\[
\begin{pmatrix}
-A_{TR} \cdot \Xi \left( \frac{A_{BR}, B_{TL}, C_{BL}}{D_{BR}, E_{TL}, F_{BL}} \right) [1] + C_{TL} \\
\Xi \left( \frac{A_{BR}, B_{TL}, C_{BL}}{D_{BR}, E_{TL}, F_{BL}} \right) [1] \\
F_{TL} \\
\Xi \left( \frac{A_{BR}, B_{TL}, C_{BL}}{D_{BR}, E_{TL}, F_{BL}} \right) [2]
\end{pmatrix}
\begin{pmatrix}
C_{TR} \\
C_{BR} \\
F_{TR} \\
F_{BR}
\end{pmatrix}
\]
Loop Inv 4

\[
\begin{align*}
-A_{TR} \cdot \left( \sum \left( \begin{array}{c} A_{BR}, B_{TL}, C_{EL} \\ D_{BR}, E_{TL}, F_{EL} \end{array} \right) \right) \left[ 1 \right] + C_{TL} & \quad C_{TR} \\
\left( \sum \left( \begin{array}{c} A_{BR}, B_{TL}, C_{EL} \\ D_{BR}, E_{TL}, F_{EL} \end{array} \right) \right) \left[ 1 \right] & \quad C_{ER} \\
-D_{TR} \cdot \left( \sum \left( \begin{array}{c} A_{BR}, B_{TL}, C_{EL} \\ D_{BR}, E_{TL}, F_{EL} \end{array} \right) \right) \left[ 2 \right] + F_{TL} & \quad F_{TR} \\
\left( \sum \left( \begin{array}{c} A_{BR}, B_{TL}, C_{EL} \\ D_{BR}, E_{TL}, F_{EL} \end{array} \right) \right) & \quad F_{ER}
\end{align*}
\]

Mathematic specification of state at top of loop
\[
\begin{pmatrix}
\left( E \left( \begin{array}{c}
\frac{A_{TL}}{D_{TL}} - A_{TR} \\
\frac{E_{TL}}{D_{TL}} - E_{TR}
\end{array} \right) [1] + C_{TL}
\right) [1] C_{TR} \\
\left( E \left( \begin{array}{c}
\frac{A_{BR}}{D_{BR}} - A_{BL} \\
\frac{E_{BR}}{D_{BR}} - E_{BL}
\end{array} \right) [1] + F_{TL}
\right) [1] C_{BR}
\end{pmatrix}
\]

Loop Inv 5
Algorithm with intermediate states generated by system
\[
\begin{align*}
C_{00} & := -A_{01} \cdot C_{10} + C_{00} \\
C_{01} & := -A_{01} \cdot C_{11} - A_{02} \cdot C_{21} + C_{01} \\
C_{10} & := \left( \mathbb{E} \left( \frac{A_{11}, B_{00}, C_{10}}{D_{11}, E_{00}, F_{10}} \right) \right) [1] \\
C_{11} & := \left( \mathbb{E} \left( \frac{A_{11}, B_{11}, F_{10} \cdot B_{01} - A_{12} \cdot C_{21} + C_{11}}{D_{11}, E_{11}, F_{10} \cdot E_{01} - D_{12}} \right) \right) [1] \\
C_{21} & := \left( \mathbb{E} \left( \frac{A_{22}, B_{11}, F_{20} \cdot B_{01} + C_{21}}{D_{22}, E_{11}, F_{20} \cdot E_{01} + F_{21}} \right) \right) [1] \\
F_{00} & := -D_{01} \cdot C_{10} + F_{00} \\
F_{01} & := -D_{01} \cdot C_{11} - D_{02} \cdot C_{21} + F_{01} \\
F_{10} & := \left( \mathbb{E} \left( \frac{A_{11}, B_{00}, C_{10}}{D_{11}, E_{00}, F_{10}} \right) \right) [2] \\
F_{11} & := \left( \mathbb{E} \left( \frac{A_{11}, B_{11}, F_{10} \cdot B_{01} - A_{12} \cdot C_{21} + C_{11}}{D_{11}, E_{11}, F_{10} \cdot E_{01} - D_{12}} \right) \right) [2] \\
F_{21} & := \left( \mathbb{E} \left( \frac{A_{22}, B_{11}, F_{20} \cdot B_{01} + C_{21}}{D_{22}, E_{11}, F_{20} \cdot E_{01} + F_{21}} \right) \right) [2]
\end{align*}
\]
Loop Inv 49

\[
\begin{align*}
\left( \sum \left( \frac{A_{BR}, E_{BR}, C_{BL}}{D_{BR}, E_{TL}, F_{BL}} \right) \right)^{[2]} & \left( \sum \left( \frac{A_{BR}, E_{BR}, C_{BL}}{D_{BR}, E_{TL}, F_{BL}} \right) \right)^{[2]} - \left( \sum \left( \frac{A_{BR}, E_{BR}, C_{BL}}{D_{BR}, E_{TL}, F_{BL}} \right) \right)^{[2]} \cdot D_{TR} + C_{BR} \\
\left( \sum \left( \frac{A_{BR}, E_{BR}, C_{BL}}{D_{BR}, E_{TL}, F_{BL}} \right) \right)^{[1]} & \left( \sum \left( \frac{A_{BR}, E_{BR}, C_{BL}}{D_{BR}, E_{TL}, F_{BL}} \right) \right)^{[1]} + 1
\end{align*}
\]

worksheet[coupledSylv,
Scope of Methodology

- Triangular Generalized Sylvester Equation
  \[ A X + Y B = C \]
  \[ D X + Y E = F \]
- A, B, D, E triangular
- X and Y overwrite C and F
- 57 algorithmic variants…
- Blocked and unblocked each
- Compose recursively for optimal performance…
The Final Generation

Mathematical specification of architecture

Mathematical specification of operation

Mechanical derivation of algorithms

Family of algorithms

Mechanical analysis of algorithms

Optimized library

Rewrite rules for language

Mechanical translation to code

Aug. 25, 2005

http://www.cs.utexas.edu/users/flame/
The Final Generation

Mathematical specification of architecture

Mathematical analysis of algorithms

Mechanical derivation of algorithms

Family of algorithms

Rewrite rules for language

Mechanical translation to code

Optimized library
Mechanical Cost Analysis

Dissertation of John Gunnels
(for parallel distributed)
Mechanical Cost Analysis
Key: Vertical Integration

- **Solvers (FLAME)**

- **BLAS (FLAME)**

- **Low level kernels (K.Goto)**
Systematic Mechanical Stability Analysis

Paolo Bientinesi (in progress)
Annotated Algorithm: \( \kappa := x^T y \)

\[ \dot{\kappa} = x^T \Delta y \]

Step

<table>
<thead>
<tr>
<th>Partition</th>
</tr>
</thead>
</table>
| \( x \rightarrow \left( \begin{array}{c}
   x_T \\
   x_B
\end{array} \right) \), \( y \rightarrow \left( \begin{array}{c}
   y_T \\
   y_B
\end{array} \right) \) |

\[ \Delta \rightarrow \left( \begin{array}{c|c}
   \Delta_T & 0 \\
   0 & \Delta_B
\end{array} \right) \]

\( \{ \hat{\kappa} = x_T y_T \} \)

\( \{ \kappa = m(x_T) \} \)

while \( m(x_B) > 0 \) do

<table>
<thead>
<tr>
<th>Repartition</th>
</tr>
</thead>
</table>
| \( \left( \begin{array}{c}
   x_T \\
   x_B
\end{array} \right) \rightarrow \left( \begin{array}{c}
   x_0  \\
   x_1  \\
   x_2
\end{array} \right) \), \( \left( \begin{array}{c}
   y_T \\
   y_B
\end{array} \right) \rightarrow \left( \begin{array}{c}
   y_0  \\
   y_1  \\
   y_2
\end{array} \right) \) |

\[ \left( \begin{array}{c|c}
   \Delta_T & 0 \\
   0 & \Delta_B
\end{array} \right) \rightarrow \left( \begin{array}{c|c|c}
   \Delta_0 & 0 & 0 \\
   0 & \delta_1 & 0 \\
   0 & 0 & \Delta_2
\end{array} \right) \]

\( \hat{\kappa} = \left[ \begin{array}{c}
   x_0^T \\
   y_0 \\
   \psi_0
\end{array} \right] \)

\( \{ \hat{\kappa} = x_0^T \Delta_0^{\{k\}} \psi_0 \} \)

\( \kappa := [\hat{\kappa} + \chi_1 \psi_1] \)

\( \hat{\kappa} := \left( \hat{\kappa} + \chi_1 \psi_1 (1 + \epsilon_+) \right) (1 + \epsilon_+) \)

\( \Delta_0 := \Delta_0 (1 + \epsilon_+) \)

\( \delta_1 := (1 + \epsilon_+) (1 + \epsilon_+) \psi_1 \)

\( \kappa := \left[ \begin{array}{c}
   x_0 \\
   \chi_1
\end{array} \right] \)

\( \hat{\kappa} := \left( \begin{array}{c}
   x_0 \\
   \chi_1
\end{array} \right)^T \left[ \begin{array}{c|c}
   \Delta_0^{\{k\}} & 0 \\
   0 & \delta_1
\end{array} \right] \left( \begin{array}{c}
   y_0 \\
   \psi_1
\end{array} \right) = \left( \begin{array}{c}
   x_0^T \Delta_0^{\{k\}} y_0 + \chi_1 \delta_1 \psi_1 \\
   \chi_1 (1 + \epsilon_+) \psi_1
\end{array} \right) \)

Continue with \( \ldots \)

\( \left( \begin{array}{c|c}
   \Delta_T & 0 \\
   0 & \Delta_B
\end{array} \right) \left( \begin{array}{c|c|c}
   \Delta_0 & 0 & 0 \\
   0 & \delta_1 & 0 \\
   0 & 0 & \Delta_2
\end{array} \right) \)
The Final Generation

- Mathematical specification of architecture
  - Mechanical analysis of algorithms

- Mechanical derivation of algorithms
  - Family of algorithms
  - Mechanical translation to code
  - Rewrite rules for language

- Optimized library
Future Challenges: Multicore Processors

- Processors with two cores available now
- All HP users will have to cope
- 32-128 cores per processor in 10 years?
- 32 processor SMP x 32 cores = 1024 way SMP parallelism
- Virtually no literature on SCALABLE linear algebra libraries for SMPs
Shared Memory Parallelism

- Traditional approach:
  - Only from multithreaded BLAS

- Observation:
  - Better speedup if parallelism is exposed at a higher level
  - Scalability requires 2D work distribution

- FLAME approach:
  - Choose the best algorithmic variant
  - Work queuing (e.g., OpenMP task queues)
LAPACK: Rich in TRMM

New Variant: Rich in GEMM

1 CPU
Performance on Itanium2 SMP

New Variant: Rich in GEMM

LAPACK: Rich in TRMM

4 CPUs
Work queuing

- Simple example:
  - Symmetric rank-k update
Work queuing

Simple example:

Symmetric rank-k update
Shared Memory Parallelism

- Example:
  - Symmetric rank-k update
while ( FLA_Obj_length( CTL ) < FLA_Obj_length( C ) ){
    b = min( FLA_Obj_length( CBR ), nb_alg );

    FLA_Report_2x2_to_3x3( CTL, /**/ CTR, &C00, /**/ &C01, &C02,
                          /***************/
                          &C10, /**/ &C11, &C12,
                          CBL, /**/ CBR, &C20, /**/ &C21, &C22,
                          b, b, FLA_BR );

    FLA_Report_2x1_to_3x1( AT,
                           &A0,
                           /* */
                           /* */
                           &A1,
                           AB,
                           &A2, b, FLA_BOTTOM );

    /**********************************************************************/

    FLA_Gemm( FLA_NO_TRANSPOSE, FLA_TRANSPOSE, ONE, A0, A1, ONE, C10 );
    FLA_Svrm( FLA_LOWER_TRIANGULAR, FLA_NO_TRANSPOSE, ONE, A1, ONE, C11 );

    /**********************************************************************/

    FLA_Cont_with_3x3_to_2x2( &CTL, /**/ &CTR, C00, C01, /**/ C02,
                               C10, C11, /**/ C12,
                               /***************/
                               &CBL, /**/ &CBL, C20, C21, /**/ C22,
                               FLA_TL );

    FLA_Cont_with_3x1_to_2x1( &AT, A0,
                               &A1,
                               /* */
                               /* */
                               &AB, A2, FLA_TOP );
}
while ( FLA_Obj_length( CTL ) < FLA_Obj_length( C ) ) {
    b = min( FLA_Obj_length( CBR ), nb Alg );

    FLA_Report_2x2_to_3x3( CTL, /* */ CTR, &C00, /* */ &C01, &C02,
        /* */ &C10, /* */ &C11, &C12,
        CBL, /* */ CBR, &C20, /* */ &C21, &C22,
        b, b, FLA_BR );
    FLA_Report_2x1_to_3x1( AT, &A0,
        /* */ &A1,
        AB, &A2, b, FLA_BOTTOM );
*/

    FLA_Gemm( FLA_NO_TRANSPOSE, FLA_TRANSPOSE, ONE, A0, A1, ONE, C10 );
    FLA_Surf( FLA_LOWER_TRIANGULAR, FLA_NO_TRANSPOSE, ONE, A1, ONE, C11 );
*/

    FLA_Cont_with_3x3_to_2x2( &CTL, /* */ &CTR, C00, C01, /* */ C02,
        C10, C11, /* */ C12,
        /* */ &CBL, /* */ &CBR, C20, C21, /* */ C22,
        FLA_TL );

    FLA_Cont_with_3x1_to_2x1( &AT, A0,
        /* */ A1,
        /* */ &AB, A2, FLA_TOP );
}
```c
#pragma intel omp parallel taskq
{
    while (FLA_Obj_length(CTL) < FLA_Obj_length(C)) {
        b = min(FLA_Obj_length(CBR), nb_alg);

        FLA_Report_2x2_to_3x3(CTL, /**/ CTR, &C00, /**/ &C01, &C02,
                            /***********/                        /***********/
                            &C10, /**/ &C11, &C12,
                            CBL, /**/ CBR, &C20, /**/ &C21, &C22,
                b, b, FLA_BR );

        FLA_Report_2x1_to_3x1(AT, &A0,
                /** */                                    /** */
                &A1,
                AB, &A2, b, FLA_BOTTOM );

    } /* end task */
} /* end of taskq */
```
#pragma intel omp parallel taskq
{
    while ( FLA_Obj_length( AT ) < FLA_Obj_length( A ) ){
        b = min( FLA_Obj_length( AB ), nb_alg );

        FLA_Repatt_2x1_to_3x1( AT, &A0,
           /* ** */ /* ** */
           &A1,
           AB, &A2, b, FLA_BOTTOM );
        FLA_Repatt_2x2_to_3x3( CTL, /**/ CTR, &C00, /**/ &C01, &C02,
           /* ************** */ /* *************** */
           &C10, /**/ &C11, &C12,
           CBL, /**/ CBR, &C20, /**/ &C21, &C22, b, b, FLA_BR );

    } /*---------------------------------------------------------------*/

#pragma intel omp task capture private(A2, A1, C11, C21)
{
    FLA_Gemm( FLA_NO_TRANSPOSE, FLA_TRANSPOSE, ONE, A2, A1, ONE, C21 );
    FLA_Syrk( FLA_LOWER_TRIANGULAR, FLA_NO_TRANSPOSE, ONE, A1, ONE, C11 );
} /*---------------------------------------------------------------*/

    FLA_Cont_with_3x1_to_2x1( &AT, A0,
       /* ** */ /* ** */
       &A1,
       A1, &AB, A2, FLA_TOP );
    FLA_Cont_with_3x3_to_2x2( &CTL, /**/ &CTR, C00, C01, /**/ C02,
       /* ************** */ /* *************** */
       C10, C11, /**/ C12,
       CBL, /**/ CBR, &C20, C21, /**/ C22, FLA_TL );
}
Performance HP 4CPU Itanium2
dsyrk variant 2
Performance HP 4CPU Itanium2
dsyrk variant 3
Performance on NEC 16 CPU Itanium2 system (1.5 GHz)

OpenFLAME syrk_in_var2 performance (one task; outer panel-panel)

matrix dimension m

GFLOPS/sec.

Reference
OpenFLAME (n_th=4)
OpenFLAME (n_th=8)
OpenFLAME (n_th=12)
OpenFLAME (n_th=16)
Work queuing

Example:

Symmetric rank-k update
#pragma intel omp parallel taskq

{  
while ( FLA_Obj_length( AT ) < FLA_Obj_length( A ) )
{
    b = min( FLA_Obj_length( AB ), nb_alg );
    FLA_Repart_2x1_to_3x1( AT,  &A0,
                       /** */  /** */
                       &A1,
                       /** */  /** */
                       AB,  &A2,  b, FLA_BOTTOM );
    FLA_Repart_2x2_to_3x3( CTL, /** */  CTR,  
                           /** */  /** */
                           C00,  /** */  C01,  C02,
                           /** */  /** */
                           C10,  /** */  C11,  C12,
                           /** */  /** */
                           CBL,  /** */  CBR,  
                           /** */  /** */
                           C20,  /** */  C21,  C22,
                           b,  b, FLA_BR );

    b2 = FLA_Obj_length( A2 )/2;
    FLA_Part_2x1( A2,  &A2_T,
                  &A2_B,  b2, FLA_TOP );
    FLA_Part_2x1( C21,  &C21_T,
                  &C21_B,  b2, FLA_TOP );

    //----------------------------------------------------------------------------*/
    #pragma intel omp task captureprivate(A2_T, A1, C21_T)
    {
        FLA_Gemm( FLA_NO_TRANSPOSE, FLA_TRANSPOSE,
                  ONE, A2_T, A1, ONE, C21_T );
    }
    #pragma intel omp task captureprivate(A2_B, A1, C21_B)
    {
        FLA_Gemm( FLA_NO_TRANSPOSE, FLA_TRANSPOSE,
                  ONE, A2_B, A1, ONE, C21_B );
    }

    //----------------------------------------------------------------------------*/
    //----------------------------------------------------------------------------*/

    FLA_Comp_with_3x1_to_2x1(  &AT,
                               &A0,
                               /** */  /** */
                               &A1,
                               /** */  /** */
                               &AB,
                               /** */  /** */
                               &A2,  FLA_TOP );
    FLA_Comp_with_3x3_to_2x2( &CTL, /** */  &CTR,  
                              /** */  /** */
                              C00,  C01,  /** */  C02,
                              /** */  /** */
                              C10,  C11,  /** */  C12,
                              /** */  /** */
                              CBL,  /** */  CBR,  
                              /** */  /** */
                              C20,  C21,  /** */  C22,
                              FLA_TL );
}
Performance on NEC 16 CPU Itanium2 system (1.5 GHz)

OpenFLAME syrk_in_var2 performance (one task; outer panel-outer panel)

Reference
- OpenFLAME (n_th=4)
- OpenFLAME (n_th=8)
- OpenFLAME (n_th=12)
- OpenFLAME (n_th=16)

OpenFLAME syrk_in_var2 performance (two loops, split column tasks; outer panel-outer panel)

Reference
- OpenFLAME (n_th=4)
- OpenFLAME (n_th=8)
- OpenFLAME (n_th=12)
- OpenFLAME (n_th=16)
Syrk performance

FLAME workqueueing (with OpenMP) syrk_in_var2 performance (m = p, k = 200)
Cholesky Factorization Performance

FLAME workqueueing (with OpenMP) chol_l_var3 performance ($m = p$)

- 16 threads (100,−16,−16,−2)
- 16 threads (200,−16,−16,−2)
- 16 threads (300,−16,−16,−2)
- 16 threads (400,−16,−16,−2)
- 16 threads (MKL 7.0)

GFLOPS/sec. vs. problem size $p$
Cholesky Factorization Performance

FLAME workqueuing (with OpenMP) chol_u_var3 performance (m = p)

- 16 threads (100,−16,−16,−2)
- 16 threads (200,−16,−16,−2)
- 16 threads (300,−16,−16,−2)
- 16 threads (400,−16,−16,−2)
- 16 threads (MKL 7.0)
Switch to Demo
Show[
  GraphicsArray[
    {ListPlot[Table[{n, MaxIndex[schedule[oneTask[n], 8]]}, {n, 300, 2000, 1}],
      PlotRange -> {0, 8}, PlotStyle -> PointSize[0.004], PlotLabel -> '1 Task', DisplayFunction -> Identity],
  100%}
Conclusion

Mathematical specification of operation

**Mechanical** derivation of algorithms

Mathematical specification of architecture

**Mechanical** analysis of algorithms

Family of algorithms

Rewrite rules for language

**Mechanical** translation to code

Optimized library
Conclusion

- Mechanical generation of libraries supports
  - New architectures
    - Architectures currently supported: Sequential, SMP parallel, distributed memory parallel
  - New languages
    - Languages supported: LaTeX, C, Matlab, Fortran, C+MPI, C+OpenMP, Mathematica, Haskell, LabView’s G
  - New datastructures
    - Datastructures supported: column-major storage, banded, dense, stored by blocks, sparse hierarchical, out-of-core
  - New operations
    - Mechanical derivation of algorithms for all BLAS3, LAPACK, many operations in control theory
More Information

http://www.cs.utexas.edu/users/flame/

http://www.cs.utexas.edu/users/flame/pubs.html

rvdg@cs.utexas.edu
What is needed?

- Project so far concentrates on the science that supports the approach and prototyping of tools
- Full-blown integration of all tools requires
  - Full-time postdoc
  - Full-time professional programmer
  - Part-time web developer
  - Several graduate students
  - Collaboration with the Texas Advanced Computing Center