Three versions of High Performance Minimal Storage Cholesky Algorithm which uses New Data Structures: Recursion, Block Packed Hybrid and Rectangular Full Packed

Jerzy Waśniewski

Informatics & Mathematical Modeling

Technical University of Denmark

DK - 2800 Lyngby, Denmark

e-mail: jw@imm.dtu.dk

Parallel Processing and Applied Mathematics (PPAM 2007)

Gdańsk, Poland

September 9–12, 2007

In collaboration with:

(•) Bjarne Sig Andersen

Danish Meteorological Institute, Denmark

(•) Fred Gustavson and John Gunnel

IBM Watson Research Center, USA

(•) John K. Reid

Rutherford Appleton Laboratory, UK

(•) Plamen Yalamov, and Minka and Alexander Karaivanov

University of Rousse, Bulgaria

Slides:

http://www.imm.dtu.dk/~jw/Lectures/ppam07.pdf
http://www.imm.dtu.dk/~jw/Lectures/ppam07.ps.gz

Outline of my talk

- **The LAPACK Standard,**
- **Recursive Algorithm,**
- **Block Packed Hybrid Algorithm,**
- **Rectangular Full Packed Algorithm,**
- **Performance results between these three NEW and LAPACK Algorithms.**

LAPACK Data Formats

Present Standards of DLA

- LAPACK with Level 1, 2, and 3 BLAS (ATLAS)
  - Usually for single processors
  - Shared memory versions
  - Open MP
  - Partly for distributed memory versions
- ScaLAPACK with PBLAS, BLACS and MPI
  - Distributed memory computers
  - Network computing
- Harwell, NAG, IMSL, NetLib and others
- Computer vendor Libraries, i.e. ESSL and SunPerf

Symmetric/Hermitian matrix

$$\mathbf{n} = 7, \quad \mathbf{lda} = 9, \quad \mathbf{memory\ needed} = \mathbf{lda} \times \mathbf{n} = 63$$

$$\begin{pmatrix}
a_{1,1_1} & \diamond & \diamond & \diamond & \diamond & \diamond & \diamond \\
a_{2,1_2} & a_{2,2_{11}} & \diamond & \diamond & \diamond & \diamond & \diamond \\
a_{3,1_3} & a_{3,2_{12}} & a_{3,3_{21}} & \diamond & \diamond & \diamond & \diamond \\
a_{4,1_4} & a_{4,2_{13}} & a_{4,3_{22}} & a_{4,4_{31}} & \diamond & \diamond & \diamond \\
a_{5,1_5} & a_{5,2_{14}} & a_{5,3_{23}} & a_{5,4_{32}} & a_{5,5_{41}} & \diamond & \diamond \\
a_{6,1_6} & a_{6,2_{15}} & a_{6,3_{24}} & a_{6,4_{33}} & a_{6,5_{42}} & a_{6,6_{51}} & \diamond \\
a_{7,1_7} & a_{7,2_{16}} & a_{7,3_{25}} & a_{7,4_{34}} & a_{7,5_{43}} & a_{7,6_{52}} & a_{7,7_{61}} \\
\oslash & \oslash & \oslash & \oslash & \oslash & \oslash & \oslash \\
\oslash & \oslash & \oslash & \oslash & \oslash & \oslash & \oslash
\end{pmatrix}$$

The mapping of 7$\times$ 7 real symmetric or complex Hermitian matrix for the LAPACK algorithm using the full storage. Lower triangular case.

Symmetric/Hermitian matrix

$$\mathbf{n} = 7, \quad \mathbf{lda} = 9, \quad \mathbf{memory\ needed} = \mathbf{lda} \times \mathbf{n} = 63$$

$$\begin{pmatrix}
a_{1,1_1} & a_{1,2_{10}} & a_{1,3_{19}} & a_{1,4_{28}} & a_{1,5_{37}} & a_{1,6_{46}} & a_{1,7_{55}} \\
\diamond & a_{2,2_{11}} & a_{2,3_{20}} & a_{2,4_{29}} & a_{2,5_{38}} & a_{2,6_{47}} & a_{2,7_{56}} \\
\diamond & \diamond & a_{3,3_{21}} & a_{3,4_{30}} & a_{3,5_{39}} & a_{3,6_{48}} & a_{3,7_{57}} \\
\diamond & \diamond & \diamond & a_{4,4_{31}} & a_{4,5_{40}} & a_{4,6_{49}} & a_{4,7_{58}} \\
\diamond & \diamond & \diamond & \diamond & a_{5,5_{41}} & a_{5,6_{50}} & a_{5,7_{59}} \\
\diamond & \diamond & \diamond & \diamond & \diamond & a_{6,6_{51}} & a_{6,7_{60}} \\
\diamond & \diamond & \diamond & \diamond & \diamond & \diamond & a_{7,7_{61}} \\
\oslash & \oslash & \oslash & \oslash & \oslash & \oslash & \oslash \\
\oslash & \oslash & \oslash & \oslash & \oslash & \oslash & \oslash
\end{pmatrix}$$

The mapping of 7$\times$ 7 real symmetric or complex Hermitian matrix for the LAPACK algorithm using the full storage. Upper triangular case.

Symmetric/Hermitian matrix

$$\mathbf{n} = 7, \quad \mathbf{memory\ needed} = \mathbf{n} \times (\mathbf{n}+1)/2 = 28$$

$$\begin{pmatrix}
a_{1,1_1} & & & & & & \\
a_{2,1_2} & a_{2,2_8} & & & & & \\
a_{3,1_3} & a_{3,2_9} & a_{3,3_{14}} & & & & \\
a_{4,1_4} & a_{4,2_{10}} & a_{4,3_{15}} & a_{4,4_{19}} & & & \\
a_{5,1_5} & a_{5,2_{11}} & a_{5,3_{16}} & a_{5,4_{20}} & a_{5,5_{23}} & & \\
a_{6,1_6} & a_{6,2_{12}} & a_{6,3_{17}} & a_{6,4_{21}} & a_{6,5_{24}} & a_{6,6_{26}} & \\
a_{7,1_7} & a_{7,2_{13}} & a_{7,3_{18}} & a_{7,4_{22}} & a_{7,5_{25}} & a_{7,6_{27}} & a_{7,7_{28}}
\end{pmatrix}$$

The mapping of 7$\times$ 7 real symmetric or complex Hermitian matrix for the LAPACK algorithm using the packed storage. Lower triangular case.

Example; packed data storage

$$A = \begin{pmatrix} \underline{a_{11}} & a_{12} & a_{13} & a_{14} \\ a_{21} & \underline{a_{22}} & a_{23} & a_{24} \\ a_{31} & a_{32} & \underline{a_{33}} & a_{34} \\ a_{41} & a_{42} & a_{43} & \underline{a_{44}} \end{pmatrix}$$

$a_{ij} = \mathsf{conjg}(a_{ji})$ for $i, j = 1, \cdots, 4.$

UPLO = 'U'

a_{11}	a_{12}	a_{22}	a_{13}	a_{23}	a_{33}	a_{14}	a_{24}	a_{34}	a_{44}

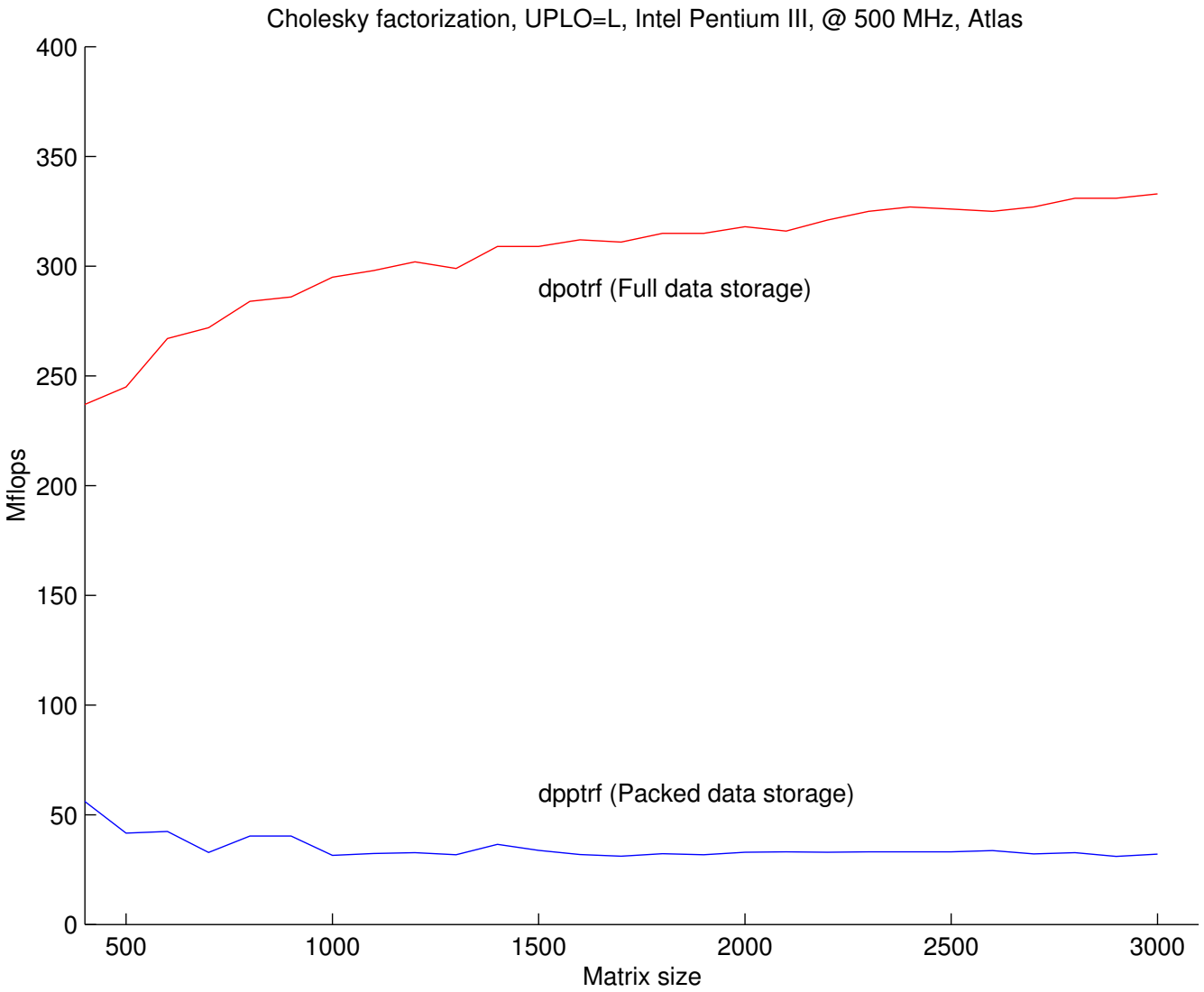
UPLO = 'L'

a_{11}	a_{21}	a_{31}	a_{41}	a_{22}	a_{32}	a_{42}	a_{33}	a_{43}	a_{44}

$size(\mathbf{AP}) = n(n+1)/2 = 10$

If $n = 1000$ **then** $size(\mathbf{AP}) = 500500$**, saving** 499500

Cholesky, a packed and full data storage



Recursive packed format

$LL^T = U^T U$ **Cholesky Decomposition**

$$AX = B$$

- A **– a symmetric or Hermitian, positive definite**
- X **and** B **– rectangular matrices or vectors**

⋆ $A = U^T U$**, if upper triangular part of** A **is given**

⋆ $A = L\,L^T$**, if lower triangular part of** A **is given**

⋆ U **– an upper triangular matrix**

⋆ L **– a lower triangular matrix**

⋆ $L = U^T$ **and** $U = L^T$

⋆ **The factored form of** A **is then used to** $A\,X = B$

Cholesky: $A = LL^T$

$$A = \begin{pmatrix} A_{11} & \\ A_{21} & A_{22} \end{pmatrix} \quad \text{and} \quad L = \begin{pmatrix} L_{11} & \\ L_{21} & L_{22} \end{pmatrix}$$

$$A = \begin{pmatrix} L_{11} & \\ L_{21} & L_{22} \end{pmatrix} \times \begin{pmatrix} L_{11}^T & L_{21}^T \\ & L_{22}^T \end{pmatrix}$$

$$A = \begin{pmatrix} A_{11} & A_{21} \\ A_{21} & A_{22} \end{pmatrix} = \begin{pmatrix} L_{11}L_{11}^T & L_{11}L_{21}^T \\ L_{21}L_{11}^T & L_{21}L_{21}^T + L_{22}L_{22}^T \end{pmatrix}$$

$$A_{11} = L_{11}L_{11}^T, \quad L_{21}L_{11}^T = A_{21} \quad \text{and} \quad \hat{A}_{22} = L_{22}L_{22}^T$$

where $\hat{A}_{22} := A_{22} - L_{21}L_{21}^T$

Cholesky: $A = LL^T$

$$A = \begin{pmatrix} A_{11} & \\ A_{21} & A_{22} \end{pmatrix} \quad \text{and} \quad L = \begin{pmatrix} L_{11} & \\ L_{21} & L_{22} \end{pmatrix}$$

Do recursion

- **if** $n > 1$ **then**
  - $L_{11} :=$ **rcholesky of** A_{11}
  - $L_{21}L_{11}^T = A_{21} \rightarrow$ **RTRSM**
  - $\hat{A}_{22} := A_{22} - L_{21}L_{21}^T \rightarrow$ **RSYRK**
  - $L_{22} :=$ **rcholesky of** $\hat{A}_{22}$
- **otherwise**
  - $L := \sqrt{A_{11}}$

End recursion

Cholesky: $A = U^T U$

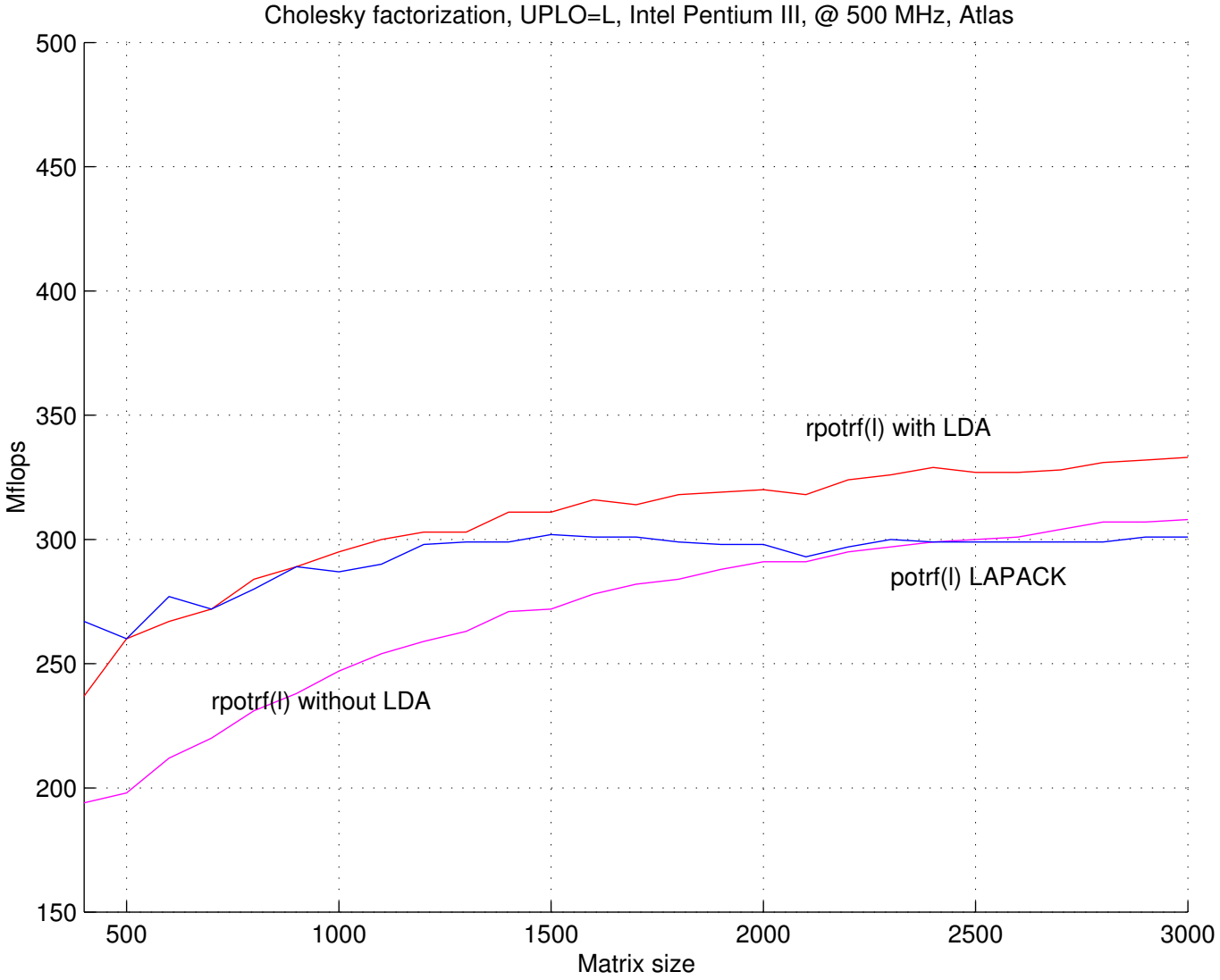
$$A = \begin{pmatrix} A_{11} & A_{12} \\ & A_{22} \end{pmatrix} \quad \text{and} \quad L = \begin{pmatrix} U_{11} & U_{12} \\ & U_{22} \end{pmatrix}$$

Do recursion

- **if** $n > 1$ **then**
  - U_{11} := **rcholesky of** A_{11}
  - $U_{11}^T U_{12} = A_{12} \rightarrow$ **RTRSM**
  - $\hat{A}_{22} := A_{22} - U_{12}^T U_{12} \rightarrow$ **RSYRK**
  - U_{22} := **rcholesky of** $\hat{A}_{22}$
- **otherwise**
  - $U := \sqrt{A_{11}}$

End recursion

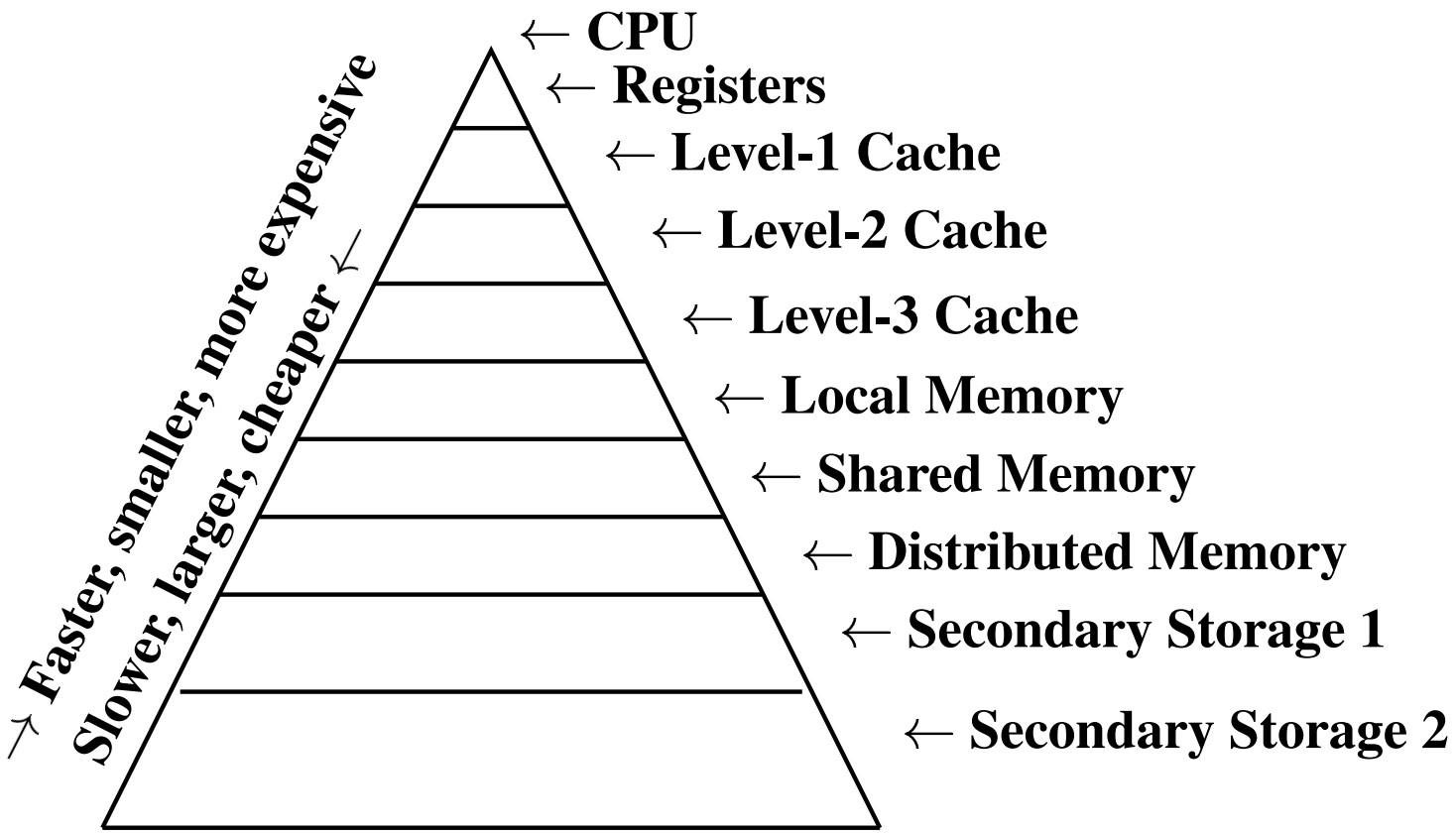
Cholesky: LAPACK full Storage



Reference

- **J. Waśniewski, B.S. Andersen and F. Gustavson. "Recursive Formulation of Cholesky Algorithm in Fortran 90". Proceedings of the Workshop on Applied Parallel Computing, Large Scale Scientific and Industrial Problems, PARA'98, 1998, Umeå, Sweden, Lecture Notes in Computer Science Number 1541, Springer, June, pages 574–578.**

A computer memory hierarchy



- **Floating point arithmetic cannot be done unless the operands involved first reside in the L1 (even L0) cache.**
- **Two dimensional Fortran and C arrays do not map nicely into L1 cache.**
  - **The best case happens when the array is contiguous and properly aligned.**
  - **At least three-way set associative cache is required when matrix multiply is being done.**

Cholesky: $A \approx LL^T$

$$A = \begin{pmatrix} A_{11} & \\ A_{21} & A_{22} \end{pmatrix} \quad \text{and} \quad L = \begin{pmatrix} L_{11} & \\ L_{21} & L_{22} \end{pmatrix}$$

Do recursion

- **if** $n > 1$ **then**
  - $L_{11} :=$ **rcholesky of** A_{11}
  - $L_{21}L_{11}^T = A_{21} \rightarrow$ **RTRSM**
  - $\hat{A}_{22} := A_{22} - L_{21}L_{21}^T \rightarrow$ **RSYRK**
  - $L_{22} :=$ **rcholesky of** $\hat{A}_{22}$
- **otherwise**
  - $L := \sqrt{A_{11}}$

End recursion

Cholesky, Packed Storage

$$\mathbf{n} = 7, \quad \textbf{memory needed} = \mathbf{n} \times (\mathbf{n}+1)/2 = 28$$

$$\begin{pmatrix}
a_{1,1_1} & & & & & & \\
a_{2,1_2} & a_{2,2_8} & & & & & \\
a_{3,1_3} & a_{3,2_9} & a_{3,3_{14}} & & & & \\
a_{4,1_4} & a_{4,2_{10}} & a_{4,3_{15}} & a_{4,4_{19}} & & & \\
a_{5,1_5} & a_{5,2_{11}} & a_{5,3_{16}} & a_{5,4_{20}} & a_{5,5_{23}} & & \\
a_{6,1_6} & a_{6,2_{12}} & a_{6,3_{17}} & a_{6,4_{21}} & a_{6,5_{24}} & a_{6,6_{26}} & \\
a_{7,1_7} & a_{7,2_{13}} & a_{7,3_{18}} & a_{7,4_{22}} & a_{7,5_{25}} & a_{7,6_{27}} & a_{7,7_{28}}
\end{pmatrix}$$

The mapping to array-subscript order of a 7×7 matrix for LAPACK Cholesky Algorithm using packed storage. Lower triangular case.

Cholesky, Recursive Packed Storage

$$\mathbf{n} = 7, \quad \textbf{memory needed} = \mathbf{n} \times (\mathbf{n}+1)/2 = 28$$

$$\left(\begin{array}{ccc|cccc}
a_{1,1_1} & & & & & & \\
a_{2,1_2} & a_{4,1_4} & & & & & \\
a_{3,1_3} & a_{5,1_5} & a_{6,1_6} & & & & \\
\hline
a_{7,1_7} & a_{5,2_{11}} & a_{4,3_{15}} & a_{4,4_{19}} & & & \\
a_{2,2_8} & a_{6,2_{12}} & a_{5,3_{16}} & a_{5,4_{20}} & a_{5,5_{23}} & & \\
a_{3,2_9} & a_{7,2_{13}} & a_{6,3_{17}} & a_{6,4_{21}} & a_{6,5_{24}} & a_{6,6_{26}} & \\
a_{4,2_{10}} & a_{3,3_{14}} & a_{7,3_{18}} & a_{7,4_{22}} & a_{7,5_{25}} & a_{7,6_{27}} & a_{7,7_{28}}
\end{array}\right)$$

The mapping to array-subscript order of a 7×7 matrix for the Cholesky Algorithm using the recursive packed storage. The recursive block division is illustrated. Lower triangular case.

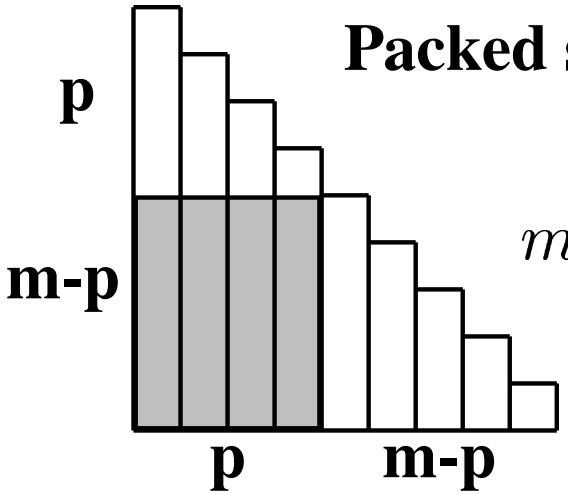
Cholesky, Recursive Packed Storage

$$\mathbf{n} = 7, \quad \textbf{memory needed} = \mathbf{n} \times (\mathbf{n}+1)/2 = 28$$

$$\left(\begin{array}{ccc|ccc|ccc|c}
a_{1,1_1} & & & & & & & & & \\
\hline
a_{2,1_2} & a_{4,1_4} & & & & & & & & \\
\hline
a_{3,1_3} & a_{5,1_5} & a_{6,1_6} & & & & & & & \\
\hline
a_{7,1_7} & a_{5,2_{11}} & a_{4,3_{15}} & a_{4,4_{19}} & & & & & & \\
\hline
a_{2,2_8} & a_{6,2_{12}} & a_{5,3_{16}} & a_{5,4_{20}} & a_{6,4_{21}} & & & & & \\
\hline
a_{3,2_9} & a_{7,2_{13}} & a_{6,3_{17}} & a_{7,4_{22}} & a_{6,5_{24}} & a_{6,6_{26}} & & & & \\
\hline
a_{4,2_{10}} & a_{3,3_{14}} & a_{7,3_{18}} & a_{5,5_{23}} & a_{7,5_{25}} & a_{7,6_{27}} & a_{7,7_{28}} & & &
\end{array}\right)$$

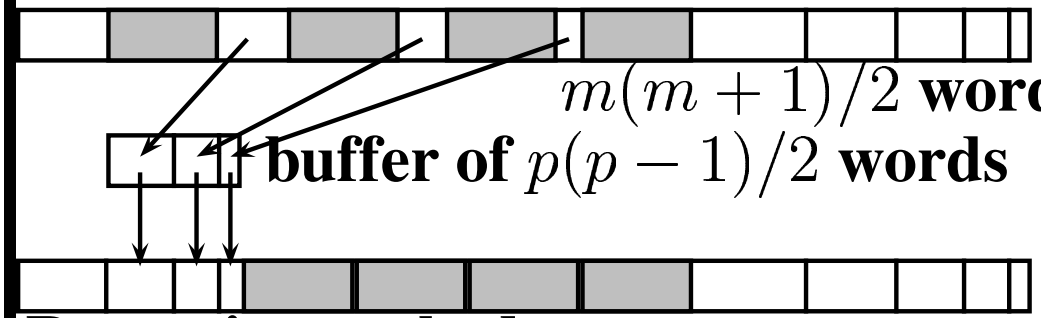
The mapping to array-subscript order of a 7×7 matrix for the Cholesky Algorithm using the recursive packed storage. The recursive block division is illustrated. Lower triangular case.

Cholesky, Recursive Packed Storage



$m = 9, \quad p = [m/2] = 4$

LAPACK packed storage memory map



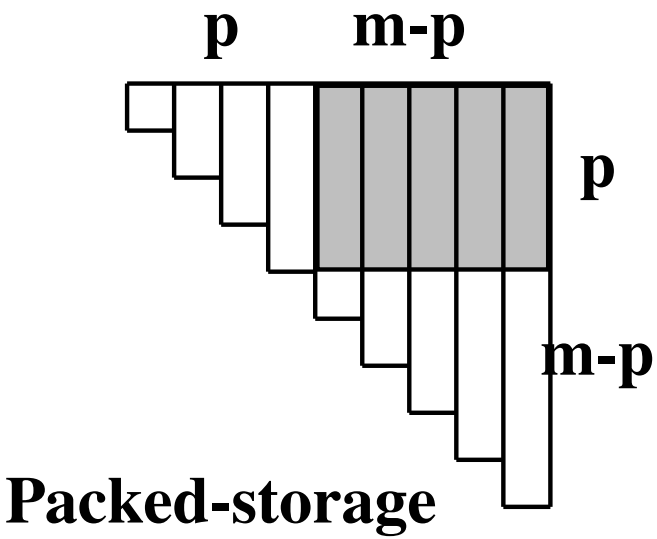
$m(m+1)/2$ **words**

buffer of $p(p-1)/2$ **words**

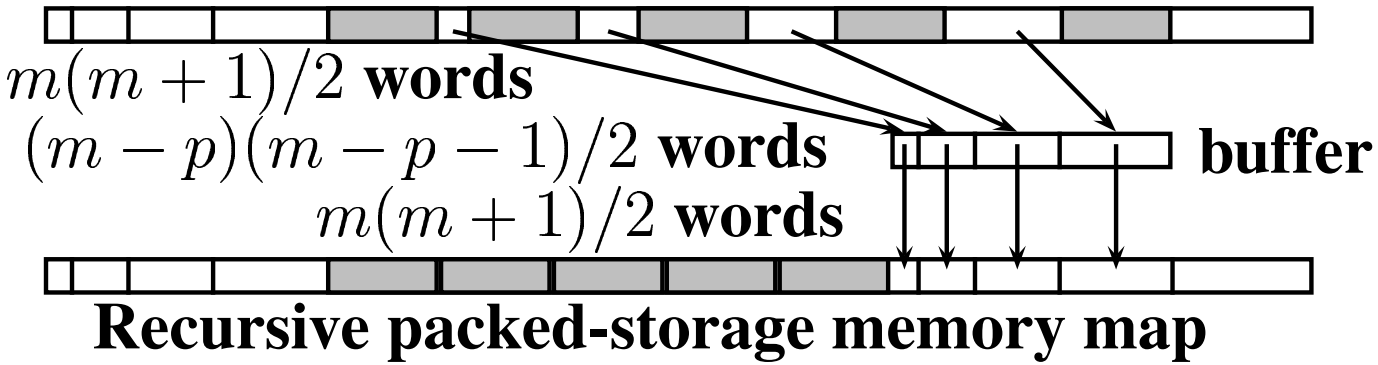
Recursive packed storage memory map

$m(m+1)/2$ **words**

Cholesky, Recursive Packed Storage



LAPACK packed-storage memory map



Cholesky: $A \approx LL^T$

$$A = \begin{pmatrix} A_{11} & \\ A_{21} & A_{22} \end{pmatrix} \quad \text{and} \quad L = \begin{pmatrix} L_{11} & \\ L_{21} & L_{22} \end{pmatrix}$$

Do recursion

- **if** $n > 1$ **then**
  - $L_{11} :=$ **rcholesky of** A_{11}
  - $L_{21}L_{11}^T = A_{21} \rightarrow$ **RTRSM**
  - $\hat{A}_{22} := A_{22} - L_{21}L_{21}^T \rightarrow$ **RSYRK**
  - $L_{22} :=$ **rcholesky of** $\hat{A}_{22}$
- **otherwise**
  - $L := \sqrt{A_{11}}$

End recursion

LAPACK Packed and Recursive Packed Storage

Two things are important:

- **The BLAS**
- **The Data storage**

- **LAPACK uses Level2 BLAS,**

  we use Level 3 BLAS.

- **We use different data storage,**

  recursive packed data storage.

Recursive Packed BLAS

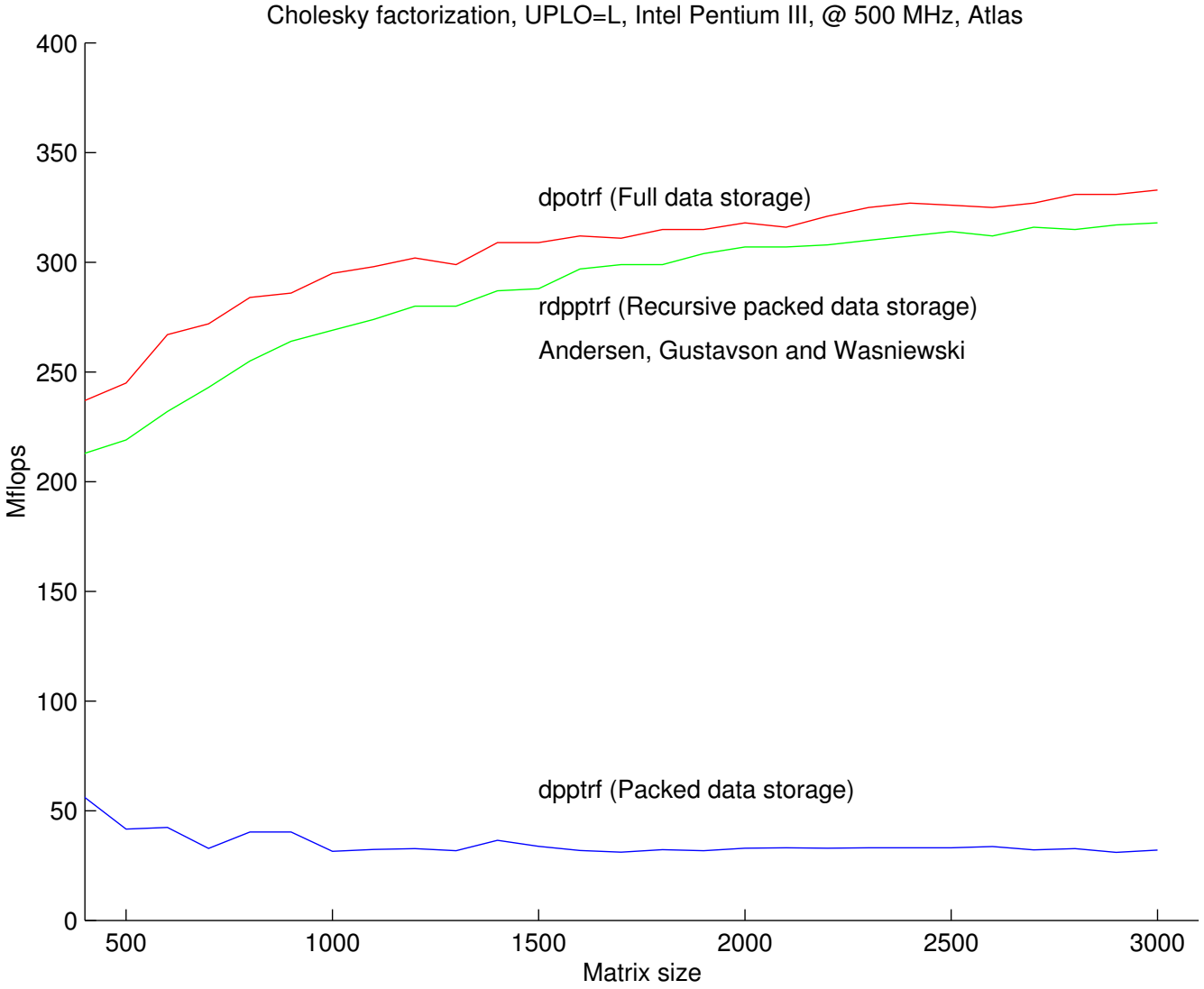
Three recursive level-3 BLAS will be explained:

- **_RPTRSM, _RPSYRK, and _RPTRMM.**

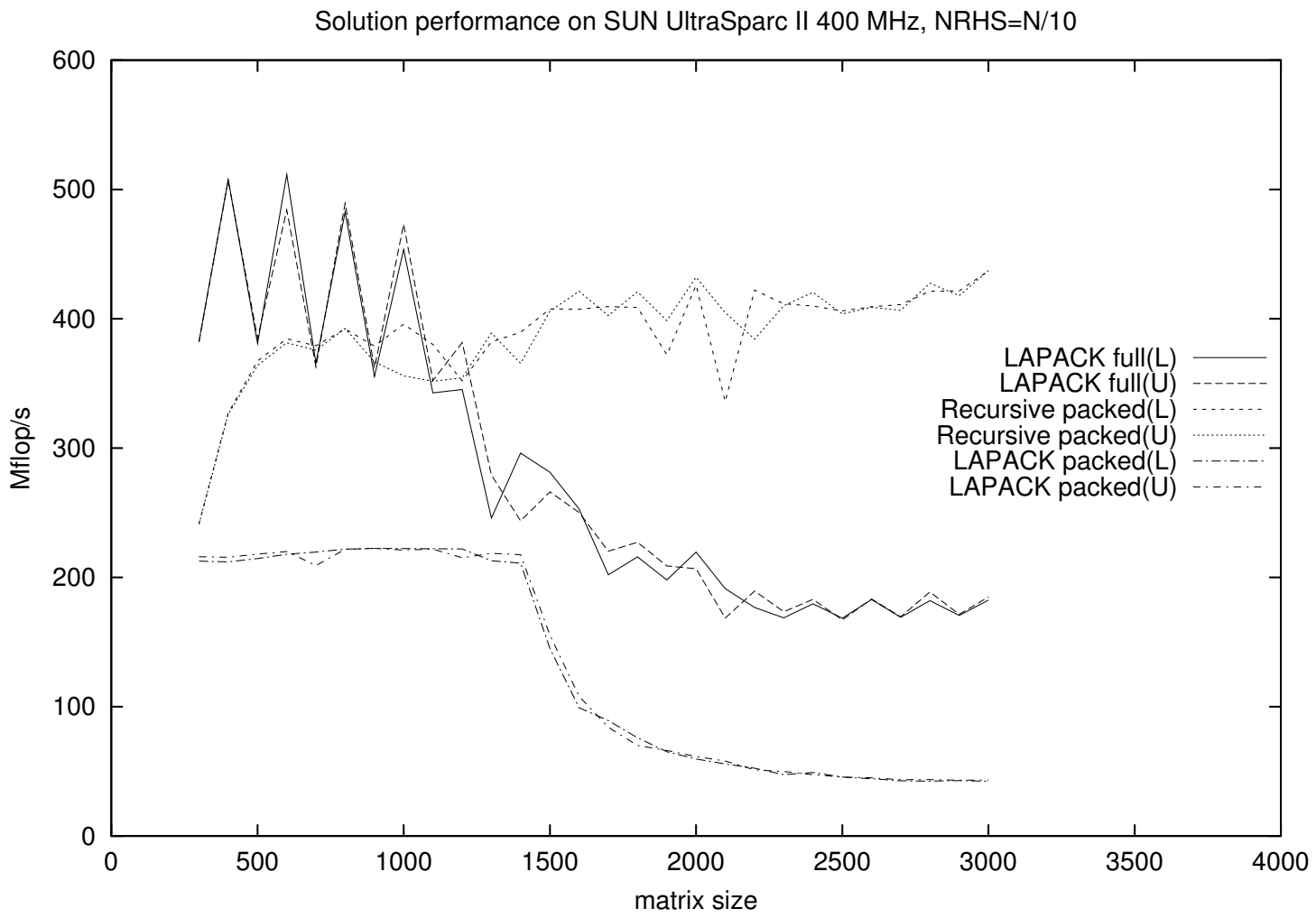
These three BLAS do all the work through:

- **_GEMM**
  - **ATLAS: Automatically tuned linear algebra software.**
  - **PHIPAC: Optimizing matrix multiply.**
  - **GOTO: Kazushige Goto at the University of Texas at Austin.**
  - **Computer vendor libraries.**

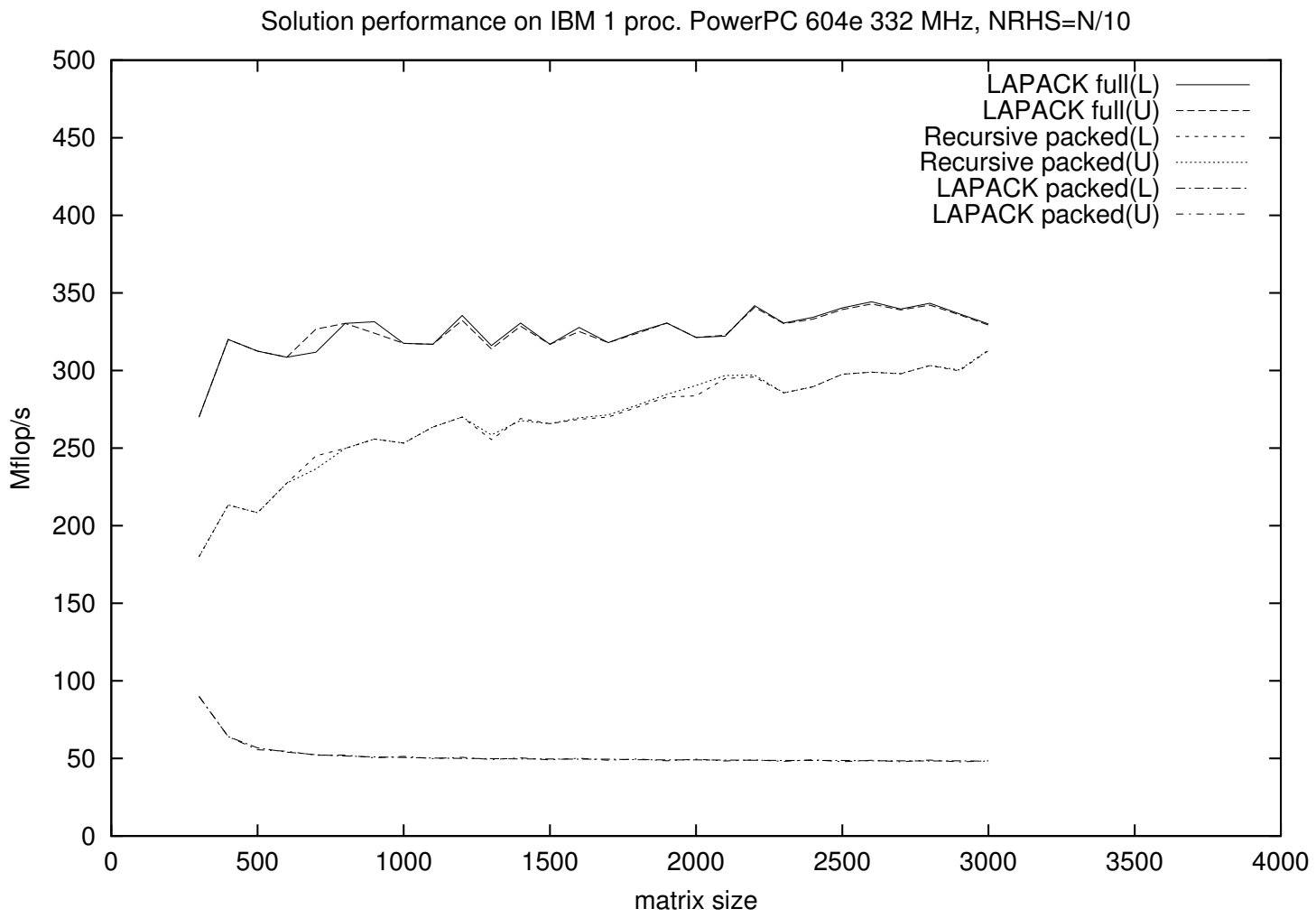
Cholesky: packed vs. full data storage



Cholesky, Recursive Packed Storage



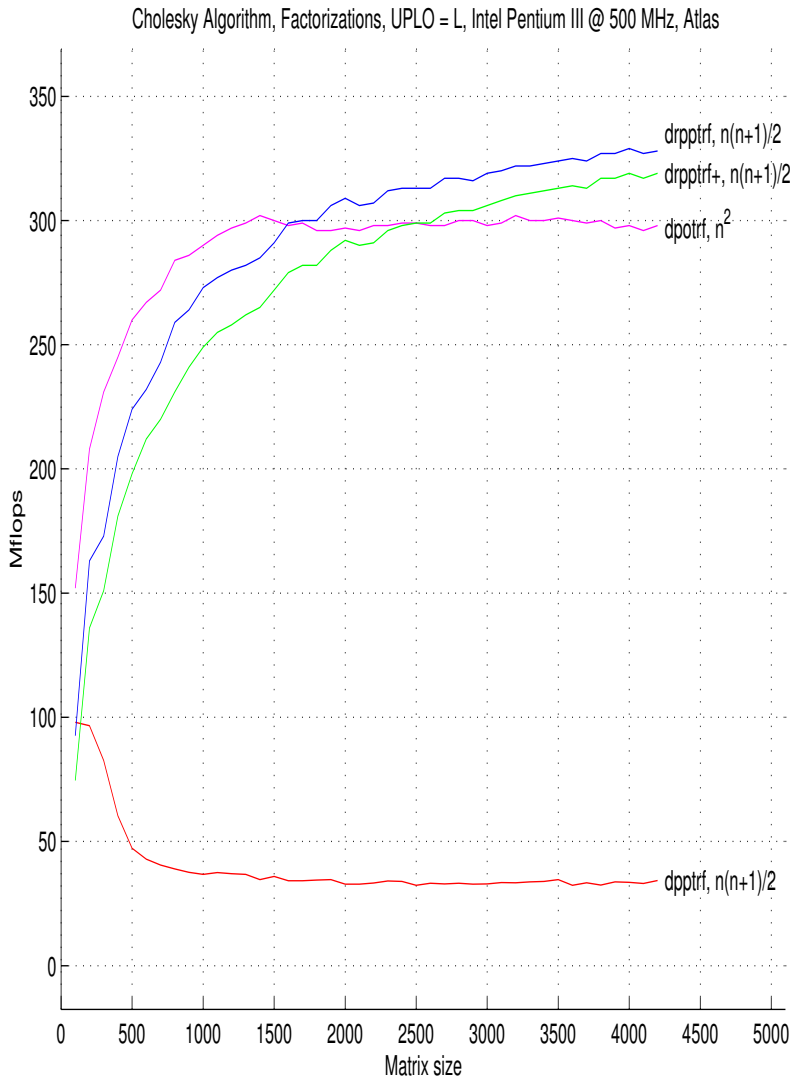
Cholesky, Recursive Packed Storage

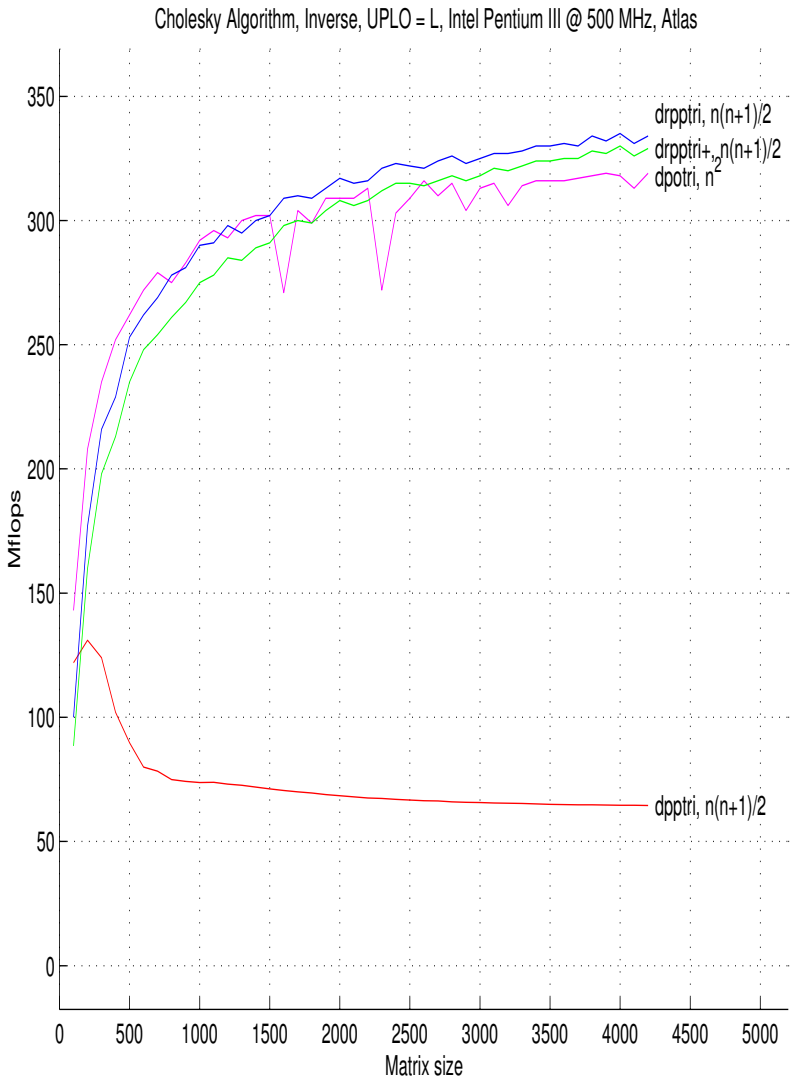


References

- **F.G. Gustavson. "Recursion Leads to Automatic Variable Blocking for Dense Linear-Algebra Algorithms". IBM Journal of Research and Development, 41(6), November 1997.**
- **Bjarne S. Andersen, Fred G. Gustavson, and Jerzy Waśniewski. "A recursive formulation of Cholesky factorization of a matrix in packed storage". ACM Transactions on Mathematical Software, 27(2):214–244, June 2001**

Intel Pentium III, @ 500 MHz, ATLAS Library





Reference

- **Bjarne S. Andersen, John A. Gunnels, Fred Gustavson, and Jerzy Waśniewski, "A Recursive Formulation of the Inversion of Symmetric Positive Definite Matrices in Packed Storage Data Format". In Para'2002 Conference Proceedings, Espoo, Finland, June 2002.**

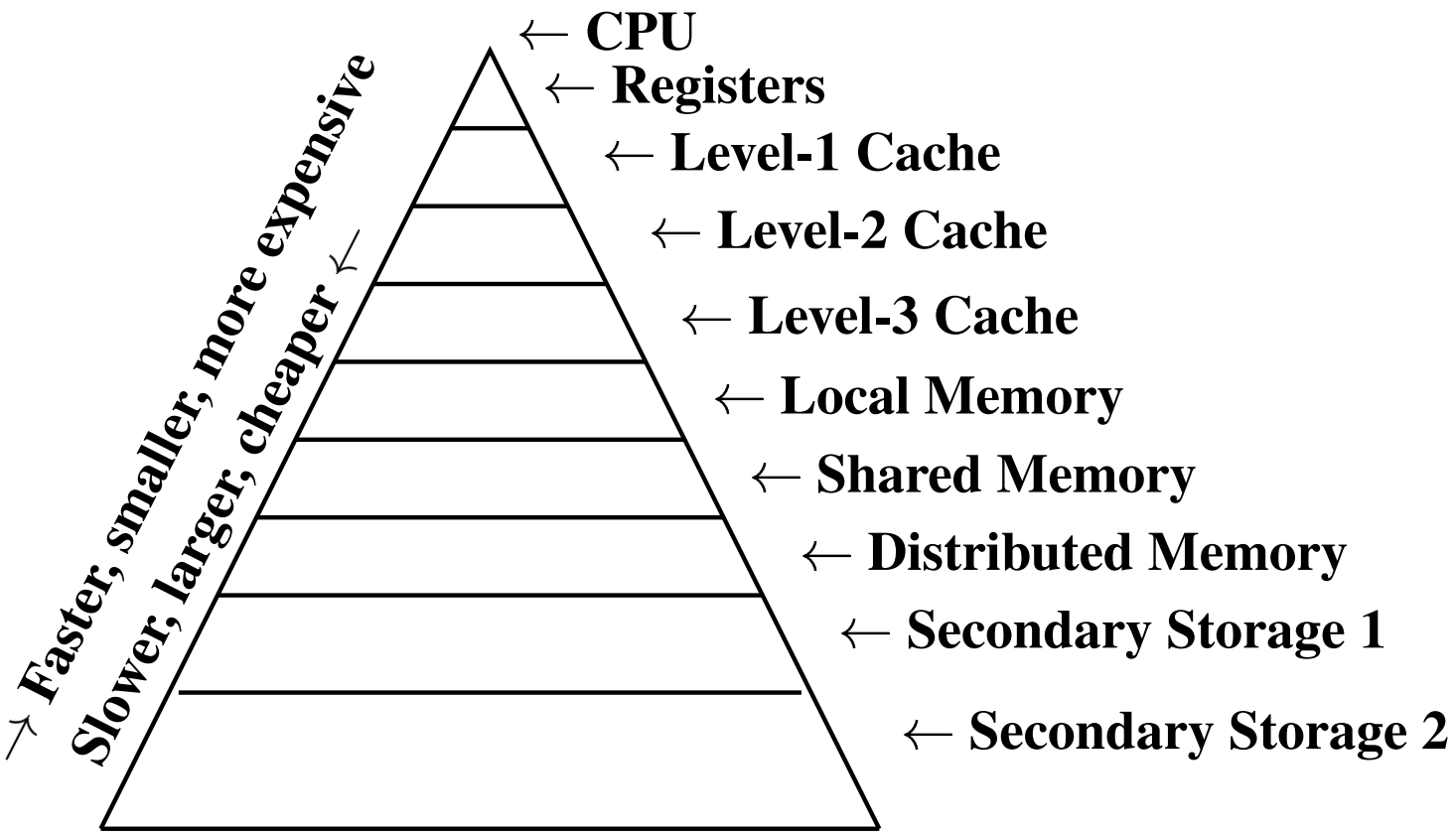
Symmetric/Hermitian Positive Definite Matrices

Block Packed Hybrid Storage Data Format

Joint work with

Fred Gustavson, John Reid,
Bjarne Andersen and John Gunnels

A computer memory hierarchy



Cholesky, Block Packed Hybrid Storage

Block size nb chosen to fit comfortably in level-1 cache.

Note that _GEMM:

$$C = aAB + bC$$

is efficient if A can be retained in cache while columns of B and C are 'streamed' in and columns of the modified C are streamed out. Important that B and C be held by columns.

The consequence for the lower blocked format is that each block should be held by rows.

For efficient rearrangement want no element to be moved out of its block column.

Cholesky, Packed Storage

$$\mathbf{n} = 7, \quad \mathbf{memory\ needed} = \mathbf{n} \times (\mathbf{n}+1)/2 = 28$$

$$\begin{pmatrix}
a_{1,1_1} & & & & & & \\
a_{2,1_2} & a_{2,2_8} & & & & & \\
a_{3,1_3} & a_{3,2_9} & a_{3,3_{14}} & & & & \\
a_{4,1_4} & a_{4,2_{10}} & a_{4,3_{15}} & a_{4,4_{19}} & & & \\
a_{5,1_5} & a_{5,2_{11}} & a_{5,3_{16}} & a_{5,4_{20}} & a_{5,5_{23}} & & \\
a_{6,1_6} & a_{6,2_{12}} & a_{6,3_{17}} & a_{6,4_{21}} & a_{6,5_{24}} & a_{6,6_{26}} & \\
a_{7,1_7} & a_{7,2_{13}} & a_{7,3_{18}} & a_{7,4_{22}} & a_{7,5_{25}} & a_{7,6_{27}} & a_{7,7_{28}}
\end{pmatrix}$$

The mapping to array-subscript order of a 7×7 matrix for LAPACK Cholesky Algorithm using packed storage. Lower triangular case.

Cholesky, Block Packed Hybrid Storage

$$\mathbf{n} = 7, \quad \textbf{memory needed} = \mathbf{n} \times (\mathbf{n}+1)/2 = 28$$

$$\left(\begin{array}{ccc|ccc|c}
a_{1,1_1} & & & & & & \\
a_{2,1_2} & a_{3,1_3} & & & & & \\
a_{4,1_4} & a_{5,1_5} & a_{6,1_6} & & & & \\
\hline
a_{7,1_7} & a_{2,2_8} & a_{3,2_9} & a_{4,4_{19}} & & & \\
a_{2,4_{10}} & a_{2,5_{11}} & a_{2,6_{12}} & a_{5,4_{20}} & a_{6,4_{21}} & & \\
a_{2,7_{13}} & a_{3,3_{14}} & a_{3,4_{15}} & a_{7,4_{22}} & a_{5,5_{23}} & a_{5,6_{24}} & \\
\hline
a_{3,5_{16}} & a_{3,6_{17}} & a_{3,7_{18}} & a_{7,5_{25}} & a_{6,6_{26}} & a_{7,6_{27}} & a_{7,7_{28}}
\end{array}\right)$$

The mapping to array-subscript order of a 7×7 matrix for LAPACK Cholesky Algorithm using block packed hybrid storage. Lower triangular case. Each block held by rows. Sort can be done using a buffer of size $n \times nb$. Here nb = 3.

Cholesky code, lower block hybrid format

do j = 1, $\lceil n/nb \rceil$

  do k = 1, j-1

    $A_{jj} = A_{jj} - L_{jk} L_{jk}^T$ **! Call of level-3 BLAS _SYRK**

    do i = j+1, $\lceil n/nb \rceil$

      $A_{ij} = A_{ij} - L_{ik} L_{jk}^T$ **! Call of level-3 BLAS _GEMM**

    end do

  end do

  $L_{jj} L_{jj}^T = A_{jj}$ **! Call of LAPACK _POTRF**

  do i = j+1, $\lceil n/nb \rceil$

    $L_{ij} L_{jj}^T = A_{ij}$ **! Call of level-3 BLAS _TRSM**

  end do

end do

LL^T Implementation for Lower Blocked Hybrid Format

do j **= 1,** l **!** $l = \lceil n/nb \rceil$
  do k **= 1,** $j-1$
    $A_{jj} = A_{jj} - L_{jk}L_{jk}^T$ **! Call of Level-3 BLAS** `_SYRK`
    do $i = j+1, l$
    $A_{ij} = A_{ij} - L_{ik}L_{jk}^T$ **! Call of Level-3 BLAS** `_GEMM`
    end do
  end do
  $L_{jj}L_{jj}^T = A_{jj}$ **! Call of Cholesky** `Kernel` **subroutine**
  do $i = j+1, l$
    $L_{ij}L_{jj}^T = A_{ij}$ **! Call of Level-3 BLAS** `_TRSM`
  end do
end do

Cholesky code, lower block hybrid format

Can implement the inner loops (index i) by a single call of _GEMM and _TRSM.

A_{jj} then L_{jj} held in buffer in full format and used for _POTRF and _TRSM.

For large problems, most of the work is in _GEMM. This is performed in contiguous memory and is actually called for the transpose:

$$A_{ij}^T = A_{ij}^T - L_{jk} L_{ik}^T$$

Note that L_{jk} is held by rows and A_{ij}^T and L_{ik}^T are held by columns. The operation can therefore be applied efficiently with streaming.

Kernel subroutines

For factorizing the diagonal blocks, LAPACK's _POTRF is unsatisfactory since it uses the level-2 BLAS _POTF2.

We can expect the matrix to be held in level-1 cache, so it is register usage that is important.

Fred proposed writing a block code in Fortran with very small block size.

All the BLAS are replaced by in-line code with inner loops unrolled.

We have found block size $kp = 2$ to be adequate.

Kernel code

The code takes the form:

do i = 1, $\lceil n/kb \rceil$

$A_{ii} = A_{ii} - \sum_{k=1}^{i-1} (U_{ki}^T U_{ki})$ **! Like level-3 BLAS _SYRK**

$U_i i^T U_{ii} = A_{ii}$ **! Cholesky factorization of block**

do j = i+1, nb

$A_{ij} = A_{ij} - \sum_{k=1}^{i-1} (U_{ki}^T U_{kj})$ **! Like level-3 BLAS _GEMM**

$U_{ii}^T U_{ij} = A_{ij}$ **! Like level-3 BLAS _TRSM**

end do

end do

The key loop is the one that corresponds to _GEMM. For this, the following code is suitable

```
do k = 1, ii - 1
   aki = a(k,ii)
   akj = a(k,jj)
   t11 = t11 - aki*akj
   aki1 = a(k,ii+1)
   t21 = t21 - aki1*akj
   akj1 = a(k,jj+1)
   t12 = t12 - aki*akj1
   t22 = t22 - aki1*akj1
end do
```

8 local variables, hopefully in registers, 4 memory accesses, 8 flops. Also tried inner loop that updates A_{ij} and $A_{i,j+1}$. aki and $aki1$ need to be loaded only once, so get 14 local variables, 6 memory accesses, 8 flops.

Performance in Mflops of the Kernel Cholesky Algorithm. Comparison between different computers and different versions of subroutines.

Order	LAPACK		Recur-	Mini-block	
	Vendor	Comp.	sive	2×2	2×2 & 2×4
IBM Power4, 1700 MHz, ESSL Library:					
40	1658	1503	707	1999	1999
72	2653	2303	1447	2751	2753
100	3037	2481	1930	2957	2945
SUN Ultra III, 900 MHz, Sunperf BLAS Library:					
40	392	427	251	803	941
72	598	664	417	1191	1012
100	619	830	589	1143	1506
HP Itanium 2, 1000 MHz, HP MLIB BLAS Library:					
40	449	448	153	1125	1133
72	597	595	266	1711	1722
100	567	559	364	2103	2103

Cholesky solution, single right-hand side

do j = 1, $\lceil n/nb \rceil$

$L_{jj}Y_j = B_j$ **! Call of level-2 BLAS _TPSV**

$B_i = B_i - L_{ij}Y_j, \forall i > j$ **! Single call of level-2 BLAS _GEMV**

end do

do j = $\lceil n/nb \rceil$**, 1, -1**

$Y_j = Y_j - \sum_{i>j}(L_{ij}^T X_i)$ **! Single call of level-2 B LAS _GEMV**

$L_{jj}X_j = Y_j$ **! Call of level-2 BLAS _TPSV**

end do

Similar code could be applied for many rhs, with BLAS3 codes replacing BLAS2. However, the blocks B_i **,** Y_i **, and** X_i **would not occupy contiguous memory.**

Cholesky solution, many right-hand sides

If the number of columns m is modest, we therefore make a copy of B as a block matrix, with each block B_i held contiguously by columns:

$$\begin{array}{cccc} 0 & 3 & 6 & 9 \\ 1 & 4 & 7 & 10 \\ 2 & 5 & 8 & 11 \\ \hline 12 & 15 & 18 & 21 \\ 13 & 16 & 19 & 22 \\ 14 & 17 & 20 & 23 \\ \hline 24 & 25 & 26 & 27 \end{array}$$

Have a single level-3 BLAS call for each block, but each matrix is held contiguously, and streaming is available.

Upper packed format

Essentially the same advantages are obtained by doing a UU^T factorization, which corresponds to a backward pivot sequence.

Equally good numerically, but rejected since not compatible with what LAPACK does.

Forced us to abandon the property of contiguous blocks forming an array. No merging of BLAS3 calls.

Computers Used

Processor	MHz	Peak Mflops	Cache sizes level-1	level-2	level-3	TLB entries
IBM Power4	1700	6800	64K	1.5M*	32M*	1024
SUN UltSparc	900	1800	64K	8M	None	512
HP Itanium 2	1000	4000	32K	256K	1.5M	128
SGI R12000	300	600	32K	8M	None	64
HP Alpha EV6	500	1000	64K	4M	None	128
INTEL Pent. 3	500	500	16K	512K	None	32

*Shared with another processor.

Mflops, Cholesky factorizations including rearrangement, different nb values, IBM Power4.

n	40	64	100	160	250	400	640	1000	1600	2500	4000
Lower Packed Hybrid											
nb=40	956	1444	2055	2679	3156	3626	3861	3933	3968	3960	4040
nb=72	957	1520	1972	2650	3100	3626	3971	4086	4162	4166	4292
nb=100	943	1494	2103	2741	3193	3715	3971	4119	4162	4117	4258
nb=200	947	1515	2116	2417	2895	3440	3786	4059	4213	4322	4500
Upper Packed Hybrid											
nb=40	1117	1425	1916	2523	2911	3404	3598	3655	3710	3720	3716
nb=72	1111	1743	2052	2590	3006	3489	3761	3947	4015	4084	4102
nb=100	1102	1732	2376	2831	3159	3678	3832	4033	4112	4150	4191
nb=200	1105	1732	2375	2584	2997	3447	3761	4033	4266	4340	4481

Mflops, Cholesky factorizations including rearrangement, different nb values, SUN Ultra III.

n	40	64	100	160	250	400	640	1000	1600	2500	4000
Lower Packed Hybrid											
nb=40	394	548	644	773	832	963	1046	1106	1110	949	842
nb=72	391	558	660	760	857	993	1107	1182	1215	1120	1045
nb=100	389	557	708	738	824	959	1095	1115	1137	1144	1083
nb=200	390	557	708	782	867	965	1080	1180	1201	1206	1254
Upper Packed Hybrid											
nb=40	526	608	680	804	830	943	1006	1040	1052	915	791
nb=72	520	769	767	838	916	1032	1104	1182	1168	1123	1004
nb=100	516	770	957	841	901	1026	1116	1177	1213	1145	1065
nb=200	518	768	954	988	983	1032	1144	1181	1240	1280	1243

Mflops, Cholesky factorizations, lower case, nb = 100, IBM Power4.

n	40	64	100	160	250	400	640	1000	1600	2500	4000
p. lapack	747	951	1043	1024	1059	1101	1037	709	638	621	635
v. p. lapack	1750	2359	2658	2346	3107	3560	3773	3870	3969	3815	3836
f. lapack	440	722	1390	2119	2562	3242	3495	3797	3901	3787	4010
v. f. lapack	1492	2165	2486	3194	3454	3677	3832	3921	4162	4037	4327
p. recursive+	170	379	593	1024	1586	2077	2621	3030	3434	3555	3943
p. recursive	181	406	618	1060	1652	2133	2700	3111	3523	3604	3980
p. hybrid+	878	1488	2085	2721	3211	3754	3974	4112	4188	4200	4275
p. hybrid	1006	1717	2334	2977	3441	3938	4149	4279	4266	4269	4309

Mflops, Cholesky Factorizations, lo. case, nb = 200, SUN UltraSPARC III.

n	40	64	100	160	250	400	640	1000	1600	2500	4000
p. lapack	183	247	296	312	321	325	328	331	315	200	169
f. lapack	299	436	637	842	933	1086	864	1173	1203	1169	1236
p. recursive+	95	202	275	426	550	727	913	1010	1093	1118	1215
p. recursive	102	219	290	454	581	760	945	1043	1146	1162	1249
p. hybrid+	390	557	708	782	867	965	1080	1180	1201	1206	1254
p. hybrid	529	778	959	973	1003	1077	1164	1267	1277	1257	1304

Mflops, Cholesky Factorizations, lower case, nb = 200, HP Itanium 2.

n	40	64	100	160	250	400	640	1000	1600	2500	4000
p. lapack	218	328	463	644	825	952	1039	660	609	595	590
f. lapack	376	581	838	1307	1728	2189	2546	2777	2904	3028	3141
p. recursive+	110	232	378	698	1079	1608	2170	2531	2758	2861	3013
p. recursive	121	256	411	742	1147	1675	2305	2666	2874	2942	3056
p. hybrid+	657	1017	1689	1731	1279	1384	1552	1829	2089	2216	2402
p. hybrid	763	1139	1849	1815	1390	1504	1661	1910	2167	2274	2438

Mflops, Solution, many right-hand sides, Notes: Results for Vendor Packed Lapack L and Vendor Full Lapack very similar to corresponding Lapack results. nb = 100, mb = 100, IBM Power4.

n	40	64	100	160	250	400	640	1000	1600	2500	4000
p. lapack L	919	1298	1577	1822	2027	2182	2088	1733	1526	1460	1456
p. lapack U	912	1296	1572	1836	2044	2197	2088	1710	1517	1467	1469
f. lapack L	3195	3368	3750	4021	4210	4266	4259	4257	4441	4320	4368
f. lapack U	3205	3392	3740	4045	4175	4238	4259	4222	4495	4320	4353
p. recursive L	1240	1510	1956	2658	3075	3306	3524	3734	4147	4166	4571
p. recursive U	1225	1493	1935	2655	3091	3306	3510	3750	4147	4166	4555
P. Hybrid L	2359	2941	3411	3778	3991	4084	4231	4285	4362	4422	4620
P. Hybrid U	2374	2927	3389	3736	3965	4132	4231	4293	4415	4380	4555

Mflops, Solution, many right-hand sides, nb = 200, mb = 100, SUN UltraSPARC III.

n	40	64	100	160	250	400	640	1000	1600	2500	4000
p. lapack L	289	341	373	265	220	209	203	199	119	63	54
p. lapack U	276	326	357	294	271	277	281	270	179	99	87
f. lapack L	901	933	1045	1089	1103	1279	1238	1300	1268	1312	1299
f. lapack U	894	974	1042	1095	1103	1276	1244	1312	1261	1390	1290
p. recursive L	280	467	489	678	849	981	1081	1241	1296	1362	1436
p. recursive U	267	466	477	659	843	968	1071	1239	1291	1360	1434
p. hybrid L	766	852	959	1023	1083	1236	1317	1334	1373	1420	1436
p. hybrid U	767	858	959	1031	1074	1234	1316	1336	1401	1453	1435

Mflops, Solution, many right-hand sides, nb = 200, mb = 100, HP Itanium 2.

n	40	64	100	160	250	400	640	1000	1600	2500	4000
p. lapack L	223	305	385	471	538	576	608	628	640	649	655
p. lapack U	225	308	385	465	532	578	614	632	642	648	655
f. lapack L	272	411	593	844	1150	1525	1878	2222	2452	2637	2813
f. lapack U	271	406	593	835	1157	1538	1881	2222	2452	2626	2813
p. recursive L	712	958	1325	1641	2071	2440	2678	2765	3015	3094	3062
p. recursive U	695	958	1289	1617	2077	2392	2636	2784	3015	3094	3062
p. hybrid L	251	396	581	817	1103	1473	1853	2196	2560	2765	2976
p. hybrid U	251	396	577	808	1103	1463	1853	2193	2544	2777	2969

Mflops, Solution, one right-hand side, nb = 200, SUN UltraSPARC III.

n	40	64	100	160	250	400	640	1000	1600	2500	4000
p. lapack L	285	339	383	268	221	208	204	199	84	56	51
p. lapack U	270	325	371	290	274	280	281	280	154	96	85
f. lapack L	363	479	575	657	665	798	798	495	370	285	286
f. lapack U	373	472	601	698	667	786	791	540	340	306	288
p. recursive L	71	113	155	229	299	404	504	560	318	258	252
p. recursive U	72	114	155	229	306	405	505	566	314	263	277
p. hybrid L	247	310	354	294	338	411	519	554	348	265	229
p. hybrid U	240	309	352	292	337	411	520	584	325	231	224

Mflops, Solution, one right-hand side, nb = 200, HP Itanium 2.

n	40	64	100	160	250	400	640	1000	1600	2500	4000
p. lapack L	221	301	382	471	518	587	655	622	640	649	656
p. lapack U	225	301	385	462	542	584	738	663	657	653	652
f. lapack L	181	263	341	452	522	625	317	307	314	285	258
f. lapack U	183	261	344	438	554	603	344	312	317	287	255
p. recursive L	76	118	173	268	370	529	615	662	779	886	999
p. recursive U	72	118	169	258	382	527	591	658	771	888	1001
p. hybrid L	194	279	371	449	618	847	743	649	624	620	789
p. hybrid U	193	280	370	450	620	855	776	670	640	633	789

Mflops, Solution, one right-hand side, nb = 100, IBM Power4.

n	40	64	100	160	250	400	640	1000	1600	2500	4000
p. lapack L	746	1145	1422	1732	1993	2117	1897	1619	1475	1432	1443
p. lapack U	746	1145	1446	1775	2009	2190	1793	1483	1415	1402	1417
V. p. lapack L	746	1185	1472	1746	1993	2182	2039	1695	1505	1448	1456
f. lapack L	706	1074	1422	1760	1907	1639	1437	1311	1399	1344	1141
f. lapack U	706	1108	1422	1775	2009	1890	1622	1251	1362	1330	1385
V. f. lapack L	746	1108	1422	1790	1993	1980	1675	1212	1400	1343	1292
V. f. lapack U	706	1074	1422	1790	1920	1586	1411	1267	1358	1314	1065
p. recursive L	268	404	552	804	880	1073	1054	1002	1150	1216	967
p. recursive U	268	414	559	817	895	1169	1188	997	1175	1247	1055
p. hybrid L	746	1074	1446	1579	1759	1451	1218	1085	1036	1012	743
p. hybrid U	746	1108	1446	1591	1771	1404	1202	1065	1008	985	720

Reference

- **B.S. Andersen, J.A. Gunnels, F. Gustavson, J.K. Reid, and J. Waśniewski. "A Fully Portable High Performance Minimal Storage Hybrid Format Cholesky Algorithm". ACM Trans. of Math. Software, 31 (2005), 201-227.**

Rectangular Full Packed Data Format (RHP)

(•) Fred Gustavson
IBM Research Center, Yorktown Height, USA
(•) Jerzy Waśniewski
Technical University of Denmark, Lyngby, Denmark

Matrices A and A^T in Storage

- **Let A be a $m \times n$ matrix**
- **Fact: A^T is a $n \times m$ matrix**
- **Fact: Both A and A^T are simultaneously represented by either A or A^T**

Triangular Matrices in Storage

- **Let A be an order n of symmetric matrix**
- **Fact: A can be represented by either a lower or upper triangular matrix**

A Triangular Matrix can be a Rectangular Matrix

- **Let A be an order n upper or lower triangular matrix**
- **Matrix A can be represented as a $(n+1) \times n/2$ or $n \times (n+1)/2$ rectangular matrix**

Cholesky: $A = LL^T$

$$A = \begin{pmatrix} A_{11} & \\ A_{21} & A_{22} \end{pmatrix} \quad \text{and} \quad L = \begin{pmatrix} L_{11} & \\ L_{21} & L_{22} \end{pmatrix}$$

$$A = \begin{pmatrix} L_{11} & \\ L_{21} & L_{22} \end{pmatrix} \times \begin{pmatrix} L_{11}^T & L_{21}^T \\ & L_{22}^T \end{pmatrix}$$

$$A = \begin{pmatrix} A_{11} & A_{21} \\ A_{21} & A_{22} \end{pmatrix} = \begin{pmatrix} L_{11}L_{11}^T & L_{11}L_{21}^T \\ L_{21}L_{11}^T & L_{21}L_{21}^T + L_{22}L_{22}^T \end{pmatrix}$$

$$A_{11} = L_{11}L_{11}^T, \quad L_{21}L_{11}^T = A_{21} \quad \text{and} \quad \hat{A}_{22} = L_{22}L_{22}^T$$

where $\hat{A}_{22} := A_{22} - L_{21}L_{21}^T$

Cholesky: $A = LL^T$

$$A = \begin{pmatrix} A_{11} & \\ A_{21} & A_{22} \end{pmatrix} \quad \text{and} \quad L = \begin{pmatrix} L_{11} & \\ L_{21} & L_{22} \end{pmatrix}$$

Do

- **if** $n > 1$ **then**
  - $L_{11} :=$ **cholesky of** A_{11}
  - $L_{21}L_{11}^T = A_{21} \rightarrow$ **TRSM**
  - $\hat{A}_{22} := A_{22} - L_{21}L_{21}^T \rightarrow$ **SYRK**
  - $L_{22} :=$ **cholesky of** $\hat{A}_{22}$
- **otherwise**
  - $L := \sqrt{A_{11}}$

End

Rectangular Full Packed Data Format, n is odd

n $= 7,$ **memory needed** $=$ **n** $\times$ **n** $= 49$

$$A = \left(\begin{array}{cccc|ccc} a_{1,1_1} & \diamond & \diamond & \diamond & \diamond & \diamond & \diamond \\ a_{2,1_2} & a_{2,2_9} & \diamond & \diamond & \diamond & \diamond & \diamond \\ a_{3,1_3} & a_{3,2_{10}} & a_{3,3_{17}} & \diamond & \diamond & \diamond & \diamond \\ a_{4,1_4} & a_{4,2_{11}} & a_{4,3_{18}} & a_{4,4_{25}} & \diamond & \diamond & \diamond \\ \hline a_{5,1_5} & a_{5,2_{12}} & a_{5,3_{19}} & a_{5,4_{26}} & a_{5,5_{33}} & \diamond & \diamond \\ a_{6,1_6} & a_{6,2_{13}} & a_{6,3_{20}} & a_{6,4_{27}} & a_{6,5_{34}} & a_{6,6_{41}} & \diamond \\ a_{7,1_7} & a_{7,2_{14}} & a_{7,3_{21}} & a_{7,4_{28}} & a_{7,5_{35}} & a_{7,6_{42}} & a_{7,7_{49}} \end{array}\right)$$

LAPACK full data format

n $= 7,$ **memory needed** $=$ **n** $\times$ **(n+1)/2** $= 28$

$$A^{rfp} = \left(\begin{array}{cccc} a_{1,1_1} & a_{5,5_8} & a_{6,5_{15}} & a_{7,5_{22}} \\ a_{2,1_2} & a_{2,2_9} & a_{6,6_{16}} & a_{7,6_{23}} \\ a_{3,1_3} & a_{3,2_{10}} & a_{3,3_{17}} & a_{7,7_{24}} \\ a_{4,1_4} & a_{4,2_{11}} & a_{4,3_{18}} & a_{4,4_{25}} \\ \hline a_{5,1_5} & a_{5,2_{12}} & a_{5,3_{19}} & a_{5,4_{26}} \\ a_{6,1_6} & a_{6,2_{13}} & a_{6,3_{20}} & a_{6,4_{27}} \\ a_{7,1_7} & a_{7,2_{14}} & a_{7,3_{21}} & a_{7,4_{28}} \end{array}\right)$$

Rectangular full packed data format

Rectangular Full Packed Data Format, n is even

$$\mathbf{n = 6, \quad memory\ needed = n \times n = 36}$$

$$A = \left(\begin{array}{ccc|ccc} a_{1,1_1} & \diamond & \diamond & \diamond & \diamond & \diamond \\ a_{2,1_2} & a_{2,2_8} & \diamond & \diamond & \diamond & \diamond \\ a_{3,1_3} & a_{3,2_9} & a_{3,3_{15}} & \diamond & \diamond & \diamond \\ \hline a_{4,1_4} & a_{4,2_{10}} & a_{4,3_{16}} & a_{4,4_{22}} & \diamond & \diamond \\ a_{5,1_5} & a_{5,2_{11}} & a_{5,3_{17}} & a_{5,4_{23}} & a_{5,5_{29}} & \diamond \\ a_{6,1_6} & a_{6,2_{12}} & a_{6,3_{18}} & a_{6,4_{24}} & a_{6,5_{30}} & a_{6,6_{36}} \end{array}\right)$$

LAPACK full data format

$$\mathbf{n = 6, \quad memory\ needed = (n+1) \times n/2 = 21}$$

$$A^{rfp} = \left(\begin{array}{ccc} a_{4,4_1} & a_{5,4_8} & a_{6,4_{15}} \\ \hline a_{1,1_2} & a_{5,5_9} & a_{6,5_{16}} \\ a_{2,1_3} & a_{2,2_{10}} & a_{6,6_{17}} \\ \hline a_{3,1_4} & a_{3,2_{11}} & a_{3,3_{18}} \\ \hline a_{4,1_5} & a_{4,2_{12}} & a_{4,3_{19}} \\ a_{5,1_6} & a_{5,2_{13}} & a_{5,3_{20}} \\ a_{6,1_7} & a_{6,2_{14}} & a_{6,3_{21}} \end{array}\right)$$

Rectangular full packed data format

LAPACK Symmetric/Hermitian and Triangular matrices

$$\mathbf{n} = 7, \quad \mathbf{memory\ needed} = \mathbf{n} \times \mathbf{n} = 49$$

$$\begin{pmatrix}
a_{1,1_1} & \diamond & \diamond & \diamond & \diamond & \diamond & \diamond \\
a_{2,1_2} & a_{2,2_9} & \diamond & \diamond & \diamond & \diamond & \diamond \\
a_{3,1_3} & a_{3,2_{10}} & a_{3,3_{17}} & \diamond & \diamond & \diamond & \diamond \\
a_{4,1_4} & a_{4,2_{11}} & a_{4,3_{18}} & a_{4,4_{25}} & \diamond & \diamond & \diamond \\
a_{5,1_5} & a_{5,2_{12}} & a_{5,3_{19}} & a_{5,4_{26}} & a_{5,5_{33}} & \diamond & \diamond \\
a_{6,1_6} & a_{6,2_{13}} & a_{6,3_{20}} & a_{6,4_{27}} & a_{6,5_{34}} & a_{6,6_{41}} & \diamond \\
a_{7,1_7} & a_{7,2_{14}} & a_{7,3_{21}} & a_{7,4_{28}} & a_{7,5_{35}} & a_{7,6_{42}} & a_{7,7_{49}}
\end{pmatrix}$$

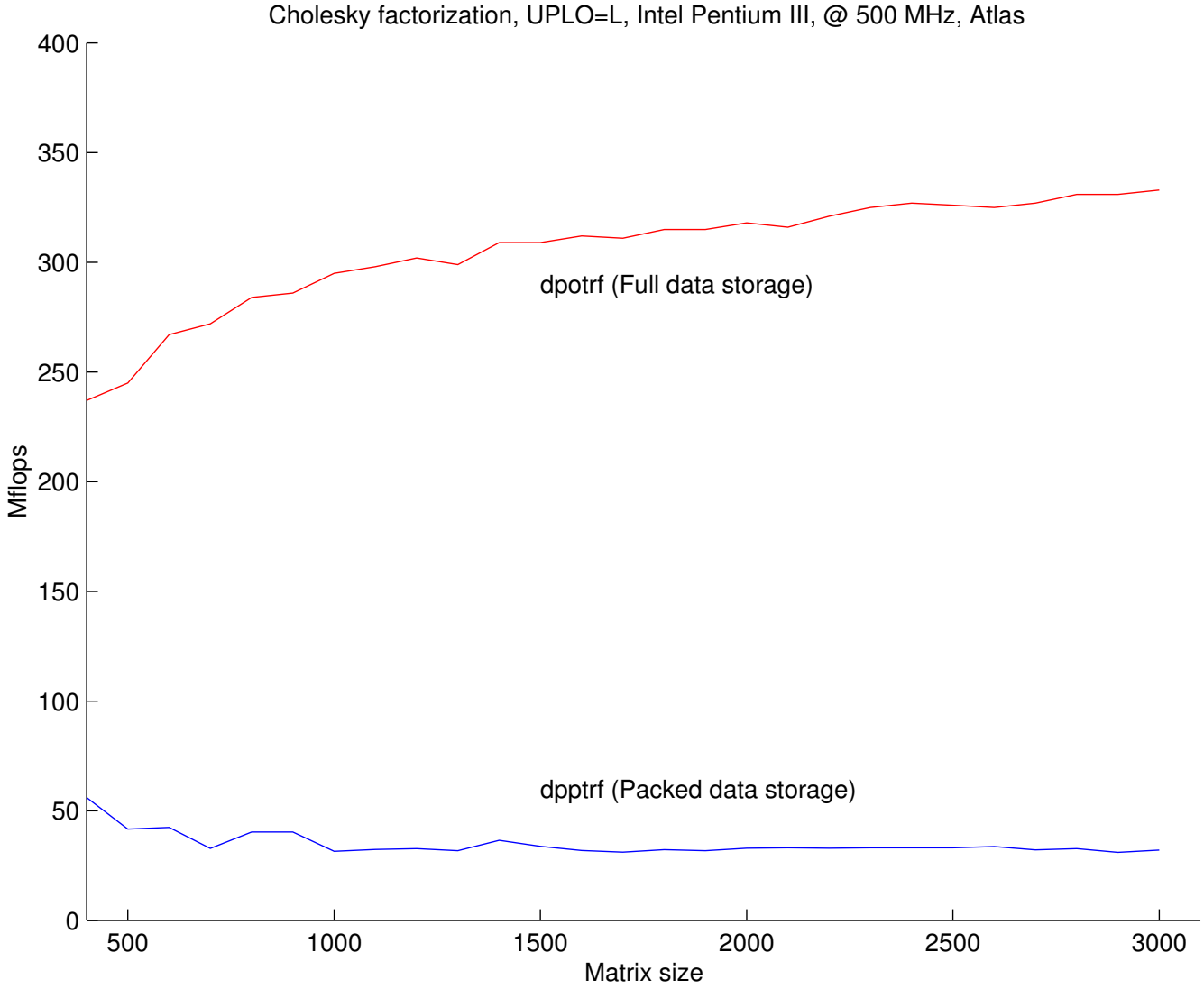
LAPACK full data format

$$\mathbf{n} = 7, \quad \mathbf{memory\ needed} = \mathbf{n} \times (\mathbf{n}+1)/2 = 28$$

$$\begin{pmatrix}
a_{1,1_1} & & & & & & \\
a_{2,1_2} & a_{2,2_8} & & & & & \\
a_{3,1_3} & a_{3,2_9} & a_{3,3_{14}} & & & & \\
a_{4,1_4} & a_{4,2_{10}} & a_{4,3_{15}} & a_{4,4_{19}} & & & \\
a_{5,1_5} & a_{5,2_{11}} & a_{5,3_{16}} & a_{5,4_{20}} & a_{5,5_{23}} & & \\
a_{6,1_6} & a_{6,2_{12}} & a_{6,3_{17}} & a_{6,4_{21}} & a_{6,5_{24}} & a_{6,6_{26}} & \\
a_{7,1_7} & a_{7,2_{13}} & a_{7,3_{18}} & a_{7,4_{22}} & a_{7,5_{25}} & a_{7,6_{27}} & a_{7,7_{28}}
\end{pmatrix}$$

LAPACK packed data format

Cholesky, a packed and full data storage



Symmetric/Hermitian matrices

n $= 7,$ **memory needed** $=$ **n** $\times$ **(n+1)/2** $= 28$

$$A^{rfp} = \left(\begin{array}{cccc} a_{1,1_1} & a_{5,5_8} & a_{6,5_{15}} & a_{7,5_{22}} \\ a_{2,1_2} & a_{2,2_9} & a_{6,6_{16}} & a_{7,6_{23}} \\ a_{3,1_3} & a_{3,2_{10}} & a_{3,3_{17}} & a_{7,7_{24}} \\ a_{4,1_4} & a_{4,2_{11}} & a_{4,3_{18}} & a_{4,4_{25}} \\ \hline a_{5,1_5} & a_{5,2_{12}} & a_{5,3_{19}} & a_{5,4_{26}} \\ a_{6,1_6} & a_{6,2_{13}} & a_{6,3_{20}} & a_{6,4_{27}} \\ a_{7,1_7} & a_{7,2_{14}} & a_{7,3_{21}} & a_{7,4_{28}} \end{array}\right)$$

Rectangular full packed data format

$$A^{rfp} = \left(\begin{array}{c} A_{11} \quad A_{22}^T \\ A_{21} \end{array}\right)$$

$$L^{rfp} = \left(\begin{array}{c} L_{11} \quad L_{22}^T \\ L_{21} \end{array}\right) ?$$

- L_{11} := **POTRF of** (A_{11})
- L_{21} := **TRSM of** $(L_{21}L_{11}^T = A_{21})$
- $\hat{L}_{22}$:= **SYRK of** $(A_{22} - L_{21}L_{21}^T)$
- L_{22} := **POTRF of** $(\hat{L}_{22})$

There are eight cases:

1. odd, lower, no transpose
2. odd, lower, transpose
3. odd, upper, no transpose
4. odd, upper, transpose
5. even, lower, no transpose
6. even, lower, transpose
7. even, upper, no transpose
8. even, upper, transpose

The Cholesky factorization algorithm using the Rectangular Full Packed format (RFP) if n is odd, uplo='lower', and trans='no transpose'.

A of Rectangular full packed
n $= 7$, memory needed
n $\times$ (n+1)/2 $= 28$

$a_{1,1_1}$	$a_{5,5_8}$	$a_{6,5_{15}}$	$a_{7,5_{22}}$			
$a_{2,1_2}$	$a_{2,2_9}$	$a_{6,6_{16}}$	$a_{7,6_{23}}$			
$a_{3,1_3}$	$a_{3,2_{10}}$	$a_{3,3_{17}}$	$a_{7,7_{24}}$			
$a_{4,1_4}$	$a_{4,2_{11}}$	$a_{4,3_{18}}$	$a_{4,4_{25}}$			
$a_{5,1_5}$	$a_{5,2_{12}}$	$a_{5,3_{19}}$	$a_{5,4_{26}}$	$a_{5,5_{33}}$		
$a_{6,1_6}$	$a_{6,2_{13}}$	$a_{6,3_{20}}$	$a_{6,4_{27}}$	$a_{6,5_{34}}$	$a_{6,6_{41}}$	
$a_{7,1_7}$	$a_{7,2_{14}}$	$a_{7,3_{21}}$	$a_{7,4_{28}}$	$a_{7,5_{35}}$	$a_{7,6_{42}}$	$a_{7,7_{49}}$

The Algorithm $(k = \lceil n/2 \rceil)$ **:**

1) factor $L_{11}L_{11}^T = A_{11}$;
call POTRF$('L', k, AR(1,1), n, \& \; info)$;

2) solve $L_{21}L_{11}^T = A_{21}$;
call TRSM$('R', 'L', 'T', 'N', k-1, \& \; k, one, AR(1,1), n, AR(k+1,1), n)$;

3) update $A_{22}^T := A_{22}^T - L_{21}L_{21}^T$;
call SYRK$('U', 'N', k-1, k, -one, \& \; AR(k+1,1), n, one, AR(2,1), n)$;

4) factor $U_{22}^T U_{22} = A_{22}^T$;
call POTRF$('U', k-1, AR(2,1), n, \& \; info)$;

SUN UltraSPARC IV dual-core CPUs (1350 MHz/ 8 MB/core L2-cache)
Sun BLAS Lib., Real long prec., Cholesky Factorization, Mflops

	New				lapack			
	rfp		hpp		po		pp	
n	u	l	u	l	u	l	u	l
50	819	909	1317	1321	910	800	431	617
100	1419	1533	1954	1953	1268	1389	591	806
200	1757	1807	1996	1997	1712	1925	708	377
400	2177	2200	2289	2316	2219	2314	793	257
500	2170	2206	2377	2415	2354	2350	812	251
800	2446	2409	2574	2635	2549	2416	800	242
1000	2512	2459	2644	2711	2617	2498	702	228
1600	2584	2531	2759	2849	2753	1950	627	217
2000	2692	2624	2804	2899	2795	2638	629	216
4000	2337	2753	2842	2944	2834	2700	509	97

SUN UltraSPARC IV dual-core CPUs (1350 MHz/ 8 MB/core L2-cache)
Sun BLAS Lib., Real long prec., Cholesky Factorization, Mflops

n	rfp u	rfp l	hpp u	hpp l	rpp u	rpp l	pp u	pp l
50	830	921	1319	1321	256	285	432	619
100	1416	1531	1950	1952	535	601	591	806
200	1760	1815	2005	2006	968	1060	708	376
400	2182	2213	2294	2321	1527	1632	793	257
500	2168	2205	2386	2422	1827	1845	812	251
800	2451	2408	2579	2640	2076	2142	784	237
1000	2513	2472	2644	2717	2361	2313	702	227
1600	2589	2538	2765	2853	2489	2497	627	216
2000	2654	2633	2810	2903	2454	2417	617	216
4000	2448	2762	2845	2942	2847	2814	492	94

SUN UltraSPARC IV dual-core CPUs (1350 MHz/ 8 MB/core L2-cache)
Sun BLAS Lib., Real long prec., Cholesky Inversion, Mflops

	rfp				lapack			
	no trans		trans		po		pp	
n	u	l	u	l	u	l	u	l
50	705	680	681	685	546	534	486	532
100	1166	1158	1108	1121	1171	1143	680	706
200	1744	1727	1720	1737	1785	1766	824	818
400	2266	2266	2242	2260	2272	2276	930	885
500	2283	2282	2285	2302	2376	2397	951	898
800	2530	2539	2501	2510	2453	2473	887	770
1000	2544	2570	2608	2578	2523	2524	739	533
1600	2561	2613	2623	2560	2233	2542	638	424
2000	2541	2610	2587	2524	2590	2601	584	407
4000	2598	2414	2396	2506	2340	2424	421	145

SUN UltraSPARC IV dual-core CPUs (1350 MHz/ 8 MB/core L2-cache)
Sun BLAS Lib., Real long prec., Cholesky Solution, Mflops

nr	n	rfp u	rfp l	hpp u	hpp l	rpp u	rpp l	pp u	pp l
100	50	1773	1763	1567	1568	669	693	559	556
100	100	2051	2051	2086	2085	1036	1066	718	713
100	200	2590	2591	2447	2448	1538	1565	693	827
100	400	2765	2725	2659	2660	2038	2045	733	887
100	500	2731	2683	2734	2732	2295	2277	742	900
100	800	2867	2865	2803	2814	2462	2464	642	692
100	1000	2918	2912	2831	2836	2610	2606	537	542
160	1600	2944	2893	2942	2941	2760	2767	445	446
200	2000	3023	3031	2915	2853	2718	2910	429	419
400	4000	2821	2832	2883	2897	2908	2921	179	175

SUN UltraSPARC IV dual-core CPUs (1350 MHz/ 8 MB/core L2-cache)
Sun BLAS Lib., Complex long prec., Cholesky Factorization, Mflops

	rfp				lapack			
	no trans		trans		po		pp	
n	u	l	u	l	u	l	u	l
50	1450	1550	1642	1436	1313	1310	893	1347
100	2046	2001	2072	1858	1564	1944	1292	1368
200	2343	2285	2346	2220	2123	2389	1657	543
400	2688	2607	2524	2602	2570	2653	2006	482
500	2794	2708	2636	2729	2707	2750	2068	478
800	2912	2794	2750	2862	2895	2246	1430	443
1000	2951	2834	2824	2880	2925	2852	1319	435
1600	2994	2900	2495	2902	3070	1499	1256	430
2000	3039	2969	2965	3033	3091	2989	1206	423
4000	3036	2942	2980	3068	3092	2144	596	146

SUN UltraSPARC-III, 1200 MHz, L1 c 64/32/2/2 Kb, L2 c 8 Mb

Cholesky: factorization, sun BLAS library, double prec.

	rfp				lapack			
	no trans		trans		po		pp	
n	u	l	u	l	u	l	u	l
50	456	520	516	482	460	464	291	294
100	753	813	829	768	612	827	399	369
200	946	979	997	955	933	1150	455	370
400	1208	1231	1158	1183	1081	1244	483	339
500	1173	1227	1138	1186	1121	1340	511	343
800	1316	1318	1189	1269	1256	1310	522	324
1000	1275	1318	1281	1303	1313	1406	530	288
1600	1350	1387	1358	1312	1405	1234	502	223
2000	1264	1367	1403	1323	1360	1491	394	163
4000	1287	1450	1537	1263	1392	1565	300	153

SUN UltraSPARC-III, 1200 MHz, L1 c 64/32/2/2 Kb, L2 c 8 Mb

Cholesky: inversion, sun BLAS library, double prec.

	rfp				lapack			
	no trans		trans		po		pp	
n	u	l	u	l	u	l	u	l
50	379	379	380	381	330	328	318	338
100	698	696	699	700	698	707	412	446
200	1012	989	1008	997	1052	1030	467	558
400	1290	1223	1229	1263	1229	1212	452	606
500	1290	1276	1238	1330	1285	1276	448	595
800	1446	1445	1330	1356	1343	1325	408	566
1000	1428	1337	1442	1436	1378	1318	404	531
1600	1418	1372	1369	1396	1142	1317	262	450
2000	1387	1333	1370	1536	1400	1366	242	394
4000	1460	1408	1389	1453	1421	1395	201	288

SUN UltraSPARC-III, 1200 MHz, L1 c 64/32/2/2 Kb, L2 c 8 Mb

Cholesky: factorization and inversion, sun BLAS library, double prec.

	rfp				lapack			
	no trans		trans		po		pp	
n	u	l	u	l	u	l	u	l
50	569	569	570	572	495	492	477	508
100	1048	1045	1049	1050	1047	1061	619	670
200	1518	1484	1512	1496	1579	1546	701	837
400	1935	1835	1844	1895	1844	1819	679	909
500	1936	1915	1858	1996	1928	1915	672	893
800	2169	2168	1995	2035	2015	1988	612	849
1000	2143	2006	2164	2154	2067	1978	607	797
1600	2127	2059	2054	2094	1713	1976	393	676
2000	2081	2000	2055	2304	2100	2049	363	592
4000	2190	2113	2084	2180	2132	2093	302	433

SUN UltraSPARC-III, 1200 MHz, L1 c 64/32/2/2 Kb, L2 c 8 Mb

Cholesky: solution, sun BLAS library, double prec.

		rfp				lapack			
		no trans		trans		po		pp	
nrhs	n	u	l	u	l	u	l	u	l
100	50	1132	1123	1153	1135	1103	1069	353	353
100	100	1193	1163	1237	1211	1262	1262	478	478
100	200	1477	1478	1500	1490	1280	1235	557	554
100	400	1494	1505	1514	1534	1150	1149	582	582
100	500	1466	1443	1436	1445	1217	1229	560	569
100	800	1503	1505	1535	1469	1151	1096	528	526
100	1000	1553	1524	1499	1576	1089	1125	513	513
160	1600	1595	1564	1603	1577	1155	1121	421	403
200	2000	1600	1636	1610	1615	1105	1087	347	338
400	4000	1666	1668	1696	1665	1080	1084	292	290

SUN UltraSPARC-III, 1200 MHz, L1 c 64/32/2/2 Kb, L2 c 8 Mb

Cholesky: factorization and solution, sun BLAS library, double prec.

		rfp				lapack			
		no trans		trans		po		pp	
nrhs	n	u	l	u	l	u	l	u	l
100	50	972	999	1006	989	985	990	349	350
100	100	1026	1058	1102	1087	1079	1095	457	440
100	200	1262	1241	1304	1310	1150	1195	518	493
100	400	1352	1402	1338	1373	1143	1174	533	459
100	500	1310	1339	1287	1270	1187	1263	544	426
100	800	1440	1397	1321	1395	1207	1226	532	385
100	1000	1408	1396	1347	1414	1243	1301	528	328
160	1600	1421	1439	1430	1450	1281	1178	453	288
200	2000	1369	1478	1480	1409	1273	1326	385	222
400	4000	1392	1542	1606	1393	1273	1304	297	188

IBM Power4 1300 MHz, caches: L1 128KB, L2 1.5MB, L3 32MB

ESSL BLAS Lib., Real long prec., Cholesky Factorization, Mflops

n	rfp u	rfp l	hpp u	hpp l	po u	po l	pp u	pp l
50	1087	1077	422	421	1500	331	437	629
100	1908	1862	840	842	2165	900	790	862
200	2606	2433	1776	1740	2624	1660	1102	907
400	2877	2740	2574	2602	2825	2359	1331	959
500	2823	2658	2759	2787	2814	2530	1344	888
800	2972	2881	2935	2976	2862	2799	1115	501
1000	2941	2813	2972	3010	2794	2813	1061	448
1600	3085	2952	3050	3034	3034	2844	961	364
2000	2777	2792	3082	3082	2930	2867	984	342
4000	3069	2975	3017	2918	2836	2646	761	304

IBM Power4 1300 MHz, caches: L1 128KB, L2 1.5MB, L3 32MB
ESSL BLAS Lib., Real long prec., Cholesky Factorization, Mflops

n	rfp u	rfp l	hpp u	hpp l	rpp u	rpp l	pp u	pp l
50	1092	1085	406	401	135	133	437	617
100	1908	1852	846	844	357	328	808	827
200	2598	2408	1765	1740	831	780	1114	886
400	2839	2677	2602	2627	1584	1470	1299	988
500	2787	2621	2705	2713	1833	1633	1333	844
800	2862	2730	2862	2650	2174	1994	1073	418
1000	2742	2642	2845	2850	2306	2083	980	354
1600	2592	2452	2592	2301	2340	2167	758	284
2000	2914	2601	2930	2914	2614	2480	857	308
4000	3091	2814	2724	2871	2954	2803	797	320

IBM Power4 1300 MHz, caches: L1 128KB, L2 1.5MB, L3 32MB

ESSL BLAS Lib., Complex long prec., Cholesky Factorization, Mflops

	rfp				lapack			
	no trans		trans		po		pp	
n	u	l	u	l	u	l	u	l
50	1826	1766	1693	1807	2302	727	1140	1479
100	2483	2407	2471	2467	2435	1484	1725	1743
200	2804	2663	2684	2776	2821	2189	2189	1907
400	2936	2807	2754	2844	2955	2669	2230	1431
500	2941	2850	2869	2922	2902	2742	2043	1028
800	2986	2913	2931	2986	2913	2878	1750	734
1000	3030	2898	2963	2996	2930	2852	1698	694
1600	2758	2664	3034	3051	2829	2690	1521	579
2000	3100	3056	2954	2890	2829	2996	1664	708
4000	3104	3000	3077	3034	2968	2867	1465	574

ia64 Itanium (DMI) @ 1300 MHz

Cholesky: factorization, NEC BLAS library, double prec.

	rfp				lapack			
	no trans		trans		po		pp	
n	u	l	u	l	u	l	u	l
50	781	771	784	771	1107	739	495	533
100	1843	1788	1848	1812	1874	1725	879	825
200	3178	2869	2963	3064	2967	2871	1323	1100
400	3931	3709	3756	3823	3870	3740	1121	1236
500	4008	3808	3883	3914	4043	3911	1032	1257
800	4198	4097	4145	4126	3900	4009	612	1127
1000	4115	4038	4015	3649	3769	3983	305	697
1600	3851	3652	3967	3971	3640	3987	147	437
2000	3899	3716	3660	3660	3865	3835	108	358
4000	3966	3791	3927	4011	3869	4052	119	398

ia64 Itanium (DMI) @ 1300 MHz

Cholesky: inversion, NEC BLAS library, double prec.

	rfp				lapack			
	no trans		trans		po		pp	
n	u	l	u	l	u	l	u	l
50	633	659	648	640	777	870	508	460
100	1252	1323	1300	1272	1573	1760	815	810
200	2305	2442	2431	2314	2357	2639	1118	1211
400	3084	3199	3188	3094	3152	3445	1234	1363
500	3204	3316	3329	3218	3400	3611	1239	1382
800	3617	3741	3720	3640	3468	3786	1182	1268
1000	3611	3716	3637	3590	3456	3790	767	946
1600	3721	3802	3795	3714	3589	3713	500	609
2000	3784	3812	3745	3704	3636	3798	473	596
4000	3822	3762	3956	3851	3760	3750	467	614

ia64 Itanium (DMI) @ 1300 MHz

Cholesky: factorization and inversion, NEC BLAS library, double prec.

	rfp				lapack			
	no trans		trans		po		pp	
n	u	l	u	l	u	l	u	l
50	950	989	972	961	1166	1306	763	691
100	1878	1985	1950	1909	2360	2641	1223	1215
200	3458	3664	3647	3472	3536	3959	1678	1817
400	4626	4799	4783	4641	4729	5168	1852	2045
500	4806	4974	4994	4827	5100	5417	1859	2073
800	5426	5612	5581	5461	5203	5680	1774	1903
1000	5417	5575	5456	5386	5184	5685	1151	1419
1600	5582	5703	5693	5572	5384	5570	750	914
2000	5677	5718	5618	5557	5455	5698	710	894
4000	5734	5644	5935	5777	5640	5626	701	922

ia64 Itanium (DMI) @ 1300 MHz

Cholesky: solution, NEC BLAS library, double prec.

		rfp				lapack			
		no trans		trans		po		pp	
nrhs	n	u	l	u	l	u	l	u	l
100	50	2409	2412	2414	2422	3044	3018	725	714
100	100	3305	3301	3303	3303	3889	3855	1126	1109
100	200	4149	4154	4127	4146	4143	4127	1526	1512
100	400	4398	4403	4416	4444	4469	4451	1097	1088
100	500	4313	4155	4374	4394	4203	4093	1054	1045
100	800	3979	3919	4040	4051	3969	4011	692	720
100	1000	3716	3608	3498	3477	3630	3645	376	372
160	1600	3892	3874	4020	3994	4001	4011	188	182
200	2000	4052	4073	4040	4020	4231	4203	119	119
400	4000	4245	4225	4275	4287	4330	4320	115	144

ia64 Itanium (DMI) @ 1300 MHz

Cholesky: factorization and solution, NEC BLAS library, double prec.

		rfp				lapack			
		no trans		trans		po		pp	
nrhs	n	u	l	u	l	u	l	u	l
100	50	2068	2069	2073	2066	2672	2427	701	695
100	100	2952	2929	2952	2930	3346	3264	1083	1059
100	200	3860	3735	3759	3799	3761	3711	1457	1371
100	400	4218	4080	4113	4124	4178	4121	1088	1142
100	500	4136	4067	4115	4152	4152	4120	1042	1132
100	800	4052	3996	4044	3976	3890	3982	688	983
100	1000	3950	3858	3761	3530	3685	3821	346	538
160	1600	3859	3715	4016	3989	3765	3985	124	292
200	2000	3952	3830	3797	3794	4002	3953	112	204
400	4000	4055	3930	4040	4088	4010	4129	142	241

SX-6 NEC (DMI) @ ... MHz, Vector option

Cholesky: factorization, NEC BLAS library, double prec.

	rfp				po		lapack	
	no trans		trans				pp	
n	u	l	u	l	u	l	u	l
50	206	200	225	225	365	353	57	238
100	721	728	789	788	1055	989	120	591
200	2028	2025	2005	2015	1380	1639	246	1250
400	3868	3915	3078	3073	1763	3311	479	1975
500	4483	4470	4636	4636	4103	4241	585	2149
800	5154	5168	4331	4261	3253	4469	870	2399
1000	5666	5654	5725	5703	5144	5689	1035	2474
1600	6224	6145	5644	5272	5375	5895	1441	2572
2000	6762	6788	6642	6610	6088	6732	1654	2598
4000	7321	7325	7236	7125	6994	7311	2339	2641

SX-6 NEC (DMI) @ ... MHz, Vector option

Cholesky: inversion, NEC BLAS library, double prec.

	rfp				lapack			
	no trans		trans		po		pp	
n	u	l	u	l	u	l	u	l
50	152	152	150	152	148	145	91	61
100	430	432	428	432	313	310	194	126
200	950	956	940	941	636	627	404	249
400	1850	1852	1804	1806	1734	1624	722	470
500	2227	2228	2174	2181	2180	2029	856	572
800	3775	3775	3668	3686	3405	3052	1186	842
1000	4346	4346	4254	4263	4273	3638	1342	985
1600	5313	5294	5137	5308	5438	4511	1690	1361
2000	6006	6006	5930	5931	5997	4832	1854	1536
4000	6953	6953	6836	6888	7041	4814	1921	2122

SX-6 NEC (DMI) @ ... MHz, Vector option

Cholesky: factorization and inversion, NEC BLAS library, double prec.

	rfp				lapack			
	no trans		trans		po		pp	
n	u	l	u	l	u	l	u	l
50	228	229	225	228	222	218	137	92
100	645	649	643	648	470	465	291	189
200	1426	1434	1410	1412	954	941	607	374
400	2775	2779	2707	2709	2602	2437	1083	706
500	3341	3342	3261	3272	3270	3044	1284	859
800	5663	5663	5502	5529	5108	4579	1779	1263
1000	6520	6519	6381	6395	6410	5458	2014	1478
1600	7970	7942	7706	7963	8158	6767	2535	2042
2000	9009	9010	8896	8897	8996	7249	2782	2304
4000	10430	10430	10254	10333	10562	7221	2882	3183

SX-6 NEC (DMI) @ ... MHz, Vector option

Cholesky: solution, NEC BLAS library, double prec.

		rfp				po		lapack	
		no trans		trans				pp	
nrhs	n	u	l	u	l	u	l	u	l
100	50	873	870	889	886	1933	1941	88	88
100	100	2173	2171	2200	2189	3216	3236	181	179
100	200	4236	4230	4253	4245	4165	4166	352	347
100	400	5431	5431	5410	5408	5302	5303	648	644
100	500	5563	5562	5568	5567	5629	5632	783	779
100	800	6407	6407	6240	6240	5569	5593	1132	1128
100	1000	6578	6578	6559	6558	6554	6566	1325	1320
160	1600	6781	6805	6430	6430	6799	6809	1732	1727
200	2000	7568	7569	7519	7519	7406	7407	1920	1914
400	4000	7858	7858	7761	7761	7626	7627	2414	2410

SX-6 NEC (DMI) @ ... MHz, Vector option

Cholesky: factorization and solution, NEC BLAS library, double prec.

		rfp				lapack			
		no trans		trans		po		pp	
nrhs	n	u	l	u	l	u	l	u	l
100	50	687	687	724	710	1523	1516	85	92
100	100	1680	1664	1772	1756	2572	2476	168	198
100	200	3347	3309	3373	3363	2767	3014	317	424
100	400	4684	4693	4167	4145	2939	4306	568	881
100	500	5038	5050	5098	5090	4812	4925	679	1097
100	800	5636	5652	4976	4928	3969	4916	966	1619
100	1000	6003	6021	6012	5989	5601	6005	1128	1863
160	1600	6422	6335	5909	5644	5828	6209	1537	2173
200	2000	7045	7048	6953	6931	6540	6977	1744	2291
400	4000	7523	7528	7424	7354	7230	7439	2366	2548

***** RFP Data Format: cpu_times and Mflops *****

Sunfire (IMM) @ 1200 MHz, Sun BLAS Lib., Double prec.

uplo	trans	n	times					Mflops
			rfptrf	potrf	trsm	syrk	potrf	
l	n	2000	1.82	0.26	0.69	0.62	0.26	1463
			100%	14%	38%	34%	14%	
l	t	2000	1.85	0.24	0.64	0.75	0.21	1443
			100%	13%	35%	41%	12%	
u	n	2000	1.92	0.26	0.56	0.85	0.26	1386
			100%	13%	29%	44%	13%	
u	t	2000	1.76	0.24	0.71	0.60	0.21	1515
			100%	13%	40%	34%	12%	

***** RFP Data Format: cpu_times and Mflops *****

Sunfire (IMM) @ 1200 MHz, Sun BLAS Lib., Double prec.

uplo	trans	n	times					Mflops
			rfptrf	potrf	trsm	syrk	potrf	
l	n	4000	13.20	1.67	5.07	4.66	1.81	1616
			100%	13%	38%	35%	14%	
l	t	4000	15.91	1.79	5.39	7.04	1.69	1341
			100%	11%	34%	44%	11%	
u	n	4000	15.23	1.68	4.49	7.25	1.81	1401
			100%	11%	29%	48%	12%	
u	t	4000	13.21	1.75	5.04	4.73	1.69	1614
			100%	13%	38%	36%	13%	

***** RFP Data Format: cpu_times and Mflops *****

Sunfire (IMM) @ 1200 MHz, Sun BLAS Lib., Double prec.

uplo	trans	n	times					Mflops
			rfptrf	potrf	trsm	syrk	potrf	
l	n	4500	19.12	2.59	7.24	6.77	2.53	1588
			100%	14%	38%	35%	13%	
l	t	4500	22.92	2.50	7.78	10.15	2.48	1325
			100%	11%	34%	44%	11%	
u	n	4500	21.76	2.64	6.43	10.14	2.55	1396
			100%	12%	30%	47%	12%	
u	t	4500	19.06	2.52	7.15	6.80	2.59	1594
			100%	13%	38%	36%	14%	

***** RFP Data Format: cpu_times and Mflops *****

Sunfire (IMM) @ 1200 MHz, Sun BLAS Lib., Double prec.

uplo	trans	n	times					Mflops
			rfptrf	potrf	trsm	syrk	potrf	
l	n	5000	25.78	3.32	9.73	9.33	3.41	1616
			100%	13%	38%	36%	13%	
l	t	5000	31.30	3.35	10.81	13.83	3.31	1331
			100%	11%	35%	44%	11%	
u	n	5000	29.97	3.36	8.76	14.33	3.51	1390
			100%	11%	29%	48%	12%	
u	t	5000	25.97	3.36	9.86	9.41	3.34	1605
			100%	13%	38%	36%	13%	

IBM Power4 1300 MHz, caches: L1 128KB, L2 1.5MB, L3 32MB

n	n proc	Mflops rfptrf		Times in rfptrf				Times lapack	
				potrf	trsm	syrk	potrf	potrf	pptrf
1000	1	2695	0.12	0.02	0.05	0.04	0.02	0.12	0.94
	5	7570	0.04	0.01	0.02	0.01	0.01	0.03	0.32
	10	10699	0.03	0.01	0.01	0.01	0.00	0.02	0.16
	15	9114	0.04	0.01	0.02	0.01	0.01	0.02	0.16
2000	1	2618	1.02	0.13	0.38	0.38	0.13	0.97	8.74
	5	10127	0.26	0.04	0.10	0.09	0.04	0.24	3.42
	10	17579	0.15	0.02	0.06	0.05	0.03	0.12	1.65
	15	23798	0.11	0.02	0.04	0.04	0.01	0.13	1.11
3000	1	2577	3.49	0.45	1.33	1.28	0.44	3.40	30.42
	5	11369	0.79	0.11	0.28	0.30	0.11	0.71	11.76
	10	19706	0.46	0.06	0.19	0.16	0.05	0.38	6.16
	15	29280	0.31	0.05	0.12	0.10	0.04	0.26	4.28
4000	1	2664	8.01	1.01	2.90	3.09	1.01	7.55	75.72
	5	11221	1.90	0.26	0.68	0.72	0.24	1.65	25.73
	10	21275	1.00	0.13	0.39	0.36	0.12	0.86	13.95
	15	31024	0.69	0.09	0.28	0.24	0.08	0.59	10.46
5000	1	2551	16.34	2.04	6.16	6.10	2.04	15.79	154.74
	5	11372	3.66	0.45	1.37	1.44	0.40	3.27	47.76
	10	22326	1.87	0.25	0.78	0.62	0.22	1.73	28.13
	15	32265	1.29	0.17	0.53	0.45	0.14	1.16	20.95

SUN UltraSPARC-IV, 1300 MHz, caches: L1 64KB, L2 8MB

n	n pr oc	Mflops rfptrf		Times in rfptrf				Times lapack	
				potrf	trsm	syrk	potrf	potrf	pptrf
1000	1	1587	0.21	0.03	0.09	0.07	0.03	0.19	1.06
	5	4762	0.07	0.02	0.02	0.02	0.02	0.07	1.13
	10	5557	0.06	0.01	0.01	0.02	0.02	0.06	1.12
	15	5557	0.06	0.02	0.01	0.01	0.02	0.06	1.11
2000	1	1668	1.58	0.22	0.63	0.52	0.22	1.45	11.20
	5	6667	0.40	0.07	0.13	0.13	0.07	0.38	11.95
	10	8602	0.31	0.06	0.07	0.11	0.07	0.25	11.24
	15	9524	0.28	0.06	0.06	0.08	0.08	0.23	11.66
3000	1	1819	4.95	0.62	1.98	1.72	0.63	4.86	45.48
	5	6872	1.31	0.20	0.42	0.48	0.20	1.38	55.77
	10	12162	0.74	0.14	0.22	0.21	0.16	0.76	46.99
	15	12676	0.71	0.14	0.16	0.30	0.16	0.61	45.71
4000	1	1823	11.70	1.52	4.62	4.01	1.55	11.86	112.52
	5	7960	2.68	0.40	0.94	0.92	0.42	2.74	112.77
	10	14035	1.52	0.26	0.47	0.49	0.30	1.61	112.53
	15	17067	1.25	0.24	0.37	0.35	0.29	1.29	111.67
5000	1	1843	22.61	2.92	8.76	8.00	2.93	23.60	218.94
	5	8139	5.12	0.77	1.81	1.80	0.74	5.45	221.58
	10	14318	2.91	0.50	0.97	0.93	0.51	3.11	214.54
	15	17960	2.32	0.45	0.72	0.68	0.47	2.40	225.08

Conclusions on Recursive Packed Algorithm:

1. The Algorithm and Data Format are very convenient.
2. The block size specification is not needed.
3. The Algorithm is fully Level 3 BLAS.
4. However it works slow for the small sizes.

Conclusions on Hybrid Blocked Packed Algorithm:

1. Mini-blocked kernel remarkable successful.
2. It is possible to get good performance with packed formats, usually comparable with full Lapack, sometimes better.
3. Packed hybrid is better than the packed recursive for modest n and for solving one rhs.
4. Packed hybrid may be slower than the packed recursive for large n.
5. The Block Packed Hybrid Format works well for the multi-core processors.

Conclusions on Rectangular Full Packed Algorithm:

1. **The Rectangular Full Packed (RFP) Algorithm can replace both, full and packed, LAPACK Algorithms for symmetric, Hermitian and triangular matrices.**
2. **RFP saves almost half $n(n-1)/2$ space comparing with LAPACK full Algorithm but it runs with almost the same speed.**
3. **RFP is many times faster than LAPACK packed Algorithm.**

General Conclusions

1. All Three Algorithms are fully Level 3 BLAS.
2. The Block Packed Hybrid Format works well for the multi-core processors.
3. The Block Packed Hybrid Algorithm can be specialized for the Cell Processor.

Three versions of High Performance Minimal Storage Cholesky Algorithm which uses New Data Structures: Recursion, Block Packed Hybrid and Rectangular Full Packed

Jerzy Waśniewski

Informatics & Mathematical Modeling

Technical University of Denmark

DK - 2800 Lyngby, Denmark

e-mail: jw@imm.dtu.dk

Parallel Processing and Applied Mathematics (PPAM 2007)

Gdańsk, Poland

September 9–12, 2007