# Pipelined Shared Memory Implementation of Linear Algebra Routines with Arbitrary Lookahead - LU, Cholesky, QR

Jakub Kurzak[1] and Jack Dongarra[2]

[1] University of Tennessee, Knoxville TN 37996, USA,
kurzak@cs.utk.edu, http://www.cs.utk.edu/~kurzak
[2] University of Tennessee, Knoxville TN 37996, USA,
dongarra@cs.utk.edu, http://www.netlib.org/utk/people/JackDongarra/

**Abstract.** Linear algebra algorithm commonly encapsulate parallelizm in Basic Linear Algebra Subroutines (BLAS). This solution relies on the fork-join model of parallel execution, which may result in suboptimal performance. To overcome the shortcomings of this approach a pipelined model of parallel execution is presented and the idea of arbitrary lookahead is utilized in order to suppress the negative effects of inherently sequential steps of the algorithms. Application to LU, Cholesky and QR factorizations is presented. The method is especially suitable for shared memory systems with small communication and synchronization overheads.

## 1 Introduction

Blocked implementation of LU, Cholesky and QR factorizations from the LAPACK library are used as a basis for development of the pipelined algorithms. In the following sections the algorithms are briefly reviewed in the formulation which is implemented in the LAPACK library. Then their pipelied implementatin for shared memory system using POSIX threads is presented altogether with the idea of a flexible lookahead. Discussion of the results follows.

## 2 Factorizations

### 2.1 LU Factorization

The LU factorization with partial row pivoting of an $m \times n$ real matrix $A$ has the form

$$A = PLU,$$

where $L$ is an $m \times n$ real unit lower triangular matrix, $U$ is an $n \times n$ real upper triangular matrix and $P$ is a permutation matrix. The derivation of the blocked algorithm is straightforward [2, 1]. In LAPACK a single step of the algorithm is implemented by the following sequence of calls to LAPACK and BLAS routines:

1. Factorize the column panel:
   – xGETF2,
2. Update the matrix $A$ to the right from the panel:
   – xLASWP, xTRSM, xGEMM.

Here we simplified the actual algorithm implemented in the LAPACK library by removing the row echanges to the left from the panel. Those can conveniently be performed at the end of the factorization.

### 2.2 Cholesky Factorization

The Cholesky factorization of an $n \times n$ real symmetric positive definite matrix $A$ has the form

$$A = LL^T,$$

where $L$ is an $n \times n$ real lower triangular matrix with positive diagonal elements. The derivation of the blocked algorithm is analogous to the one for LU factorization. In LAPACK a single step of the algorithm is implemented by the following sequence of calls to LAPACK and BLAS routines:

1. Factorize the row panel:
   – xSYRK, xPOTF2,
2. Update the matrix $A$ down from the panel:
   – xGEMM, xTRSM.

### 2.3 QR Factorization

The QR factorization of an $m \times n$ real matrix $A$ has the form

$$A = QR,$$

where Q is an $m \times m$ real orthogonal matrix and $R$ is an $m \times n$ real upper triangular matrix. The traditional algorithm for $QR$ factorization applies a series of elementary Householder matrices of the general form

$$H = I - \tau v v^T,$$

where $v$ is a column vector and $\tau$ is a scalar. In the block form of the algorithm a product of $nb$ elementary Householder matrices is represented in the form [3, 4]

$$H_1 H_2 \ldots H_{NB} = I - VTV^T,$$

where $V$ is an $n \times n$ real matrix those columns are the individual vectors $v$ and $T$ is an $nb \times nb$ real upper triangular matrix. For the derivation of the blocked algorithm the reader is referred to the original papers mentioned above. In LAPACK a single step of the algorithm is implemented by the following sequence of calls to LAPACK and BLAS routines:

1. Factorize the column panel:
   – xGEQR2, xLARFT,
2. Update the matrix $A$ to the right from the panel:
   – xLARFB.

## 3 Pipeline Organization

The approach is presented in the context of shared memory programming model. By the same token, no data partitioning or explicit communication is present. Parallelization relies of the notion of *w*ork partitioning, which is an analogue of data partitioning on distributed memory systems. In this work 1D block cyclic work partitioning is utilized. The input matrix is partitioned in block columns for the LU and QR factorizations and in block rows for the Cholesky factorization of a lower triangular matrix - the natural choice for each algorithm given the LAPACK formulations. In practice each thread of execuiton follows the same algorithmic path and skips tasks involving data which is not assigned to that thread (Fig. 1).

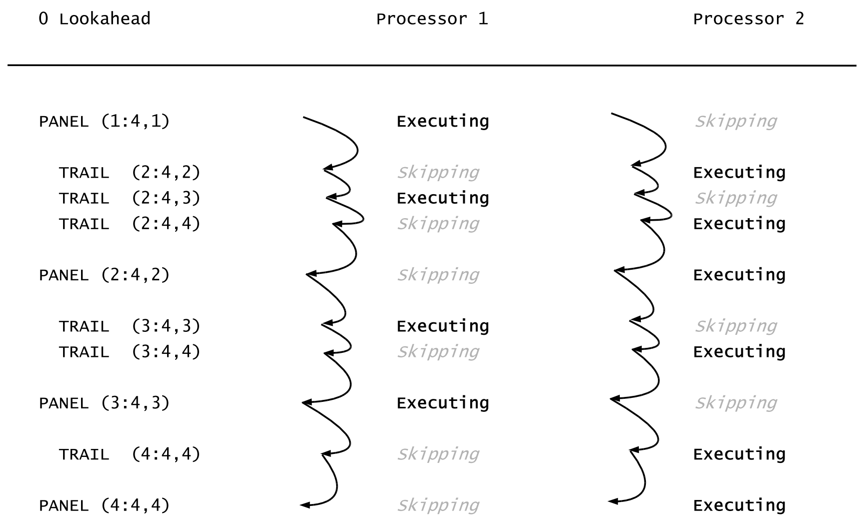

**Fig. 1.** Shared memory parallelization through *w*ork partitioning.

The three algorithms introduced above share the common behaviour of factorizing a panel and then updating the remaining part of the matrix $A$. In all cases panel factorization is an inherently sequential task and is most efficiently performed on a single processor. By the same token panel factorization becomes the least efficient tasks in parallel execuiton, with all processors blocking until its completion. We may note, however, that each factorization has many algorithmic variants. In particular panel factorizations can be initiated before updating of the trailing matrix is completed, what is the essence of the idea of lookahead. Fig. 2 shows the steps of LU or QR factorizations for a matrix of size $4 \times 4$

blocks with a lookahead of depth 0 (no lookahead), depth 1, and an infinite depth (maximum possible depth).

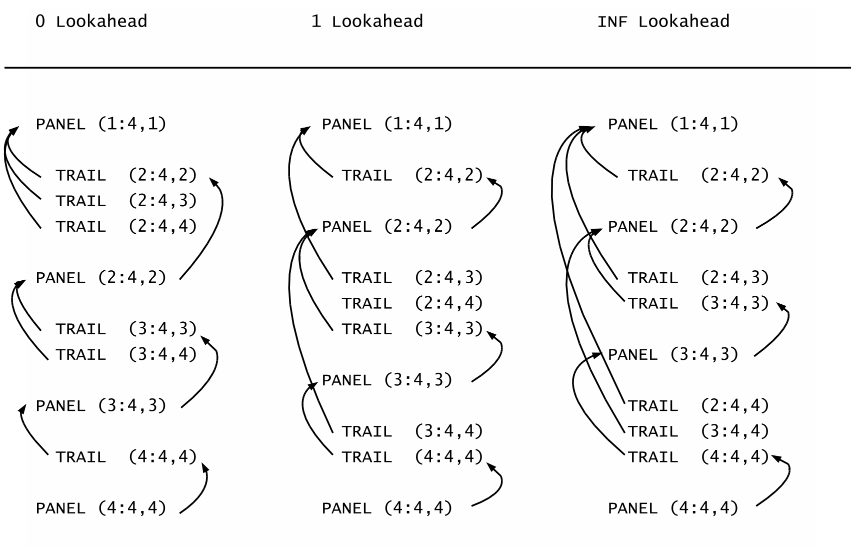

**Fig. 2.** Variants of the LU and QR factorizations. Arrows show data dependencies.

## 4 Results

In most cases any lookahead of a fixed size substantially improves the performance of the discussed algorithms.

It is shown, however, that the use of fixed lookahead is suboptimal and substantial performance gains can be achieved by dynamically controlling the lookahead behaviour at runtime.

## References

1. J. J. Dongarra, L. S. Duff, D. C. Sorensen, H. Van Der Vorst. *Numerical Linear Algebra for High Performance Computers.* SIAM 1999.
2. J. W. Demmel. *Applied Numerical Linear Algebra.* SIAM 1997.
3. C. Bischof, C. Van Loan. *The WY Representation for Products of Householder Matrices.* SIAM J. of Sci. Stat. Comput., **8** (1987) 2–13.
4. R. Schreiber, C. Van Loan. *A Storage Efficient WY Representation for Products of Householder Transformations.* SIAM J. of Sci. Stat. Comput., **10** (1991) 53–57.