Parallel LL^T for dense symmetric matrix divided by columns

P. Pajewski

April 2020

1 Introduction

In this paper we present parallel Cholesky LL^T factorization. This method is widely used for numerical solutions inter alia of linear equations. Realization of decomposition will be done with a use of the MPICC.

2 Cholesky decomposition

If matrix A is hermitian and positive definite its Cholesky decomposition is of the form:

$$A = LL^{\dagger}, \tag{1}$$

where L is a lower triangular matrix and $L^{\dagger}$ is hermitian conjugate of L.
For positive-definite A the Cholesky decomposition is unique.
If matrix A is symmetric we have

$$A = LL^{T}. \tag{2}$$

Example

$$A = \begin{pmatrix} 4 & 12 & -16 \\ 12 & 37 & -43 \\ -16 & -43 & 98 \end{pmatrix} = \begin{pmatrix} 2 & 0 & 0 \\ 6 & 1 & 0 \\ -8 & 5 & 3 \end{pmatrix} \begin{pmatrix} 2 & 6 & -8 \\ 0 & 1 & 5 \\ 0 & 0 & 3 \end{pmatrix},$$

hence

$$L = \begin{pmatrix} 2 & 0 & 0 \\ 6 & 1 & 0 \\ -8 & 5 & 3 \end{pmatrix}.$$

3 Cholesky decomposition algorithm

One may figure out Cholesky factorization algorithm simply solving the equation:

$$\begin{pmatrix} a_{11} & a_{21} & \cdots & a_{n1} \\ a_{21} & a_{22} & \cdots & a_{n2} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & \cdots & a_{nn} \end{pmatrix} = \begin{pmatrix} l_{11} & 0 & \cdots & 0 \\ l_{21} & l_{22} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ l_{n1} & l_{n2} & \cdots & l_{nn} \end{pmatrix} \begin{pmatrix} l_{11} & l_{21} & \cdots & l_{n1} \\ 0 & l_{22} & \cdots & l_{n2} \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & l_{nn} \end{pmatrix} =$$

$$\begin{pmatrix} l_{11}^2 & l_{11}l_{21} & \cdots & l_{11}l_{n1} \\ l_{11}l_{21} & l_{21}^2 + l_{22}^2 & \cdots & l_{21}l_{n1} + l_{22}l_{n2} \\ \vdots & \vdots & \ddots & \vdots \\ l_{11}l_{n1} & l_{21}l_{n1} + l_{22}l_{n2} & \cdots & l_{n1}^2 + ... + l_{nn}^2 \end{pmatrix}$$

Therefore if we compare the terms in general we have

$$\begin{aligned} a_{ii} &= \sum_{j=1}^{i} l_{ij}^2 \\ a_{ki} &= \sum_{j=1}^{i} l_{ij} l_{kj} \end{aligned} \tag{3}$$

For L matrix terms we obtain (we take the convention that second index of the term is always less or equal first that is for every l_{ij} we have $j \leq i$):

$$\begin{aligned} l_{ii} &= \sqrt{a_{ii} - \sum_{j=1}^{i-1} l_{ij}^2} \\ l_{ij} &= \frac{1}{l_{jj}} \left(a_{ij} - \sum_{k=1}^{j-1} l_{jk} l_{ik} \right) \end{aligned} \tag{4}$$

From above equations one immediately notices that calculation must be done in a sequential order:

$$
\begin{aligned}
& l_{11} = \sqrt{a_{11}}, \\
& l_{21} = \frac{a_{21}}{l_{11}}, \ldots, l_{n1} = \frac{a_{n1}}{l_{11}}, \\
& l_{22} = \sqrt{a_{22} - l_{21}^2}, \\
& l_{32} = \frac{1}{l_{22}} \left(a_{32} - l_{21} l_{31} \right), \ldots, l_{n2} = \frac{1}{l_{22}} \left(a_{n2} - l_{21} l_{n1} \right), \\
& \vdots \\
& l_{nn} = \sqrt{a_{nn} - \sum_{j=1}^{n-1} l_{nj}^2}
\end{aligned}
\tag{5}
$$

Let us now emphasize that solving for L must be done sequentially-column after column (from left to right) but in a given column, subtracting terms apparent in sums can be done concurrently which is the most important source of parallelism in column-oriented factorization Cholesky algorithm. In other words, looking at the above formulae calculations are done line by line but for a given line all calculations may be done concurrently.

4 Machine language algorithm

Translating 5 to the machine language we obtain:

for $j = 1$ **to** n
  for $k = 1$ **to** $j - 1$
    for $i = j$ **to** n
    $a_{ij} = a_{ij} - a_{ik} a_{jk}$
    end
  end
  $a_{jj} = \sqrt{a_{jj}}$
  for $i = j + 1$ **to** n
    $a_{ij} = a_{ij} / a_{jj}$
  end
end

We can modify/compactify above algorithm declaring functions:
div(j)- which takes a square root of a diagonal element of the column j and divides non-diagonal elements by a diagonal one:

div(j):
$a_{jj} = \sqrt{a_{jj}}$
for $i = j + 1$ **to** n
$a_{ij} = \frac{a_{ij}}{a_{jj}}$
end

and **mod(j, k)**- which subtracts from columns j elements multiples of columns k elements:

mod(j, k):
for $i = j$ **to** n
$a_{ij} = a_{ij} - a_{ik}a_{jk}$
end

Using above definitions algorithm for Cholesky decomposition may be rewritten as:

for $j = 1$ **to** n
for $k = 1$ **to** $j - 1$
mod(j, k)
end
div(j)
end

Studying data dependencies of the above one sees that **mod(j, k)** functions for a given j and all $k \in 1, 2..., n$ can be done simultaneously (hence parallelism in our algorithm) but **div(j)** function for a given j depends on **mod(j, k)** for $k \in 1, 2..., n$ so this part must be done in an ordered way.

5 Parallel machine language algorithm

In the parallel implementation of the Cholesky decomposition we chose 0-th process to be the one which stores calculated data and organizes calculations. To briefly describe the way code works: 0=th process first calculates a_{11} then sends it to every of 1, 2... n-1 (where n is rank of a matrix) processes which than use it to calculate simultaneously every element of first of first column and sends it back to 0=th. Next 0-th process calculates next diagonal element that is a_{22} and sends it to processes 1, 2...n-2, than every of them calculates different element of the second column. Source of parallelism in our implementation is calculating all below diagonal elements concurrently. All procedure repeats until last column. Than 0-th process prints to the screen L matrix that is Cholesky decomposition. Algorithm in a machine language and matrix M with elements $M[i][j]$ is presented below:

for $k = 0$ **to** $n - 1$
  if process=0
    for $j = 0$ **to** $k - 1$
      $M[k][k] = M[k][k] - M[k][j]^2$
    end
    $M[k][k] = \sqrt{M[k][k]}$
    for $p = 1$ **to** $n - k$
      for $c = 0$ **to** k
        for $r = c$ **to** $n - 1$
          Send $M[r][c]$ **to** process p
          end
        end
        Receive $M[p + k][k]$ **from** process p
  else if $process < n - k$
    for $c = 0$ **to** k
      for $r = c$ **to** $n - 1$
        Receive $M[r][c]$ **from** process 0
      end
      end
    for $g = 0$ **to** $k - 1$
      $M[k + process][k] = M[k + process][k] - M[k][g]M[k + process][g]$
    end
    $M[k + process][k] = \frac{M[k+process][k]}{M[k][k]}$
    Send $M[k + process][k]$ **to** process 0
  end
end

where *process* is an index number of process. In above algorithm matrix indexation is moved to start from 0 because indexation of arrays in c++ begins with 0. Algorithm is meant to be run with n=rank(M) processes. This number maximizes parallelism. Due to the nature of problem more processes are redundant and less cannot fully use parallel power of computations.

References

[1] Parallel Numerical Algorithms, Prof. Michael T. Heath,

[2] Parallel Implementation of Cholesky LLT-Algorithm in FPGA-Based Processor. Oleg Maslennikow1, Volodymyr Lepekha, Anatoli Sergiyenko, Adam Tomas, and Roman Wyrzykowski

[3] Wikipedia: "Cholesky decomposition".