Choleski decomposition in HPJava

Guansong Zhang, Bryan Carpenter

September 23, 1997

1 Introduction

If A is symmetric positive definite, then an often used algorithm for direct solving linear equation is Choleski decomposition.

$$A = LL^T$$

where L is a lower triangular matrix, followed by the forward and back substitutions

$$L\mathbf{y} = \mathbf{b},\ L^T\mathbf{x} = \mathbf{y}$$

to complete the solution $A\mathbf{x} = \mathbf{b}$.

In [?] different kinds of algorithm organization are discussed. Among them, column interleaved *outer product* form and row interleaved *column Choleski* form are considered more promising ones on a parallel computer system with distributed memory.

2 Outer product form

The outer product form of Choleski algorithm is based on the following mathematic equation:

$$A_1 = \begin{bmatrix} a_{11} & \mathbf{a_1}^T \\ \mathbf{a_1} & A_s \end{bmatrix} = \begin{bmatrix} a_{11} & a_{11}^{1/2}\mathbf{l_1}^T \\ a_{11}^{1/2}\mathbf{l_1} & A_2 + \mathbf{l_1}\mathbf{l_1}^T \end{bmatrix}$$

$$= \begin{bmatrix} a_{11}^{1/2} & 0 \\ \mathbf{l_1} & I \end{bmatrix} \begin{bmatrix} 1 & 0 \\ 0 & A_2 \end{bmatrix} \begin{bmatrix} a_{11}^{1/2} & \mathbf{l_1}^T \\ 0 & I \end{bmatrix}$$

the updating of the submatix A_2 is accomplished by subtracting the outer product of $\mathbf{l_1}\mathbf{l_1}^T$.

A simple description of the algorithm in a sequential code may be,

$$
\begin{array}{l}
l_{11} = a_{11}^{1/2} \\
For\ k = 1\ to\ n-1 \\
\quad For\ s = k+1\ to\ n \\
\quad\quad l_{sk} = a_{sk}/l_{kk} \\
\quad For\ j = k+1\ to\ n \\
\quad\quad For\ i = j\ to\ n \\
\quad\quad\quad a_{ij} = a_{ij} - l_{ik}l_{jk} \\
\quad l_{k+1,k+1} = a_{k+1,k+1}^{1/2}
\end{array}
$$

On a parallel computer with a distributed memory, the array may have a column-interleaved storage, then each processor can update its own column after geting the first updated column of the matrix or submatrix.

We write the above algorithm in the following HPJava program. In the code, a cyclic range is used to allocate the orignal array on multiprocessors, the result L can overwrite part of its storage. During the computation, collective communication **remap** is used for broadcasting updated columns.

```
public class ChKjiC {
  public static void main() {

    final int size=100;
    Range x = new CyclicRange(size, id);

    float a[[*,]] = new float [[size, x]];
    // initialize the array;

    float b[[*]] = new float [[size]]; // used as a buffer

    at (x|0)
      a[[0,x]]=sqrt(a[[0,x]]);

    for (int k=0; k<size-1; k++) {

      for (int s=k+1; s<size; s++)
        at (x|k)
          a[[s,x]]/=a[[x,x]];

      Adlib.remap(b[k+1:],a[k,x|k+1:]);

      over (x|k+1:)
        for (int i=x; i<size; i++)
          a[[i,x]]-=b[[i]]*b[[x]];

      at (x|k+1)
        a[[k+1,x]]=sqrt(a[[k+1,x]]);
```

```
    }
  }
}
```

3 Column Choleski form

This form is based on the following:

if $A_{i-1} = L_{i-1}L_{i-1}^T$ is the decomposition of the $(i-1)st$ principal submatrix, then the decomposition of A_i is

$$A_i = \begin{bmatrix} A_{i-1} & \mathbf{a_i} \\ \mathbf{a_i}^T & a_{ii} \end{bmatrix} = \begin{bmatrix} L_{i-1} & 0 \\ \mathbf{l_i}^T & l_{ii} \end{bmatrix} \begin{bmatrix} L_{i-1}^T & \mathbf{l_i} \\ 0 & l_{ii} \end{bmatrix}$$

Thus, the new ith row of L is obtained from $L_{i-1}\mathbf{l_i} = \mathbf{a_i}$, $l_{ii} = (a_{ii} - \mathbf{l_i}^T\mathbf{l_i})^{1/2}$.

A description of this form of algorithm in a sequential code can be,

$$
\begin{array}{l}
l_{11} = a_{11}^{1/2} \\
For\ j = 2\ to\ n \\
\quad For\ s = j\ to\ n \\
\quad\quad l_{sj-1} = a_{sj-1}/l_{j-1,j-1} \\
\quad For\ k = 1\ to\ j-1 \\
\quad\quad For\ i = j\ to\ n \\
\quad\quad\quad a_{ij} = a_{ij} - l_{ik}l_{jk} \\
\quad\quad l_{jj} = a_{jj}^{1/2}
\end{array}
$$

Written in HPJava with row-interleaved storage, the program should be,

```
public class ChJkiR {
  public static void main() {

    final int size=100;

    Range x = new CyclicRange(size, id);

    float a[[,*]] = new float [[x, size]];

    // init array ;

    float b[[*]] = new float [[size]];

    at (x|0)
      a[[x,0]]=sqrt(a[[x,0]]);

    for (int j=1; j<size; j++) {

      Adlib.remap(b[:j-1], a[x|j-1,:j-1]);
```

```
      over(x|j:)
        a[[x, j-1]]/=b[[j-1]];

      for (int k=0; k<j-1; k++) {
        over (x|j:)
          a[[x,j]]-=a[[x,k]]*b[[k]];

        at (x|j)
          a[[x,j]]=sqrt(a[[x,j]]);
      }
    }
  }
}
```

4 Performance

There are three major loops in the above algorithms, with each index as `i`, `j` and `k`. After proper arrangement, we may get 6 different orders of the combination of these loops. Then in each permutation, we can either use row interleaved storage or column interleaved storage. So there will be total 12 different oganizition of the Choleski decomposition. The above two are considered the best.

testing

(to be added)