FACTORIZACIÓN QR POR TILES

Análisis de los Artículos Working Notes 190 y 191

Mihaita Alexandru Lupoiu

18 Enero 2016

Algoritmos Paralelos Matriciales en Ingeniería

UNIVERSITAT POLITÈCNICA DE VALÈNCIA

TRABAJO PROPUESTO

1. Introdución
2. Factorización QR por Bloques
3. Tiled QR Factorization
4. Implementación
5. Conclusion y Trabajos Futuros
6. Resumen

INTRODUCIÓN

La factorización QR es una transformación que factoriza una matriz inicial A de tamaño $m \times n$ en dos matrices Q y R donde Q es una matriz unitaria de tamaño $n \times n$ y R es una matriz triangular de tamaño $m \times m$.

$$A = Q * R$$

Esta factorizacion se realiza aplicando $min(m, n)$ reflexiones de Householder a la matriz A. Como las reflexiones de Householder son transformaciones ortogonales, esta factorización es más estable comparado con la LU pero a cambio de una mayor coste computacional. La factorización QR tiene un coste de $2n^2(m - n/3)$ Flops, mientras que la LU tiene un coste de $n^2(m - n/3)$ FLops.

FACTORIZACIÓN QR POR BLOQUES

FACTORIZACIÓN QR POR BLOQUES

En la librería de LAPACK se utiliza una versión peculiar de la factorización QR que consigue mejores prestaciones en arquitecturas con varios niveles de memória gracias a la división por bloques.

Este algoritmo se basa en acumular una cantidad de transformaciónes de Householder y se denominado *panel factorization*, que luego se aplica todas a la vez aprovechando las rutinas de BLAS de nivel 3. Esta técnica utilizada para acumular las transformaciones de householder se denomina técnica compacta WY.

La rutina de LAPACK que realiza la factorización QR se denomina `DGEQRF`.

FACTORIZACIÓN QR POR BLOQUES

Consideranto una matriz A de tamaño $m \times n$ que se puede representar como:

$$A = \begin{pmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{pmatrix}$$

Donde:

- A_{11} es de tamaño $b \times b$
- A_{12} es de tamaño $b \times (n - b)$
- A_{21} es de tamaño $(m - b) \times b$
- A_{22} es de tamaño $(m - b) \times (n - b)$

PASO1: Factorización del Panel. En este paso se realiza una transformación QR del panel (A_{*1}) como se puede observar en la Ecuación:

$$\begin{pmatrix} A_{11} \\ A_{21} \end{pmatrix} \Longrightarrow \begin{pmatrix} V_{11} \\ V_{21} \end{pmatrix}, (T_{11}), (R_{11}) \tag{1}$$

($V_{*,1}$): matriz con las b reflexiones de Householder.
R_{11}: matriz triangular superior de tamaño $b \times b$ y parte de la matriz final R.
T_{11}: matriz triangular superior de tamaño $b \times b$.

PASO2: Actualización de la submatriz. En este paso la transformación realizada en el apartado anterior es aplicado al resto de matriz como se puede observar en la ecuación:

$$\begin{pmatrix} R_{12} \\ \tilde{A}_{22} \end{pmatrix} = \left(I - \begin{pmatrix} V_{11} \\ V_{21} \end{pmatrix} \cdot (T_{11}) \cdot \begin{pmatrix} V_{11}^T & V_{21}^T \end{pmatrix} \right) \begin{pmatrix} A_{12} \\ A_{22} \end{pmatrix} \quad (2)$$

R_{12}: de tamaño $b \times (n-b)$, parte de la matriz final de R.
$\tilde{A}_{22}$: Matriz restante sobre la cual se vuelve a repetir el algoritmo.

$$A = \begin{pmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{pmatrix} \Longrightarrow \begin{pmatrix} V_{11} \\ V_{21} \end{pmatrix}, \begin{pmatrix} R_{11} & R_{12} \\ 0 & \tilde{A}_{22} \end{pmatrix} \tag{3}$$

TILED QR FACTORIZATION

La idea de distribución dinámica y la ejecución fuera de orden de las tareas se puede aplicar a toda clase de algoritmos que permiten la paralelización de las operaciones de Álgebra Lineal comunes.

Un ejemplo es la factorización de Cholesky, que no es necesario realizar ningún cambio algorítmico ya que se puede paralelizar de forma natural por ”tiles”. Mientras que el algoritmo por bloque de la factorizacion QR tiene un cuello de botella computacional por el tipo de matriz que se emplean.

Con el fin de tener una mayor granularidad en la QR, las operaciones se tienen que paralelizar por tareas y por lo tanto se va a necesitar un cambio algorítmico importante en la QR.

TILED QR FACTORIZATION

El algoritmo que se propone en el artículo "Working Notes 191" es el siguiente:

Algorithm 1: ALGORITMO DE LA FACTORIZACIÓN QR POR TILES.

for $k = 1, 2..., min(p, q)$ **do**
    DGEQRT(A_{kk}, V_{kk}, R_{kk}, T_{kk});
    for $j = k + 1, k + 2, ..., q$ **do**
        DLARFB(A_{kj}, V_{kk}, T_{kk}, R_{kj});
    end
    for $i = k + 1, k + 1, ..., p$ **do**
        DTSQRT(R_{kk}, A_{ik}, V_{ik}, T_{ik});
        for $i = k + 1, k + 1, ..., p$ **do**
            DSSRFB(R_{kj}, A_{ij}, V_{ik}, T_{ik});
        end
    end
end

DGEQRT(A_{kk}, V_{kk}, R_{kk}, T_{kk}):

$$A_{kk} \longrightarrow (V_{kk}, R_{kk}, T_{kk}) = QR(A_{kk})$$

DLARFB(A_{kj}, V_{kk}, T_{kk}, R_{kj}):

$$A_{kj}, V_{kk}, T_{kk} \longrightarrow R_{kj} = (I - V_{kk} T_{kk} V_{kk}^T) A_{kj}$$

DTSQRT(R_{kk}, A_{ik}, V_{ik}, T_{ik}):

$$\begin{pmatrix} R_{kk} \\ A_{ik} \end{pmatrix} \longrightarrow (V_{ik}, T_{ik}, R_{kk}) = QR \begin{pmatrix} R_{kk} \\ A_{ik} \end{pmatrix}$$

DSSRFB(R_{kj}, A_{ij}, V_{ik}, T_{ik}):

$$\begin{pmatrix} R_{kj} \\ A_{ij} \end{pmatrix}, V_{ik}, T_{ik} \longrightarrow \begin{pmatrix} R_{kj} \\ A_{ij} \end{pmatrix} = (I - V_{ik} T_{ik} V_{ik}^T) \begin{pmatrix} R_{kj} \\ A_{ij} \end{pmatrix}$$

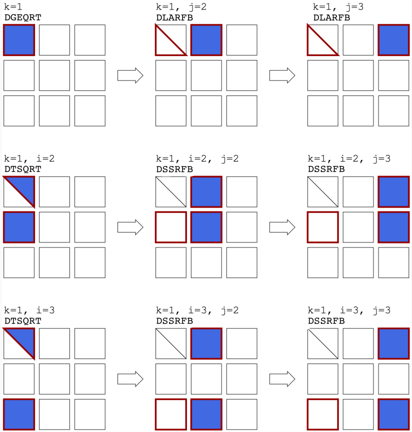

Figure 1: Graphical representation of one repetition of the outer loop in Algorithm(1) on a matrix with $p = q = 3$.

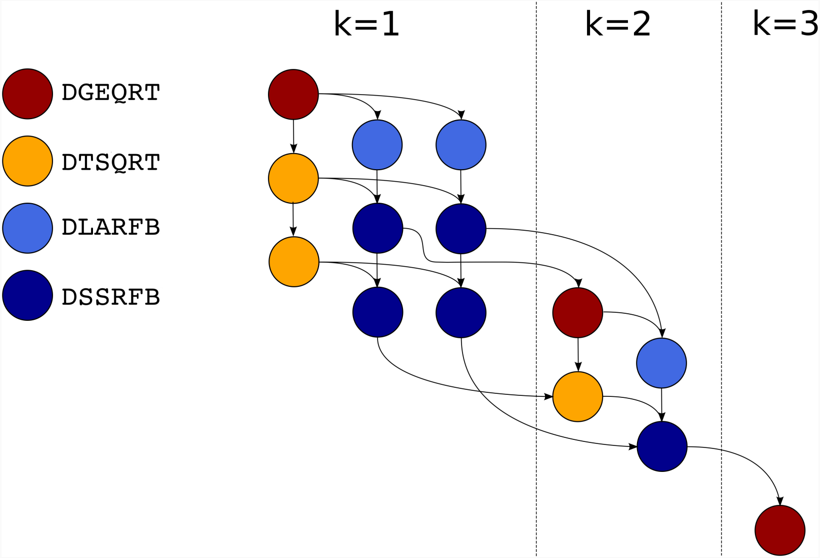

Figure 2: The dependency graph of Algorithm(1) on a matrix with $p = q = 3$.

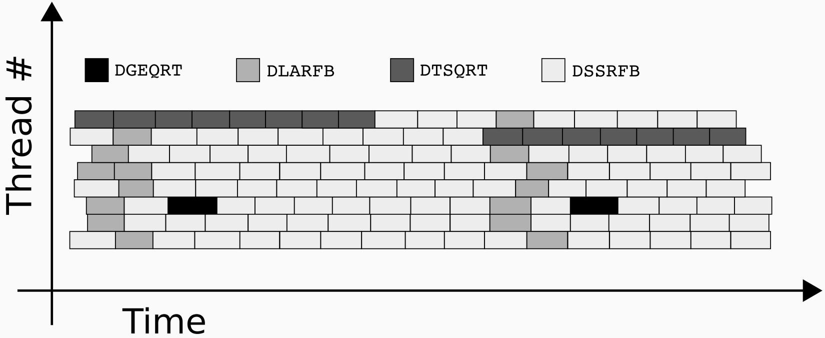

Figure 3: The execution flow for dynamic scheduling, out of order execution of Algorithm(1).

IMPLEMENTACIÓN

A la hora de realizar la implementación fue el hecho de encontrar las rutinas del LAPACK para poder utilizarlas. El nombre de estas rutinas se habían cambiado y tras hablar con Alfredo Buttari[1], descubrí que las rutinas que tengo que utilizar son las siguientes:

- DGEQRT
- DGEMQRT que coresponde a DLARFB
- DTPQRT que coresponde a DTSQRT
- DTPMQRT que coresponde a DSSRFB

También cabe destacar que estas funciones requieren que la librería de LAPACK este en una versión más recientes(3.5.0).

[1] Uno de los autores de los artículos

Una vez instalado y comprobado que se encuentran las rutinas, se tuvo que preparar la matriz de entrada para que se pueda procesar por bloques. El código correspondiente sería el siguiente:

```
void convtile(double *A, double *B, int n, int bs, int tofrom)
{
  int i, j, ii, jj, i2, j2, nb=n/bs;

  for (i=0; i<nb; i++) {
    for (j=0; j<nb; j++) {
      for (jj=0; jj<bs; jj++) {
        if (tofrom)
        ^^Imemcpy(B+i*bs+(j*bs+jj)*n, A+(i+j*nb)*bs*bs+jj*bs, bs*sizeof(double));
        else
        ^^Imemcpy(B+(i+j*nb)*bs*bs+jj*bs, A+i*bs+(j*bs+jj)*n, bs*sizeof(double));
      }
    }
  }
}
```

```
284.000000  87.000000  16.000000  87.000000  63.000000  27.000000
87.000000  278.000000  94.000000  93.000000  28.000000  41.000000
16.000000  94.000000  236.000000  50.000000  91.000000  27.000000
87.000000  93.000000  50.000000  222.000000  60.000000  73.000000
63.000000  28.000000  91.000000  60.000000  264.000000  37.000000
27.000000  41.000000  27.000000  73.000000  37.000000  212.000000
```

```
284.000000  87.000000  87.000000  278.000000  16.000000  87.000000
94.000000  93.000000  63.000000  27.000000  28.000000  41.000000
16.000000  94.000000  87.000000  93.000000  236.000000  50.000000
50.000000  222.000000  91.000000  27.000000  60.000000  73.000000
63.000000  28.000000  27.000000  41.000000  91.000000  60.000000
27.000000  73.000000  264.000000  37.000000  37.000000  212.000000
```

CÓDIGO TILED QR FACTORIZATION

El código del algoritmo secuencial:

```
void QR_LAPACK_Tile(double *A, int lA, int bs, int *info)
{
    int i=0, j=0, k=0, nb=lA/bs, lda=bs, m=bs, n=bs, bs2=bs*bs, ldt=bs, K=m, ldc = lda;
    double * T = (double *) calloc(ldt * bs, sizeof( double ) );
    double *work = (double *) calloc(n*bs, sizeof( double ) );

    for (k=0; k<nb; k++) { //min(p, q)
        //DGEQRT(Akk, Vkk, Rkk, Tkk)
        dgeqrt_(&m, &n, &bs, A+(k+k*nb)*bs2, &lda, T, &ldt, work, info);
        if (*info != 0) return;
        for (j=k+1; j<nb; j++){ //q
            //DLARFB(Akj , Vkk, Tkk, Rkj )
            dgemqrt_("L", "T", &m, &n, &K, &bs, A+(k+k*nb)*bs2, &lda, T, &ldt, A+(j+k*nb)*bs2, &ldc, work, &info
        }
        for (i=k+1; i<nb; i++){ //p
            DTSQRT(Rkk, Aik, Vik, Tik)
            dtpqrt_(&m, &n, &n, &bs, A, dla, B, ldb, T, ldt, work, info);

            for (j=k+1; j<nb; j++){ //q
                DSSRFB(Rkj , Aij , Vik, Tik)
                dtpmqrt_("L", "T", &m,&n,&k,&l,&nb,v,&ldv,t,&ldt,A,&lda,b,&ldb,work,info);
            }
        }
    }
}
```

Para la comprobación de los resultados obtenidos se pensaba utilizar la rutina del LAPACK y comparar los resultados.

```
void QR_LAPACK(double *A, int lA){
    int n = lA;
    int m = n;

    int info = 0;
    int k = n;          /* k = min(m,n);       */
    int lda = m;        /* lda = max(m,1);     */
    int lwork = n;      /* lwork = max(n,1);   */
    int max = lwork;    /* max = max(lwork,1); */

    double *work;
    double *tau;
    double *vec;
    work = (double *) calloc(max, sizeof( double ) );
    tau  = (double *) calloc( k, sizeof( double ) );
    vec  = (double *) calloc( m, sizeof( double ) );

    dgeqrf_(&m, &n, A, &lda, tau, work, &lwork, &info);

    free(work);
    free(tau);
    free(vec);
}
```

CONCLUSION Y TRABAJOS FUTUROS

El algoritmo de la factorización *QR* se puede implementar de manera concurrente por tiles y con unas mejoras considerables si se observan en los artículos Working Notes 190 y 191.

Como trabajo futuro se debería de acabar la implementación del algoritmo secuencial. Para posteriormente utlizar OpenMP 4.4 o Quarck para paralelizar las tareas.

Una vez funcionando se debería de comparar las prestaciones que se han obtenido y compararlos con las de la facteoriazación QR que ofrece LAPACK.

RESUMEN

Pueden conseguir todo el código fuente en

`github.com/MihaiLupoiu/AMPI`

¡Gracias!

¿Alguna pregunta?