Relazione Progetto SCPD

Michele Valfrè

Ottobre 2022

1 Introduzione: Metodo di Jacobi

Il metodo di Jacobi è un metodo iterativo per la soluzione di sistemi di equazioni lineari nella forma $Ax = b$ tali che A sia diagonalmente dominante, ovvero

$$\forall i,\ |a_{ii}| >= \sum_{j \neq i} |a_{ij}| \tag{1}$$

Il passo k-esimo dell' iterazione è descritto dall' equazione

$$x_i^k = \frac{1}{a_{i,i}}[b_i - \sum_{j \neq i} a_{i,j} x_j^{k-1}]. \tag{2}$$

La complessità del metodo di Jacobi dipende dal numero di iterazioni e dalla precisione richiesta.

2 Algoritmo Sequenziale

L' algoritmo, preso un sistema di equazioni lineari *system* dotato di attributi A, b, x, *rows* e *cols*, ad ogni iterazione, scorre le righe della matrice $system.A$ ed applica l' equazione (2):

Algorithm 1 Jacobi Sequenziale

1: **function** JACOBI_SEQ(*system*,*iterations*)
2: **while** $iterations > 0$ **do**
3: $old\_x \leftarrow system.x$
4: $i \leftarrow 0$
5: **while** $i < system.rows$ **do**
6: $sum \leftarrow \sum_{j \neq i} system.A[i] - old\_x[j]$
7: $system.x[i] \leftarrow \frac{1}{system.A[i][i]}(system.b[i] - sum)$
8: $i \leftarrow i + 1$
9: $iterations \leftarrow iterations - 1$

Nell' implementazione sequenziale, si è scelto di utilizzare la seguente classe:

```
typedef struct _lin_sys{
    long double * a;
    long double * b;
    long double * x;
    int rows;
    int cols;
} linear_equation_system;
```

Come si può notare, sono mantenuti sia il numero di righe che il numero di colonne di A. Si tratta di un' informazione ridondante nel caso dell' implementazione sequenziale(una delle precondizioni è che A sia quadrata), tuttavia, tornerà utile nell' implementazione parallela dove, nel caso generale, le singole UC non opereranno su matrici quadrate.

3 Algoritmo Parallelo

Siccome, entro la medesima iterazione, non vi sono dipendenze fra le operazioni, il metodo di Jacobi è una computazione di tipo *embarassingly parallel*: il problema può essere scomposto in più parti che non sono legate da una relazione di dipendenza. In particolare, definendo P_i il processo con $rank = i$, l' implementazione parallela è riassunta dai seguenti punti:

3.1 Caso 1: il numero di processi divide esattamente il numero di righe

1. P_0 distribuisce fra se stesso e gli altri processi la matrice A e il vettore b. Dunque, il processo P_i avrà una matrice A_i di dimensione $\frac{numero\_righe}{numero\_processi} \times numero\_colonne$ contenente le righe comprese fra l' i-esima e la $(i +$

$\frac{numero\_righe}{numero\_processi}$)-esima e i relativi valori di b;

2. P_0 distribuisce ad ogni processo una copia del vettore x tramite una $Bcast$;

3. ogni processo applica l' equazione (2) solo alle proprie variabili(il processo P_i modificherà le variabili comprese fra $x[i]$ e $x[i + \frac{numero\_righe}{numero\_processi}]$ ma avrà bisogno anche delle altre per eseguire la sommatoria).

4. i risultati vengono raccolti nel vettore x di P_0 con una $Gather$(ogni processo contribuirà solo con la porzione di vettore di propria competenza);

5. ritorna a (2).

3.2 Caso 2: il numero di processi non divide esattamente il numero di righe

TODO

4 Sperimentazione