

OpenMp

Introduzione agli strumenti

Esempio: Prodotto Matrice-Vettore

Prof. G. Laccetti

Esempio: Hello World

Libreria

```
#include <omp.h>
#include <stdio.h>
int main()
{
```

Creazione di un
team di thread

```
    #pragma omp parallel
```

```
    printf("Hello from thread %d, nthreads %d\n", omp_get_thread_num(), omp_get_num_threads());
    return 0;
```

```
}
```

Library function
per conoscere l'id
del thread
chiamante

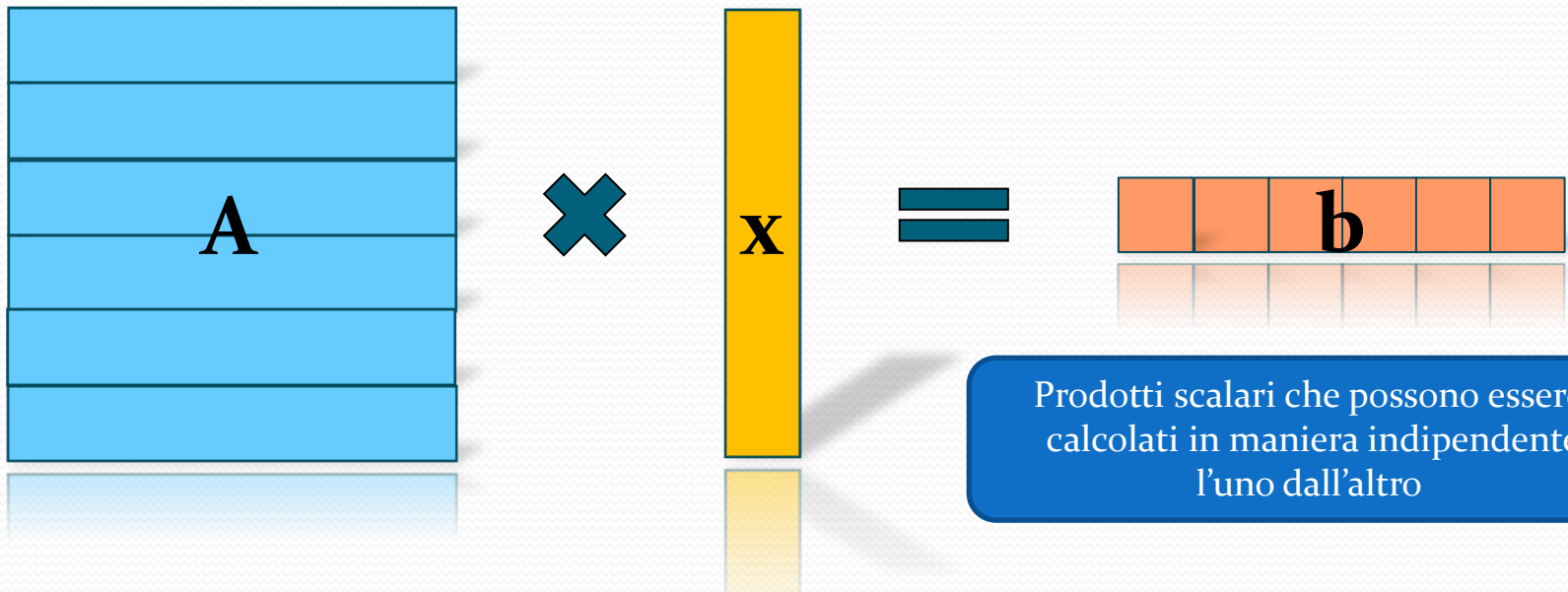
Library function
per conoscere il
numero di thread
attivi

Esempio: Prodotto Mat-Vet

Problema

Prodotto $Ax=b$

$$A \in \mathbb{R}^{n \times m} \quad x \in \mathbb{R}^m \quad b \in \mathbb{R}^n$$



Esempio: Prodotto Mat-Vet

Programma chiamante

```
main()  
begin
```

dichiarazione delle variabili

allocazione della memoria

*assegnazione di valori alle dimensioni della matrice (**n,m**)*

*assegnazione dei valori alla matrice **A** ed al vettore **x***

*inizializzazione del vettore risultato **b***

```
b:=matxvet(m,n,x,A)
```

```
for i:=0 to n-1
```

```
begin
```

```
    stampa b[i]
```

```
end
```

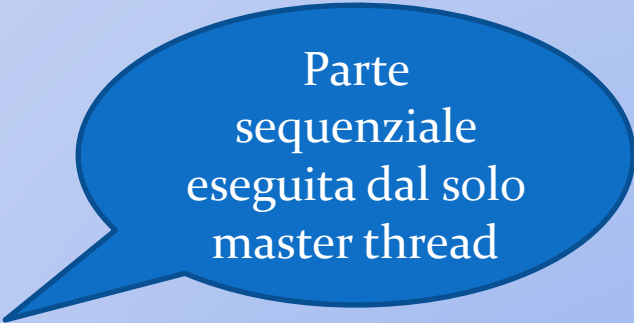
```
end
```

Funzione che realizza il
prodotto della matrice **A** per il
vettore **x**

Esempio: Prodotto Mat-Vet

```
#include <omp.h>
#include <math.h>
#include <stdlib.h>
...

double * matxvet(int m, int n, double *x, double **A){
    int i,j;
    double *b;
    ...
    /*allocazione memoria per b*/
    ...
}
```



Parte
sequenziale
eseguita dal solo
master thread

Esempio: Prodotto Mat-Vet

```
#include <omp.h>
#include <math.h>
#include <stdio.h>
...
double b, x, double **A){
    double b,
    ...
    /*allocazione memoria per b*/
    ...
    #pragma omp parallel for default(none) shared(m,n,A,x,b) private(i,j)
    for (i=0; i<n; i++){
        for (j=0; j<m; j++){
            b[i] += A[i][j]*x[j];
        }
    }
}
```

Creazione di un
team di thread

i-mo prodotto
scalare

Esempio: Prodotto Mat-Vet

```
#include <omp.h>
#include <math.h>
#include <stdlib.h>
...
```

```
double * matxvet(int m, int n, double *x, double *b)
{
    int i,j;
    double *b;
    ...
    /*allocazione memoria per b*/
    ...
    #pragma omp parallel for default(none) shared(m,n,A,x,b) private(i,j)
    for (i=0; i<n; i++){
        for (j=0; j<m; j++)
            b[i] += A[i][j]*x[j];
    }
}
```

Distribuzione
delle iterazioni
del for tra i
thread

Con la clausola
schedule si può
scegliere la
distribuzione

i-mo prodotto
scalare

Esempio: Prodotto Mat-Vet

```
#include <omp.h>
#include <math.h>
#include <stdlib.h>
...
```

```
double * matxvet
```

```
    int i,j;
    double *b;
```

```
    ...
    /*allocazione memoria per b*/
```

```
    ...
    #pragma omp parallel for default(none) shared(m,n,A,x,b) private(i,j)
```

```
    for (i=0; i<n; i++){
```

```
        for (j=0; j<m; j++){
```

```
            b[i] += A[i][j]*x[j];
```

```
        }
```

```
}
```

Sarà il
programmatore a
stabilire cosa è
condiviso e cosa
non lo è

i-mo prodotto
scalare

Esempio: Prodotto Mat-Vet

```
#include <omp.h>
#include <math.h>
#include <stdlib.h>
...
```

```
double * matxvet(int m, int n, double *x, double **A){
    int i,j;
    double *b;
```

```
...
    /*allocazione memoria per b*/
```

```
...
    #pragma omp parallel for default(none) shared(m,n,A,x,b) private(i,j)
```

```
    for (i=0; i<n; i++){
```

```
        for (j=0; j<m; j++){
```

```
            b[i] += A[i][j]*x[j];
```

```
        }
```

```
}
```

Variabili
condivise tra
i thread

Tutti devono
poterle
leggere

Se fossero private
m, n, A e x
verrebbero
de-inizializzate

i-mo prodotto
scalare

Esempio: Prodotto Mat-Vet

```
#include <omp.h>
#include <math.h>
#include <stdlib.h>
...
```

```
double * matxvet(int m, int n, double *x, double **A){
    int i,j;
    double *b;
```

```
...
    /*allocazione memoria per b*/
```

```
...
    #pragma omp parallel for default(none) shared(m,n,A,x,b) private(i,j)
```

```
    for (i=0; i<n; i++){
```

```
        for (j=0; j<m; j++){
```

```
            b[i] += A[i][j]*x[j];
```

```
        }
```

```
}
```

Variabili
condivise tra
i thread

Tutti devono
poterle
leggere

Se fosse privata b,
non sarebbe
accessibile fuori
dalla regione
parallela

i-mo prodotto
scalare

Esempio: Prodotto Mat-Vet

```
#include <omp.h>
#include <math.h>
#include <stdlib.h>
...
```

```
double *
```

**Comportamento
imprevedibile**

```
...
/*allocazione memoria per b*/
```

```
...
#pragma omp parallel for default(none) shared(m,n,A,x,b) private(i,j)
for (i=0; i<n; i++){
    for (j=0; j<m; j++)
        b[i] += A[i][j]*x[j];
}
```

```
}
```

Se i e j fossero
condivise, un
thread potrebbe
modificare il
contatore per un
altro

Variabili
private per
ogni thread

i-mo prodotto
scalare

Esempio: Prodotto Mat-Vet

```
#include <omp.h>
#include <math.h>
#include <stdlib.h>
...

double * matxvet(int m, int n, double *x, double **A){
    int i,j;
    double *b;
    ...
    /*allocazione memoria per b*/
    ...
    #pragma omp parallel for default(none) shared(m,n,A,x,b) private(i,j)
    for (i=0; i<n; i++){
        for (j=0; j<m; j++){
            b[i] += A[i][j]*x[j];
        }
    }
    /*-- Fine del for parallelo--*/
    return b;
}
```

Barriera di
sincronizzazione

Parte
sequenziale
eseguita dal solo
master thread

Esempio: Prodotto Mat-Vet

```
#include <omp.h>
#include <math.h>
#include <stdlib.h>
...
```

```
double * matxvet(int m, int n, double *x, double **A){
    int i,j;
    double *b;
    ...
    /*allocazione memoria per b*/
    ...
    #pragma omp parallel for default(none) shared(m,n,A,x,b) private(i,j)
    for (i=0; i<n; i++){
        for (j=0; j<m; j++){
            b[i] += A[i][j]*x[j];
        }
    }
    /*-- Fine del for parallelo--*/
    return b;
}
```

A come array
lineare

Ottimizzabile con
la cura di alcuni
dettagli

vettori *restricted*
Es: `double * restrict x`

Forzare i vettori ad
occupare regioni
disgiunte di memoria

Riferimenti bibliografici

Using OpenMp

B. Chapman, G. Jost, R. van der Pas

The MIT Press

<http://openmp.org/>

<https://computing.llnl.gov/tutorials/openMP/>