Producto $Ax = b$

Producto $Ax = b$

- Una matriz de m-filas y n-columnas
- Multiplicación con un vector de n-filas
- Un vector de m-filas
  - Producto punto de elementos de una fila de la matriz por el vector x

$$\begin{bmatrix} a_{1,1} & a_{1,2} & \cdots & a_{1,n} \\ a_{2,1} & & & \vdots \\ \vdots & & \ddots & \\ a_{m,1} & \cdots & & a_{m,n} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} = \begin{bmatrix} a_{1,1}x_1 + a_{1,2}x_2 + \ldots + a_{1,n}x_n \\ a_{2,1}x_1 + a_{2,2}x_2 + \ldots + a_{2,n}x_n \\ \ldots \\ a_{m,1}x_1 + a_{m,2}x_2 + \ldots + a_{m,n}x_n \end{bmatrix}$$

Producto $Ax = b$

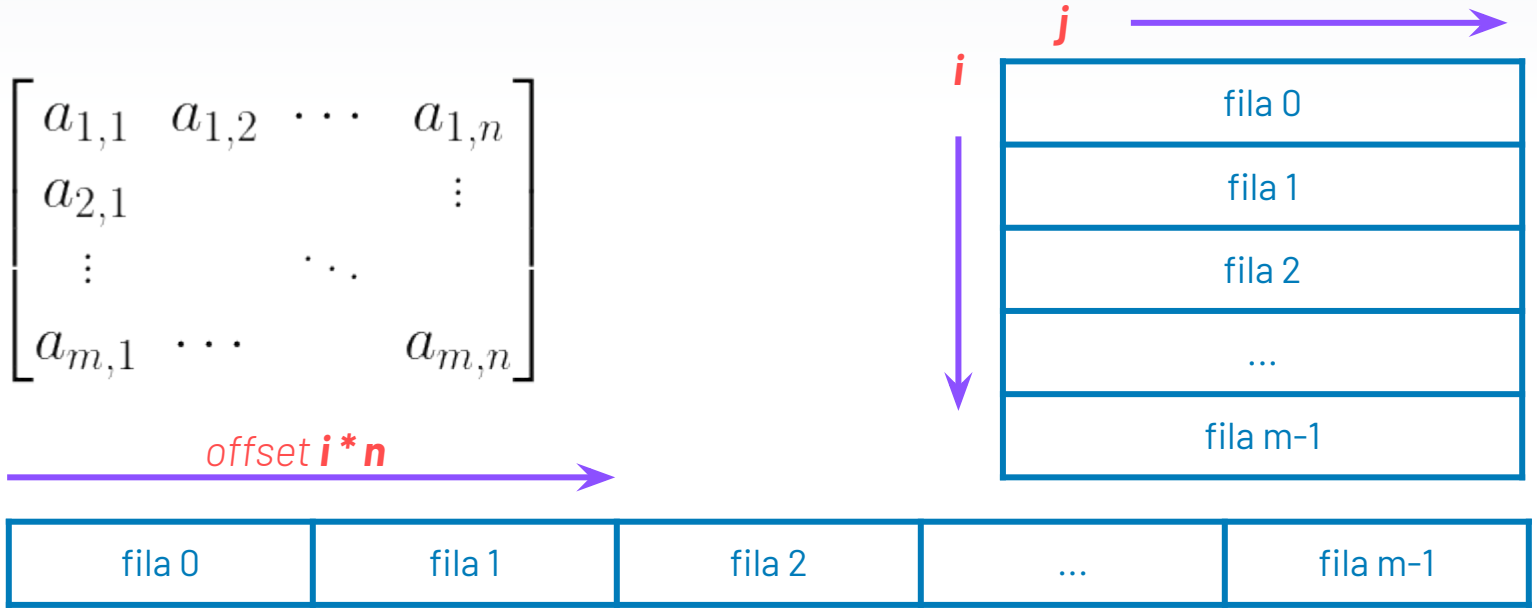
$$\begin{bmatrix} a_{1,1} & a_{1,2} & \cdots & a_{1,n} \\ a_{2,1} & & & \vdots \\ \vdots & & \ddots & \\ a_{m,1} & \cdots & & a_{m,n} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} = \begin{bmatrix} a_{1,1}x_1 + a_{1,2}x_2 + \ldots + a_{1,n}x_n \\ a_{2,1}x_1 + a_{2,2}x_2 + \ldots + a_{2,n}x_n \\ \ldots \\ a_{m,1}x_1 + a_{m,2}x_2 + \ldots + a_{m,n}x_n \end{bmatrix}$$

$$\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 6 & 5 & 4 \\ 3 & 2 & 1 \end{bmatrix} \begin{bmatrix} 4 \\ 3 \\ 2 \end{bmatrix} = \begin{bmatrix} 1(4) + 2(3) + 3(2) \\ 4(4) + 5(3) + 6(2) \\ 6(4) + 5(3) + 4(2) \\ 3(4) + 2(3) + 1(2) \end{bmatrix} = \begin{bmatrix} 16 \\ 43 \\ 44 \\ 20 \end{bmatrix}$$

Representación row-variant

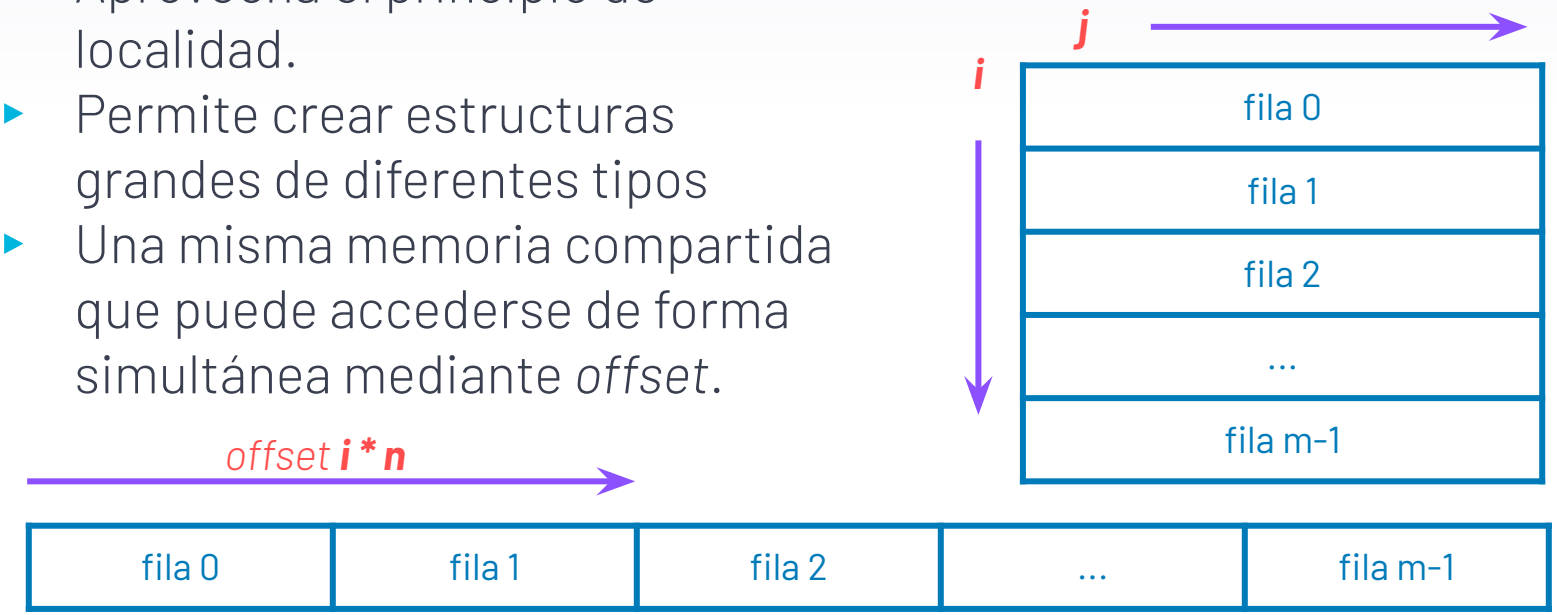
Representamos la matriz como una consecución de elementos de filas (rows) consecutivas en un arreglo de longitud *mxn*.

$$\begin{bmatrix} a_{1,1} & a_{1,2} & \cdots & a_{1,n} \\ a_{2,1} & & & \vdots \\ \vdots & & \ddots & \\ a_{m,1} & \cdots & & a_{m,n} \end{bmatrix}$$



Ventajas de row-variant

- Aprovecha el principio de localidad.
- Permite crear estructuras grandes de diferentes tipos
- Una misma memoria compartida que puede accederse de forma simultánea mediante *offset*.



Producto *Ax=b* en row-variant

```
void Ax_b(int m, int n, double* A, double* x, double* b) {
    int i, j;
    for(i = 0; i < m; i++) {
        b[i] = 0.0;  //inicialización elemento i vec. res.
        for(j = 0; j < n; j++) {
            b[i] += A[i*n + j] * x[j]; //producto punto
        }
    }
}
```

Producto *Ax=b* paralelo

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

    #pragma omp parallel for
    for(i = 0; i < m; i++) {
        b[i] = 0.0;  //inicialización elemento i del vec.
        for(j = 0; j < n; j++) {
            b[i] += A[i*n + j] * c[j]; //producto punto
        }
    }/*---Fin de parallel for---*/
}
```

Ejercicio 3

- Descargue el programa prodAx.c
- Inicie el programa con 10,000 x 10,000 elementos y córralo
- Duplique y agregue parallel for (prodAx_for.c)
- Ejecute ambos programas con

```
time ./ejecutable
```

- Compare los tiempos de ejecución

Producto *Ax=b* paralelo

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

    #pragma omp parallel for shared(m,n,A,x,b) private(i,j)
    for(i = 0; i < m; i++) {
        b[i] = 0.0;  //inicialización elemento i del vec.
        for(j = 0; j < n; j++) {
            b[i] += A[i*n + j] * c[j]; //producto punto
        }
    }/*---Fin de parallel for---*/
}
```

Ejercicio 4

- Copie el programa paralelo (prodAx_for_scope.c)
- Modifique el entorno de las variables
- Ejecute ambos programas
  - Compare los tiempos de ejecución e intente calcular el producto para una matriz 30,000 x 30,000

Producto *Ax=b* paralelo

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

    #pragma omp parallel for shared(m,n,A,x,b) private(i,j)\
            schedule(static,100)
    for(i = 0; i < m; i++) {
        b[i] = 0.0;  //inicialización elemento i del vec.
        for(j = 0; j < n; j++) {
            b[i] += A[i*n + j] * c[j]; //producto punto
        }
    }/*---Fin de parallel for---*/
}
```

Ejercicio 5

- Duplique y modifique la estrategia de planificación en el último programa (prodAx_for_schedule.c) con los siguientes block_size:
  - Static (100,000, 10,000, 1,000)
  - Dynamic (100,000, 10,000, 1,000)
  - Guided (1,000, 100, 10)
- Compare los tiempos y resultados al usar diferentes estrategias
- Cree una tabla con sus resultados

Ejercicio Final

- Ahora, trate de optimizar y obtener mejor performance que la version mas rapida que tengan.
- Pueden utilizar cualquier técnica y constructo conocido, o bien alguno que hayan leído.
- Sugerencia: Revisar Barlas 4.6 donde hablan de constructos de Sync (tema que hablaremos viernes).