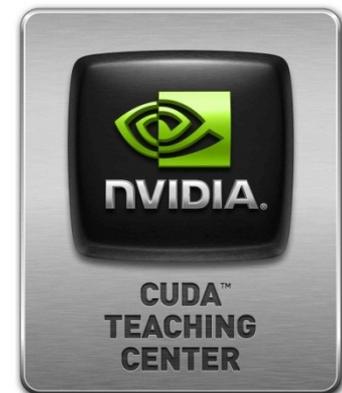


# Computación de Propósito General en Unidades de Procesamiento Gráfico (GPGPU)

**E. Dufrechou, P. Ezzatti, M. Pedemonte**



UNIVERSIDAD  
DE LA REPUBLICA  
URUGUAY



# Clases 3

## Programación paralela

# Contenido

- **Conceptos básicos de programación paralela**

# Conceptos básicos de Programación Paralela

# Conceptos Básicos de Programación Paralela

- **Múltiples unidades de cómputo trabajando en forma coordinada**
  - **Comunicación**
  - **Sincronización**
- **División**
  - **de tareas**
  - **de datos**
  - **híbrida**

# Conceptos Básicos de Programación Paralela

- **Paralelismo de**
  - **Memoria compartida (multi-core)**
  - **Memoria distribuida (cluster)**
- **Contexto**
  - **Homogéneo**
  - **Heterogéneo**
- **Balance de carga**

# Conceptos Básicos de Programación Paralela

En este caso (lo que vemos en el curso):

- Muchas unidades de cómputo trabajando en forma coordinada.
- División de datos.
- Paralelismo “similar” a paralelismo a memoria compartida (multi-core).
- Unidades de procesamiento homogéneas.

# Conceptos Básicos de Programación Paralela

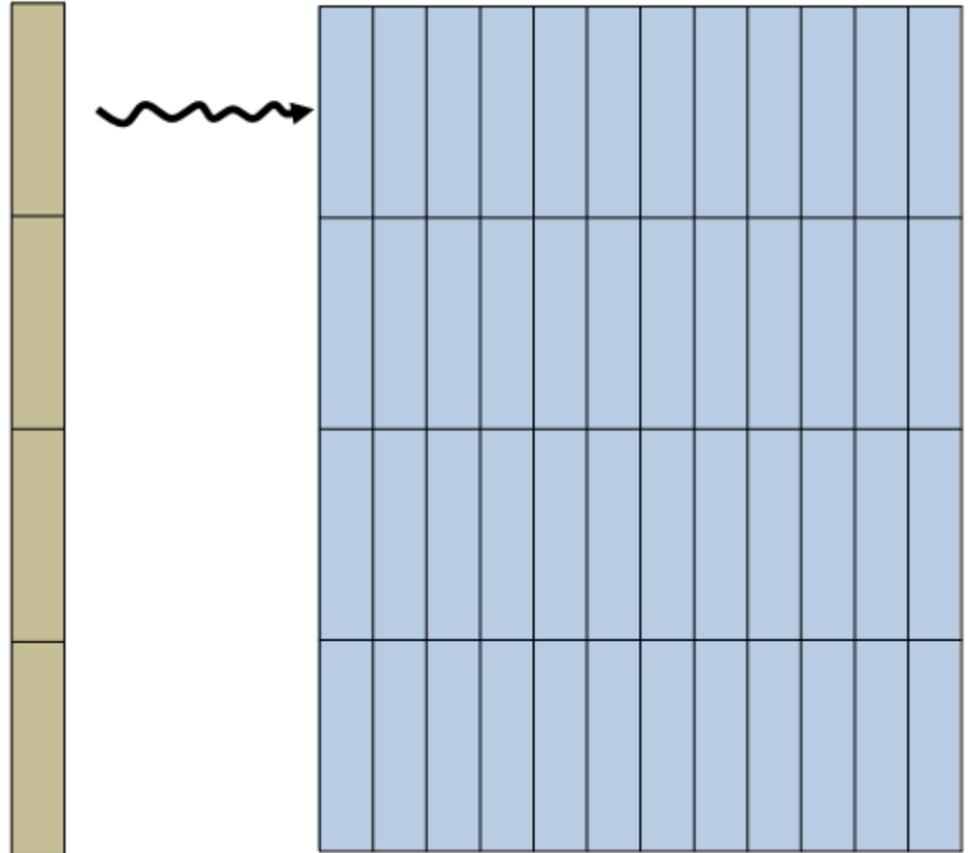
En los ejemplos que se verán a continuación:

- Se utilizará una sintaxis inspirada en la de Matlab/Octave.
- El código escrito en color negro ejecuta en forma secuencial.
- El código escrito en color azul indica palabras reservadas que son directivas para el manejo de paralelismo.
- El código escrito en color rojo indica el punto de “ramificación” del paralelismo.
- El código escrito en color verde es el código que es ejecutado en forma paralela por cada una de las unidades de procesamiento.

# Conceptos Básicos de Programación Paralela

## Ejemplo 0:

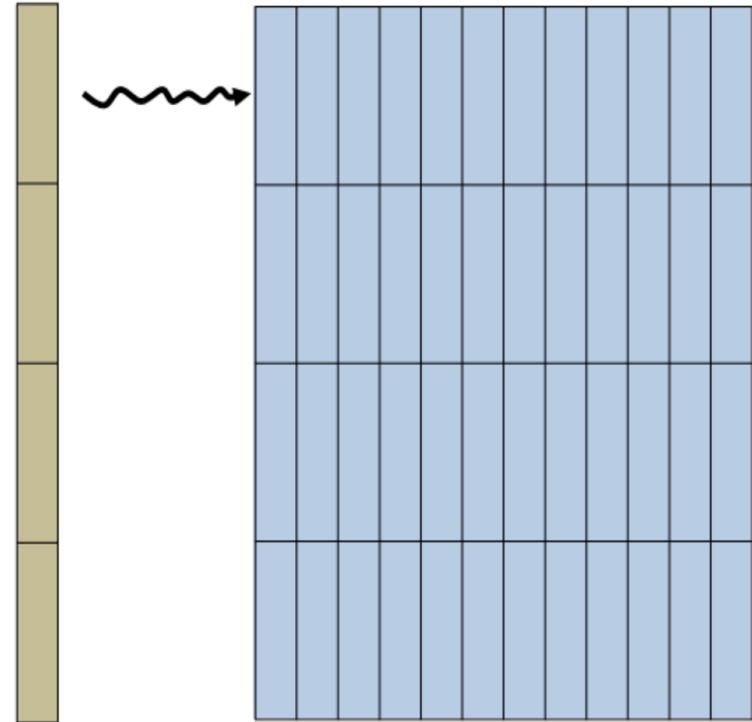
- Se tiene una matriz con 4 filas y se quiere obtener la suma por filas de la matriz.
- Se dispone de una única unidad de procesamiento.



# Conceptos Básicos de Programación Paralela

## Ejemplo 0:

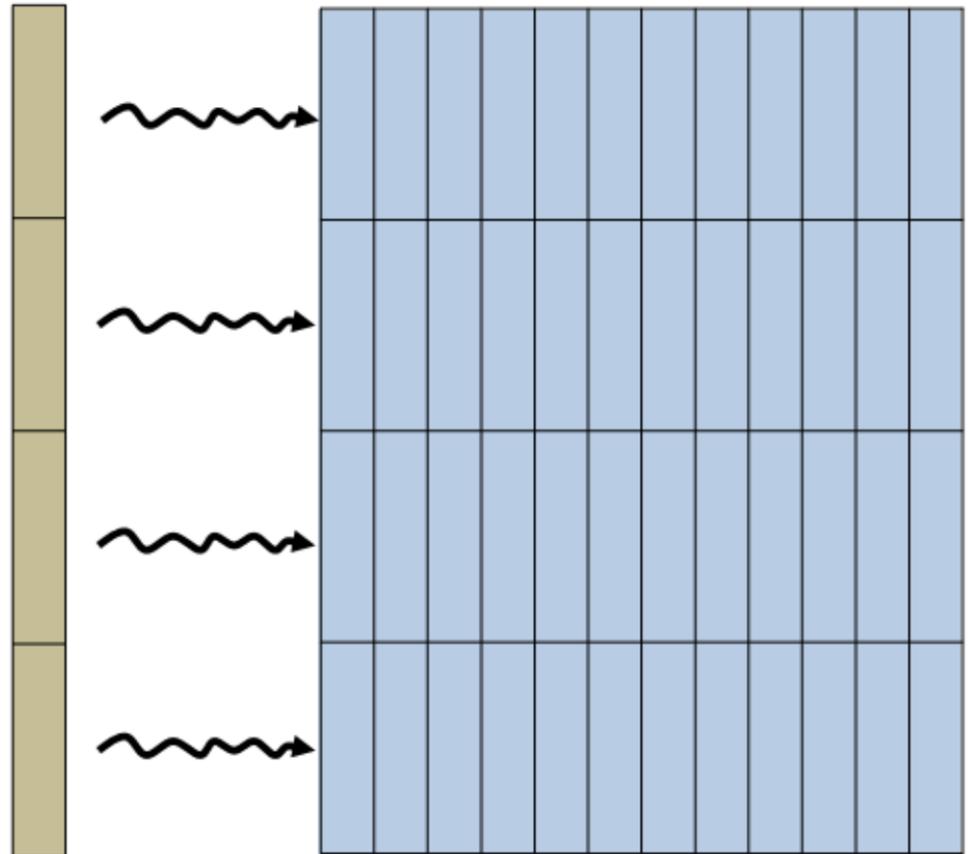
```
Res = zeros(1,4)
[m,n] = size(Mat)
For i = 1:4
    For j = 1 : n
        Res(i) = Res(i) + Mat(i,j)
```



# Conceptos Básicos de Programación Paralela

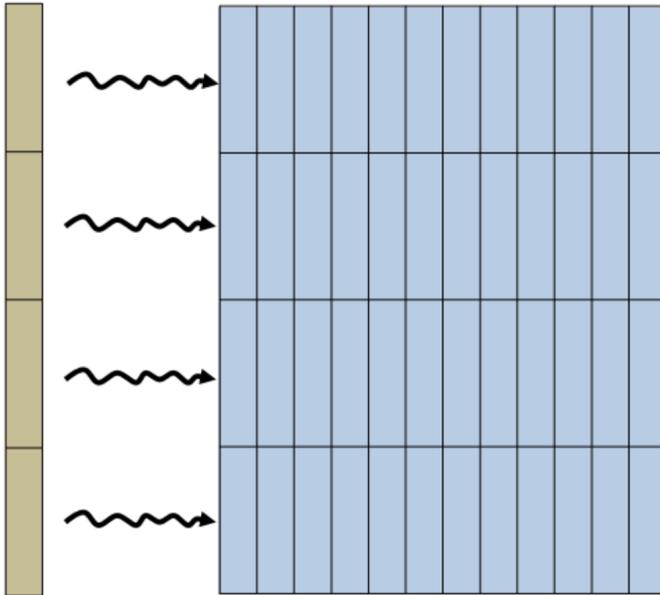
## Ejemplo 1:

- Se tiene una matriz con 4 filas y se quiere obtener la suma por filas de la matriz.
- Se dispone de cuatro unidades de procesamiento.



# Conceptos Básicos de Programación Paralela

## Ejemplo 1:

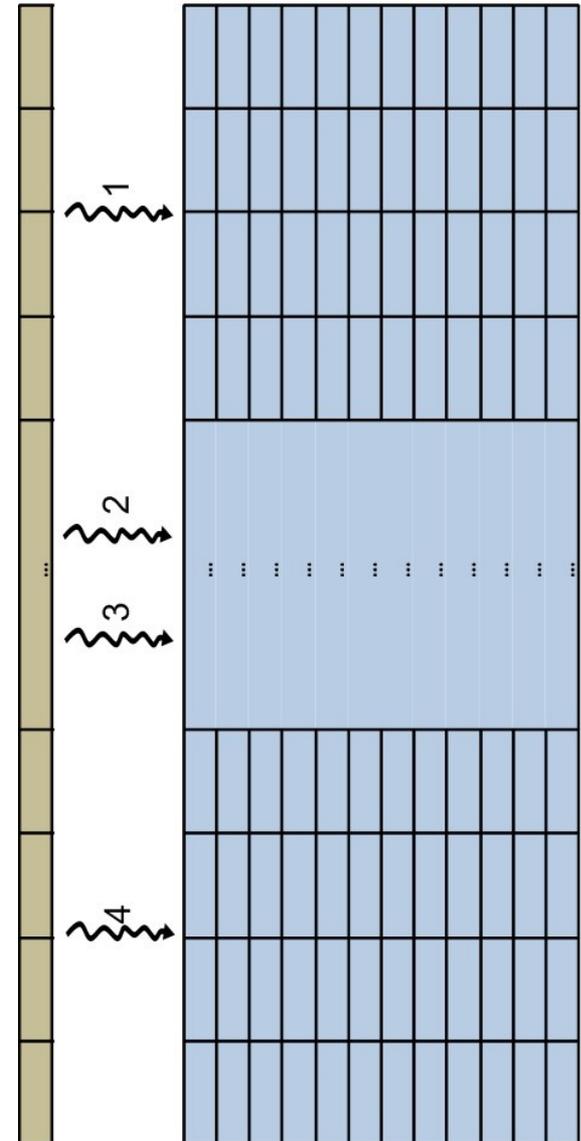


```
Res = zeros(1,4)
[m,n] = size(Mat)
begin_parallel(4)
For i = 1:4
    For j = 1:n
        Res(i) = Res(i) + Mat(i,j)
    end_parallel
end_parallel
```

# Conceptos Básicos de Programación Paralela

## Ejemplo 2:

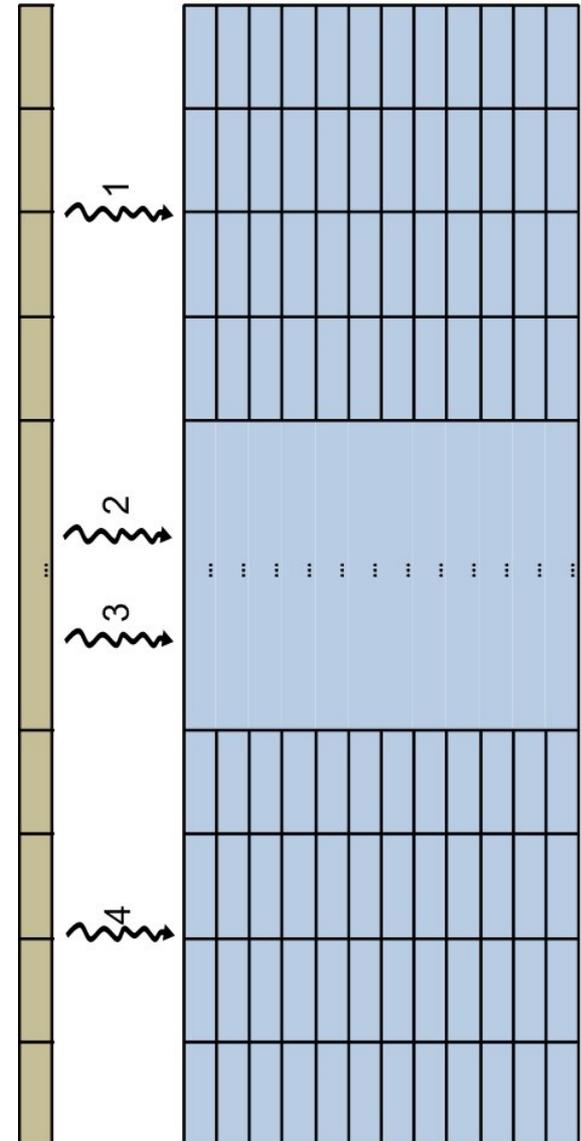
- Se tiene una matriz con 400 filas y se quiere obtener la suma por filas de la matriz.
- Se dispone de cuatro unidades de procesamiento.



# Conceptos Básicos de Programación Paralela

## Ejemplo 2:

```
Res = zeros(1,400)
[m,n] = size(Mat)
begin_parallel(4)
For i = 1:400
    For j = 1 : n
        Res(i) = Res(i) + Mat(i,j)
    end_parallel
end_parallel
```



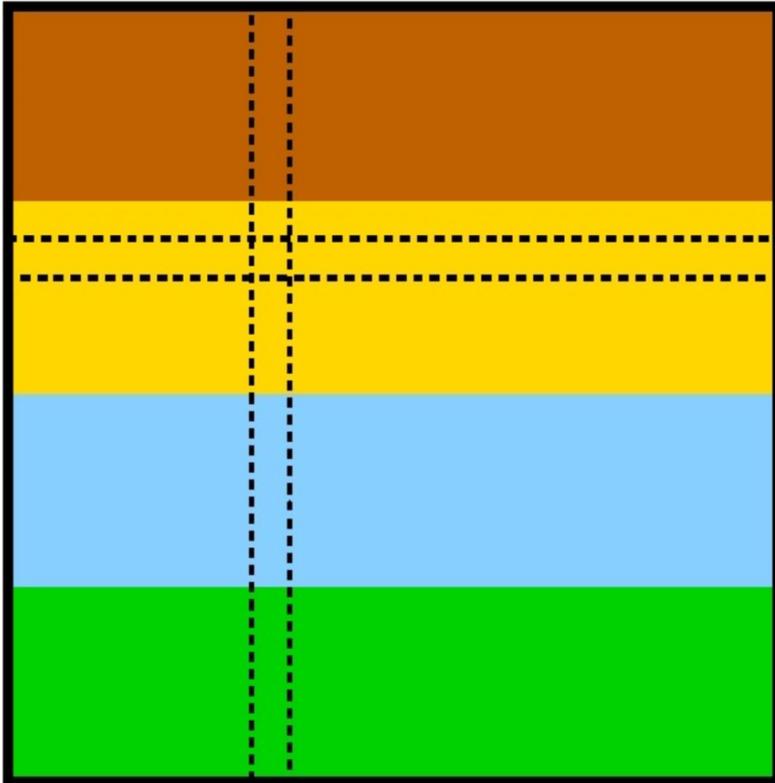
# Conceptos Básicos de Programación Paralela

## Ejemplo 3:

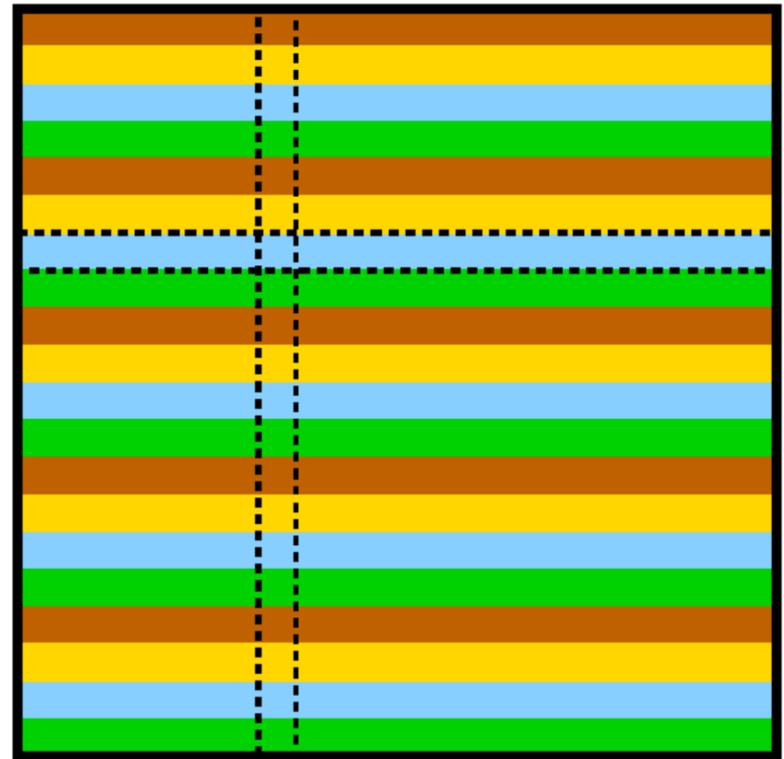
- Se tiene una matriz cuadrada con 400 filas y se quiere obtener la suma por filas de la matriz triangular superior.
- Se dispone de cuatro unidades de procesamiento.

# Conceptos Básicos de Programación Paralela

## Distribución de datos/cálculos



**Distribución por bloque**



**Distribución cíclica**

# Conceptos Básicos de Programación Paralela

## Ejemplo 3:

- Es posible achicar el tamaño de los bloques.
- Por ejemplo: de 100 filas pasar a 10 filas por bloque.

```
Res = zeros(1,400)
[m,n] = size(Mat)
begin_parallel(4, [40,cyc])
For i = 1:400
    For j = i : n
        Res(i) = Res(i) + Mat(i,j)
    end_parallel
end_parallel
```

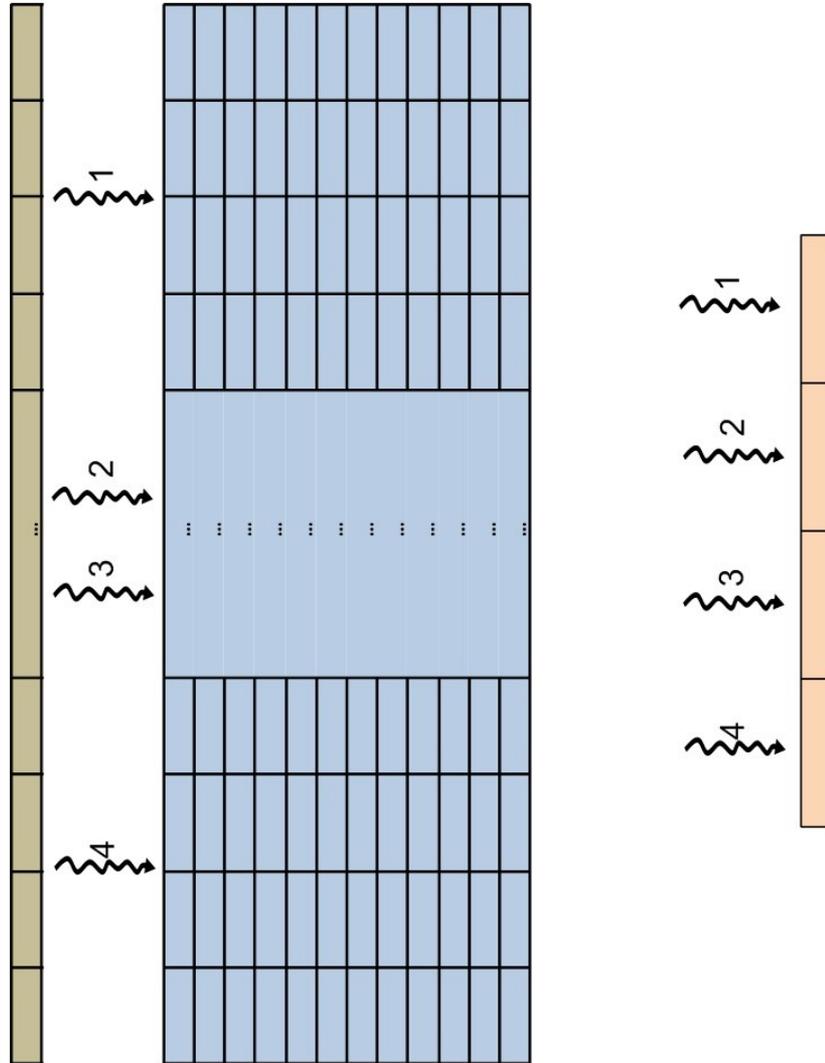
# Conceptos Básicos de Programación Paralela

## Ejemplo 4:

- Se tiene una matriz con 400 filas y se quiere obtener la suma por filas de la matriz.
- Se dispone de cuatro unidades de procesamiento.
- En este caso, además se quiere obtener la suma parcial por unidad de procesamiento.

# Conceptos Básicos de Programación Paralela

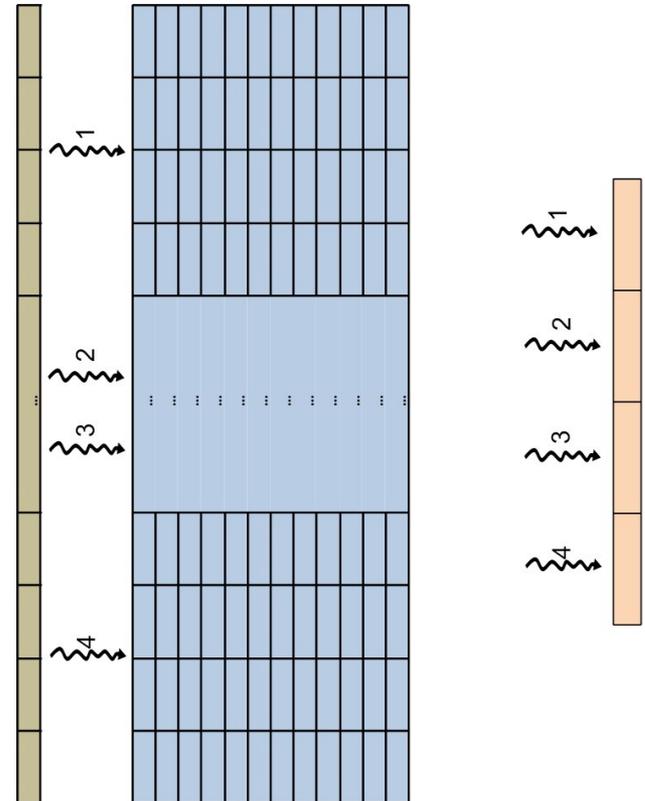
## Ejemplo 4:



# Conceptos Básicos de Programación Paralela

## Ejemplo 4:

```
Res = zeros(1,400)
Res2 = zeros(1,4)
[m,n] = size(Mat)
begin_parallel(4)
For i = 1:400
    private idP = myId()
    For j = 1 : n
        Res(i) = Res(i) + Mat(i,j)
        Res2(idP) = Res2(idP) + Res(i)
    end_parallel
end_parallel
```



# Conceptos Básicos de Programación Paralela

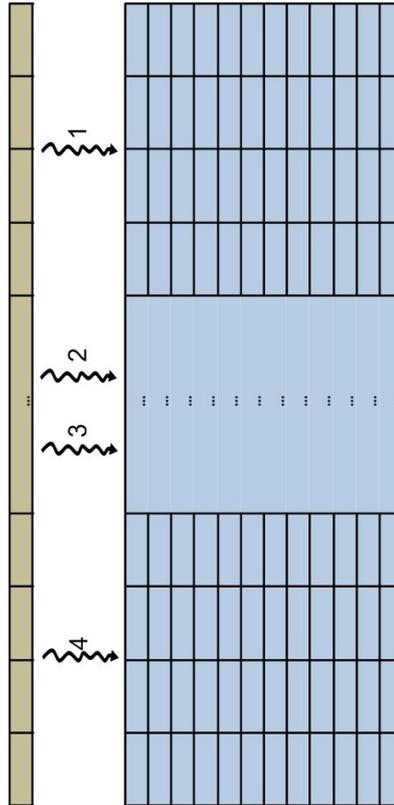
## Ejemplo 5:

- Se tiene una matriz con 400 filas y se quiere obtener la suma de todos los elementos de la matriz.
- Se dispone de cuatro unidades de procesamiento.

# Conceptos Básicos de Programación Paralela

## Solución 1

- Se suma por fila



- Luego se suma todo el vector intermedio



# Conceptos Básicos de Programación Paralela

## Solución 1

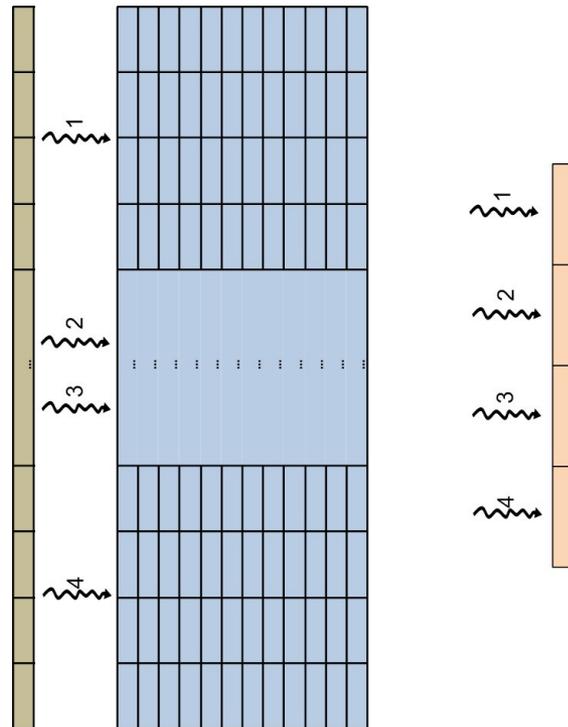
- Se suma por fila y luego se suma el vector intermedio.

```
Res = zeros(1,400)
[m,n] = size(Mat)
begin_parallel(4)
For i = 1:400
    For j = 1 : n
        Res(i) = Res(i) + Mat(i,j)
    end_parallel
end_parallel
Sum = 0
For i = 1:400
    Sum = Sum + Res(i)
```

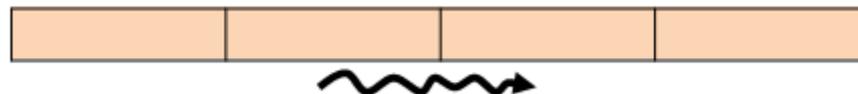
# Conceptos Básicos de Programación Paralela

## Solución 2

- Se suma por filas para cada unidad de procesamiento.



- Luego se suma el vector intermedio.



# Conceptos Básicos de Programación Paralela

## Solución 2

- Se suma por filas para cada unidad de procesamiento y luego se suma el vector intermedio.

```
Res = zeros(1,4)
[m,n] = size(Mat)
begin_parallel(4)
For i = 1:400
    private idP = myId ()
    For j = 1 : n
        Res(idP) = Res(idP) +
        Mat(i,j)
    end_parallel
Sum = 0
For i = 1:4
    Sum = Sum + Res(i)
```

# Conceptos Básicos de Programación Paralela

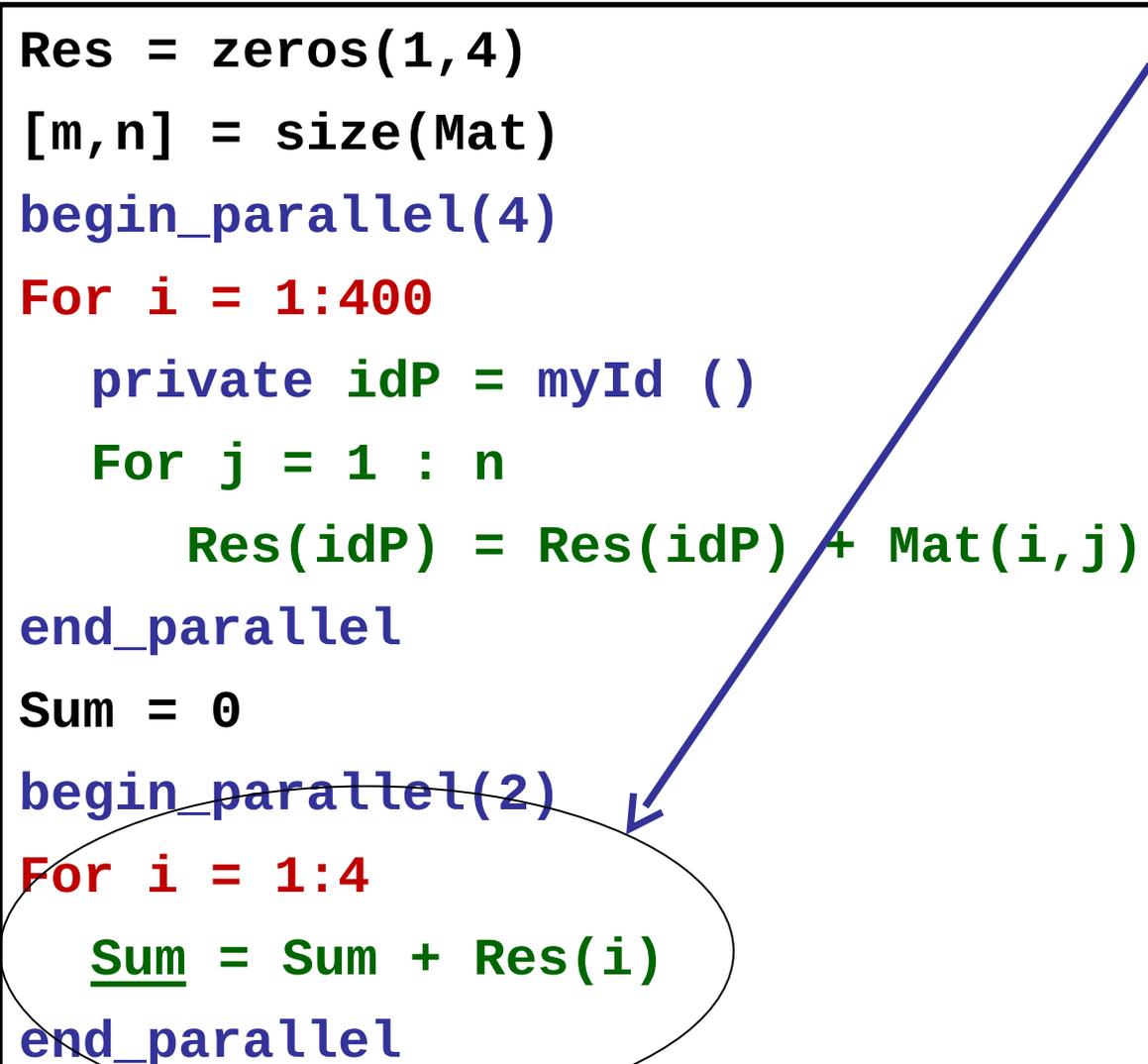
## Solución 3

- Sumo por fila y actualizo el resultado.
- Concepto de reduce

```
Res = zeros(1,4)
[m,n] = size(Mat)
begin_parallel(4)
  For i = 1:400
    private idP = myId ()
    For j = 1 : n
      Res(idP) = Res(idP) + Mat(i,j)
    end_parallel
  end_parallel
Sum = 0
begin_parallel(2)
  For i = 1:4
    Sum = Sum + Res(i)
  end_parallel
```

# Conceptos Básicos de Programación Paralela

```
Res = zeros(1,4)
[m,n] = size(Mat)
begin_parallel(4)
For i = 1:400
    private idP = myId ()
    For j = 1 : n
        Res(idP) = Res(idP) + Mat(i,j)
    end_parallel
end_parallel
Sum = 0
begin_parallel(2)
For i = 1:4
    Sum = Sum + Res(i)
end_parallel
```



Race condition:

- Los hilos compiten por el acceso a los registros.
- Pueden haber distintos ordenes de ejecución.
- Distinto orden de ejecución produce resultados diferentes!!!
- Este problema también aparece en circuitos, base de datos, etc.

# Conceptos Básicos de Programación Paralela

## Race condition:

Valor en Memoria	Hilo 1	Hilo 2
0		
0	Lee valor	
0	Incrementa valor	
1	Escribe valor	
1		Lee valor
1		Incrementa valor
2		Escribe valor

Valor en Memoria	Hilo 1	Hilo 2
0		
0	Lee valor	
0		Lee valor
0	Incrementa valor	
0		Incrementa valor
1	Escribe valor	
1		Escribe valor

# Conceptos Básicos de Programación Paralela

```
[m,n] = size(Mat)
begin_parallel(4)
private Res
For i = 1:400
    %private idP = myId ()
    For j = 1 : n
        Res = Res + Mat(i,j)
    end_parallel(reduce(+, Res))
end_parallel(reduce(+, Res))
```

## Reduce:

- Generalmente están disponibles diversas operaciones:

- +
- \*
- min
- max

# Conceptos Básicos de Programación Paralela

Consideremos el siguiente código:

- Hay problemas por el acceso concurrente a la variable aux.

```
Res = zeros(1,11)
aux = 11
begin_parallel(400)
  For i = 1:400
    if (aux == 1)
      aux = aux + 10
    else
      aux = aux - 1
    end
    Res(aux) = Res(aux) + Mat(i,aux)
  end_parallel
```

# Conceptos Básicos de Programación Paralela

Diferentes formas de resolverlo:

- Semáforos
- Zonas mutuo excluidas
- Operaciones atómicas

```
Res = zeros(1,11)
aux = 11
begin_parallel(400)
For i = 1:400
    if (aux == 1)
        aux = aux + 10
    else
        aux = aux - 1
    end
    Res(aux) = Res(aux) + Mat(i,aux)
end_parallel
```

# Conceptos Básicos de Programación Paralela

Con zonas mutuo  
excluidas

Solo un “hilo de  
ejecución” entra en la  
zona.

```
Res = zeros(1,11)
aux = 11
begin_parallel(400)
For i = 1:400
    begin_mutex()
    if (aux == 1)
        aux = aux + 10
    else
        aux = aux - 1
    end
    Res(aux) = Res(aux) + Mat(i,aux)
end_mutex()
end_parallel
```

# Conceptos Básicos de Programación Paralela

## Operaciones atómicas

```
Res = zeros(1,400)
[m,n] = size(Mat)
begin_parallel(4)
For i = 1:400
    For j = 1 : n
        Res(i) = Res(i) + Mat(i,j)
        Sum = Sum + Res(i)
    end_parallel
end_parallel
```

# Conceptos Básicos de Programación Paralela

## Operaciones atómicas

```
Res = zeros(1,400)
[m,n] = size(Mat)
Sum = 0;
begin_parallel(4)
For i = 1:400
    For j = 1 : n
        Res(i) = Res(i) + Mat(i,j)
        begin_atomic()
        Sum = Sum + Res(j)
        end_atomic()
    end_parallel
end_parallel
```

# Conceptos Básicos de Programación Paralela

Diferentes caminos dependiendo del thread ..

```
begin_parallel(400)
For i = 1:400
    private idP = myId ()
    if (mod(idP,2) == 0)
        Mat(i) = Mat(i) + 4
    else
        Mat(i) = Mat(i) + 2
    end
end_parallel
```

- Estrategia comúnmente utilizada para implementar el modelo de paralelismo Maestro-Esclavo: if soyMaestro()

# Conceptos Básicos de Programación Paralela

## Conceptos vistos

- **Paralelismo**
- **Estrategias de scheduling y balance de carga**
- **Distribución de datos/cálculos**
- **Utilizar información del thread/proceso/etc.**
- **Reducciones**
- **Race conditions**
- **Operaciones atómicas**

# Conceptos Básicos de Programación Paralela

Es conveniente tener en cuenta:

- La importancia del scheduling, distribución de datos/cálculos y control de sincronizaciones para un **buen desempeño**.
- Si bien parece obvio, además de buen desempeño es necesario **correctitud**. Sin embargo, esto es más complejo de comprobar que en programas secuenciales.