

# Programación Paralela en Fortran95 usando OpenMP (Parte I)

**Miguel Hermanns**

Universidad Politécnica de Madrid, España

26 de abril de 2007

# Estructura de la clase

Programación  
Paralela en  
Fortran95  
usando OpenMP

Miguel Hermanns

- 1 Introducción y motivación de OpenMP
- 2 Características fundamentales de OpenMP
- 3 Regiones paralelas y regiones seriales
- 4 Conceptos generales y terminología habitual
- 5 Variables compartidas y variables privadas
- 6 Distribución de la carga computacional
- 7 Recomendaciones personales

Introducción y  
motivación

Regiones paralelas

Conceptos y  
terminología

Cláusulas de datos

Directivas de  
distribución de  
trabajo

Directivas  
combinadas

Recomendaciones

Resumen

# ¿Qué es OpenMP?

Programación  
Paralela en  
Fortran95  
usando OpenMP

Miguel Hermanns

Introducción y  
motivación

Regiones paralelas

Conceptos y  
terminología

Cláusulas de datos

Directivas de  
distribución de  
trabajo

Directivas  
combinadas

Recomendaciones

Resumen

El *OpenMP Application Program Interface (API)* es:

Conjunto de **directivas de compilación, librerías y variables de entorno** que permiten expresar el paralelismo de tareas en programas escritos en C, C++ y Fortran



Toda la información está disponible en la página web:

[www.openmp.org](http://www.openmp.org)

# ¿Quién define OpenMP?

El estándar es definido y revisado por el comité

## OpenMP Architecture Review Board

compuesto actualmente por: (1/2)

- U.S. DoD: Aeronautical Systems Center
- cOMPunity (Comunidad de usuarios de OpenMP)
- Edinburgh Parallel Computing Centre (EPCC)
- Laboratorio NASA Ames
- ST Microelectronics: The Portland Group
- Fujitsu
- Hewlett Packard

# ¿Quién define OpenMP?

El estándar es definido y revisado por el comité

## OpenMP Architecture Review Board

compuesto actualmente por: (2/2)

- International Business Machines (IBM)
- Intel Corporation
- KAI Software Lab (KSL)
- NEC Corporation
- Silicon Graphics Inc. (SGI)
- Sun Microsystems
- Universidad RWTH Aachen

# Historia de OpenMP

## Antes de la introducción de OpenMP:

- Cada fabricante tenía su propio “estándar”
- Los programas tenían **portabilidad nula**
- El estándar ANSI X3H5 fracasó por falta de apoyo
- Se perdió interés en las máquinas SMP

El estándar **OpenMP** se introdujo para resolver esto:

- En 1997 se publicó la versión 1.0
- En 1999 se publicó la versión 1.1
- En 2000 se publicó la versión 2.0
- En 2005 se publicó la **versión 2.5**

# Características fundamentales de OpenMP

OpenMP se **fundamenta** en directivas y cláusulas:

- Las directivas especifican qué hacer
- Las cláusulas indican cómo hacerlo
- Se añaden como **comentarios** al código

Tipos de **directivas** en OpenMP:

- Directivas de distribución de trabajo
- Directivas de sincronización de tareas

Tipos de **cláusulas** en OpenMP:

- Cláusulas de datos
- Cláusulas de comportamiento
- Cláusulas de sincronía

# Características fundamentales de OpenMP

Programación  
Paralela en  
Fortran95  
usando OpenMP

Miguel Hermanns

Introducción y  
motivación

Regiones paralelas

Conceptos y  
terminología

Cláusulas de datos

Directivas de  
distribución de  
trabajo

Directivas  
combinadas

Recomendaciones

Resumen

Se definen dos tipos de **“comentarios OpenMP”**:

**!\$OMP**: precede a todas las directivas de OpenMP:

```
!$OMP PARALLEL DEFAULT(NONE) SHARED(A,B) &  
!$OMP PRIVATE(C,D) REDUCTION(+:A)
```

**!\$**: precede a líneas que sólo se compilan con OpenMP:

```
Interval = (b - a)  
!$ Interval = (b - a) * OMP_get_thread_num() / &  
!$ (OMP_get_num_threads() - 1)
```

Ambos “centinelas” **no pueden ir precedidos de código**:

```
Interval = (b - a) !$ * OMP_get_thread_num() ...
```



# Características fundamentales de OpenMP

Programación  
Paralela en  
Fortran95  
usando OpenMP

Miguel Hermanns

## Ventajas de OpenMP:

- Sencillo: no requiere el envío de mensajes como en MPI
- La descomposición de los datos es automática
- Admite la **paralelización incremental** del código
- Mismo código fuente para las versiones serial y paralela
- Permite implementar granularidad gruesa y fina
- Elevada portabilidad

Introducción y  
motivación

Regiones paralelas

Conceptos y  
terminología

Cláusulas de datos

Directivas de  
distribución de  
trabajo

Directivas  
combinadas

Recomendaciones

Resumen

## Inconvenientes de OpenMP:

- **Requiere** un compilador que entienda OpenMP
- El estándar sólo contempla **C y Fortran**
- Sólo es eficiente en máquinas de memoria compartida
- En general la eficiencia paralela es baja
- La escalabilidad está limitada por el acceso a memoria
- Aumentar la eficiencia requiere reestructurar el código

# Compiladores que soportan OpenMP

- SGI MIPSpro
- IBM XL
- Sun Studio
- Portland Group Compilers and Tools
- Absoft Pro FortranMP
- Lahey/Fujitsu Fortran 95
- Fujitsu-Siemens Fortran 95 (Solaris)
- Intel Software Development Products
- PathScale
- **GNU GCC (a partir de la versión 4.2)**
- HP

# Compiladores que soportan OpenMP

Programación  
Paralela en  
Fortran95  
usando OpenMP

Miguel Hermanns

El **soporte para OpenMP** ha de ser activado:

**Versión serial** de un programa:

```
hermanns@bender:~$ ifort ejemplo.f90
```

**Versión paralela** de un programa:

```
hermanns@bender:~$ ifort -openmp ejemplo.f90
```

En casi todos los compiladores la opción es `-openmp`

Introducción y  
motivación

Regiones paralelas

Conceptos y  
terminología

Cláusulas de datos

Directivas de  
distribución de  
trabajo

Directivas  
combinadas

Recomendaciones

Resumen

# Crear regiones paralelas con !\$OMP PARALLEL

Las directivas !\$OMP PARALLEL/ !\$OMP END PARALLEL:

- Definen una **región paralela**
- !\$OMP PARALLEL crea un conjunto de **tareas**
- Las **cláusulas** especifican condiciones a cumplir
- Todas las tareas ejecutan las instrucciones
- !\$OMP END PARALLEL finaliza la región paralela

**Sintáxis** de la directiva:

```
!$OMP PARALLEL clause1 clause2 ...
```

```
...instructions...
```

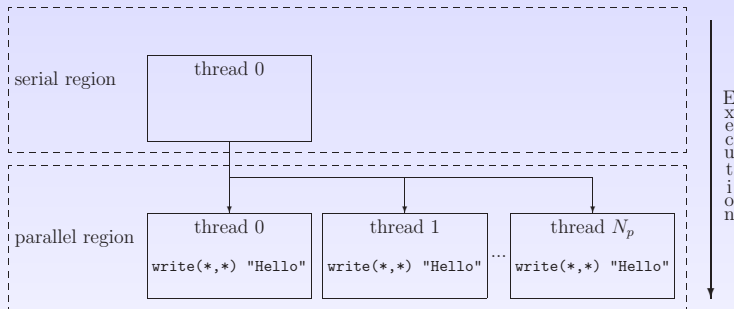
```
!$OMP END PARALLEL
```

# Crear regiones paralelas con !\$OMP PARALLEL

**Ejemplo** de región paralela:

```
!$OMP PARALLEL  
  write(*,*) "Hello"  
!$OMP END PARALLEL
```

**Representación gráfica del ejemplo:**



# Maneras de especificar el número de tareas

- La **variable de entorno** OMP\_NUM\_THREADS:

```
hermanns@bender:~$ export OMP_NUM_THREADS=4
```

- La **subrutina** OMP\_set\_num\_threads:

```
use omp_lib

call OMP_set_num_threads(4)

!$OMP PARALLEL
...
!$OMP END PARALLEL
```

- La **cláusula** NUM\_THREADS:

```
!$OMP PARALLEL NUM_THREADS(4)
...
!$OMP END PARALLEL
```

Cada una de ellas **sobreescribe** la especificación anterior

# La cláusula IF

## Características de la cláusula:

- Crear una región paralela es algo **costoso**
- La opción *condition* permite sopesarlo:
  - Si *condition* = TRUE entonces ejecuta en paralelo
  - Si *condition* = FALSE entonces ejecuta en serie
- Sólo se puede imponer una condición

```
!$OMP PARALLEL IF(condition)
```

```
...instructions...
```

```
!$OMP END PARALLEL
```



# Regiones paralelas anidadas

Son **regiones paralelas creadas en regiones paralelas**:

- Crea un nuevo conjunto de tareas
- Esta funcionalidad puede **habilitarse** y **deshabilitarse**
- Lo controla la variable de entorno OMP\_NESTED, que puede ser OMP\_NESTED=TRUE u OMP\_NESTED=FALSE
- El compilador **no está obligado** por el estándar a soportarlo

```
!$OMP PARALLEL
  write(*,*) "Hello"

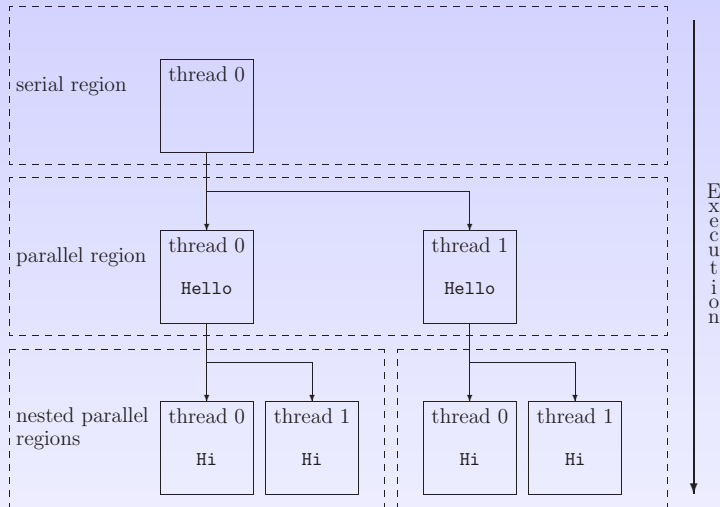
!$OMP PARALLEL
  write(*,*) "Hi"
!$OMP END PARALLEL

!$OMP END PARALLEL
```

# Regiones paralelas anidadas

Programación  
Paralela en  
Fortran95  
usando OpenMP

Miguel Hermanns



Introducción y  
motivación

Regiones paralelas

Conceptos y  
terminología

Cláusulas de datos

Directivas de  
distribución de  
trabajo

Directivas  
combinadas

Recomendaciones

Resumen

# Regiones paralelas anidadas

Programación  
Paralela en  
Fortran95  
usando OpenMP

Miguel Hermanns

Introducción y  
motivación

Regiones paralelas

Conceptos y  
terminología

Cláusulas de datos

Directivas de  
distribución de  
trabajo

Directivas  
combinadas

Recomendaciones

Resumen

**Utilidad** y **finalidad** de las regiones anidadas:

- Permite desarrollar códigos paralelos y acoplarlos
- Permite usar **librerías paralelas** en programas paralelos
- Ejemplos de librerías paralelizadas con OpenMP:
  - Intel Math Kernel Library
  - AMD Core Math Library
  - Pardiso
  - GOTO Blas
  - FFTW
  - LAPACK
  - ...

# Conceptos generales y terminología habitual

- **Directiva:** instrucción de OpenMP que especifica la acción a realizar
- **Cláusula:** instrucción de OpenMP que impone condiciones a la directiva que le precede
- **Tarea:** unidad de ejecución con memoria y stack propios
- **Región serial:** parte de un programa que es ejecutada por una única tarea
- **Región paralela:** parte de un programa que es ejecutada por un equipo de tareas que trabajan conjuntamente

# Conceptos generales y terminología habitual

Programación  
Paralela en  
Fortran95  
usando OpenMP

Miguel Hermanns

Introducción y  
motivación

Regiones paralelas

Conceptos y  
terminología

Cláusulas de datos

Directivas de  
distribución de  
trabajo

Directivas  
combinadas

Recomendaciones

Resumen

- **Tarea maestra:** tarea que crea a las demás tareas al inicio de una región paralela y cuyo número de tarea siempre es el 0
- **Bloque estructurado:** bloque de líneas de código con una única entrada lógica en su parte superior y una única salida lógica en su parte inferior
- **Rutina thread-safe:** una rutina es *thread-safe* cuando es capaz de ser utilizada de manera simultánea e independiente por varias tareas ejecutadas en paralelo
- **Race condition:** situación no deseable en la que el resultado de un código depende del orden en el que se ejecutan las tareas

## Características generales de estas cláusulas:

- Especifican propiedades “*paralelas*” de las variables
- Pueden aparecer tantas veces como sea necesario
- Pueden usarse con cualquier directiva OpenMP

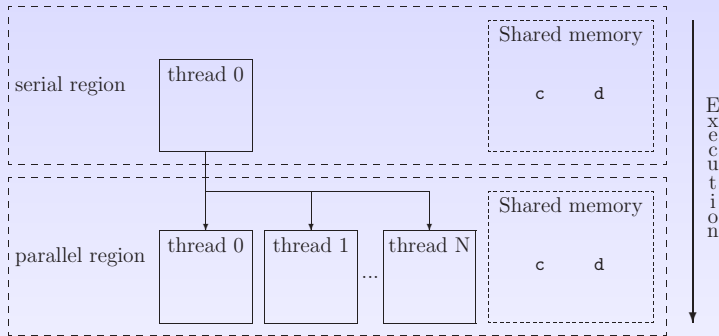
## Restricciones generales de estas cláusulas:

- Una variable no puede aparecer en más de una cláusula
- Sólo admiten variables que sean *visibles* desde la subrutina o función que incluya la cláusula

# La cláusula SHARED

Especifica que una variable es **compartida** por las tareas:

```
!$OMP PARALLEL SHARED(c,d)  
...  
!$OMP END PARALLEL
```



## Características de las variables compartidas:

- Todas las tareas ven su contenido
- Todas las tareas pueden cambiar su contenido
- Todas las tareas ven los cambios realizados

## Peculiaridades de las variables compartidas:

- Los cambios no son inmediatos (*cache coherency*)
- Pueden requerir de sincronía entre tareas
- Existen algunas limitaciones en Fortran (pág. 72)



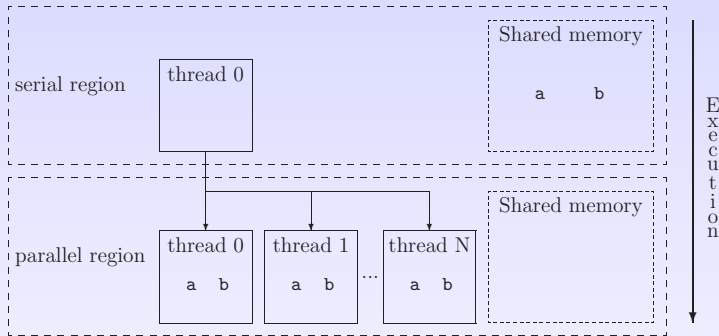
# La cláusula PRIVATE

Especifica que una variable es **privada** a cada tarea:

```
!$OMP PARALLEL PRIVATE(a,b)
```

```
...
```

```
!$OMP END PARALLEL
```



## Características de las variables privadas:

- Cada tarea tiene su *propia versión de la variable*
- Eso se consigue **replicando** la variable en memoria
- Las otras tareas no pueden ver su contenido

## Peculiaridades de las variables privadas:

- Su valor inicial **no está definido**
- Tras la región paralela su valor **tampoco está definido**
- Existen algunas limitaciones (pág. 73-75)

# La cláusula DEFAULT

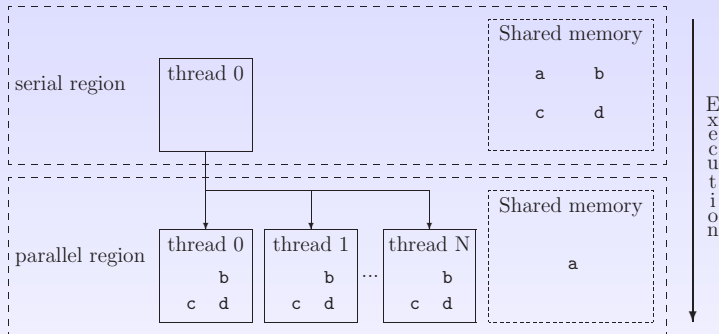
Especifica el **comportamiento por defecto**:

```
integer :: a,b,c,d
```

```
!$OMP PARALLEL DEFAULT(PRIVATE) SHARED(a)
```

```
...
```

```
!$OMP END PARALLEL
```



# La cláusula DEFAULT

## Características de la cláusula:

- Sólo puede aparecer una única vez en cada directiva
- Admite tres opciones: PRIVATE, SHARED y NONE
- Con NONE todas las variables **deben** ser definidas
- Sólo afecta a las variables directamente visibles:

```
!$OMP PARALLEL DEFAULT(SHARED)
```

```
  a = 1
```

```
  b = 2
```

```
  call ejemplo(a,b)
```

```
!$OMP END PARALLEL
```

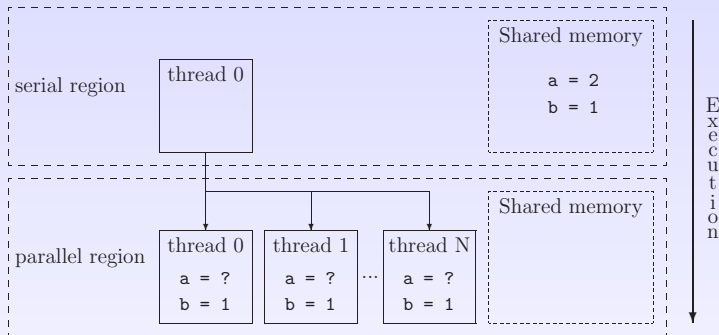
Las variables definidas dentro de la subrutina `ejemplo` **no se ven afectadas** por la cláusula DEFAULT

# La cláusula FIRSTPRIVATE

Es una **extensión** de la cláusula PRIVATE:

```
a = 2  
b = 1
```

```
!$OMP PARALLEL PRIVATE(a) FIRSTPRIVATE(b)  
...  
!$OMP END PARALLEL
```



## Características de las variables FIRSTPRIVATE:

- Cada tarea tiene su *propia versión de la variable*
- Eso se consigue **replicando** la variable en memoria
- Las otras tareas no pueden ver su contenido
- Su valor **es inicializado** con la variable original

## Peculiaridades de las variables FIRSTPRIVATE:

- Tras la región paralela su valor **no está definido**
- Existen algunas limitaciones (pág. 76-77)

# Resumen de cláusulas de datos

## Cláusulas de datos **vistas**:

SHARED, PRIVATE, DEFAULT, FIRSTPRIVATE

## Cláusulas de datos **adicionales**:

LASTPRIVATE, REDUCTION, COPYIN, COPYPRIVATE

## Condiciones **generales** a tener en cuenta:

- Una variable sólo puede aparecer en una cláusula, salvo en FIRSTPRIVATE y LASTPRIVATE a la vez
- El contador de un bucle **siempre** es privado
- Cada cláusula tiene sus limitaciones (pág. 63-79)

# Directivas de distribución de trabajo

Programación  
Paralela en  
Fortran95  
usando OpenMP

Miguel Hermanns

Introducción y  
motivación

Regiones paralelas

Conceptos y  
terminología

Cláusulas de datos

Directivas de  
distribución de  
trabajo

Directivas  
combinadas

Recomendaciones

Resumen

## Características de estas directivas:

- Distribuyen una carga de trabajo sobre varias tareas
- No crean nuevas tareas, usan las ya existentes
- Funcionan en regiones seriales y en regiones paralelas
- Incluyen al final una sincronización implícita

## Restricciones de estas directivas:

- Deben ser encontradas por todas las tareas
- Sólo pueden contener **bloques estructurados** de código



# La directiva !\$OMP DO

## Características de la directiva:

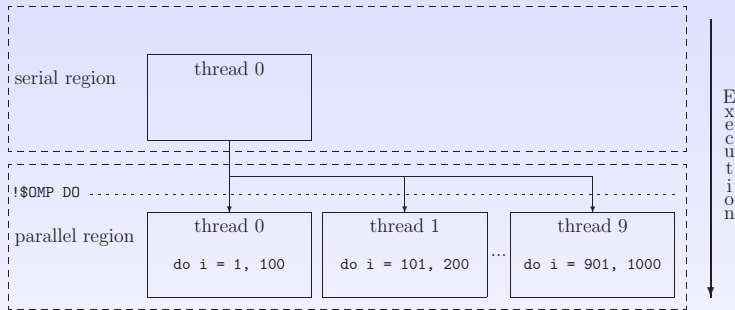
- Distribuye la carga de un bucle entre las tareas
- La descomposición se hace en el espacio de iteraciones
- La descomposición es modificable mediante cláusulas
- El contador es implícitamente declarado PRIVATE

```
!$OMP DO clause1 clause2 ...  
  do i = 1, 1000  
    ...  
  enddo  
!$OMP END DO end_clause
```

# La directiva !\$OMP DO

## Ejemplo de distribución de carga de un bucle:

```
!$OMP PARALLEL NUM_THREADS(10)
!$OMP DO
  do i = 1, 1000
    ...
  enddo
!$OMP END DO
!$OMP END PARALLEL
```



# La directiva !\$OMP DO

## Restricciones de la directiva !\$OMP DO:

- En C, el bucle debe ser de un tipo determinado
- En Fortran, el bucle no puede ser del tipo `do while`
- El contador debe ser el mismo para todas las tareas
- Sólo admite bloques estructurados de código
- Las variables `SHARED` sólo se **actualizan al final**

# La directiva !\$OMP SECTIONS

## Características de la directiva:

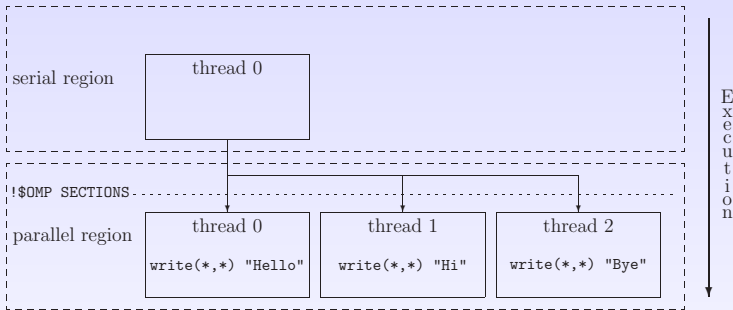
- Cada tarea ejecuta una de las secciones
- Cada sección es ejecutada una única vez
- Permite implementar el paralelismo tipo MIMD

```
!$OMP SECTIONS clause1 clause2 ...  
!$OMP SECTION  
...  
!$OMP SECTION  
...  
...  
!$OMP END SECTIONS end_clause
```

# La directiva !\$OMP SECTIONS

## Ejemplo de asignación de tareas:

```
!$OMP SECTIONS
!$OMP SECTION
    write(*,*) "Hello"
!$OMP SECTION
    write(*,*) "Hi"
!$OMP SECTION
    write(*,*) "Bye"
!$OMP END SECTIONS
```



# La directiva !\$OMP SECTIONS

## Restricciones de la directiva:

- La manera de asignar secciones a tareas no está definida
- Cada sección debe ser un bloque estructurado de código
- No se crean nuevas tareas, se usan las existentes
- Las variables SHARED sólo se **actualizan al final**

# La directiva !\$OMP WORKSHARE

## Características de la directiva:

- Es **específica** para Fortran 95
- Explota las capacidades paralelas de Fortran 95
- Distribuye las instrucciones matriciales entre las tareas
- Soporta las siguientes **instrucciones matriciales**:

matmul, dot\_product, sum, product, maxval, minval, count,  
any, all, spread, pack, unpack, reshape, transpose, eoshift,  
cshift, minloc, maxloc, forall, where

- Las instrucciones **deben comportarse** como si estuvieran en una región serial

```
!$OMP WORKSHARE  
...  
!$OMP END WORKSHARE end_clause
```

# La directiva !\$OMP WORKSHARE

## Ejemplo de distribución de instrucciones matriciales:

```
real :: A(100,100), B(100,100), C(100,100)
```

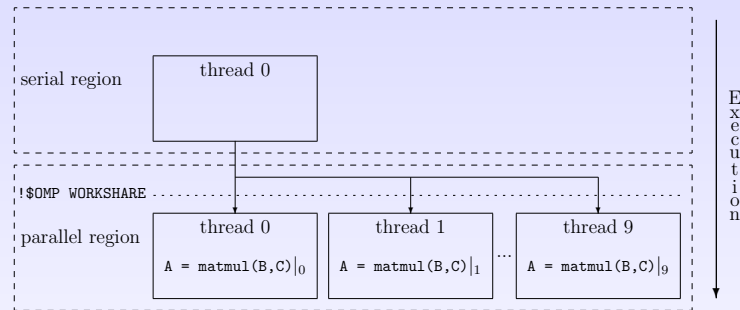
```
!$OMP PARALLEL DEFAULT(SHARED)
```

```
!$OMP WORKSHARE
```

```
  A = matmul(B,C)
```

```
!$OMP END WORKSHARE
```

```
!$OMP END PARALLEL
```





# La directiva !\$OMP WORKSHARE

## Restricciones de la directiva:

- La asignación del trabajo a las tareas no está definida
- El compilador debe insertar sincronizaciones suficientes para que el comportamiento sea el correcto
- Sólo se pueden hacer llamadas a funciones que sean del tipo elemental
- Existen restricciones al uso de variables PRIVATE

## Peculiaridad de la directiva:

- Las variables SHARED se **actualizan constantemente**

## Características de las directivas combinadas:

- Forma abreviada de escribir dos directivas consecutivas
- Equivalentes a una región paralela con una única directiva de distribución de trabajo
- Admiten las cláusulas de ambas directivas

## Restricciones de las directivas combinadas:

- El comportamiento de un programa no está definido si éste depende de la asociación de una cláusula con la región paralela o con la directiva de trabajo

# Directivas combinadas de distribución de trabajo

Programación  
Paralela en  
Fortran95  
usando OpenMP

Miguel Hermanns

Introducción y  
motivación

Regiones paralelas

Conceptos y  
terminología

Cláusulas de datos

Directivas de  
distribución de  
trabajo

Directivas  
combinadas

Recomendaciones

Resumen

## Equivalencia de las directivas combinadas:

!\$OMP PARALLEL DO	!\$OMP PARALLEL
...	!\$OMP DO
!\$OMP END PARALLEL DO	...
	!\$OMP END DO
	!\$OMP END PARALLEL

---

!\$OMP PARALLEL SECTIONS	!\$OMP PARALLEL
...	!\$OMP SECTIONS
!\$OMP END PARALLEL SECTIONS	...
	!\$OMP END SECTIONS
	!\$OMP END PARALLEL

---

!\$OMP PARALLEL WORKSHARE	!\$OMP PARALLEL
...	!\$OMP WORKSHARE
!\$OMP END PARALLEL WORKSHARE	...
	!\$OMP END WORKSHARE
	!\$OMP END PARALLEL

# Recomendaciones personales

- ➊ Revisar el estándar a la hora de programar con OpenMP
- ➋ Buscar sencillez y no optar por soluciones rebuscadas
- ➌ Especificar el número de tareas con `OMP_NUM_THREADS`
- ➍ Usar las cláusulas de datos sólo con `!$OMP PARALLEL`
- ➎ Declarar todas las variables como `SHARED` y sólo las mínimas necesarias como `PRIVATE`
- ➏ No usar directivas combinadas de distribución de trabajo
- ➐ No escribir a disco desde una región paralela

- 1 Introducción y motivación de OpenMP
- 2 Características fundamentales de OpenMP
- 3 Regiones paralelas y regiones seriales
- 4 Conceptos generales y terminología habitual
- 5 Variables compartidas y variables privadas
- 6 Distribución de la carga computacional
- 7 Recomendaciones personales

**Oscar Flores**

Universidad Politécnica de Madrid