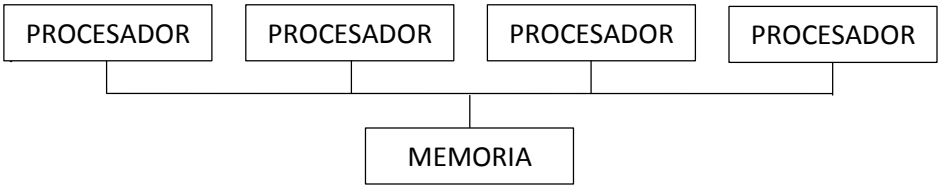

Figura 17.1 El modelo de memoria compartida de la computación en paralelo. Los procesadores se sincronizan y comunican entre sí a través de las variables compartidas.

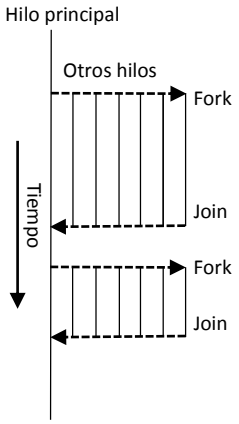

Figura 17.2 El modelo de memoria compartida se caracteriza por un paralelismo fork / join, en la que el paralelismo va y viene. Al comienzo de la ejecución sólo un único hilo, el hilo llamado maestro, está activo. El hilo principal ejecuta las porciones de serie del programa. Se fork hilos adicionales para ayudar a ejecutar partes paralelas del programa. Estos hilos se desactivan cuando se reanuda la ejecución en serie.

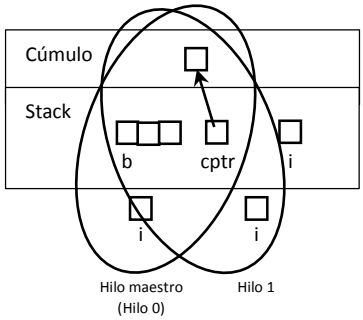

Figura 17.4 Durante la ejecución en paralelo del bucle, el índice i es una variable privada, mientras que b, cptr y el cúmulo de datos son compartidos.

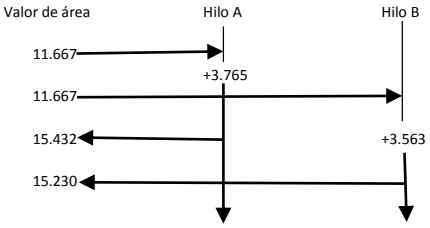

Figura 17.5 Ejemplo de una condición de corrida. Cada hilo añade un valor al área. Sin embargo el hilo B recupera el valor original del área antes que el hilo A pueda escribir un nuevo valor. Por lo tanto el valor final del área es incorrecto. Si el hilo B leyera el valor del área después de que el hilo A se actualizara, entonces el valor final del área sería correcto. En definitiva, la ausencia de una sección crítica puede provocar la ejecución determinista.

Tabla 17.1 Operadores de reducción de OpenMP para C y C++

Operador	Significado	Tipos permitidos	Valor inicial
+	Suma	float, int8	0
*	Multiplicación	float, int	1
&	Bitwise AND	int	all bits 1
|	Bitwise OR	int	0
^	Bitwise OR exclusiva	int	0
&&	AND Lógica	int	1
||	OR Lógica	int	0

Tabla 17.2 Tiempos de ejecución de dos programas en un Servidor Sun Enterprise 4000 que calculan π usando la regla de rectángulo.

	Tiempo de ejecución del programa (seg)	
Hilos	**Utilización de pragma crítico**	**Utilización de cláusula de reducción**
1	0.0780	0.0273
2	0.1510	0.0146
3	0.3400	0.0105
4	0.3608	0.0086
5	0.4710	0.0076

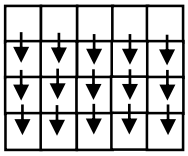

Figura 17.6 Diagrama de dependencia de datos para un determinado par de bucles anidados donde se muestra que mientras las columnas se pueden actualizar de forma simultánea, las filas no.

Tabla 17.3 Tiempo de ejecución en un servidor Sun Enterprise Server 4000 de un programa C en paralelo que calcula π usando la regla de rectángulo, como una función del número de rectángulos y el número de hilos.

	Tiempo de ejecución (mseg)	
Hilos	**n=100**	**n=100,000**
1	0.964	27.288
2	1.436	14.598
3	1.732	10.506
4	1.990	8.648

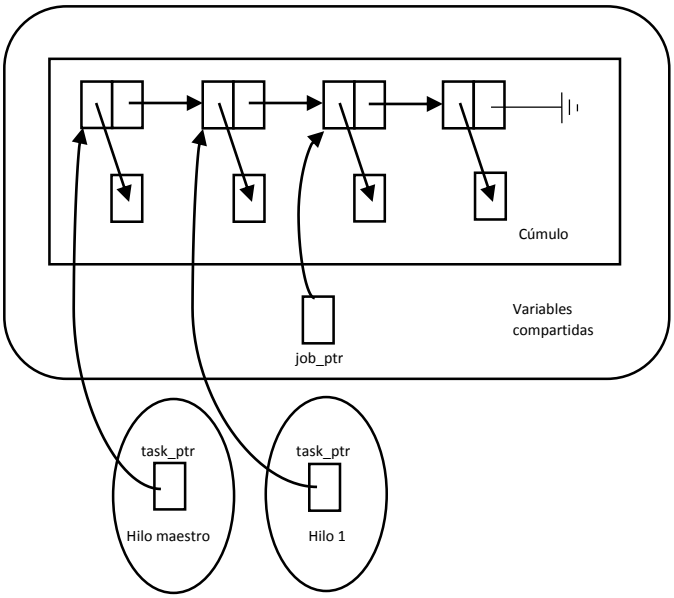

Figura 17.7 Two threads work their way through a singly linked "to do" list. Variable Job_ptr must be shared, while task_ptr must be a private variable.

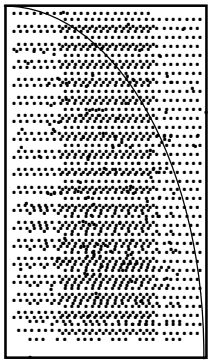

Figura 17.8 Ejemplo del algoritmo Monte Carlo para calcular π. En este ejemplo hemos generado 1000 pares de una distribución uniforme entre 0 y 1. Hay desde 773 pares dentro del circulo, nuestra estimación de π es 4(773/1000), o 3.092

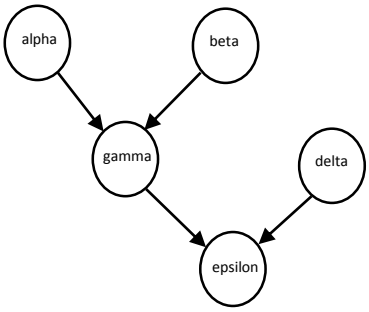

Figura 17.10 Diagrama de dependencia de datos para el segmento de código de la sección 17.9.