Soporte del Lenguaje para Mejorar el Rendimiento - FORTRAN 90*

José Enrique Alvarez Estrada

Translated By:

José Enrique Alvarez Estrada

Based on *Language Support for Performance - FORTRAN 90*† by

Charles Severance

Kevin Dowd

This work is produced by OpenStax-CNX and licensed under the
Creative Commons Attribution License 3.0‡

La versión previa del estandar de FORTRAN liberada por ANSI (el American National Standard Institute), FORTRAN 77 (X3.9-1978) se escribió para promover la portabilidad de los programas FORTRAN entre plataformas distintas. No se inventaron nuevos componentes del lenguaje, más bien se incorporaron buenas características que ya estaban disponibles en compiladores en ambientes de producción. Al contrario que FORTRAN 77, el FORTRAN 90 (ANSI X3.198-1992) añade nuevas extensiones y características al lenguaje. Algunas de ellas sólo ponen al día a FORTRAN respecto a lenguajes como C (asignación dinámica de memoria, reglas de alcance) y C++ (interfaces de funciones con genericidad). Pero algunas de estas nuevas características son únicas de FORTRAN (operaciones sobre arreglos). Es interesante notar que mientras se desarrollaba la especificación de FORTRAN 90, las arquitecturas de cómputo de alto rendimiento dominantes eran los sistemas SIMD escalables como la Connection Machine, así como los sistemas de procesadores vectoriales paralelos de compañías como Cray Research.

FORTRAN 90 hace un trabajo sorprendentemente bueno en cubrir las necesidades de estas arquitecturas tan diferentes. Sus características también se adaptan razonablemente bien a los nuevos multiprocesadores con memoria compartida uniforme. Sin embargo, como veremos después, FORTRAN 90 por sí solo todavía no es suficiente para cumplir las necesidades de los sistemas distribuidos escalables con acceso a memoria no uniforme, que se están volviendo dominantes en la computación de extremo superior.

Las extensiones de FORTRAN 90 a FORTRAN 77 incluyen:

- Constructos para arreglos
- Asignación dinámica de memoria y variables automáticas
- Apuntadores

*Version 1.1: May 14, 2011 11:25 am -0500

†http://cnx.org/content/m33757/1.3/

‡http://creativecommons.org/licenses/by/3.0/

- Nuevos tipos de datos, estructuras
- Nuevas funciones intrínsecas, incluyendo muchas que operan sobre vectores y matrices
- Nuevas estructuras de control, tales como la sentencia WHERE
- Interfaces de procedimientos mejoradas

1 Constructos para Arreglos en FORTRAN 90

Con los constructos para arreglos de FORTRAN 90, puede usted especificar arreglos completos o secciones de los mismos como participantes en operaciones unarias y binarias. Tales constructos son una característica clave para "deserializar" las aplicaciones, de forma que puedan adaptarse mejor a computadoras vectoriales o procesadores paralelos. Por ejemplo, digamos que desea usted sumar dos vectores A y B. En FORTRAN 90, puede expresarlo como una simple operación de suma, en vez de usando el bucle tradicional. Esto es, puede usted escribir:

```
A = A + B
```

en vez del tradicional bucle en FORTRAN 77:

```
DO I=1,N
  A(I) = A(I) + B(I)
ENDDO
```

Puede que el código generado por el compilador en su estación de trabajo no se vea muy diferente, pero en algunas máquinas de arquitectura paralela o en algunas estaciones de trabajo a la vuelta de la esquina, la diferencia es significativa. La versión FORTRAN 90 declara explícitamente que los cálculos pueden realizarse en cualquier orden, incluyendo todos en paralelo simultáneamente.

Un efecto importante de ello es que si la versión de FORTRAN 90 experimenta una falla de punto flotante sumando el elemento 17, y revisa usted la memoria en un depurador, encontrará que el elemento 27 ya fue calculado y tiene un valor perfectamente válido.

Y no está usted limitado tan sólo a arreglos unidimensionales. Por ejemplo, la suma a nivel de elementos de dos arreglos bidimensionales puede realizarse así:[1]

```
A = A + B
```

en vez de:

[1] Sólo en caso de que esté usted sorprendido, A*B regresa el producto de los miembros del arreglo a nivel de elementos, no el producto de dos matrices. Tal operación la cubre FORTRAN 90 mediante una función intrínseca.

```
DO J=1,M
  DO I=1,N
    A(I,J) = A(I,J) + B(I,J)
  END DO
END DO
```

Naturalmente, si desea combinar dos arreglos en una operación, sus tamaños tienen que ser compatibles. Sumar un vector de siete elementos a uno de ocho no tiene sentido. Tampoco lo tiene multiplicar un arreglo de 2*4 a uno de 3*4.. Cuando los dos arreglos tienen tamaños compatibles, relativos a la operación a que quiere usted someterlos, decimos que tienen *concordancia de tamaños*, como en el siguiente código:

```
DOUBLE PRECISION A(8), B(8)
 ...
A = A + B
```

Siempre se considera que los escalares tienen concordancia de tamaños con los arreglos (y con otros escalares). En una operación binaria con un arreglo, un escalar se trata como un arreglo del mismo tamaño, con el único elemento duplicado en el resto de las posiciones.

Aun así, existen limitaciones. Cuando haga referencia a un arreglo particular, por ejemplo A, está haciendo referencia a todo él, del primer al último elemento. Seguro puede usted imaginar casos donde le interese especificar un subconjunto de un arreglo. Puede tratarse de un grupo de elementos consecutivos o algo como "cada octavo elemento" (i.e. un recorrido de salto no unitario a través del arreglo). A tales partes de los arreglos, posiblemente no contiguas, se les denomina *secciones del arreglo*.

Para especificar secciones de arreglos en FORTRAN 90, se reemplazan los índices tradicionales con tripletas de la forma `a:b:c`, que significan "los elementos desde `a` hasta `b`, tomados con un incremento de `c`." Pueden omitirse partes de la tripleta, siempre y cuando el significado permanezca claro. Por ejemplo, `a:b` significa "los elementos desde a hasta `b`;" `a:` significa "los elementos desde `a` hasta el extremo superior del arreglo con un incremento de 1." Recuerdo que una tripleta reemplaza a un único índice, así que un arreglo n-dimensional puede tener n tripletas.

Puede usted usar tripletas en expresiones, nuevamente asegurándose de que las partes de las expresiones estén en concordancia. Considere estos ejemplos:

```
REAL X(10,10), Y(100)
    ...
X(10,1:10)   = Y(91:100)
X(10,:)      = Y(91:100)
```

La primera sentencia asigna los últimos 10 elementos de `Y` al décimo renglón de `X`. La segunda sentencia expresa lo mismo de modo ligeramente diferente. El " : " solitario le indica al compilador que está implícito el rango completo (del 1 al 10).

2 Intrínsecos en FORTRAN 90

FORTRAN 90 extiende la funcionalidad de los intrínsecos de FORTRAN 77, y además agrega muchos nuevos, incluyendo algunas subrutinas intrínsecas. La mayoría pueden ser *valoradas mediante arreglos*:

pueden regresar secciones de arreglos o escalares, dependiendo de cómo fueron invocadas. Por ejemplo, he aquí un nuevo uso del intrínseco SIN valorado mediante arreglos:

```
REAL A(100,10,2)
 ...
A = SIN(A)
```

Cada elemento del arreglo A es reemplazado por su seno. Los intrínsecos de FORTRAN 90 también trabajan con secciones de arreglos, con tal de que la variable que reciba los resultados esté en concordancia de tamaño con la que se pasó:

```
REAL A(100,10,2)
REAL B(10,10,100)
 ...
B(:,:,1) = COS(A(1:100:10,:,1))
```

También se han extendido otros intrínsecos tales como SQRT, LOG, etc. Entre los nuevos intrínsecos se encuentran:

Reducciones: FORTRAN 90 tiene reducciones de vectores tales como MAXVAL, MINVAL y SUM. Para arreglos de orden superior (cualquier cosa mayor que un vector) tales funciones pueden realizar una reducción a lo largo de una dimensión en particular. Adicionalmente, existe una función DOT_PRODUCT (producto punto) para los vectores.

Manipulación de matrices: Los intrínsecos MATMUL y TRANSPOSE pueden manipular matrices completas.

Construir o redimensionar arreglos: RESHAPE le permite crear un nuevo arreglo a partir de los elementos de uno antiguo con diferente tamaño. SPREAD replica un arreglo a lo largo de una nueva dimensión. MERGE copia porciones de un arreglo en otro, bajo el control de una máscara. CSHIFT permite desplazar un arreglo en una o más dimensiones.

Funciones de consulta: SHAPE, SIZE, LBOUND, y UBOUND le permiten averiguar cómo está construido un arreglo.

Pruebas en paralelo: Dos intrínsecos nuevos de reducción, ANY y ALL, sirven para probar en paralelo varios elementos de un arreglo.

3 Nuevas Características de Control

FORTRAN 90 incluye algunas nuevas características de control, entre ellas una *primitiva de asignación* condicional llamada WHERE, que pone las asignaciones de un arreglo con concordancia de tamaños bajo el control de una máscara, tal como en el ejemplo siguiente. He aquí un ejemplo de la primitiva WHERE:

```
REAL A(2,2), B(2,2), C(2,2)
DATA B/1,2,3,4/, C/1,1,5,5/
```

```
 ...
WHERE (B .EQ. C)
  A = 1.0
  C = B + 1.0
ELSEWHERE
  A = -1.0
ENDWHERE
```

En aquellos casos donde la expresión lógica es `TRUE`, `A` obtiene `1.0` y `C` obtiene `B+1.0`. En la claúsula `ELSEWHERE`, `A` obtiene `-1.0`. El resultado de la operación anterior serán los arreglos `A` y `C` con los elementos:

```
A =  1.0  -1.0          C =  2.0    5.0
    -1.0  -1.0               1.0    5.0
```

Nuevamente, no hay ningún orden implícito en estas asignaciones condicionales, lo cual significa que pueden realizarse en paralelo. La ausencia de orden implícito es crítica para permitir a los sistemas de cómputo SIMD y a los ambientes SPMD tener flexibilidad en la ejecución de tales cálculos.

4 Arreglos Automáticos y Asignables

Cada programa requiere variables temporales o espacio de trabajo. En el pasado, los programadores de FORTRAN a menudo administraban su propio espacio libre, declarando un arreglo lo suficientemente grande para manejar cualquier requerimiento temporal. Esta práctica se traga la memoria (aunque la memoria virtual, por lo general), e incluso puede tener efectos indeseados en el rendimiento. Con la habilidad de asignar la memoria dinámicamente, los programadores pueden esperar más tiempo para decidir cuánto espacio vacío deben obtener. FORTRAN 90 soporta la asignación dinámica de memoria mediante dos nuevas características del lenguaje: arreglos automáticos y arreglos asignables.

Tal como sucede con las variables locales de un programa en C, los arreglos automáticos de FORTRAN 90 sólo tienen asignado espacio de almacenamiento durante la vida de la subrutina o función que los contiene. Esto es diferente del almacenamiento local de arreglos tradicional de FORTRAN, donde algo de espacio se apartaba a tiempo de compilación o enlace. El tamaño y forma de los arreglos automáticos puede esculpirse mediante ua combinación de constantes y argumentos. Por ejemplo, he aquí una declaración de un arreglo automático, `B`, usando la nueva especificación de sintaxis de FORTRAN 90:

```
SUBROUTINE RELAX(N,A)
INTEGER N
REAL, DIMENSION (N) :: A, B
```

Se declararon dos arreglos: `A`, el argumento de prueba, y `B`, un arreglo automático de tamaño explícito. Cuando se regresa de la subrutina, `B` deja de existir. Observe que el tamaño de `B` se toma de uno de los argumentos, `N`.

Los arreglos asignables le dan a usted la flexibilidad de elegir el tamaño de un arreglo después de examinar otras variables en el programa. Por ejemplo, puede que desee determinar la cantidad de datos de entrada antes de asignar los arreglos. Este pequeño programa pregunta al usuario el tamaño de la matriz antes de asignarle espacio de almacenamiento:

```
INTEGER M,N
REAL, ALLOCATABLE, DIMENSION (:,:) :: X
 ...
WRITE (*,*) 'INTRODUZCA LAS DIMENSIONES DE X'
READ (*,*) M,N
ALLOCATE (X(M,N))
 ...
 hacer algo con X
 ...
DEALLOCATE (X)
 ...
```

La sentencia `ALLOCATE` crea un arreglo de `M` $\times$ `N` que posteriormente se libera mediante la sentencia `DEALLOCATE`. Tal como sucede con los programas en `C`, es importante devolver la memoria asignada cuando se ha terminado de usar; de otra forma, su programa pudiera consumir toda la memoria virtual de almacenamiento disponible.

5 Flujo Calórico en FORTRAN 90

El problema de flujo calórico es un programa ideal para demostrar cuán agradablemente puede expresar FORTRAN 90 los problemas que emplean arreglos regulares:

```
      PROGRAM HEATROD
      PARAMETER(MAXTIME=200)
      INTEGER TICKS,I,MAXTIME
      REAL*4 ROD(10)
      ROD(1) = 100.0
      DO I=2,9
        ROD(I) = 0.0
      ENDDO
      ROD(10) = 0.0
      DO TICKS=1,MAXTIME
        IF ( MOD(TICKS,20) .EQ. 1 ) PRINT 100,TICKS,(ROD(I),I=1,10)
        ROD(2:9) = (ROD(1:8) + ROD(3:10) ) / 2
      ENDDO
100   FORMAT(I4,10F7.2)
      END
```

El programa es idéntico, excepto por el hecho de que el bucle interno ha sido reemplazado por una sola sentencia que calcula la "nueva" sección, al promediar una tira de los elementos "izquierdos" y una tira de los elementos "derechos".

La salida de este programa luce así:

```
E6000: f90 heat90.f
E6000:a.out
   1 100.00   0.00   0.00   0.00   0.00   0.00   0.00   0.00   0.00   0.00
  21 100.00  82.38  66.34  50.30  38.18  26.06  18.20  10.35   5.18   0.00
  41 100.00  87.04  74.52  61.99  50.56  39.13  28.94  18.75   9.38   0.00
  61 100.00  88.36  76.84  65.32  54.12  42.91  32.07  21.22  10.61   0.00
  81 100.00  88.74  77.51  66.28  55.14  44.00  32.97  21.93  10.97   0.00
 101 100.00  88.84  77.70  66.55  55.44  44.32  33.23  22.14  11.07   0.00
 121 100.00  88.88  77.76  66.63  55.52  44.41  33.30  22.20  11.10   0.00
 141 100.00  88.89  77.77  66.66  55.55  44.43  33.32  22.22  11.11   0.00
 161 100.00  88.89  77.78  66.66  55.55  44.44  33.33  22.22  11.11   0.00
 181 100.00  88.89  77.78  66.67  55.55  44.44  33.33  22.22  11.11   0.00
E6000:
```

Si lo observa detenidamente, verá que la salida es la misma que la de la implementación rojo-negro. Ello se debe a que en FORTRAN 90:

```
ROD(2:9) = (ROD(1:8) + ROD(3:10) ) / 2
```

es una *única* sentencia de asignación. Como se muestra en Figure 1 (Alineación de datos y cálculos), la parte derecha se evalúa completamente antes de que la sección del arreglo resultante se le asigne a `ROD(2:9)`. A primera vista puede parecer poco natural, pero considere la siguiente sentencia:

```
I = I + 1
```

Sabemos que si `I` comienza en 5, esta sentencia incrementará su valor a 6. Ello sucede porque el lado derecho (5+1) se evalúa antes de ejecutar la asignación del 6 a `I`. En FORTRAN 90, una variable puede ser un arreglo completo. Así que ésta *is* una operación rojo-negro. ¡Hay un `ROD` "antiguo" en la parte derecha, y un `ROD` "nuevo" en la parte izquierda!

Para realmente "pensar" al estilo FORTRAN 90, es bueno suponer que está usted usando un sistema SIMD con millones de pequeñas CPUs. Primero alineamos cuidadosamente los datos, deslizándolos alrededor, y entonces... ¡zas! en una sola instrucción, sumamos todos los valores alineados en un solo instante. Figure 1 (Alineación de datos y cálculos) muestra gráficamente este acto de "alinear" los valores y luego sumarlos. El grafo de flujo de datos es extremadamente simple. Los dos renglones superiores son de sólo lectura, y los datos fluyen de arriba hacia abajo. Usar el espacio temporal elimina la dependencia aparente. Este enfoque de "pensar SIMD" es una de dos formas posibles de obligarnos a enfocar nuestro pensamiento en los datos, en vez de en el control. Puede que la SIMD no sea una buena arquitectura para nuestro problema, pero si es capaz de expresarlo de forma tal que pueda trabajarse en SIMD, un buen ambiente SPMD puede tomar ventaja del paralelismo a nivel de datos que usted ha identificado.

El ejemplo siguiente pone de manifiesto uno de los retos que aparecen al producir una implementación eficiente de FORTRAN 90. Si estos arreglos contienen 10 millones de elementos, y el compilador usa un enfoque simple, necesitará 30 millones de elementos para los valores "izquierdos" viejos, los valores "derechos" viejos y para los nuevos. Se requiere optimizar el flujo de datos sólo para determinar para determinar cuántos

datos extras deben mantenerse para obtener los resultados apropiados. Si el compilador es inteligente, la memoria extra puede ser muy poca:

Alineación de datos y cálculos

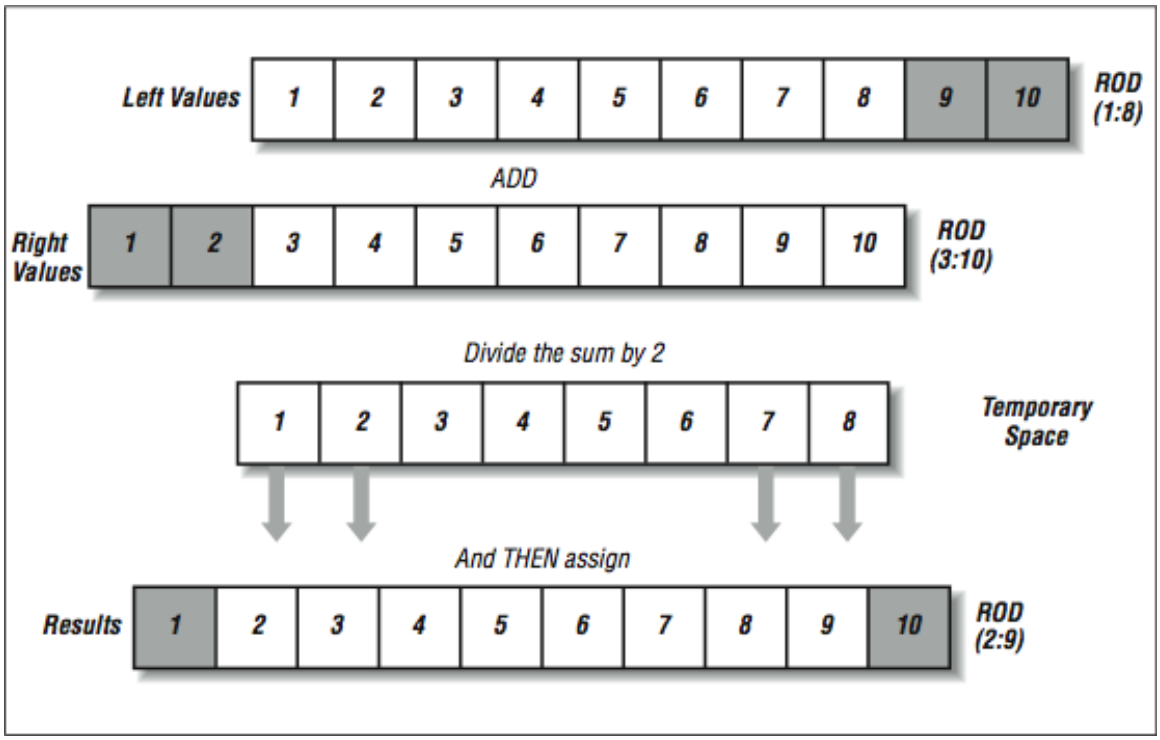

Figure 1

```
SAVE1 = ROD(1)
DO I=2,9
  SAVE2 = ROD(I)
  ROD(I) = (SAVE1 + ROD(I+1) ) / 2
  SAVE1 = SAVE2
ENDDO
```

Si bien esta implementación no tiene el paralelismo de una implementación rojo-negro completa, produce los resultados correctos con sólo dos elementos de datos extras. El truco consiste en guardar el valor "izquierdo" viejo justo antes de borrarlo. Un buen compilador de FORTRAN 90 usa análisis de flujo de datos, en busca de una plantilla sobre cómo se mueven los cálculos a lo largo de los datos para ver si puede guardar unos

pocos elementos durante un corto periodo de tiempo, para aliviar la necesidad de una copia extra completa de los datos.

La ventaja del lenguaje FORTRAN 90 es que depende del compilador si usa una copia completa del arreglo, o unos pocos elementos de datos para asegurar que el programa se ejecute apropiadamente. Y lo que es más importante, puede cambiar su enfoque conforme cambia de una arquitectura a otra.

6 FORTRAN 90 Versus FORTRAN 77

Es interesante señalar que FORTRAN 90 nunca fue completamente adoptado por la comunidad de cómputo de alto rendimiento. Existen algunas razones para ello:

- Existía la preocupación de que el uso de apuntadores y estructuras de datos dinámicas arruinase el rendimiento, y se perdieran las ventajas de optimización de FORTRAN sobre C. Algunos dijeron que FORTRAN 90 estaba tratando de ser un mejor C que C. Otros decían, "¿quién quiere parecerse más a un lenguaje más lento?" Sea cual fuere la razón, hubo algo de controversia cuando se implementó FORTRAN 90, que derivó en algo de reluctancia de los programadores para adoptarlo. Algunos vendedores decían "Puede usted usar FORTRAN 90, pero FORTRAN 77 siempre será más rápido."
- Y como los vendedores a menudo implementaron diferentes subconjuntos de FORTRAN 90, el código no fue tan transportable como el de FORTRAN 77. Por tal motivo, los usuarios que requerían máxima transportabilidad continuaron con FORTRAN 77.
- A veces los vendedores compraban sus compiladores que cumplían completamente con FORTRAN 90 a terceros, quienes cobraban altas tarifas de licenciamiento. Así, podía usted obtener un FORTRAN 77 gratuito (y más rápido, de acuerdo con el vendedor) o pagar por el compilador de FORTRAN 90 más lento (guiño, guiño).
- Por tales factores, el número de aplicaciones serias desarrolladas en FORTRAN 90 fue pequeño. Así que los bancos de pruebas usados para comprar nuevos sistemas estaban desarrollados casi exclusivamente en FORTRAN 77. Ello motivó a los vendedores a mejorar sus compiladores de FORTRAN 77 en vez de sus compiladores de FORTRAN 90.
- Conforme los compiladores de FORTRAN 77 se hacían más sofisticados, usando análisis de flujo de datos, fue relativamente fácil escribir código "paralelo" transportable en FORTRAN 77, usando las técnicas que hemos discutido en este libro.
- Uno de los mayores beneficios potenciales de FORTRAN 90 era la transportabilidad entre SIMD y las supercomputadoras paralelas/vectoriales. Conforme ambas arquitecturas fueron reemplazadas con multiprocesadores con memoria uniforme compartida, FORTRAN 77 se convirtió en el lenguaje que ofrecía la máxima transportabilidad entre las computadoras usadas típicamente por los programadores de cómputo de alto rendimiento.
- Los compiladores de FORTRAN 77 soportaban directivas que permitían a los programadores afinar finamente el rendimiento de sus aplicaciones, al tomar completo control del paralelismo. Ciertos dialectos de FORTRAN 77 esencialmente se convirtieron en un "lenguaje ensamblador" para programación paralela. Incluso las versiones altamente afinadas de tales códigos eran relativamente transportables entre multiprocesadores de memoria compartida uniforme de diferentes vendedores.

Así que los eventos conspiraron contra FORTRAN 90 en el corto plazo. Sin embargo, FORTRAN 77 no está bien adaptado a los sistemas de memoria distribuida porque no se presta bien para las directivas de disposición de datos. Conforme requerimos particionar y distribuir los datos cuidadosamente sobre esos nuevos sistemas, debemos dar al compilador *gran cantidad* de flexibilidad. FORTRAN 90 es el lenguaje mejor adaptado para tal propósito.

7 Resumen de FORTRAN 90

Bueno, este es una visita apresurada a FORTRAN 90. Probablemente no le hayamos hecho justicia al lenguaje cubriéndolo tan brevemente, pero queremos que usted se lleve una probada del mismo. Hay muchas

características que no hemos discutido. Si quiere aprender más, le recomendamos *FORTRAN 90 Explained*, de Michael Metcalf y John Reid (Oxford University Press).

FORTRAN 90 no es suficiente por sí mismo para que obtengamos rendimiento escalable en sistema de memoria distribuida. Es más, ningún compilador es todavía capaz de realizar suficiente análisis de datos para decidir dónde almacenar los datos y cuándo recuperarlos de la memoria. Así, por ahora, nosotros como programadores debemos preocuparnos por la disposición de los datos. Debemos descomponer el problema en fragmentos paralelos que puedan procesarse individualmente. Tenemos varias opciones. Podemos usar High Performance FORTRAN y delegar algunos detalles al compilador, o podemos usar paso de mensajes explícito y ocuparnos nosotros mismos de *todos* los detalles del paralelismo.