Soporte del Lenguaje para Mejorar el Rendimiento - FORTRAN de Alto Rendimiento (HPF)*

José Enrique Alvarez Estrada

Translated By:

José Enrique Alvarez Estrada

Based on *Language Support for Performance - High Performance FORTRAN (HPF)*† by

Charles Severance

Kevin Dowd

En marzo de 1992, el Foro de Fortran de Alto Rendimiento (HPFF por sus siglas en inglés) comenzó a reunirse para discutir y definir un conjunto de agregados a FORTRAN 90 para hacer más práctico su uso en ambientes de cómputo escalable. El plan era desarrollar una especificación durante ese año, de forma que los vendedores rápidamente comenzaran a implementar el estándar. El alcance de este esfuerzo incluía lo siguiente:

- Identificar escalares y arreglos que puedan distribuirse a lo largo de la máquina paralela.
- Indicar cómo se distribuirán. ¿Serán tiras, bloques o de alguna otra forma?
- Especificar cómo se alinearán dichas variables las unas con respecto a las otras.
- Redistribuir y realinear estructuras de datos a tiempo de ejecución.
- Añadir una estructura de control `FORALL` para las asignaciones paralelas que sean difíciles o imposibles de construirse usando la sintaxis de arreglos de FORTRAN.
- Hacerle mejoras a la estructura de control `WHERE` de FORTRAN 90.
- Añadir funciones intrínsecas para las operaciones en paralelo más comunes.

Existían varias fuentes de inspiración para el esfuerzo HPF. Las directivas de disposición ya eran parte del ambiente de programación de FORTRAN 90 en algunas computadoras SIMD (i.e., la CM-2). También se había liberado el año anterior PVM, el primer ambiente de paso de mensajes transportable, y los usuarios

*Version 1.1: May 14, 2011 2:48 pm -0500
†http://cnx.org/content/m33765/1.3/
‡http://creativecommons.org/licenses/by/3.0/

tenían un año de experiencia tratando de descomponer programas a mano. Habían desarrollado algunas técnicas básicas utilizables para la descomposición de datos que funcionaban muy bien, pero requerían de mucha más contabilidad.[1]

El esfuerzo HPF agrupó un conjunto diverso de intereses de todos los principales vendedores de cómputo de alto rendimiento. Estaban representadas las principales compañías. Como resultado, HPF se diseñó para implementarse en casi todo tipo de arquitecturas.

Hay un esfuerzo en marcha para producir el siguiente estándar de FORTRAN, FORTRAN 95, que se espera adopte algunas pero no todas las modificaciones de HPF.

1 Programando en HPF

HPF usa FORTRAN 90 como su núcleo. Si un programa en FORTRAN 90 se pasa por un compilador HPF, debe producir los mismos resultados que si se hubiera compilado con FORTRAN 90. Asumiendo que un programa HPF sólo use los constructos de FORTRAN 90 y las directivas HPF, un compilador FORTRAN 90 puede ignorar las directivas y debe producir los mismos resultados que un compilador HPF.

Conforme el usuario agrega directivas al programa, la semántica de éste no cambia. Si el usuario no entiende en absoluto la aplicación e inserta directivas extremadamente mal pensadas, el programa produce resultados correctos aunque muy lentamente. Un compilador HPF no trata de "mejorar" las directivas del usuario. Asume que el programador es omniscente.[2]

Una vez que el usuario ha determinado cómo distribuir los datos entre los procesadores, el compilador de HPF trata de usar el mínimo de comunicaciones necesario y traslapa la comunicación con los cálculos siempre que sea posible. HPF generalmente usa la regla de "el propietario calcula" para la ubicación de los cálculos. Un elemento particular en un arreglo se calcula mediante el procesador que almacena dicho elemento del arreglo.

Si es necesario, se recogen de procesadores remotos todos los datos necesarios para realizar el cálculo. Si el programador hace una descomposición y alineación inteligentes, muchos de los datos requeridos estarán en la memoria local, en vez de en una memoria remota. El compilador HPF también es responsable de asignar cualquier estructura de datos temporal necesaria para soportar las comunicaciones a tiempo de ejecución.

En general, el compilador HPF no es mágico -simplemente hace un muy buen trabajo con los detalles de comunicación cuando el programador es capaz de diseñar una buena descomposición de datos. Al mismo tiempo, retiene la transportabilidad con máquinas de una sola CPU y sistemas de memoria uniforme compartida usando FORTRAN 90.

2 Directivas HPF de Disposición de Datos

Tal vez las contribuciones más importantes de HPF sean las directivas de disposición de datos. Usándolas, el programador puede controlar cómo se acomodan los datos, basándose en su conocimiento acerca de las interacciones entre datos. Una directiva de ejemplo es la que sigue:

```
      REAL*4 ROD(10)
!HPF$ DISTRIBUTE ROD(BLOCK)
```

El prefijo `!HPF$` es visto como un comentario por cualquier compilador que no sea HPF, de forma que un compilador FORTRAN 90 común pueda ignorarla de forma segura. La directiva `DISTRIBUTE` indica que el arreglo `ROD` debe distribuirse entre múltiples procesadores. Si no se usase dicha directiva, el arreglo `ROD` se

[1] Como pronto veremos.

[2] Lo cual siempre es bueno de asumirse.

ubicaría en un procesador y se le comunicaría a los otros procesos conforme se necesitara. Existen varios esquemas de distribución que pueden hacerse en cada dimensión:

```
      REAL*4 BOB(100,100,100),RICH(100,100,100)
!HPF$ DISTRIBUTE BOB(BLOCK,CYCLIC,*)
!HPF$ DISTRIBUTE RICH(CYCLIC(10))
```

Estas distribuciones operan como sigue:

`BLOCK` El arreglo se distribuye a lo largo de los procesadores usando bloques contiguos del valor del índice. Los bloques se hacen tan grandes como sea posible.

`CYCLIC` El arreglo se distribuye a lo largo de los procesadores, mapeando cada elemento sucesivo al "siguiente" procesador, y cuando se llega al último procesador, la ubicación comienza nuevamente en el primero.

`CYCLIC(n)` El arreglo se distribuye del mismo modo que en `CYCLIC` excepto que se colocan `n` elementos sucesivos en cada procesador antes de pasarse al siguiente.

NOTE: Todos los elementos en esta dimensión se colocan en el mismo procesador. Esto es mayormente útil para arreglos multidimensionales.

Distribuyendo los elementos del arreglo entre procesadores

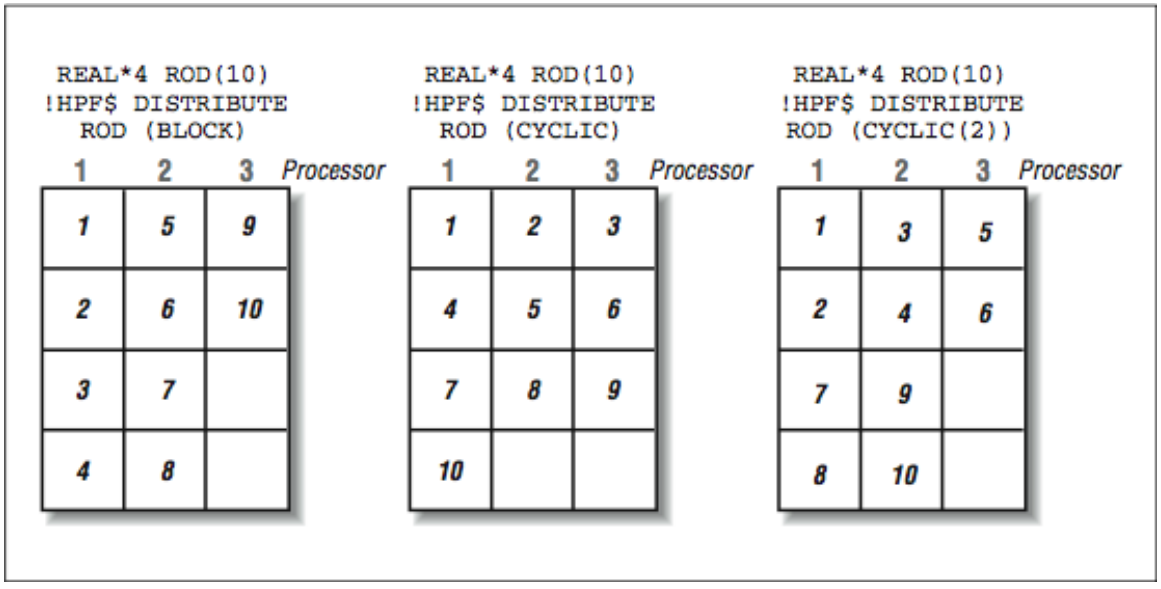

Figure 1

La Figure 1 (Distribuyendo los elementos del arreglo entre procesadores) muestra cómo se mapean los elementos de un arreglo sencillo entre tres procesadores con diferentes directivas.

Deben ubicarse cuatro elementos en los Procesadores 1 y 2 porque no hay un Procesador 4 disponible para el elemento más a la izquierda si se ubican tres elementos en los Procesadores 1 y 2. En Figure 1 (Distribuyendo los elementos del arreglo entre procesadores), los elementos se ubican en procesadores sucesivos, regresando al Procesador 1 tras el último procesador. En Figure 1 (Distribuyendo los elementos del arreglo entre procesadores), usar un tamaño de bocado de `CYCLIC` es un compromiso entre `BLOCK` puro y `CYCLIC` puro.

Para explorar el uso de `*`, debemos observar un simple arreglo bidimensional mapeado entre cuatro procesadores. En Figure 2 (Distribuciones bidimensionales), mostramos la distribución del arreglo y cada celda indica cuál procesador almacenará el dato para dicha celda en un arreglo bidimensional. En Figure 2 (Distribuciones bidimensionales), la directiva lo descompone en ambas dimensiones simultáneamente. Este enfoque resulta en unos parches aproximadamente cuadrados en el arreglo. Sin embargo, puede que no sea el mejor enfoque. En el siguiente ejemplo, usamos el `*` para indicar que queremos que todos los elementos de una columna particular sean ubicados sobre le mismo procesador. Así, los valores de columna distribuyen equitativamente las columnas entre los procesadores. Entonces, todos los renglones en cada columna siguen donde ha sido colocada la columna. Ello permite un salto unitario para las porciones ubicadas adentro de los procesadores, y resulta benéfico en algunas aplicaciones. La sintaxis de `*` también se conoce como distribución `sobre el procesador`.

Distribuciones bidimensionales

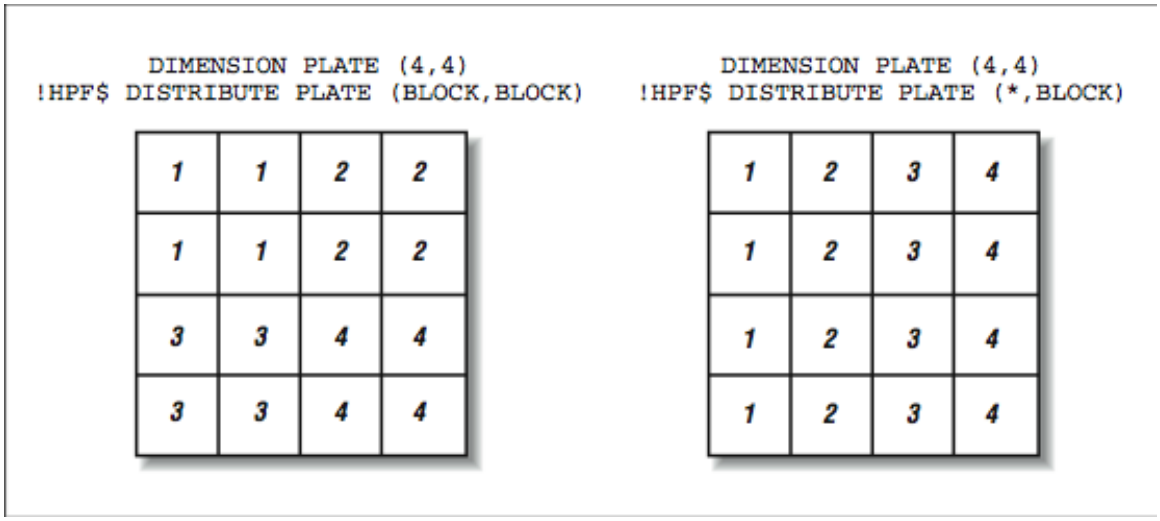

Figure 2

Cuando lidie con más de una estructura de datos para realizar un cálculo, puede bien sea distribuirlas separadamente , o bien usar la directiva `ALIGN` para asegurarse de que los elementos correspondientes de ambas estructuras estén alojados juntos. En el siguiente ejemplo, tenemos un arreglo de placa y un factor de escala que debemos aplicar a cada columna de la placa durante el cálculo:

```
      DIMENSION PLATE(200,200),SCALE(200)
!HPF$ DISTRIBUTE PLATE(*,BLOCK)
```

```
!HPF$ ALIGN SCALE(I) WITH PLATE(J,I)
```

O bien:

```
      DIMENSION PLATE(200,200),SCALE(200)
!HPF$ DISTRIBUTE PLATE(*,BLOCK)
!HPF$ ALIGN SCALE(:) WITH PLATE(*,:)
```

En ambos ejemplos, las variables `PLATE` y `SCALE` están ubicadas en los mismos procesadores que las columnas correspondientes de `PLATE`. La sintaxis `*` y `:` comunican la misma información. Cuando se usa `*`, esa dimensión se colapsa y no participa en la distribución. Cuando se usa `:`, significa que la esa dimensión sigue a la dimensión correspondiente en la variable que ya ha sido distribuida.

También puede usted especificar el acomodo de la variable `SCALE` y hacer que la variable `PLATE` "siga" la distribución de la variable `SCALE`:

```
      DIMENSION PLATE(200,200),SCALE(200)
!HPF$ DISTRIBUTE SCALE(BLOCK)
!HPF$ ALIGN PLATE(J,I) WITH SCALE(I)
```

Puede agregar expresiones aritméticas simples en la directiva `ALIGN`, sujetas a ciertas limitaciones. Las otras directivas incluyen:

`PROCESSORS` Le permite crear una forma de configuración de los procesos que pueda usarse para alinear otras estructuras de datos.

`REDISTRIBUTE y REALIGN` Le permite cambiar dinámicamente la forma de las estructuras de datos a tiempo de ejecución, conforme cambian los patrones de comunicación durante el curso de la misma.

`TEMPLATE` Le permite crear un arreglo que no usa espacio. En vez de distribuir una estructura de datos y alinear todas las demás, algunos usuarios crearán y distribuirán una plantilla y luego alinearán todas las estructuras de datos reales de acuerdo a esa plantilla.

El uso de directivas puede fluctuar desde lo muy simple a lo muy complejo. En algunas situaciones, usted distribuirá la única estructura grande compartida, alineando unas pocas estructuras relacionadas y habrá terminado. En otras, los programadores intentan optimizar las comunicaciones basándose en la topología de la red de interconexión (hypercubo, red de interconexión multietapa, malla o toroide) usando directivas muy detalladas. También pueden redistribuir cuidadosamente los datos durante las varias fases del cómputo.

Con suerte, su aplicación logrará un buen rendimiento sin demasiado esfuerzo.

3 Estructuras de control en HPF

Mientras los diseñadores de HPF estaban enmedio de definir un nuevo lenguaje, se dieron a la tarea de mejorar aquello que habían visto como una limitación de FORTRAN 90. Es interesante que tales modificaciones son las que se están considerando como parte del nuevo estándar de FORTRAN 95.

La sentencia `FORALL` permite al usuario expresar operaciones iterativas sencillas que se aplican al arreglo completo, sin descansar en un ciclo do-loop (recuerde, los ciclos do-loop fuerzan un orden). Por ejemplo:

```
FORALL (I=1:100, J=1:100) A(I,J) = I + J
```

Ello puede expresarse en FORTRAN 90 nativo, pero es mucho más feo, contraintuitivo y propenso a errores.

Otra estructura de control proporciona la habilidad de declarar una función como "PURE." Una función PURE no tiene efectos secundarios mas que a través de sus parámetros. El programador garantiza que una función PURE puede ejecutarse simultáneamente en muchos procesadores sin efectos indeseables. Esto le permite a HPF asumir que dicha función sólo operará sobre datos locales y que no requiere de ninguna comunicación de datos durante el tiempo que dure su ejecución completa. El programador también puede declarar cuáles parámetros de la función son de entrada, cuáles de salida y cuáles de ambos.

4 Intrínsecos de HPF

Las compañías que venden computadoras SIMD requieren entregarlas con herramientas que permitan la ejecución de operaciones colectivas eficientes sobre todos los procesadores. Un ejemplo perfecto de esto es la operación SUM. Para `SUM` el valor de un arreglo a lo largo de N procesadores, el enfoque más simplista toma N pasos. Sin embargo, es posible lograrlo en log(N) pasos usando una técnica denominada *suma prefija en paralelo*. Al momento en que se estaba desarrollando HPF se habían identificado e implementado varias de tales operaciones. HPF dio la oportunidad de definir una sintaxis estandarizada para ellas.

Una muestra de tales operaciones incluye:

`SUM_PREFIX` Realiza varios tipos de sumas prefijas en paralelo.
`ALL_SCATTER` Distribuye un único valor a un conjunto de procesadores.
`GRADE_DOWN` Ordena en orden decreciente.
`IANY` Calcula el OR lógico de un conjunto de valores.

Aun cuando existe un gran número de tales funciones intrínsecas, la mayoría de las aplicaciones sólo usa unas pocas de tales operaciones.

5 Extrínsecos de HPF

Con el objeto de permitir a los vendedores de arquitecturas diversas, proporcionar sus ventajas particulares, HPF incluyó la capacidad de enlazarse con funciones "extrínsecas". Tales funciones no requieren de reescribirse en FORTRAN 90/HPF, y realizan varias capacidades sólo soportadas por los vendedores. Esta capacidad permite a los usuarios realizar tales tareas, como la creación de aplicaciones híbridas, con algo de HPF y algo de paso de mensajes.

Los programadores de cómputo de alto rendimiento siempre gustan de tener la habilidad de hacer cosas a su propio modo, con el objetivo de exprimir hasta la última gota de rendimiento.

6 Flujo Calórico en HPF

Para transportar nuestra aplicación de flujo calórico a HPF, realmente sólo requerimos de agregar una línea de código. En el ejemplo siguiente, lo hemos cambiado a un arreglo bidimensional más grande:

```
      INTEGER PLATESIZ,MAXTIME
      PARAMETER(PLATESIZ=2000,MAXTIME=200)
!HPF$ DISTRIBUTE PLATE(*,BLOCK)
```

```
      REAL*4 PLATE(PLATESIZ,PLATESIZ)
      INTEGER TICK
      PLATE = 0.0

* Sumar las fronteras
      PLATE(1,:) = 100.0
      PLATE(PLATESIZ,:) = -40.0
      PLATE(:,PLATESIZ) = 35.23
      PLATE(:,1) = 4.5

      DO TICK = 1,MAXTIME
        PLATE(2:PLATESIZ-1,2:PLATESIZ-1) = (
     +    PLATE(1:PLATESIZ-2,2:PLATESIZ-1) +
     +    PLATE(3:PLATESIZ-0,2:PLATESIZ-1) +
     +    PLATE(2:PLATESIZ-1,1:PLATESIZ-2) +
     +    PLATE(2:PLATESIZ-1,3:PLATESIZ-0) ) / 4.0
        PRINT 1000,TICK, PLATE(2,2)
1000    FORMAT('TICK = ',I5, F13.8)
      ENDDO
*
      END
```

Notará que la directiva HPF distribuye las columnas del arreglo usando el enfoque `BLOCK`, manteniendo todos los elementos de una columna sobre un mismo procesador. A primera vista, pudiera parecer que (`BLOCK`,`BLOCK`) es la mejor distribución. Sin embargo, hay dos ventajas de una distribución (`*`,`BLOCK`). Primero, recorrer hacia abajo una columna es una operación de paso único, y de este modo puede usted procesar una columna completa. El aspecto más significativo de la distribución es que una distribución (`BLOCK`,`BLOCK`) fuerza a cada procesador a comunicarse con hasta ocho procesadores más para obtener los valores de sus vecinos. Usando la distribución (*,`BLOCK`), cada procesador tendrá que intercambiar datos con cuando mucho dos procesadores en cada ciclo.

Cuando veamos PVM, veremos el mismo programa implementado al estilo de paso de mensajes SPMD. En este ejemplo, verá usted algunos de los detalles que HFP debe manejar para ejecutar apropiadamente este código. Tras revisarlo ¡probablemente usted elija implementar todas sus aplicaciones futuras de flujo calórico en HPF!

7 Resumen de HPF

En ciertas cosas HPF ha sido mejor que FORTRAN 90. Compañías como IBM con su SP-1 requieren de proporcionar algún lenguaje de alto nivel para el que sus usuarios no quieren escribir código de paso de mensajes. Por tal motivo, IBM ha invertido una gran cantidad de esfuerzo en implementar un HPF optimizado. Resulta interesante que mucho de su esfuerzo beneficiará directamente la habilidad de desarrollar compiladores de FORTRAN 90 más sofisticados. El extensivo análisis de flujo de datos requerido para minimizar las comunicaciones y administrar las estructuras de datos dinámicos recaerá en los compiladores de FORTRAN 90 incluso sin usar las directivas de HPF.

El tiempo dirá si ya no serán necesarias las directivas de distribución de datos de HPF, y si los compiladores serán capaces de realizar suficiente análisis sobre el código plano de FORTRAN 90 para optimizar la ubicación y movimiento de los datos.

En su forma actual, HPF es un vehículo excelente para expresar las aplicaciones altamente paralelas a nivel de datos, y basadas en retículas. Sus debilidades son las comunicaciones irregulares y el balance de cargas dinámicas. Un nuevo esfuerzo para desarrollar la siguiente versión de HPF está en camino, para

manejar algunos de estos temas. Desafortunadamente, es más difícil resolver estos problemas a tiempo de ejecución a la vez que se mantiene un buen rendimiento a través de una amplia variedad de arquitecturas.