SEMINARIO 4B

Modelos Computacionales de simulación y dinámica de fluidos

Fortran

SERGIO HERAS ALVAREZ
UNIVERISDAD POLITECNICA DE MADRID

INDICE

¿Qué es Fortran?

FORTRAN (Formula Translation) es un lenguaje de programación de alto nivel utilizado en la computación científica y de ingeniería desde 1950. Fue el primer lenguaje de programación de alto nivel ampliamente utilizado y ha sido muy influyente en el desarrollo de otros lenguajes de programación.

FORTRAN fue diseñado originalmente por IBM para la programación de computadoras científicas y ha evolucionado a lo largo de los años en diferentes versiones. Actualmente, la versión más utilizada es Fortran 90/95, que incluye características avanzadas como la programación orientada a objetos, el manejo de punteros y la programación paralela.

FORTRAN se utiliza principalmente para aplicaciones de computación numérica y científica, incluyendo simulaciones de ingeniería, análisis de datos, modelado de sistemas físicos, procesamiento de imágenes y modelos computacionales. Es conocido por su capacidad para manejar grandes conjuntos de datos y realizar cálculos complejos con alta precisión.

Entornos de programación

En Fortran existen dos tipos de entornos para la programación mas relevantes: entorno secuencial y entorno paralelo.

Un entorno secuencial es aquel en el que los procesos o tareas se ejecutan de forma secuencial, es decir, uno después del otro en orden. En este entorno, solo se puede ejecutar una tarea a la vez, lo que puede retrasar la realización de tareas complejas y limitar la capacidad de procesamiento.

Por otro lado, un entorno paralelo es aquel en el que varias tareas se pueden ejecutar simultáneamente, lo que permite un mayor rendimiento y velocidad de procesamiento. En un entorno paralelo, las tareas se distribuyen entre varios procesadores o núcleos de procesamiento, lo que permite que se ejecuten de forma independiente y en paralelo.

La programación paralela se refiere a la técnica de diseñar programas para aprovechar el entorno paralelo y distribuir las tareas entre los procesadores disponibles para mejorar la velocidad y el rendimiento. La programación paralela es particularmente importante en la computación de alto rendimiento y en la resolución de problemas complejos que requieren grandes cantidades de cálculos y datos. Existen dos clases de paralelismo:

- Implícito
- Explicito

Paralelismo implícito

El paralelismo implícito se refiere a la capacidad de un compilador o entorno de programación para detectar automáticamente secciones de código que se pueden ejecutar en paralelo y optimizar su ejecución sin necesidad de que el programador indique explícitamente cómo realizar la paralelización.

En otras palabras, el paralelismo implícito permite que el compilador o el entorno de programación analicen el código en busca de secciones que puedan ejecutarse en paralelo y que automáticamente generen el código necesario para aprovechar el hardware paralelo disponible.

El paralelismo implícito se utiliza comúnmente en lenguajes de programación de alto nivel, como Fortran, C++ o Java . Estas herramientas ofrecen al programador una manera fácil de aprovechar el paralelismo implícito al proporcionar directivas de compilación y funciones que permiten la creación de programas paralelos sin tener que escribir todo el código explícitamente.

El paralelismo implícito puede mejorar significativamente el rendimiento y la escalabilidad de las aplicaciones, especialmente en sistemas de alto rendimiento y en la computación distribuida. Sin embargo, también puede tener algunas limitaciones y desafíos, como la necesidad de optimizar la distribución de datos y la coordinación entre los hilos de ejecución en paralelo.

Paralelismo Explicito

El paralelismo explícito se refiere a la técnica de diseño de programas que implica que el programador indique explícitamente qué secciones del código se ejecutarán en paralelo y cómo se distribuirán los datos y las tareas entre los hilos o procesos en paralelo.

En otras palabras, en el paralelismo explícito, el programador tiene el control total sobre la ejecución en paralelo y debe escribir código adicional para gestionar la distribución de tareas y datos entre los hilos de ejecución.

El paralelismo explícito se utiliza comúnmente en la programación de sistemas distribuidos y de alto rendimiento, donde se requiere un control detallado de los recursos y la comunicación entre procesos y nodos.

Algunas herramientas comunes para programación paralela explícita incluyen bibliotecas de paso de mensajes como MPI, bibliotecas de subprocesos como Pthreads y herramientas de programación para procesamiento en paralelo como OpenMP.

El paralelismo explícito puede proporcionar una mayor flexibilidad y control sobre la ejecución en paralelo, pero también puede ser más complejo y requerir más tiempo y esfuerzo por parte del programador para implementar y optimizar el código.

Lenguajes Paralelos

HPF (High Performance Fortran), Occam , Linda , PCP (Parallel C Preprocessor) y Fortran 90 son lenguajes de programación paralelos que se utilizan en la computación de alto rendimiento.

HPF es adecuado para aplicaciones Fortran existentes y permite una fácil paralelización mediante la especificación de la distribución de datos y tareas de manera declarativa. PCP proporciona una gran flexibilidad para la programación paralela explícita, permitiendo la optimización detallada de la distribución de tareas y datos. Fortran 90 es un lenguaje maduro y ampliamente utilizado que proporciona características para la programación paralela de manera clara y concisa.

Occam es adecuado para sistemas distribuidos y de memoria compartida y proporciona construcciones de programación sencillas para la comunicación y sincronización de procesos en paralelo. Linda es adecuado para sistemas de memoria compartida distribuida y proporciona una memoria compartida distribuida y operaciones de sincronización para la comunicación y coordinación de procesos en paralelo. Ambos lenguajes son herramientas útiles para la programación paralela, pero cada uno tiene sus propias fortalezas y debilidades en términos de su adecuación para diferentes aplicaciones y sistemas.

Paralelismo de Datos

El paralelismo de datos es una técnica de programación paralela que se utiliza para acelerar el procesamiento de grandes conjuntos de datos mediante la división del trabajo en tareas más pequeñas y su distribución en múltiples núcleos de procesamiento.

La ventaja del paralelismo de datos es que permite la reducción del tiempo de procesamiento de grandes conjuntos de datos sin tener que aumentar la velocidad del procesador o la memoria. Además, permite la utilización eficiente de múltiples núcleos de procesamiento para el procesamiento de datos en paralelo, lo que puede mejorar significativamente el rendimiento de las aplicaciones.

El paralelismo de datos puede implementarse en diferentes arquitecturas de computadoras, como SIMD (Single Instruction Multiple Data) y SPMD (Single Program Multiple Data).

Ambas arquitecturas SIMD y SPMD pueden utilizar el paralelismo de datos para procesar grandes conjuntos de datos en paralelo y mejorar el rendimiento de las aplicaciones. En general, las arquitecturas SIMD son adecuadas para aplicaciones que involucran cálculos de punto flotante intensivos, como la simulación física y la gráfica por computadora, mientras que las arquitecturas SPMD son adecuadas para aplicaciones que requieren la comunicación y sincronización de procesos, como el procesamiento de señales y la minería de datos.

Las Tablas en Fortran

Fortran es un lenguaje de programación que permite la creación y manipulación de tablas o matrices. En Fortran, las tablas se pueden definir como arreglos multidimensionales. La sintaxis para definir una tabla en Fortran es la siguiente:

- DIMENSION tabla(n, m)

Una tabla en fortran debe seguir las siguientes características:

- Un índice por dimensión
- El numero de dimensiones es <=7
- El numero total de elementos es >=0
- Tiene que haber un vector con tamaño de cada dimensión
- Toda tabla tiene que ser declarada expresando su rango

Ejemplo de Tabla de 2 dimensiones

```
INTEGER :: n, m, i, j
INTEGER, DIMENSION(:,:), ALLOCATABLE :: tabla

n = 3
m = 4

ALLOCATE(tabla(n, m))

DO i=1,n
```

```
  DO j=1,m
    tabla(i,j)  = i + j
  END DO
END DO

WRITE(*,*) 'Tabla de enteros:'
DO i=1,n
  WRITE(*,*) (tabla(i,j),  j=1,m)
END DO

DEALLOCATE(tabla)
```

Cálculos con Tablas

Fortran permite realizar cálculos con tablas utilizando operaciones aritméticas y funciones matemáticas básicas. Por ejemplo, para sumar dos tablas "tabla1" y "tabla2" de igual tamaño, se puede escribir:

tabla3 = tabla1 + tabla2

Para multiplicar una tabla "tabla" por un escalar "escalar", se puede escribir:

*tabla = escalar * tabla*

Para calcular la suma de los elementos de una tabla "tabla", se puede utilizar la función intrínseca "SUM". Por ejemplo:

suma = SUM(tabla)

A continuación, se muestra un ejemplo completo de cómo realizar cálculos con tablas en Fortran:

```
PROGRAM CalculosTabla
  IMPLICIT NONE

  INTEGER :: n, m, i, j
  REAL, DIMENSION(:,:), ALLOCATABLE :: tabla1, tabla2, tabla3
  REAL :: escalar, suma

  n = 3
  m = 3

  ALLOCATE(tabla1(n, m))
  ALLOCATE(tabla2(n, m))
  ALLOCATE(tabla3(n, m))
```

```
DO i=1,n
  DO j=1,m
    tabla1(i,j)  = i*j
    tabla2(i,j)  = i+j
  END DO
END DO

WRITE(*,*) 'Tabla1:'
DO i=1,n
  WRITE(*,*) (tabla1(i,j),  j=1,m)
END DO

WRITE(*,*) 'Tabla2:'
DO i=1,n
  WRITE(*,*) (tabla2(i,j),  j=1,m)
END DO

escalar = 2.0
tabla3 = tabla1 + tabla2
tabla1 = escalar * tabla1
suma = SUM(tabla3)

WRITE(*,*) 'Tabla3 (suma = ', suma, '):'
DO i=1,n
  WRITE(*,*) (tabla3(i,j),  j=1,m)
END DO

WRITE(*,*) 'Tabla1 (escalar = ', escalar, '):'
DO i=1,n
```

```
  WRITE(*,*) (tabla1(i,j),  j=1,m)
 END DO

 DEALLOCATE(tabla1)
 DEALLOCATE(tabla2)
 DEALLOCATE(tabla3)

END PROGRAM CalculosTabla
```

Este programa crea dos tablas de 3x3, las inicializa con valores y realiza varios cálculos con ellas, incluyendo suma, multiplicación por un escalar y cálculo de la suma de los elementos. Luego, muestra los resultados en la consola.

Tablas dinámicas

En Fortran, es posible crear tablas dinámicas utilizando la función intrínseca "ALLOCATE". La función "ALLOCATE" permite asignar memoria dinámicamente a una tabla, lo que significa que se puede cambiar el tamaño de la tabla durante la ejecución del programa.

Para crear una tabla dinámica en Fortran, primero se debe declarar la tabla como una variable allocatable, utilizando la palabra clave "ALLOCATABLE" en la declaración de la variable. Luego, se utiliza la función "ALLOCATE" para asignar la memoria necesaria a la tabla.

A continuación, se muestra un ejemplo de cómo crear una tabla dinámica en Fortran:

```
PROGRAM TablaDinamica
 IMPLICIT NONE

 INTEGER :: n, m, i, j
 REAL, DIMENSION(:,:), ALLOCATABLE :: tabla

 n = 3
 m = 3

 ALLOCATE(tabla(n, m))

 DO i=1,n
  DO j=1,m
   tabla(i,j) = i*j
  END DO
```

```
 END DO

 WRITE(*,*) 'Tabla:'
 DO i=1,n
   WRITE(*,*) (tabla(i,j), j=1,m)
 END DO

 DEALLOCATE(tabla)

END PROGRAM TablaDinamica
```

Es importante tener en cuenta que el uso de tablas dinámicas en Fortran puede requerir un manejo cuidadoso de la memoria y una planificación adecuada para evitar errores de asignación de memoria o fugas de memoria.

Sentencia FORALLL

La sentencia "FORALL" en Fortran es una característica importante para el procesamiento de matrices grandes en paralelo. Permite especificar un conjunto de operaciones que se deben realizar en paralelo en varios elementos de una matriz, lo que puede mejorar significativamente el rendimiento del programa en comparación con el uso de un bucle convencional "DO".

La sentencia "FORALL" es especialmente útil cuando se trabaja con matrices grandes que pueden ser procesadas en paralelo por múltiples procesadores o núcleos. Al utilizar "FORALL", se puede aprovechar al máximo el potencial de paralelismo disponible en el hardware, lo que puede conducir a una mejora significativa del rendimiento.

Además, la sintaxis de "FORALL" es más simple y legible que la de un bucle "DO", lo que puede facilitar la programación y el mantenimiento del código.

Es importante tener en cuenta que la sentencia "FORALL" no siempre mejora el rendimiento del programa. Depende de varios factores, como el tamaño de la matriz, la cantidad de procesadores disponibles y la complejidad de las operaciones a realizar en cada elemento de la matriz. Por lo tanto, es recomendable hacer pruebas y mediciones para determinar si el uso de "FORALL" es beneficioso en un caso particular.

Conclusiones

En conclusión, Fortran (Formula Translation) es un lenguaje de programación de alto nivel, diseñado para aplicaciones científicas y de ingeniería en la década de 1950. A lo largo de los años, ha evolucionado y se ha adaptado para satisfacer las necesidades cambiantes de los usuarios y de la industria, lo que ha llevado al desarrollo de Fortran 90, 95, 2003, 2008 y 2018.

Fortran es utilizado en aplicaciones científicas y de ingeniería que requieren cálculos numéricos intensivos y alto rendimiento en cálculo paralelo. Además, muchos programas de simulación, modelado y análisis de datos utilizan Fortran debido a su capacidad para procesar grandes conjuntos de datos de manera eficiente.

A pesar de la evolución de otros lenguajes de programación y del aumento en popularidad de lenguajes de programación de propósito general como Python, Fortran sigue siendo un lenguaje importante para los usuarios que requieren de una alta precisión y eficiencia en cálculos numéricos y científicos.

En resumen, Fortran sigue siendo una herramienta valiosa en la comunidad científica y de ingeniería debido a su capacidad para manejar grandes conjuntos de datos y cálculos numéricos de alta precisión, y su continua evolución asegura su relevancia en el futuro.