ARQUITECTURA DE COMPUTADORES II

AUTORES:
David Expósito Singh
Florin Isaila
Daniel Higuero Alonso-Mardones
Javier García Blas
Borja Bergua Guerra

Área de Arquitectura y Tecnología de Computadores
Departamento de Informática
Universidad Carlos III de Madrid

Julio de 2012

TEMA 2:
PROGRAMACIÓN PARALELA (I)

Índice (I)

1. Modelos de programación paralela:

   - Introducción
   - Modelo de Paralelismo sobre Datos
   - Modelo de Paso de Mensajes
   - Modelo de Memoria Compartida
   - Otros Modelos

Índice (I)

1. Modelos de programación paralela:

   - Introducción
   - Modelo de Paralelismo sobre Datos
   - Modelo de Paso de Mensajes
   - Modelo de Memoria Compartida
   - Otros Modelos

Introducción

Modelo de programación:

Colección de abstracciones de programación que le brindan al programador una visión simplificada y transparente de la arquitectura.

Modelo de programación vs. Arquitectura

- Con frecuencia un **modelo de programación** se **ajusta** mejor a un tipo de **arquitectura** que a otra.

- Sin embargo, un modelo programación puede usarse sobre una arquitectura diferente a la que le es "natural".

Índice (I)

1. Modelos de programación paralela:

- Introducción
- Modelo de Paralelismo sobre Datos
- Modelo de Paso de Mensajes
- Modelo de Memoria Compartida
- Otros Modelos

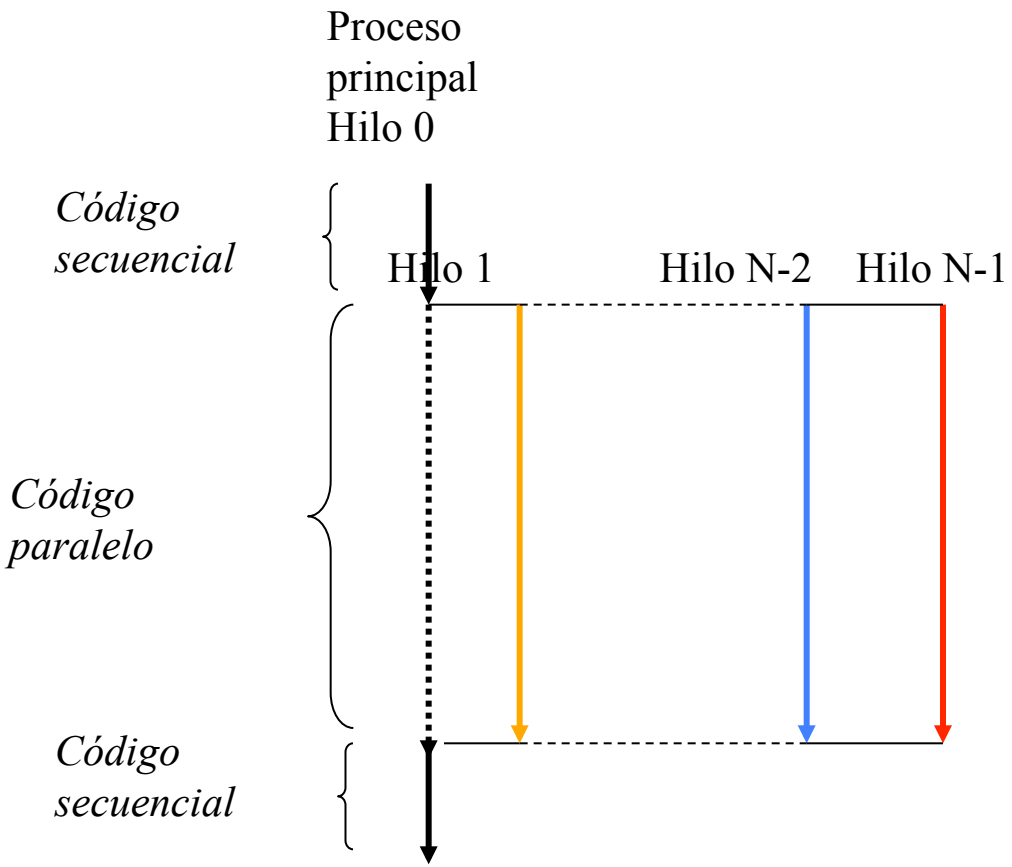
Modelo 1: *Paralelismo sobre datos*

Paralelismo sobre datos:

Concurrencia *que se obtiene cuando se aplica la* ***misma operación*** *sobre* ***todos los elementos*** *de un conjunto.*

- Se corresponde con SIMD donde cada procesador ejecuta el mismo código sobre diferentes datos.
- Dos alternativas de programación: concurrencia explícita e implícita.

Paralelismo sobre datos: Funcionamiento interno para memoria compartida



Paralelismo sobre datos: Concurrencia explícita

- El **programador** aporta:
  - Algoritmo.
  - Directivas para distribuir datos.
  - Directivas que guíen la paralelización.

- **Entorno de desarrollo** (p.ej., compilador HPF, librería OpenMP) se encarga:
  - Descubrir paralelismo potencial en el algoritmo.
  - Uso de las directivas para asignar datos y trabajo a los procesadores (SPMD).
  - Manejar la comunicación y la sincronización.

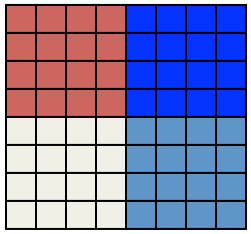
Paralelismo sobre datos: Concurrencia explícita

- Modelo aplicable en múltiples arquitecturas paralelas:
  - Thinking Machine.
  - Máquinas vectoriales.
  - Arquitecturas UMA/NUMA.
  - Multicomputadores.

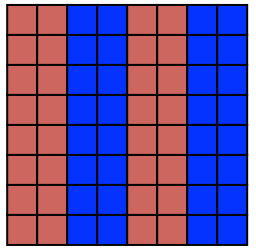
- SIMD / SPMD.

Paralelismo sobre datos: Ejemplo: HPF

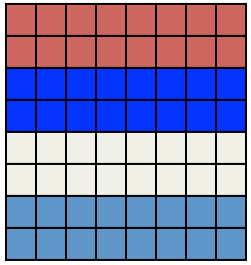
!HPF$ DISTRIBUTE New (BLOCK,BLOCK)



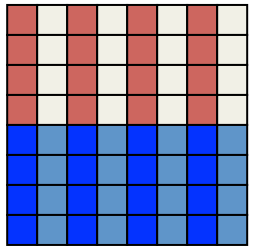
!HPF$ DISTRIBUTE New (*,CYCLIC(2))



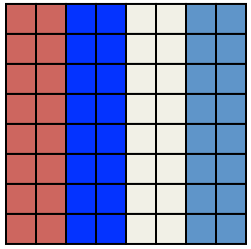
!HPF$ DISTRIBUTE New (BLOCK,*)



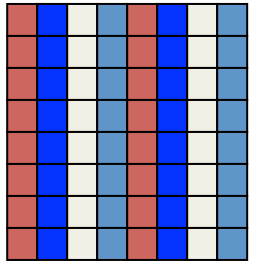
!HPF$ DISTRIBUTE New (BLOCK,CYCLIC)



!HPF$ DISTRIBUTE New (*,BLOCK)



!HPF$ DISTRIBUTE New (*,CYCLIC)



Paralelismo sobre datos: Concurrencia implícita

- El compilador es capaz de detectar iteraciones **independientes** que pueden ser realizadas concurrentemente. En este caso, todas las interaciones son independientes, ya que el resultado de una iteración no es necesario para realizar el cálculo de la siguiente iteración.

Ejemplo:

```
do i = 1,m
    do j = 1, n
        A(i,j) = B(i,j) x C(i,j)
    enddo
enddo
```

Paralelismo sobre datos: Ejemplo: versión secuencial

Ejemplo: Fortran77: programa secuencial

```
program f77_finite_difference
real X(100,100), New(100,100)
do i = 2,99
  do j = 2,99
    New(i,j) = (X(i-1,j) + X(i+1,j) + X(i,j-1) + X(i,j+1))/4
  enddo
Enddo

diffmax = 0.0
do i = 1, 100
  do j = 1, 100
    diff = abs(New(I,j) - X(I,j))
   if (diff.gt.diffmax) diffmax = diff
  enddo
enddo
```

Paralelismo sobre datos: Ejemplo: OpenMP

Fragmento de la codificación en OpenMP:

```
#pragma omp for
   for (i=1; i<n-1; i++ )
        for( j=1, j<n-1, j++ )
        temp[i][j] =0.25*(X[i-1][j]+X[i+1][j]+X[i][j-1]+X[i][j+1]);
```

```
#pragma omp for
diffmax = 0.0
do i = 1, 99
  do j = 1, 99
    diff(rank) = MAX(temp(I,j) - X(I,j))
  enddo
enddo
```

```
#pragma omp Parallel
Diffmax=MAX(diff)
```

Índice (I)

1. Modelos de programación paralela:

- Introducción
- Modelo de Paralelismo sobre Datos
- Modelo de Paso de Mensajes
- Modelo de Memoria Compartida
- Otros Modelos

Modelo 2: *Paso de mensajes*

Paso de mensajes

Un cómputo está compuesto de uno o más ***procesos*** *que se* ***comunican*** *enviando y recibiendo* ***mensajes*** *de otros procesos.*

- Se corresponde con modelo de *hardware* MIMD

Paso de mensajes: Uso

- Librerías de paso de mensajes:
  - **PVM**: pionero: desarrollado para su empleo en "*sistemas heterógeneos*".
  - **MPI**: más reciente y actualmente más difundido.
- Las librerías se usan desde diferentes lenguajes de alto nivel:
  - Fortran, C, C++, Java, etc.

Paralelismo MIMD sobre memoria distribuida:

Fragmento de la codificación en MPI (no eficiente):

```
PROGRAMA PROCESADOR 1

    for (i=0; i<n; i++ )
          for( j=0, j<n, j++ )
          temp[i][j] =0.25*(X[i-1][j]+X[i+1][j]+X[i][j-1]+X[i][j+1]);
    endfor
   COMM_SEND(X)
   COMM_SEND(TEMP)


PROGRAMA PROCESADOR 2
   COMM_RECEIVE(X)
   COMM_RECEIVE(TEMP)
    diffmax = 0.0
    do i = 1, 100
      do j = 1, 100
        diff = MAX(temp(I,j) - X(I,j))
          if (diff.gt.diffmax) diffmax = diff
      enddo
    enddo
```

Índice (I)

1. Modelos de programación paralela:

   - Introducción
   - Modelo de Paralelismo sobre Datos
   - Modelo de Paso de Mensajes
   - Modelo de Memoria Compartida
   - Otros Modelos

Modelo 3: *Memoria compartida*

Memoria compartida

- Los procesos comparten direcciones de memoria.
- **Comunicación** mediante las variables que comparten.
- **Sincronización** a través de mecanismos para el acceso exclusivo a los datos.

- Se corresponde con arquitecturas de Memoria Compartida.

- Se puede aplicar a arquitecturas de memoria distribuida con soporte software de memoria compatida.

Memoria compartida: Programación

- Programación, fases:

1. Decidir cómo **descomponer** el problema en partes.
2. Crear y destruir los **procesos** que soportan esta descomposición.
3. Añadir **sincronización** para asegurarse de que las dependencias de acceso a datos se mantienen.

Memoria compartida: Uso

- En general, se dice que este modelo permite una forma de programar más **cómoda** o "natural" que el paso de mensajes.

- Estándares de Memoria Compartida:
  - *Threads:* Práctica 3.
  - OpenMP: Práctica 2.

Paralelismo MIMD sobre memoria compartida:

Fragmento de la codificación en OpenMP (no eficiente):

```
PROGRAMA PROCESADOR 1
    shared=0
    for (i=0; i<n; i++ )
          for( j=0, j<n, j++ )
          temp[i][j] =0.25*(X[i-1][j]+X[i+1][j]+X[i][j-1]+X[i][j+1]);
    endfor
    shared=1


PROGRAMA PROCESADOR 2
    while(shared==0) do
          nothing;

    diffmax = 0.0
    do i = 1, 100
      do j = 1, 100
        diff = MAX(temp(I,j) - X(I,j))
          if (diff.gt.diffmax) diffmax = diff
      enddo
    enddo
```

Índice (I)

1. Modelos de programación paralela:

- Introducción
- Modelo de Paralelismo sobre Datos
- Modelo de Paso de Mensajes
- Modelo de Memoria Compartida
- Otros Modelos

Otros modelos

- Modelos de Programación Lógica: Prolog Concurrente, Parlog, etc.
- Modelos de Programación Funcional: lenguajes *data-flow*.
- Arquitecturas sistólicas.
- Modelos de Programación Orientada a Objetos.

Bibliografía

- ***Parallel Computer Architectures: a Hardware/Software Approach.*** *D.E. Culler, J.P. Singh, with A. Gupta*
  - Capítulo 1, sección 1.2