Computaciones de tipo stencil: Solución secuencial

Curso 2013-2014
Computación paralela
Universidad de Valladolid

1 Introducción

En el campo de la ciencia e ingeniería existen numerosas aplicaciones que implican la resolución de grandes sistemas de ecuaciones. Los métodos de resolución directos (como resolver un sistema de ecuaciones $Ax = b$ encontrando la inversa de la matriz A) no se pueden aplicar cuando los sistemas son muy grandes. En la práctica se utilizan métodos iterativos que calculan la solución de forma aproximada partiendo de una solución inicial que se refina en cada iteración del método.

1.1 Computación tipo Stencil

La simulación de muchos fenómenos físicos se calculan utilizando las ecuaciones de Laplace o de Poisson, en general ecuaciones diferenciales parciales (PDE, *Partial Differential Equation*) de tipo elíptico. Por ejemplo, el cálculo del punto de estabilidad de difusión de calor en un medio.

En este tipo de simulaciones, el procedimiento numérico se implementa utilizando una matriz multidimensional en la que cada elemento representa el valor en un punto del espacio. El algoritmo resultante tras la discretización de la ecuaciones y su resolución por un método iterativo (por ejemplo Jacobi) consiste en actualizar el valor de cada posición con los valores anteriores de los puntos más cercanos (vecinos). Esta actualización se repite en cada iteración.

Para cada ecuación los vecinos que intervienen se escogen de forma diferente. El patrón con el que se escogen los vecinos se denomina *stencil*. En concreto, para la ecuación de Poisson se utiliza un stencil conocido como *5-point star* stencil, que utiliza la media de los cuatro vecinos inmediatamente adyacentes para actualizar el quinto valor en la posición central.

$$v_{i,j} = \frac{x_{i-1,j} + v_{i+1,j} + v_{i,j-1} + v_{i,j+1}}{4}$$

De forma general en las operaciones tipo stencil, los valores de la matriz de datos se actualizan con los valores correctamente ponderados de los elementos vecinos. Los coeficientes usados para ponderar los valores de los diferentes elementos vecinos se definen según el problema.

$$m_{x,y} = \sum_{\substack{x-1 \le i \le x+1 \\ y-1 \le j \le y+1}} \frac{n_{i,j}}{9}$$

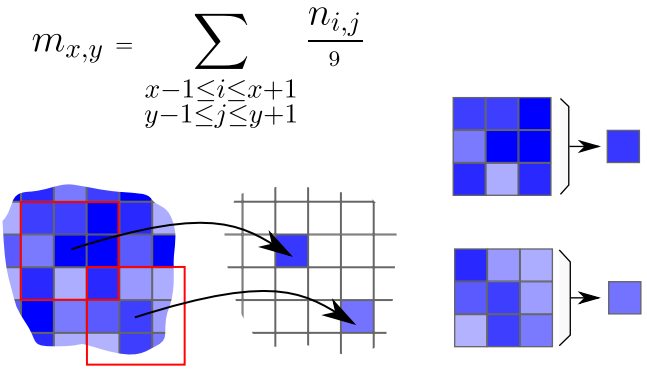
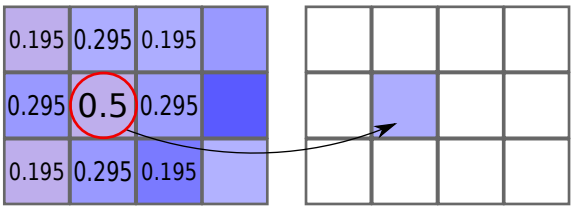


Figure 1: Stencil bidimensional de tamaño 9.



$$n[1,1] = m[1,1] * \mathbf{0.5} +$$

$$(m[0,1] + m[1,2] + m[2,1] + m[1,0]) * \tfrac{1}{4} * \mathbf{0.295} +$$

$$(m[0,0] + m[0,2] + m[2,0] + m[2,2]) * \tfrac{1}{4} * \mathbf{0.195}$$

Figure 2: Ejemplo de Stencil bidimensional de tamaño 9 con pesos asignados.

En la Figura 1 se puede observar un ejemplo de operación stencil. Se trata de un stencil bidimensional de tamaño 9. Esto quiere decir que son 2 las dimensiones sobre las que se extiende el cálculo de la operación y 9 el número de elementos de la matriz que involucrados en el mismo. Por lo tanto, cada posición de la matriz se actualiza tomando como referencia su propio valor y el valor de las 8 posiciones que la rodean. Cada uno de los valores involucrados en el cálculo son ponderados de forma diferente según el problema, y se obtiene un número que será el nuevo valor asignado a esa posición de la matriz, tal y como puede verse en la Fig. 2. Cuando se han actualizado todos los elementos de la matriz se dice que se ha completado una iteración.

1.2 Calculo de la convergencia

Los métodos iterativos se puede aplicar durante un número determinado de iteraciones o se hasta que se produzca una cierta situación de estabilidad en las posiciones de la matriz. La estabilidad de una operación se puede determinar mediante el residuo (diferencia entre el resultado de dos iteraciones consecutivas).

En la práctica se suele calcular el residuo de cada uno de los puntos de matriz y se utiliza el valor máximo de estos valores como valor representativo de la estabilidad. De manera que si el residuo representativo obtenido es menor que un valor prefijado consideramos que la operación ha alcanzado el grado de estabilidad deseado. En este punto, la ejecución de más iteraciones de la operación apenas produciría cambios y por tanto la operación global puede detenerse.

1.3 Condiciones de contorno

Las matrices sobre las que se realiza las operaciones stencil tienen un tamaño finito. Las actualizaciones de los valores que se encuentran en el borde no disponen de valores adyacentes para actualizar su valor. Para solucionar esto, se suele extender la matriz añadiendo una *frontera* o *halo* que permite realizar el cálculo de la misma manera que con las posiciones interiores. Esta frontera puede tener valores prefijados (condición de contorno constante), o se utilizar los valores opuestos en la dimensión (condiciones de contorno periódicas). Es decir, las celdas del borde utilizan como vecinas a las celdas del extremo opuesto, como se muestra en la Fig. 3.

2 Descripción de la versión secuencial

Se proporciona el código de un programa secuencial que simula una operación sencilla de mecánica de fluidos. El programa utiliza una matriz de datos con los valores de calor de una superficie bidimensional. En cada iteración del programa se calcula la transferencia de calor producida al aplicar una operación stencil bidimensional de tamaño 9, utilizando un espacio con fronteras periódicas. La condición de terminación corresponde a alcanzar el estado de estabilidad mediante cálculo de residuos.

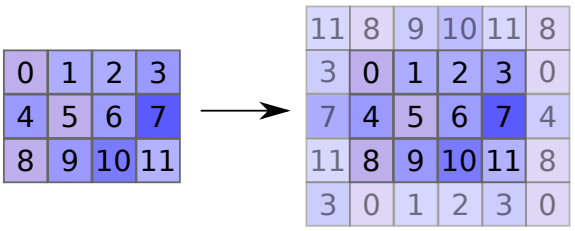


Figure 3: Stencil con fronteras periódicas.

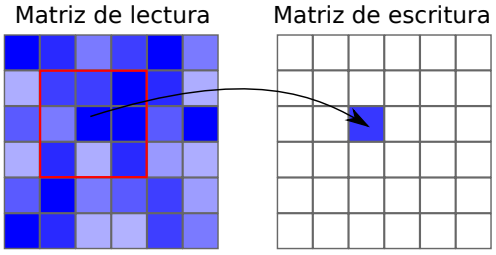


Figure 4: Los valores se leen de una matriz y se escriben en otra.

Este programa será paralelizado con las diferentes tecnologías en cada una de las prácticas propuestas.

Los coeficientes utilizados y el residuo considerado para que haya estabilidad son los siguientes:

- Peso de la celda = 0.5
- Peso de las celdas adyacentes = 0.295
- Peso de las celdas esquina = 0.195
- Condición de estabilidad: residuo $\leq$ 0.001

La operación stencil se implementa mediante dos matrices. La Fig. 4 muestra esta solución, que consiste en llevar a cabo iteraciones en una matriz de lectura, escribiendo el resultado en una matriz de escritura. Al final de cada iteración es necesario copiar los valores de la matriz de escritura en la matriz de lectura, de modo que la siguiente iteración tenga los valores actualizados. Una alternativa a esta operación de copia consiste en intercambiar las matrices, por lo que en la siguiente iteración la matriz de lectura pasará a ser la matriz de escritura y viceversa.

El pseudocódigo de la operación stencil implementada puede verse en la Fig. 5.

2.1 Datos de entrada

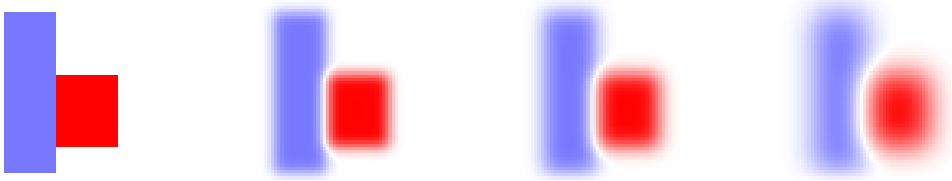
Hay varias matrices de entrada, de distintos tamaños, que se encuentra en el directorio `inputset/`. La salida esperada para cada una de las matrices de entrada está almacenada en el directorio `inputset/output/`.

2.2 Ejecución

Para compilar la práctica se dispone de un fichero Makefile que facilita la tarea. Si se ejecuta `make` sin opciones, se genera un ejecutable con el nombre `practica`. Para ejecutar el fichero es necesario pasarle como entrada una de las matrices proporcionadas. Este ejecutable es el que deberéis utilizar para obtener los tiempos de ejecución.

```
matriz_original = leer_matriz ();
do {
  iteraciones++;
  copia_frontera_periodica(matriz_original);
  matriz_actualizada = calculo_stencil(matriz_original);
  residuo = calcular_residuo(matriz_original, matriz_actualizada);
  matriz_original = matriz_actualizada;
} while (residuo > MAX_RESIDUO)
```

Figure 5: Pseudocódigo de la operación stencil implementada.



Iteración 0 Iteración 10 Iteración 20 Iteración 50

Figure 6: Estado intermedio de la matriz mat2.txt en las iteraciones 0, 10, 20 y 50.

```
$ make
$ ./practica ../inputset/mat15x20.txt
Matriz ../inputset/mat15x20.txt: 15x20
Check sum: -20.941669
Iteraciones: 139
Tiempo de ejecución: 0.001696
```

Si se ejecuta `make` con la opción `display`, se genera un ejecutable `practica_display`. Este fichero se ejecuta de la misma manera que el anterior, pero en su ejecución se genera un gráfico en el que podéis contemplar la evolución del programa. Cuando el tamaño de la matriz es muy grande no se genera el gráfico. Este ejecutable no es apropiado para la medición de tiempos.

```
$ make display
$ ./practica_display ../inputset/mat15x20.txt
```

3 Bibliotecas de utilidades

Junto al código secuencial del stencil se proporciona también una biblioteca que incluye una serie de tipos y funciones para leer los datos de entrada, medir el tiempo y otras que puede servir de utilidad al desarrollar los programas.

Para utilizarla hay que incluir el fichero de cabecera `cputils.h`, el Makefile que se proporciona con el programa secuencial está configurado para compilar con la biblioteca. A continuación se explican algunos de los elementos:

Tipo de datos de las matrices

La biblioteca puede manejar matrices de entrada de tipo flotante de simple precisión (`float`) o doble precisión (`double`). El comportamiento se puede modificar cambiando el macro `CP_MATRIX_TYPE` del fichero de cabecera `cputils.h` a `CP_FLOAT` (por defecto) o `CP_DOUBLE`. El tipo usado por la

matrices se define en el datatype `m_type`. Se puede utilizar este datatype y el macro para generar un código que se adapte en función del tipo. Por ejemplo:

```
// Reservo memoria para una matriz
m_type * matrix = malloc(sizeof(m_type) * (size_t) (row * cols));

// Comprobaciones extra en función del tipo
#if CP_MATRIX_TYPE == CP_FLOAT
// m_type is a float
#
```

Medición de tiempos

Para proporcionar una manera de medir el tiempo en el código secuencial similar a las funciones de OpenMP (`omp_get_wtime`) y MPI (`MPI_Wtime`), en la biblioteca hemos añadido la función:

```
double cp_Wtime();
```

Es una función que devuelve el tiempo en segundos desde un instante dato y se utiliza para medir el tiempo de ejecución del código. En la versión secuencial, la medición de tiempos está colocada al principio y al final de la región de código que se os pide paralelizar. Se utiliza de la siguiente manera:

```
// Se obtiene el tiempo al inicio
t_begin = cp_Wtime();
// Código a medir
code_to_measure();
// Se obtiene el tiempo al final
t_end = cp_Wtime();
// Se imprime el tiempo
printf(" Tiempo de ejecución: %f\n", t_end-t_begin);
```

Lectura de las matrices

Existen dos funciones para leer las matrices que se facilitan como datos de entrada:

```
void cp_read_matrix_size(char * filename, int * rows, int * cols);
```

y

```
void cp_read_matrix(char * filename, m_type * matrix, int border);
```

La primera función lee el tamaño de la matriz, devuelve el número de filas y columnas en los punteros correspondientes. La segunda, lee los datos de la matriz y los guarda en la variable `matrix` que debe ser reservada por el usuario. El parámetro `border` indica si la matriz destino tiene o no borde. Si `border` es 0, se supone que la matriz reservada es justo de tamaño rows $\times$ cols. Si `border` es 1, se supone que la matriz reservada es de tamaño rows $+ 2 \times$ cols $+ 2$ y los datos se colocan a partir de la posición [1,1].

Desactivar Xlib

Si tu distribución no tiene instalada las librerías Xlib para la generación del gráfico que muestra la evolución del Stencil, en el Makefile se puede desactivar su uso:

```
CP_DISPLAY:=no
```

Opciones adicionales del Makefile

Para eliminar los ejecutables de la práctica:

```
$ make clean
```

Si cambiáis de máquina y movéis los ficheros es necesario que volváis a compilar a librería:

```
$ make cleanutils
$ make
```