Práctica de MPI

Tecnología de la Programación Paralela (TPP)

José E. Román Paulo B. Vasconcelos

Esta práctica se compone de dos partes. En la primera se trabaja con una operación muy común como es el producto matriz-vector, utilizando primitivas de comunicación punto a punto en la paralelización. La segunda parte aborda un algoritmo algo más complejo, y además se trabaja la parte de topologías.

Producto Matriz-Vector

En los archivos `matvec.c` y `matvecmpi.c` se proporciona la versión secuencial y paralela del producto matriz-vector de una matriz banda cuadrada $A \in \mathbb{R}^{N\times N}$ con ancho de banda b. La estructura banda de la matriz se refiere a que el elemento a_{ij} es cero si $|i-j| > b$.

En la versión paralela, la distribución de la matriz se realiza por bloques de filas contiguas, siendo el número de filas local a cada proceso, n, aproximadamente el mismo pero no necesariamente igual en todos los procesos. Para simplificar la paralelización, se asume el caso sencillo en que el ancho de banda es siempre menor que el tamaño local de la matriz, n.

En la Figura 1 (izquierda) se muestra un esquema para ayudar a entender el código paralelo.

E1 Análisis de la Versión Paralela

Este primer ejercicio consiste en comparar la versión secuencial y paralela para entender perfectamente la forma en que se ha realizado la paralelización. Realizar algunas medidas de tiempos para analizar las prestaciones (*speedup* y eficiencia para diferente número de procesos). Instrumentar apropiadamente el código para tomar tiempos (`MPI_Wtime`). Los valores de N y b deben ser suficientemente grandes para que los tiempos sean significativos.

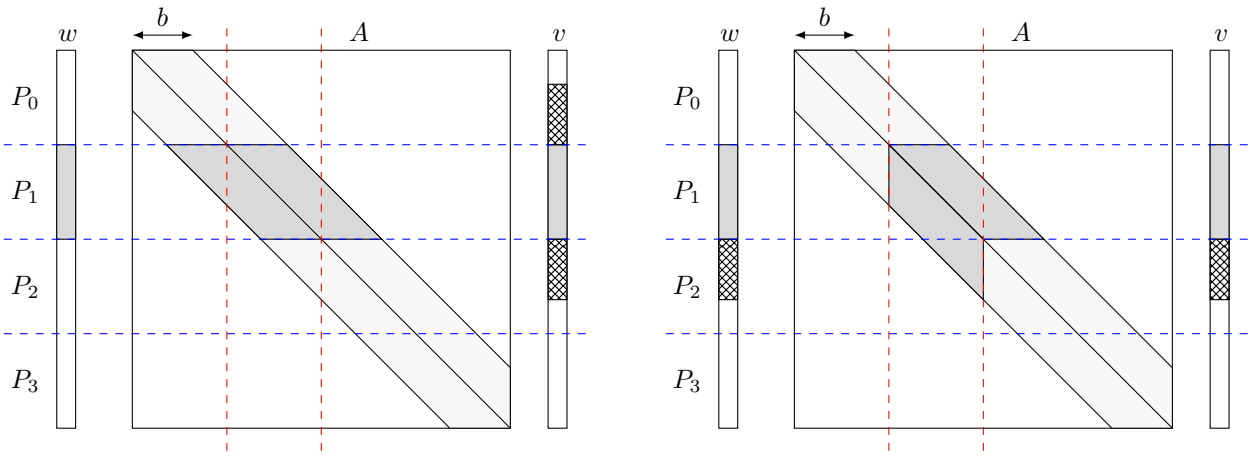

Figura 1: Esquema del producto matriz-vector $w = Av$, donde A es una matriz banda con ancho de banda b. Se ilustra el caso de 4 procesos. La parte sombreada indica los datos almacenados localmente en el proceso P_1, y las zonas rayadas indican datos pertenecientes a otros procesos que implican comunicación con P_1. El esquema de la derecha es una modificación para el caso de almacenamiento simétrico.

E2 **Versión con Comunicación No Bloqueante**

El objetivo de este ejercicio es hacer una versión que intente mejorar las prestaciones paralelas solapando la comunicación con parte de la computación. Esto es posible porque hay parte del cálculo (correspondiente a las columnas que coinciden con el rango local de filas) que no depende de los datos de otros procesos. Para ello, hay que utilizar primitivas de comunicación no bloqueantes.

E3 **Versión para Almacenamiento Simétrico (opcional)**

En matrices simétricas es posible almacenar solo la parte triangular superior, por ejemplo. Cada proceso posee localmente un rango de filas (parte superior) y le corresponden también los elementos simétricos, por lo que el esquema de comunicación cambia (ver Figura 1, derecha). Adaptar el código paralelo a este caso.

Métodos Estacionarios para la Ecuación de Poisson

El programa `poisson.c` resuelve la ecuación de Poisson $\nabla^2 u = f$ en un dominio rectangular mediante el método de diferencias finitas centradas, utilizando una malla de discretización en la que se divide el eje horizontal en $M+1$ subintervalos y el vertical en $N+1$ subintervalos, resultando en un total de $M \times N$ puntos interiores (incógnitas). El sistema de ecuaciones resultante, $Ax = b$, se resuelve mediante el método iterativo estacionario de Jacobi sin construir explícitamente la matriz A. En cada iteración de Jacobi, se actualiza el valor en cada uno de los puntos de la malla haciendo una media ponderada con el valor de los cuatro vecinos.

E4 **Caso Particionado Unidimensional**

El objetivo de este ejercicio es realizar la paralelización con MPI.

En primer lugar, se recomienda modificar el programa para que las condiciones de contorno se almacenen explícitamente, de forma que no sea necesario considerar cada caso de forma particular. Para ello, los vectores deberán tener longitud $(M+2) \times (N+2)$, almacenando ceros en los valores de la frontera.

Paralelizar el cálculo dividiendo el dominio en bandas horizontales, una por procesador (ver Figura 2). Cada subdominio deberá tener también una orla de valores alrededor (*ghost values*) para almacenar los valores de la frontera real o bien los valores recibidos de otro procesador (frontera ficticia).

Nota: Usar comunicación punto a punto para intercambio de valores de la malla, intentando resolver el problema de la secuencialización. Para el criterio de parada, utilizar una primitiva de comunicación colectiva.

E5 **Caso Particionado Bidimensional con Topología Cartesiana (opcional)**

El objetivo es particionar el dominio tanto en vertical como en horizontal, por lo que cada proceso podrá tener hasta 4 vecinos. Para simplificar la programación, utilizar la utilidad de topologías cartesianas de MPI, con `MPI_Cart_shift` para obtener el identificardor de los vecinos. Usar tipos de datos derivados de MPI.

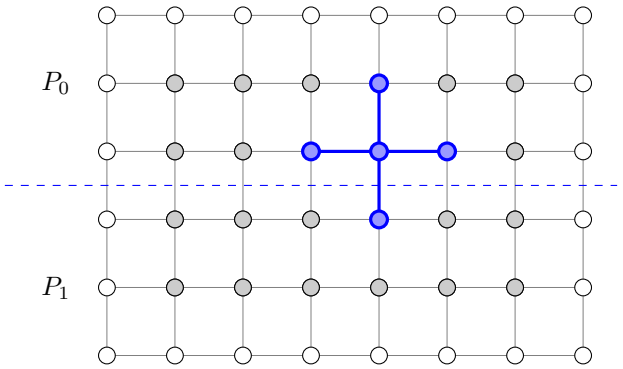

Figura 2: Malla repartida entre dos procesos, donde se indica un caso en que es necesaria la comunicación.