Multiplicación de una matriz por un vector – MPI

David Mejía Restrepo

C.c. 1039474824

# Introducción

A la hora de paralelizar un código existen múltiples opciones, por ejemplo, se puede paralelizar un código para ser ejecutado en un único computador utilizando los recursos del mismo o distribuir el las tareas en múltiples computadores (cluster) y para este ultimo caso podemos utilizar MPI, a continuación, se muestra la paralelización de un código utilizando MPI.

El código que se desea paralelizar se encarga de tomar una matriz nxn y multiplicarla por un vector de tamaño n iterativamente, por ejemplo, tenemos una matriz A de tamaño nxn y un vector X de tamaño n, se desea realizar la multiplicación de manera iterativa m veces, para esto se multiplica A por X y se obtiene un vector Y, para la siguiente iteración el vector X toma los valores del vector Y, de modo que la matriz A se multiplica por el resultado del vector Y de la iteración anterior.

# Método

Para la paralelización del programa se utilizó la librería de OpenMPI como herramienta para así ejecutar el código en un cluster de manera óptima.

Dado que cada nodo en el cluster ejecuta todo el código, se opto por que todos los nodos generaran el total de la información con la misma semilla aleatoria (seed) pero repartiendo la información que se encarga cada nodo de procesar, esto con el fin de evitar hacer muchos más llamados entre los nodos, que puedan tomar más tiempo de ejecución. El nodo encargado de tomar la información es el nodo 0, el cual después de recibida la información la reparte a los otros nodos mediante la función MPI\_Bcast(), luego de esto se genera una barrera para asegurar que todos tengan los datos antes de continuar con la ejecución del programa. Esto se evidencia en la figura 1.

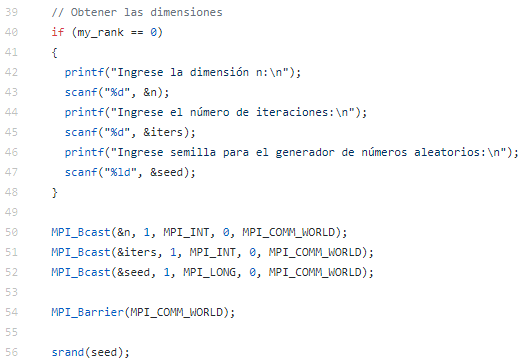


Figura 1. Sección de código donde se toman y distribuyen los datos a los múltiples nodos.

En el método mat\_vect\_mult() donde se realizó la mayor parte del procesamiento, se crearon múltiples variables para identificar algunos datos necesarios para identificar las secciones que analizara cada nodo de la matriz, específicamente se decide hacer la separación por filas, asignando la misma cantidad de filas a cada nodo, y luego uniendo los resultados de todos los nodos. Esto anterior se evidencia en la figura 2.

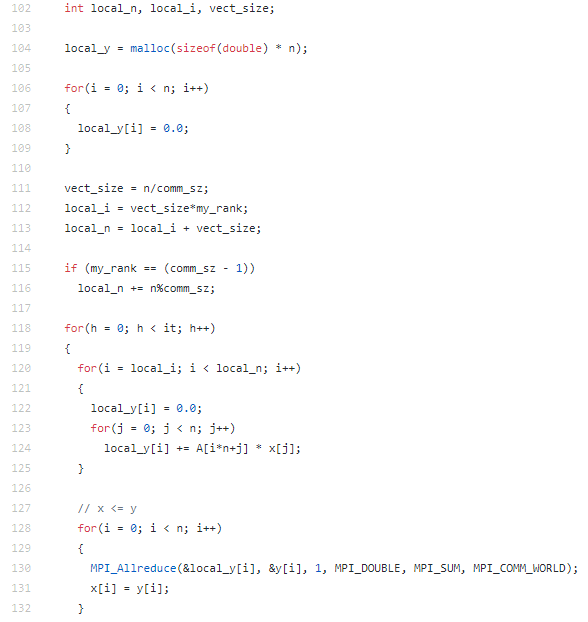


Figura 2. Código donde se paraleliza la multiplicación de la matriz por el vector de manera iterativa.

# Resultados

Se presenta actualmente solo resultados realizados en una máquina (mi equipo de cómputo personal), dado que para las fechas donde se tenía previsto tomar los resultados y hacer las pruebas finales en él cluster se han tenido complicaciones con el acceso a internet.

Se realizaron 30 ejecuciones del programa empleando desde 1, 2, 4 y 8 nodos, cada ejecución se realizó con un vector de 2000 (dos mil) posiciones (matriz de tamaño 2000x2000) y 100 iteraciones, los resultados promedios, SpeedUp y la Efficiency se presentan a continuación.

|  |  |  |  |
| --- | --- | --- | --- |
| Nodos | Tiempo | SpeedUp | Efficiency |
| 1 | 1.165 | 1 | 1 |
| 2 | 0.663 | 1.755 | 0.877 |
| 4 | 0.548 | 2.124 | 0.531 |
| 8 | 0.776 | 1.5 | 0.187 |

# Conclusiones

Se hace evidente que, incluso siendo ejecutado en una única máquina, en lugar de un cluster, se presentan eficiencias sobre 0.5 para 2 y 4 nodos, se infiere que empleando un cluster donde cada nodo corresponde a un computador independiente se obtengan mejores resultados tanto para 2 y 4, como para 8 que ya es un resultado donde sin un cluster se presenta una desmejora en el tiempo obtenido.