Programación Paralela

Práctica 2. Implementación en Memoria Distribuida de un Algoritmo Paralelo de Datos

Juan Francisco Díaz Moreno

3 de mayo de 2020

1. Índice

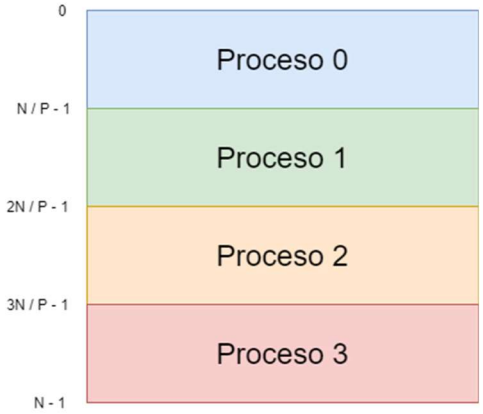
2. Introducción

En esta segunda práctica, se trabaja una nueva versión del algoritmo de Floyd, que encuentra los caminos mínimos en grafos dirigidos ponderados.

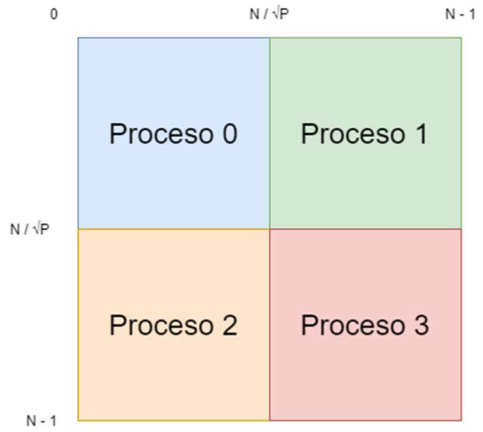
En este caso se estudia la utilidad de *Message Passing Interface* (MPI), una especificación para programación de paso de mensajes que permite comunicar datos entre procesos que ejecutan un mismo programa.

Para ello se van a analizar los beneficios de dos versiones distintas que utilizan MPI frente a una versión secuencial.

La primera versión, proporcionada por el profesor, divide la matriz de entrada (y de salida) en bloques de filas contiguas que son distribuidas entre los procesos de la siguiente manera, utilizando 4 procesos para el ejemplo:

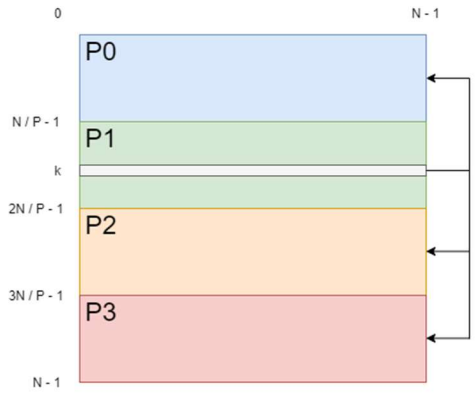


La segunda versión, desarrollada por el alumno, divide la matriz de entrada (y de salida) en bloques bidimensionales de elementos contiguos, siendo la división ahora de la siguiente forma:

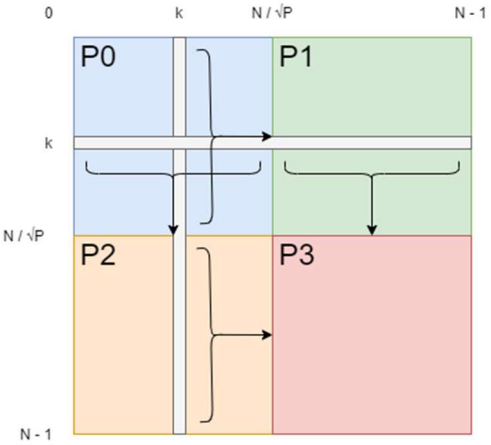


Hay que tener en cuenta que para el cálculo de cada $A_{i,j}$ el algoritmo de Floyd requiere también conocer los elementos $A_{i,k}$ y $A_{k,j}$, los cuales debido a las divisiones no existirán en las submatrices de los procesos para algunas iteraciones (k). Por ello, es necesaria la comunicación de estos valores entre los procesos.

En el primer caso, los procesos van a tener los valores de la columna k-ésima necesaria para el cálculo en todo momento. Sin embargo, solo van a poseer N / P de las filas k-esimas necesarias. Esto se resuelve haciendo que, en cada iteración k, el proceso que posea la fila k-ésima la envíe a los demás:



En el segundo caso esto se vuelve aún más complejo, ya que los procesos no siempre tendrán los valores de la columna k-ésima necesarios. Para resolverlo, cada proceso que posea un fragmento de la fila k-ésima tendrá que mandarlo a los procesos que se encuentren sobre y bajo él en la malla lógica de procesos. De la misma forma, un proceso que tenga un fragmento de la columna k-ésima lo mandará a los procesos que estén a su izquierda y a su derecha. El esquema sería el siguiente:



Para facilitar las comunicaciones en este segundo caso se han creado dos nuevos comunicadores, uno que comunica las columnas y otro que comunica las filas, de tal forma que, si un proceso quiere mandar su segmento de fila, con hacerlo por el comunicador vertical le llegaría sólo a los procesos interesados. Igual pasaría si un proceso quiere mandar su segmento de columna, lo haría por el comunicador horizontal y le llegaría a los procesos que lo necesitan.

3. Tablas de Tiempo

Vamos a representar en tablas los tiempos de ejecución y las ganancias en velocidad de cada una de las versiones del algoritmo paralelo frente al algoritmo secuencial.

En esta primera tabla se muestran los datos de la versión de bloques unidimensionales:

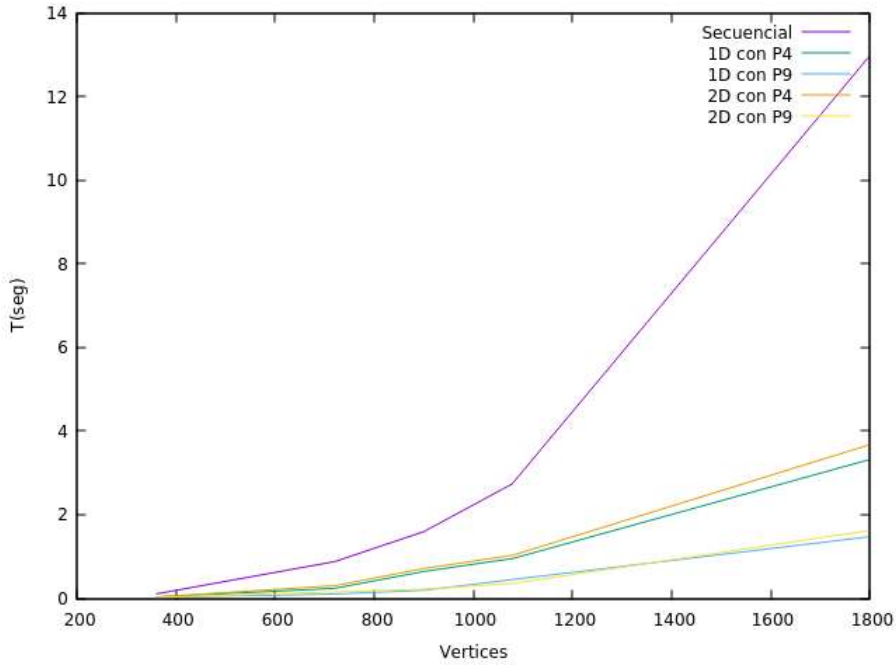
	T_{SEQ}	T_{P1D} 4P	S_{P1D} 4P	T_{P1D} 9P	S_{P1D} 9P
N = 360	0.10299	0.0287105	3.5872	0.029918	3.4424
N = 720	0.876859	0.238238	3.6806	0.102158	8.5833
N = 900	1.5948	0.642588	2.4818	0.19151	8.3275
N = 1080	2.74862	0.954085	2.8809	0.450769	6.0976
N = 1800	12.967	3.31952	3.9063	1.47256	11.3491

En la segunda tabla, se muestran los datos de la versión bidimensional:

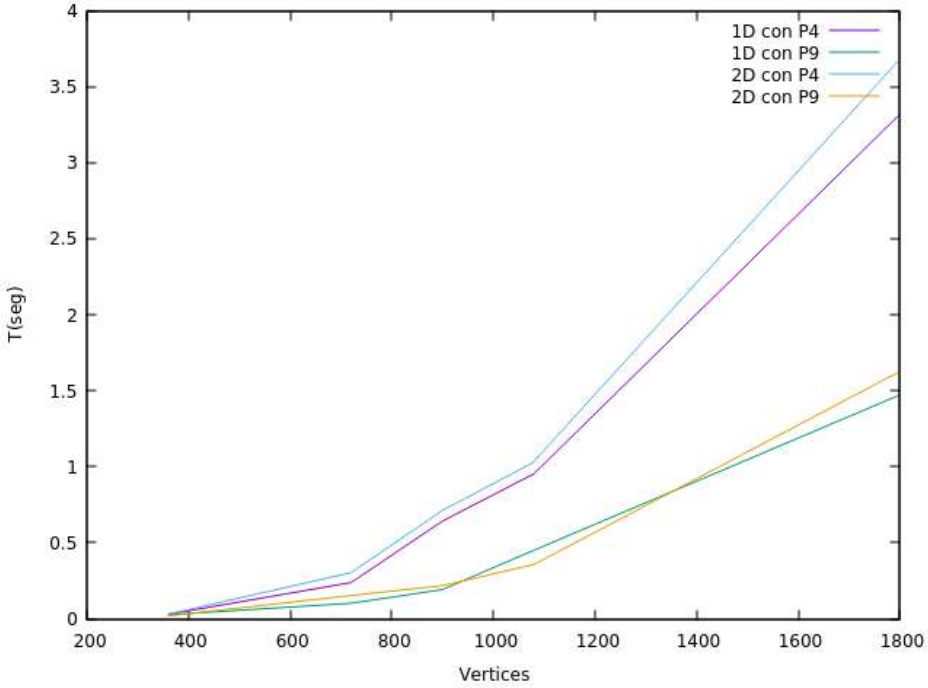
	T_{SEQ}	T_{P2D} 4P	S_{P2D} 4P	T_{P2D} 9P	S_{P2D} 9P
N = 360	0.10299	0.0293959	3.5032	0.0186234	5.5301
N = 720	0.876859	0.302929	2.8946	0.152134	5.7637
N = 900	1.5948	0.712623	2.2379	0.217455	7.3339
N = 1080	2.74862	1.03077	2.6666	0.357271	7.6934
N = 1800	12.967	3.67747	3.5261	1.62686	7.9705

4. Gráficas

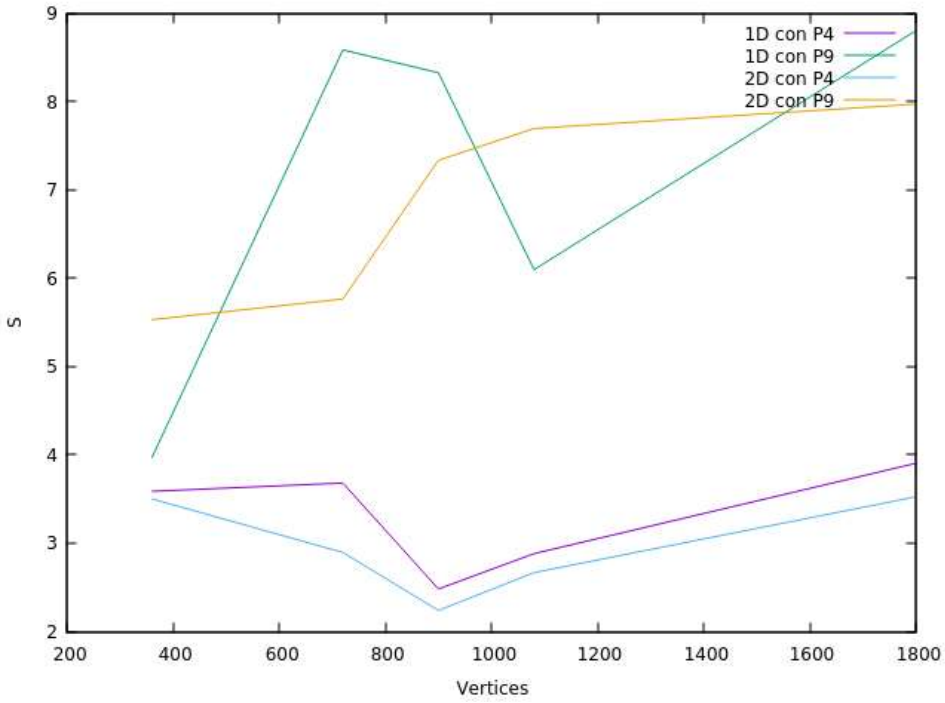
A partir de los datos obtenidos de las distintas ejecuciones de las tres versiones del programa (secuencial, paralela unidimensional y paralela bidimensional), se han elaborado las siguientes gráficas para representar de forma visual la comparativa:



En esta primera gráfica del tiempo de ejecución podemos ver, sobre todo en el tiempo del programa secuencial (morado), el carácter cúbico del algoritmo de Floyd. También, a simple vista percibimos cómo todas las versiones paralelas se ejecutan en un tiempo bastante menor que la versión secuencial, por lo que serían más eficientes.



Si nos centramos sólo en los tiempos de las versiones paralelas podemos observar claramente que los tiempos de ejecución de los programas con 9 procesos son menores que los de los programas con 4. Esto se puede comprobar también comparando sus ganancias frente a los tiempos del programa secuencial:



La peor correspondería a la versión bidimensional ejecutada por 4 procesos, seguida de cerca por la versión unidimensional ejecutada por 4 procesos. Las mejores son sin duda las versiones ejecutadas con 9 procesos, siendo la primera la versión unidimensional.

5. Conclusiones finales

Mediante la realización de esta práctica hemos podido comprobar de nuevo la utilidad de los algoritmos paralelos frente a los secuenciales, esta vez utilizando MPI.

Además, en este caso concreto, hemos notado también que las versiones unidimensionales son levemente mejores que las bidimensionales.

Esto puede deberse a que el tiempo de latencia en las versiones bidimensionales es mucho mayor al haber más comunicaciones que en las versiones unidimensionales, a pesar de que en estas se mandan menos datos por mensaje y hay más paralelismo en los cálculos.