Programación Paralela

Práctica 2: Implementación en memoria compartida y en memoria distribuida un algoritmo paralelo de datos

UNIVERSIDAD DE GRANADA

Realizado por: Daniel Díaz Pareja
Fecha: 08/05/2018
Universidad de Granada

Índice

1. Resumen.....3

2. Algoritmo de Floyd Paralelo 1. Descomposición unidimensional (por bloques de filas)....3

2.1 Pruebas y resultados de la implementación en OpenMP.....4

3. Algoritmo de Floyd Paralelo 2. Descomposición bidimensional (por bloques 2D).....5

3.1 Pruebas y resultados de la implementación en OpenMP.....6

3.2 Pruebas y resultados de la implementación en MPI.....9

4. Speedup comparativo de todas las versiones.....10

1. Resumen.

En esta práctica se aborda la implementación paralela en plataformas multiprocesador de memoria compartida y de memoria distribuida del algoritmo de Floyd para el cálculo de todos los caminos más cortos en un grafo etiquetado. Se plantearán dos versiones paralelas del algoritmo que difieren en el enfoque seguido para repartir los datos entre las tareas. Por simplicidad, se asume que las etiquetas de las aristas del Grafo de entrada son números enteros.

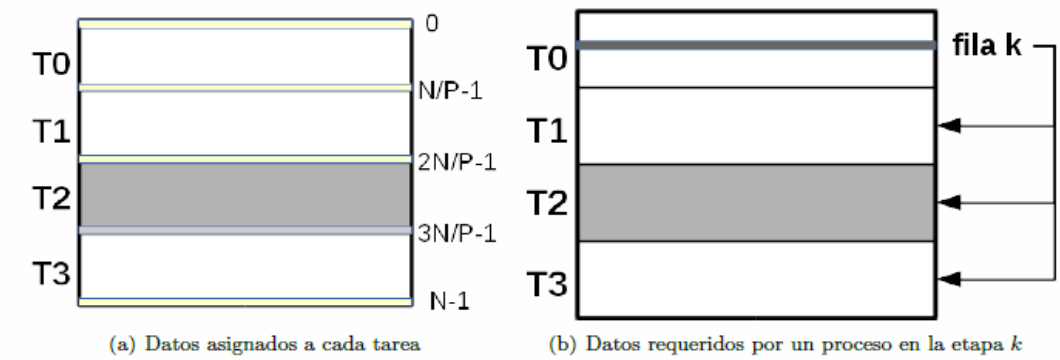
Las pruebas se han realizado en un equipo con un procesador Intel(R) Core(TM) i7-4770k @ 3.50Ghz (8 CPUs).

2. Algoritmo de Floyd Paralelo 1. Descomposición unidimensional (por bloques de filas)

Asumimos que el número de vértices N es múltiplo del número de tareas P.
En esta versión, cada tarea procesa un bloque contiguo de filas de la matriz I por lo que cada tarea procesa N=P filas de I.
Se podrán utilizar hasta N tareas como máximo. Cada tarea es responsable de una o más filas adyacentes de I y ejecutará el siguiente algoritmo:

```
procedure floyd paralelo 1
begin
I_{i,j} = A
  for k := 0 to N-1
    for i := local_i_start to local_i_start
      for j := 0 to N-1
        I^{k+1}_{i,j} = min{I^k_{i,j}, I^k_{i,k} + I^k_{k,j}}
  end;
```

En la iteración k, cada tarea, además de sus datos locales, necesita la fila k de I:



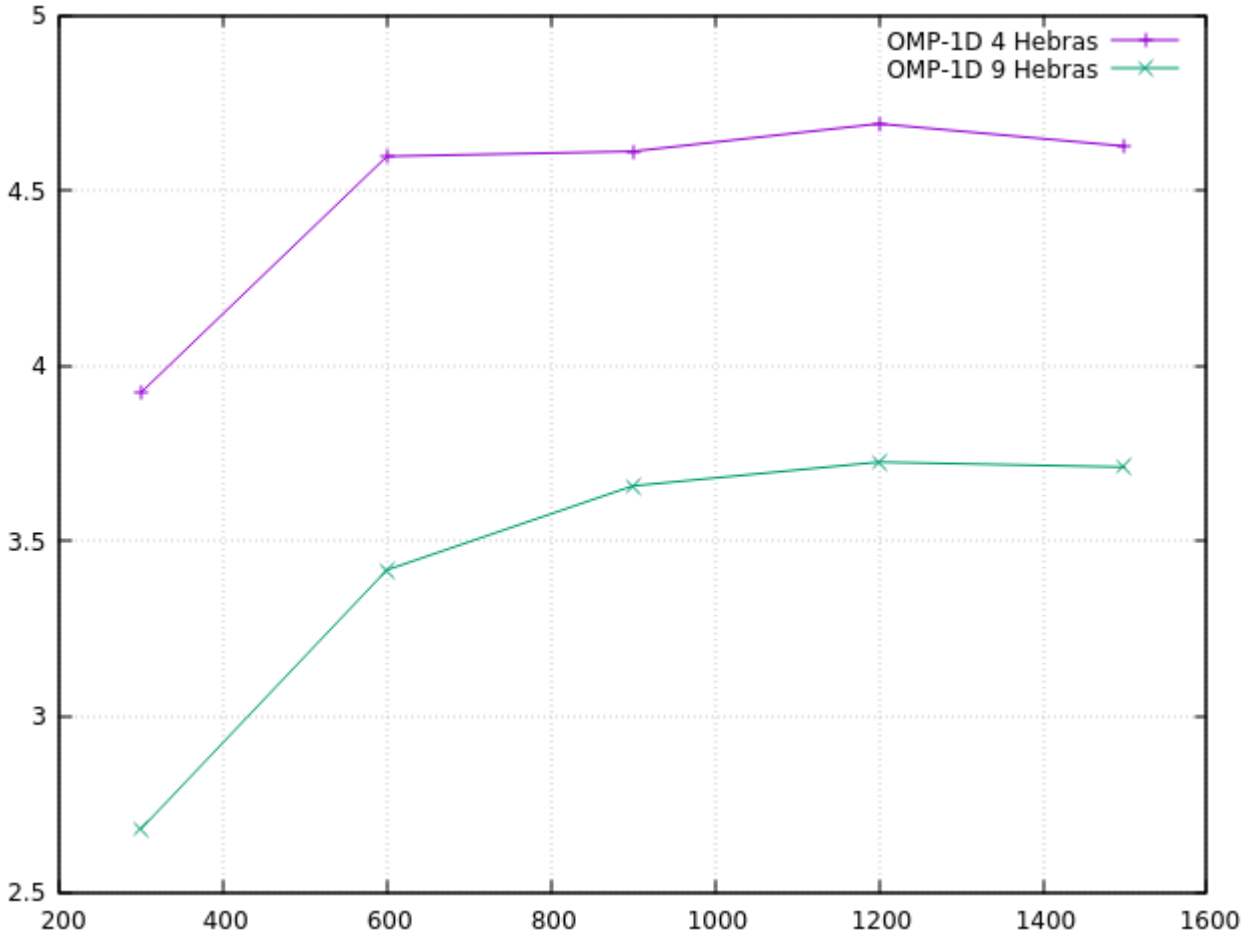
(a) Datos asignados a cada tarea (b) Datos requeridos por un proceso en la etapa k

2.1 Pruebas y resultados de la implementación en OpenMP.

Medidas:

	Tsec	TP (P = 4)	S (P = 4)	T_P (P = 9)	S (P = 9)
n = 300	0.127176	0.0324063	3.92442	0.0474527	2.68006
n = 600	0.994302	0.216276	4.59738	0.290893	3.4181
n = 900	3.35516	0.727514	4.61182	0.917208	3.65801
n = 1200	8.09614	1.72596	4.6908	2.17298	3.72582
n = 1500	15.9193	3.44072	4.62674	4.28907	3.7116

Speedup:

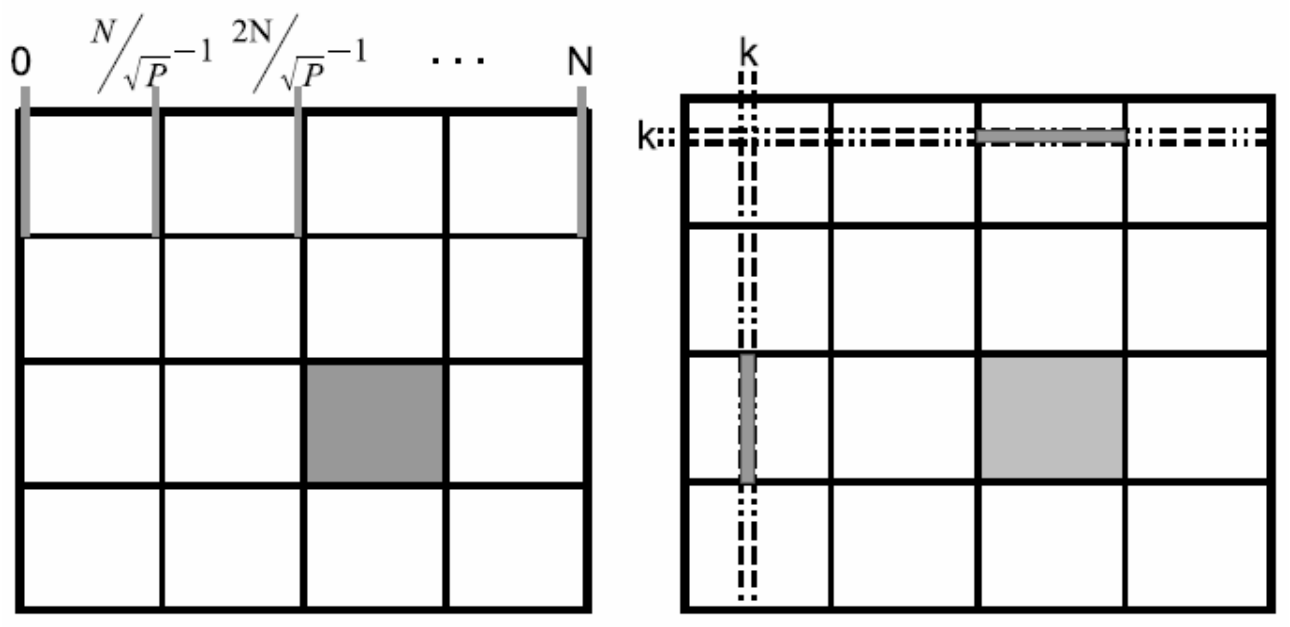


3. Algoritmo de Floyd Paralelo 2. Descomposición bidimensional (por bloques 2D)

Por simplicidad, se asume que el número de vértices N es múltiplo de la raíz del número de tareas
P.

Esta versión del algoritmo de Floyd utiliza un reparto por bloques bidimensionales de la matriz I entre las tareas, pudiendo utilizar hasta N2 procesos. Suponemos que las tareas se organizan lógicamente como una malla cuadrada con raiz de P tareas en cada fila y columna. Cada tarea trabaja con un bloque de N/(raiz de P) subfilas alineadas (cubren las mismas columnas contiguas) con N/(raiz de P) elementos cada uno.

En cada paso, además de los datos locales, cada tarea necesita N/(raiz de P) valores de dos procesos localizados en la misma fila y columna respectivamente:



(a) Datos asignados a cada tarea

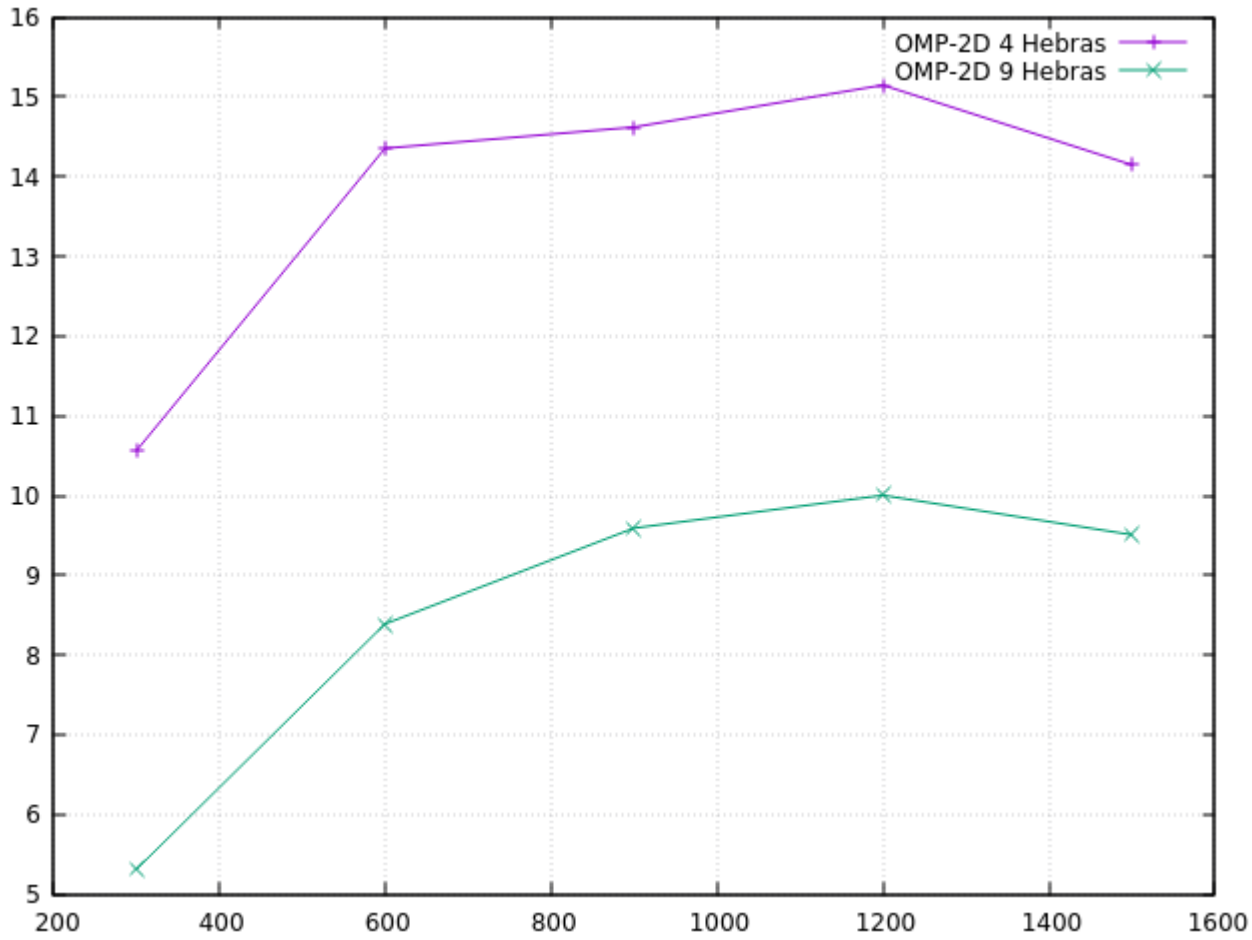
(b) Datos requeridos por una tarea en la etapa k con $P = 4$

3.1 Pruebas y resultados de la implementación en OpenMP.

Medidas:

	Tsec	TP (P = 4)	S (P = 4)	T_P (P = 9)	S (P = 9)
n = 300	0.127176	0.0120466	10.557	0.0239798	5.30346
n = 600	0.994302	0.0692612	14.3558	0.11858	8.38507
n = 900	3.35516	0.229464	14.6217	0.349899	9.58894
n = 1200	8.09614	0.534498	15.1472	0.809581	10.0004
n = 1500	15.9193	1.12513	14.1489	1.67438	9.50758

Speedup:

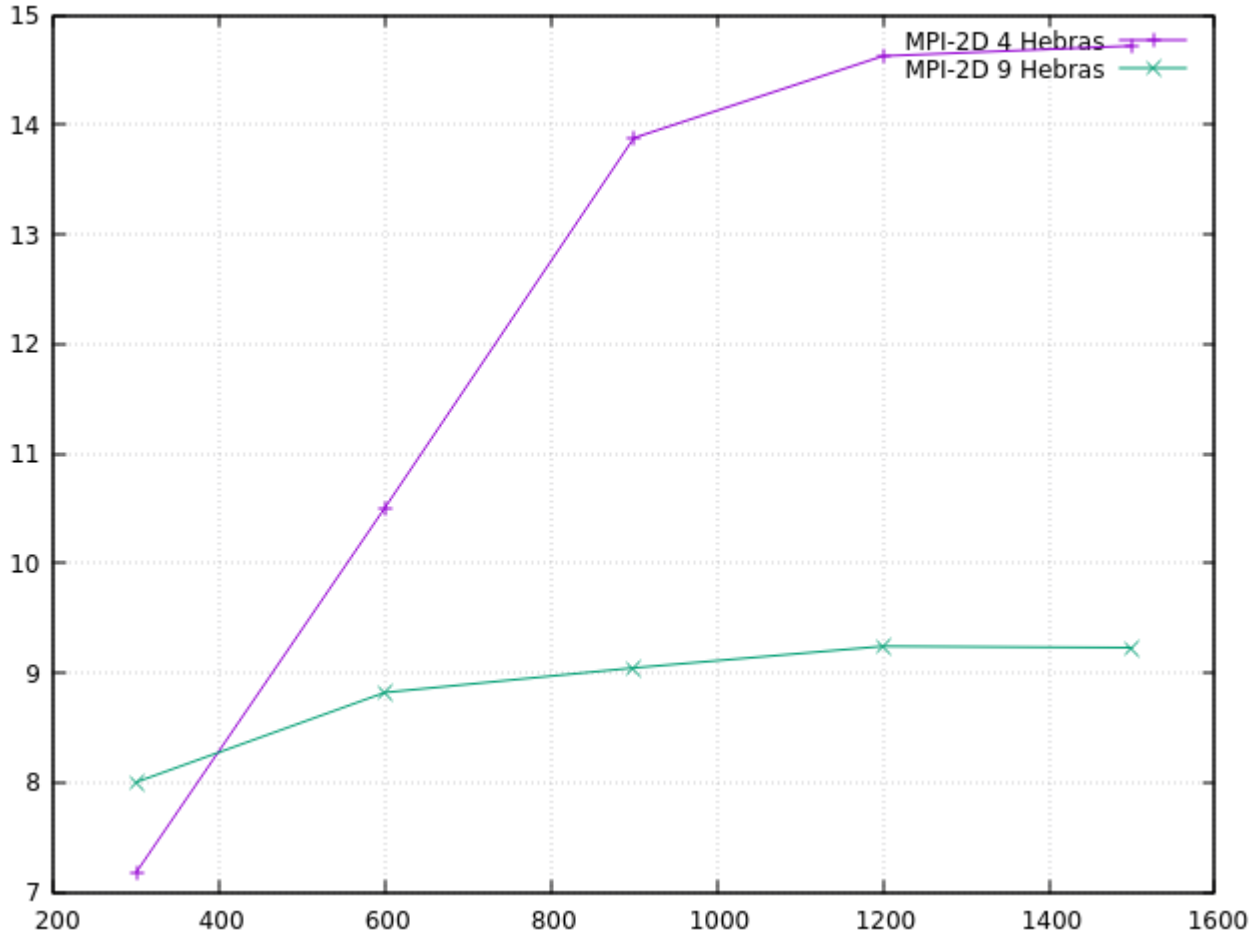


3.2 Pruebas y resultados de la implementación en MPI.

Medidas:

	Tsec	TP (P = 4)	S (P = 4)	T_P (P = 9)	S (P = 9)
n = 300	0.127176	0.0177188	7.17746	0.0158958	8.0006
n = 600	0.994302	0.0946412	10.50601	0.11273	8.82021
n = 900	3.35516	0.241723	13.8802	0.370964	9.04444
n = 1200	8.09614	0.553531	14.6264	0.876026	9.24189
n = 1500	15.9193	1.0814	14.72101	1.72488	9.22922

Speedup:



4. Speedup comparativo de todas las versiones.