

Informe

Trabajo Práctico Final

Sistemas Distribuidos y Paralelos

Morán Marcos – Abraham Aylen

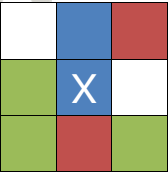
**Introducción**

Hemos simulado la permutación de la opinión política de una población. La simulación se realiza sobre una matriz, de modo que en un determinado momento las celdas se encuentran “pintadas” de un determinado color distinguiendo con ese color la opinión política de un ciudadano. Los colores posibles son cuatro: rojo, azul, verde y blanco. La matriz representa a la población y cada celda representa a una persona en dicha población, usamos diferentes tamaños de matriz en las diferentes simulaciones para poder observar el tiempo de cómputo de las simulaciones acorde a los tamaños del problema, donde los tamaños que usamos para la evaluación son 2500, 5000 y 7500.

Tenemos que en un paso de tiempo cada persona (celda) cambia su opinión política (color) a una nueva (el cambio puede ser que mantenga la misma opinión, cambiarla o no tener opinión política), basándose en la opinión de sus vecinos adyacentes, es decir, por cada celda formamos el vecindario de Moore. El reloj virtual se simula en base a 1000 iteraciones. Cada celda tiene la siguiente información:

* Opinión política: representada por los valores Azul, Verde, Rojo y Blanco. El colorblanco indica que no se tiene una opinión política formada.
* Convicción: variable que indica el porcentaje de convicción respecto a sus creencias políticas. Una persona con convicción cercana al 100% difícilmente cambie sus creencias, mientras que alguien con una convicción menor tendrá mayor tendencia a cambiar su ideología política, tomando como referencia a su entorno (tomamos porcentajes de convicción entre 15% y 90%).
* Grados de creencia: porcentaje para cada una de las ideologías políticas existentes, que indica el grado de creencia en la misma. Cada uno de los porcentajes de creencia puede variar a medida que transcurre el tiempo.

Las reglas para que una persona cambie de opinión en un determinado momento se detallan en el siguiente ejemplo. Si la situación en el tiempo actual T es la descripta por la siguientefigura, la celda central (X) cambiará su estado en el siguiente ciclo de tiempo T+1, mediante los siguientes pasos:



En este ejemplo la celda X tiene las siguientes características:

* Opinión política Azul.
* Porcentaje de convicción del 85%.
* Grado de creencia en Blanco del 5%, en Azul del 65%, en Verde del 10% y enRojo del 20%.

La celda X cuenta con una vecindad de 8 celdas, donde:

* = 0.25 de los vecinos tiene una opinión política blanco.
* = 0.125 de los vecinos tiene una opinión política azul.
* = 0.375 de los vecinos tiene una opinión política verde.
* = 0.25 de los vecinos tiene una opinión política roja.

Se debe calcular la diferencia entre las opiniones políticas de las celdas vecinas y la celdaX. Para cada opinión política, se calcula la diferencia como el grado de creencia de lavecindad menos el grado de creencia de la celda X. Por ejemplo:

* Diferencia de creencias en Blanco = 0.25 – 0.05 = 0.20
* Diferencia de creencias en Azul = 0.125 – 0.65 = -0.525
* Diferencia de creencias en Verde = 0.375 – 0.10 = 0.275
* Diferencia de creencias en Rojo = 0.25 – 0.20 = 0.05

Teniendo en cuenta el grado de convicción de la celda X (85%), se calcula el porcentajede cambio como 100% menos el grado de convicción. Para la celda X el porcentaje decambio es del 15% (0.15).

Los grados de creencia de la celda X para cada una de las opiniones políticas se ajustanen cada paso del tiempo de acuerdo a un valor de ajuste obtenido como la multiplicaciónentre la diferencia de creencias en cada opinión política y el porcentaje de cambio de lacelda X.

En nuestro ejemplo, los valores de ajuste resultantes son:

* Valor de Ajuste para Blanco = 0.20 0.15 = 0.03
* Valor de Ajuste para Azul = -0.525 0.15 = -0.07875
* Valor de Ajuste para Verde = 0.275 0.15 = 0.04125
* Valor de Ajuste para Rojo = 0.05 0.15 = 0.0075

Finalmente, los grados de creencia en cada opinión política de la celda X son actualizadossumando el correspondiente valor de ajuste. Los nuevos grados de creencia son:

* Grado de creencia en Blanco = 0.05+0.03 = 0.08
* Grado de creencia en Azul = 0.65 + (-0.07875) = 0.57125
* Grado de creencia en Verde = 0.10+0.04125 = 0.14125
* Grado de creencia en Rojo = 0.20+0.0075 = 0.2075

En el siguiente ciclo de tiempo T+1 la celda X tendrá una opinión política Azul, ya quees la opinión política con mayor grado de creencia.

Realizamos la simulación del problema de cuatro maneras diferentes, la primera consiste en un programa secuencial en lenguaje C, las otras dos versiones las implementamos de manera paralela usando paso de mensajes en memoria distribuida (MPI) con cuatro procesos, y en memoria compartida usando threads (OpenMP), usamos cuatro threads para esta última opción. Por último implementamos lasimulación de manera paralela híbrida usando paso de mensajes (MPI) y threads (OpenMP), con cuatro procesos de cuatro threads cada uno.

Para cada una de las cuatro implementaciones realizamos tres ejecuciones con los diferentes tamaños de matriz antes planteados con el fin de analizar el tiempo de cómputo de los programas que resuelven el problema planteado.

**Implementaciones**

**Programa secuencial**

La primera simulación se realiza en un programa secuencial escrito en lenguaje C, donde se mantienen dos matrices cuadradas con elementos de tipo “celda” (estructura que contiene los valores necesarios para realizar el cambio de política), en donde una matriz es copia de la otra. En el tiempo t una matriz actualiza los valores correspondientes al tiempo t+1 de cada una de las celdas, de acuerdo a las reglas anteriormente mencionadas mientras que en la otra matriz se mantienen los datos correspondientes al tiempo t sin modificarse. Finalmente antes de evaluar el cambio del tiempo t+1 al t+2 se actualizan los valores de la matriz original (tiempo t) con los valores de la copia (correspondientes al tiempo t+1). La simulación consiste en los cambios ocurridos en la matriz durante una iteración de 1000 ciclos sobre los tamaños de matriz 2500, 5000, y 7500, para ello se realizaron tres corridas y el tiempo obtenido para estas variantes fueron:

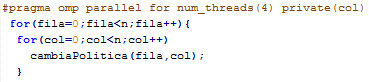
|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Corridas | Tiempo de matriz 2500 | | Tiempo de matriz 5000 | | Tiempo de matriz 7500 | | |
| 1000 ciclos | Segundos | Hra-min-seg | Segundos | Hra-min-seg | | Segundos | Hra-min-seg |
| 1 | 2726 | 00:45:26 | 10890 | 03:01:30 | | 24550 | 06:49:10 |
| 2 | 2725 | 00:45:25 | 10964 | 03:02:44 | | 24532 | 06:48:52 |
| 3 | 2722 | 00:45:22 | 10906 | 03:01:46 | | 24601 | 06:50:01 |
| Promedio | 2724 | 00:45:24 | 10920 | 03:02:00 | | 24561 | 06:49:21 |

La curva correspondiente a los tiempos promedios del programa secuencial es la siguiente:

**Programa paralelo en memoria compartida usando threads**

La segunda simulación se realiza en un programa paralelo con memoria compartida usando cuatro threads, y haciendo uso de la biblioteca OpenMP en C.

Para esta ocasión realizamos un análisis del programa secuencial para identificar qué tareas de las que se llevaban a cabo en la resolución del problema, podían paralelizarse. Como el mismo se centra básicamente en el cambio de la opinión política de los ciudadanos, el cual es representado en una doble iteración que muta cada una de las celdas de la matriz, concluimos paralelizar esta tarea del siguiente modo:



Es decir, paralelizamos sólo las filas de la matriz. Notar que la matriz se divide en cuatro horizontalmente, y cada porción es ejecutada por un solo thread. En este caso no hay inconvenientes con datos compartidos ya que las actualizaciones se realizan sobre otra matriz auxiliar.

A continuación se muestran los tiempos obtenidos para este caso:

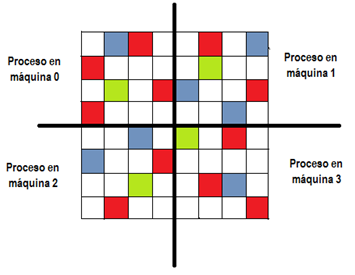
|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Corridas | Tiempo de matriz 2500 | | Tiempo de matriz 5000 | | Tiempo de matriz 7500 | | |
| 1000 ciclos | Segundos | Hra-min-seg | Segundos | Hra-min-seg | | Segundos | Hra-min-seg |
| 4threads |
| 1 | 842 | 0:32:26 | 3358 | 2:05:58 | | 18324 | 5:05:24 |
| 2 | 841 | 0:32:18 | 3358 | 2:05:41 | | 18332 | 5:05:32 |
| 3 | 843 | 0:32:29 | 3357 | 2:05:37 | | 18322 | 5:05:22 |
| Promedio | 842 | 0:32:24 | 3358 | 2:05:45 | | 18326 | 5:05:26 |

La curva correspondiente a los tiempos promedios del programa es la siguiente:

**Programa paralelo en memoria distribuida con paso de mensajes**

La tercera simulación se realiza en un programa paralelo con memoria distribuida haciendo uso de la biblioteca MPI en C.

Para esta ocasión se utilizaron cuatro computadoras distintas para resolver el problema en donde consideramos que cada computadora trabaja sobre una partición de la matriz (Figura 1).Cabe destacar que cada partición va a necesitar información de celdas que no se encuentran en la memoria de la computadora que lleva a cabo la ejecución, por lo que se requiere la comunicación a través del paso de mensajes para obtener esos valores de los demás procesos (Figura 2). Notar que cada computadora opera sobre la partición de la matriz de manera secuencial.



***Figura 1***

En nuestra implementación el procedimiento “completarMatriz”, es el encargado de realizar la comunicación punto a punto entre los procesos utilizando los “MPI\_Send” y “MPI\_Recv” para completar con los valores correctos las fronteras de la matriz de cada proceso. Una vez que cada proceso tiene su matriz en condiciones para operar, se llama al procedimiento “cambiaPolitica” donde cada proceso operara sobre su matriz de manera secuencial. Una vez finalizado el cambio de política en la matriz de cada proceso, todos ellos enviaran su matriz al proceso 0 a través de una comunicación colectiva usando un “MPI\_Gather”. De esta forma, el proceso 0 será el encargado de armar la matriz completa.



***Figura 2***

Los resultados obtenidos son los siguientes:

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Corridas | Tiempo de matriz 2500 | | Tiempo de matriz 5000 | | Tiempo de matriz 7500 | | |
| 1000 ciclos | Segundos | Hra-min-seg | Segundos | Hra-min-seg | | Segundos | Hra-min-seg |
| 1 | 1736 | 0:28:56 | 6918 | 1:55:18 | | 15553 | 4:19:13 |
| 2 | 1737 | 0:28:57 | 6917 | 1:55:17 | | 15591 | 4:19:51 |
| 3 | 1738 | 0:28:58 | 6920 | 1:55:20 | | 15561 | 4:19:21 |
| Promedio | 1737 | 0:28:57 | 6918 | 1:55:18 | | 15568 | 4:19:28 |

La curva correspondiente a los tiempos promedios del programa MPI es la siguiente:

**Programa paralelo en memoria distribuida con paso de mensajes y threads: Híbrido**

Para esta versión de la simulación combinamos el programa realizado en MPI con el programa OpenMP, de modo que paralelizamos el cómputo secuencial que realizamos en cada una de las máquinas.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Corridas | Tiempo de matriz 2500 | | Tiempo de matriz 5000 | | Tiempo de matriz 7500 | | |
| 1000 ciclos | Segundos | Hra-min-seg | Segundos | Hra-min-seg | | Segundos | Hra-min-seg |
| 4threads |
| 1 |  |  |  |  | |  |  |
| 2 |  |  |  |  | |  |  |
| 3 |  |  |  |  | |  |  |
| Promedio |  |  |  |  | |  |  |

La curva correspondiente a los tiempos promedios del programa híbrido es la siguiente:

Conclusión

En base a los tiempos obtenidos, analizamos el desempeño de cada una de las aplicaciones realizadas, tomando como medidas de performance el Speed Up y la eficiencia de cada uno de ellos. Primero se muestra una gráfica comparativa de los tiempos obtenidos de las diferentes optativas.

Para la medida de Speed Up, consideramos el tiempo del mejor secuencial para los diferentes valores de tamaño de matriz, estos tiempos son, para la matriz de 2500 celdas 2722 segundos, para la de 5000 el mejor tiempo fue 10890 segundos, y finalmente para la de 7500 fue de 24532 segundos. En donde el cálculo para la obtención del Speed Up es:

Entonces, los valores de Speed Up para las diferentes implementaciones son:

|  |  |  |  |
| --- | --- | --- | --- |
| Algoritmo  Paralelo | Speed Up con matriz de 2500 celdas | Speed Up con matriz de5000 celdas | Speed Up con matriz de7500 celdas |
| MPI |  |  |  |
| OpenMP-4th |  |  |  |
| Hibrido |  |  |  |

Ahora para obtener la eficiencia de los algoritmos calculamos:

Donde el costo del algoritmo paralelo es:

Entonces, los valores de eficiencia para las diferentes implementaciones son:

|  |  |  |  |
| --- | --- | --- | --- |
| Algoritmo  Paralelo | Eficiencia con matriz de 2500 celdas | Eficiencia con matriz de 5000 celdas | Eficiencia con matriz de 7500 celdas |
| MPI |  |  |  |
| OpenMP-4th |  |  |  |
| Hibrido |  |  |  |

En resumen:

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Implementación | Matriz 2500 | | Matriz 5000 | | Matriz 7500 | | |
|  | Speed Up | Eficiencia | Speed Up | Eficiencia | | Speed Up | Eficiencia |
| MPI | 1.5670 | 0.3917 | 1.5741 | 0.3935 | | 1.5758 | 0.3939 |
| OpenMP-4 threads | 1.4002 | 0.3500 | 1.4433 | 0.3608 | | 1.3386 | 0.3346 |
| Hibrido-4th |  |  |  |  | |  |  |

Si comparamos la eficiencia de estas implementaciones podemos observar:

Para el caso de OpenMP con dos threads notamos que el Speed Up es muy cercano a su límite superior, por lo que su aceleración es considerable. Además la eficiencia para esta implementación es muy buena, ya que es cercana a uno.

Para el caso de OpenMP con cuatro, seis y ocho threads observamos que el Speed Up no es muy bueno, y que mientras se agregan threads degrada su aceleración, esto puede deberse a la falta de trabajo, es decir, el trabajo es poco para el número de procesadores utilizados por lo que la aceleración es baja.

Para la implementación realizada en MPI notamos que su Speed Up es bastante bueno ya que es cercano a su límite superior (4), y que esto se logra con una amplia eficiencia.

Podemos concluir que existe una gran mejora en cuanto al tiempo de ejecución al hacer uso de algoritmos en paralelo, ya que existe una amplia diferencia en los tiempos obtenidos respecto al secuencial, pero también observamos que no sólo hay que tener en cuenta la aceleración del mismo si no también cuál es su eficiencia, es decir, cuánto nos cuesta obtener esa aceleración.