COMPUTACIÓN PARALELA
SIMULACIÓN DE EXTINCIÓN DE INCENDIOS (MPI)

Andrés Cabero Mata
Fernando Gigosos Ronda

Planteamiento inicial:

En un principio se planteó hacer la práctica utilizando las mejoras secuenciales ya conocidas y entendidas de la práctica hecha en OpenMP, añadiendo la estructura de halos por bandas, que parecía mejor opción que dividir la tabla en cuadros por simplicidad de programación y menor cantidad de comunicaciones entre procesos. Desde el primer momento se programó la paralelización con la idea de hacer halos de diez filas de tamaño debido a cómo funciona el algoritmo, donde en cada iteración se debería comunicar 10 veces el halo, reduciendo así a solo una comunicación. Este planteamiento no nos dió resultado así que al final reducimos el halo a una fila de tamaño y la realización de las 10 comunicaciones.

Apartado 3:

Antes de empezar la paralelización lo primero fue la creación de las regiones. Para ello se decide el tamaño de estas según el número de filas que tenga la región total. Se reparte equitativamente a todos los procesos el mismo número de filas de la región, y si sobran filas que no se puedan repartir equitativamente entre todos, se va añadiendo una fila a cada proceso hasta que estén todas asignadas. El orden de posición de las regiones en la región total es proporcional al número de proceso. Como hay que añadir los halos correspondientes a cada región se crean variables enteras que indican el tamaño de la región, y de la región con halos incluídos, así como la posición en la región completa. Se asigna ahora la tabla del tamaño necesario en memoria (tanto *surface* como *surfaceCopy del mismo tamaño)*. Las dos tablas se inicializan a cero completamente, incluyendo los halos.

Apartado 4:

Apartado 4.1:

Este apartado lo hacen todos los procesos por igual. Pensamos que una mejora podría haber sido dividir los focos por posición en la tabla y comunicar el resultado entre los procesos, pero no lo implementamos debido a que la mejora, si la hubiera, posiblemente no habría sido sustancial.

Apartado 4.2:

En este apartado es donde se hacen todas las comunicaciones de los halos en el programa. En el momento de hacerlo con diez filas por halo se hacía la comunicación entre los

```
if( rank==0 ){
  MPI_Send( &accessMat( surfaceCopy, p_rows-1, 0 ), columns, MPI_FLOAT, rank+1, 1, MPI_COMM_WORLD );
} else if( rank==nprocs-1 ) {
  MPI_Send( &accessMat( surfaceCopy, 1, 0 ), columns, MPI_FLOAT, rank-1, 2, MPI_COMM_WORLD );
} else {
  MPI_Send( &accessMat( surfaceCopy, p_rows, 0 ), columns, MPI_FLOAT, rank+1, 1, MPI_COMM_WORLD );
  MPI_Send( &accessMat( surfaceCopy, 1, 0 ), columns, MPI_FLOAT, rank-1, 2, MPI_COMM_WORLD );
}
// RECV DE HALOS
if( rank==0 ){
  MPI_Recv( &accessMat( surfaceCopy, p_rows, 0 ), columns, MPI_FLOAT, rank+1, 2, MPI_COMM_WORLD, MPI_STATUS_IGNORE );
} else if( rank==nprocs-1 ) {
  MPI_Recv( &accessMat( surfaceCopy, 0, 0 ), columns, MPI_FLOAT, rank-1, 1, MPI_COMM_WORLD, MPI_STATUS_IGNORE );
} else {
  MPI_Recv( &accessMat( surfaceCopy, p_rows+1, 0 ), columns, MPI_FLOAT, rank+1, 2, MPI_COMM_WORLD, MPI_STATUS_IGNORE );
  MPI_Recv( &accessMat( surfaceCopy, 0, 0 ), columns, MPI_FLOAT, rank-1, 1, MPI_COMM_WORLD, MPI_STATUS_IGNORE );
}
```

procesos una única vez justo antes del bucle con 10 pasos de propagación. Al hacer el cambio a solo una fila de halo se puso esta comunicación justo al principio del bucle. La comunicación consiste de un send para el primer y último proceso, y dos send para el resto. El primer y último proceso sólo necesitan hacer un envío ya que solo constan de un halo. De igual manera ocurre con las recepciones de los halos. Se envía la última (primera) fila dela región y se recibe en el halo del principio (final) de la tabla. Este bucle (step) sufrió la misma modificación según si era una iteración par o impar, para evitar hacer la copia entre surface y surfaceCopy y simplemente utilizar uno de los dos.

A pesar de que al final lo que conseguimos hacer se basó en un halo de tamaño 1, intentamos plantear una comunicación mucho más reducida con un halo de tamaño 10, el cual reproduciría los cálculos de 10 steps, sin necesidad de comunicación entre procesos

Según vimos y como nos confirmó un compañero más tarde, utilizar un halo de tamaño 10 cargaba demasiado las comunicaciones, ralentizandolas mas allá de que su uso supusiese una optimización

Echando un breve vistazo a las posibilidades, no tuvimos el tiempo necesario para reaccionar y utilizar un halo menor de 10 pero en todo caso mayor que 1, utilizando por supuesto una anidación del bucle step para que se realizasen los 10 necesarios, pero en grupos de 2 o 3 iteraciones, dependiendo del tamaño del halo

Asimismo, pero sin capacidad de llegar a implementarlo (debido a nuestra ofuscación con el pésimo funcionamiento del halo 10) tuvimos una idea de implementación de conjuntos de comunicación de procesos, restringidos mayoritariamente a zonas de la misma máquina.
Esto se descartó porque aún con esas restricciones, las dependencias entre iteraciones y cálculos requerían comunicación entre distintos ordenadores

Sin embargo, y también de la mano de la información dada por el mismo compañero, con el cual colaboramos en consultas y ocurrencias (Eduardo García Asensio), nos confirmó que era posible retener el cálculo de tooodo el programa no solamente restringido a unos ordenadores, sino solamente a uno, el hermes, con 64 núcleos
Así pues nuestra idea, aparte de no llegar a ser implementada, demostró ser parcialmente correcta, únicamente era necesario hacer trabajar a un ordenador, pobres de nosotros no darnos cuenta antes.

Consideramos estas dos pequeñas implementaciones, que no llegamos a realizar, como nuestra mejor parte en inspiración.
Por supuesto con la ayuda de las múltiples consultas realizadas al Profesor, tanto por nuestra parte, como por la de nuestro compañero Eduardo, con el cual mutuamente hemos colaborado, siempre con las picarescas pistas incompletas, que hacían de encontrar la solución adecuada, un desafío.

Apartado 4.2.1:

Este bucle actualiza la temperatura de los focos (los activados) en cada paso. Como solo es necesario que se activen los focos de la tabla que se utiliza (incluyendo a los halos), cada proceso actualiza únicamente los que se encuentran en esa región.

Apartado 4.2.2:

Este apartado ha sido eliminado al hacer el desdoble del bucle según el paso sea par o impar.

Apartado 4.2.3:

La única modificación que sufre este bucle hasta que posición hacen la actualización, lo cual resulta muy fácil utilizando las variables creadas al principio indicando el final de la tabla total que incluye el halo (f_total).

Apartado 4.2.4:

En este bucle además del cambio de recorrido de la tabla a solo la región real de cada proceso (sin halo), utilizando como base todas las mejoras secuenciales utilizadas en la práctica con OpenMP, añadimos la condición de que fuera el primer paso para ejecutar el bucle, ya que es el caso de mayor diferencia de temperaturas equilibrandose con cada paso siguiente.

Al terminar el bucle, la comprobación de los procesos de que la variable global_residual sea mayor que el límite se hace utilizando una reducción en la misma variable. Se utiliza la instrucción MPI_Allreduce (equivalente a MPI_Reduce seguido de MPI_Broadcast) en conjunto con MPI_IN_PLACE, para que no sea necesaria la creación de una nueva variable para el envío.

Apartado 4.3:

Este apartado no ha sufrido ninguna paralelización en absoluto. Creíamos que era necesario añadir una barrera al final para que no hubiera cambios de decisión al cambiar en el siguiente bucle el estado de los focos. Pero esto no es así ya que cada uno de ellos modifica también el estado de cada foco para cada equipo en el siguiente bucle. Este fallo es una confusión producida por pensar la programación para OpenMP y no MPI.

Apartado 4.4:

Este apartado, a la hora de paralelizar solo es necesario hacer que cada proceso modifique exclusivamente los puntos que afecten a las zonas de su región (sin halo). Para ello se comprueba si entra en la región con las variables de posición (i_row, f_row). Además también se añaden mejoras secuenciales, más allá de las ya hechas entonces; si el radio de influencia de un equipo nunca entra en la región del proceso, se salta al siguiente equipo.

Para finalizar el programa, pasamos los resultados obtenidos de los focos de las zonas de cada proceso y hacemos la reducción sumando, para asegurar que solo la información de los procesos es la enviada.

Bibliografía:

- Transparencias 1 a 4 de MPI, Grupo Trasgo, Departamento de Informática, Universidad de Valladolid