# ARQUITECTURA E INGENIERÍA DE COMPUTADORES. SIMULACIÓN DE UNA CACHÉ DE DATOS MEDIANTE MPI (ver. 0.4)

Guillermo González Talaván

18 de noviembre de 2009

## 1. Objetivo de la práctica

Realizar una simulación del comportamiento de una caché de datos con ayuda de la biblioteca de paso de mensajes MPI.

## 2. Descripción de la práctica

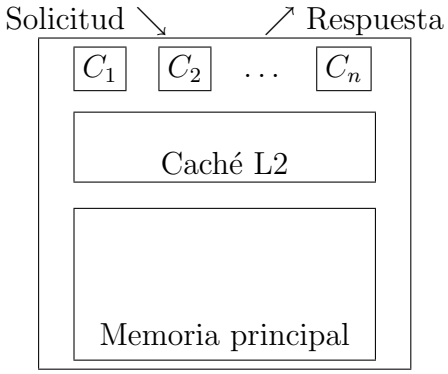
Esta práctica tiene dos partes, una estándar, que cada grupo deberá realizar obligatoriamente siguiendo las especificaciones aquí descritas, y una parte más libre, que el grupo deberá realizar opcionalmente de un modo original, según su iniciativa.

La práctica constará de una serie de procesos que realizarán en paralelo un cálculo. Este cálculo necesitará utilizar memoria. La memoria será simulada por otro proceso adicional que se encargará del módulo de simulación de memoria, gestión de la caché simulada y evaluación del rendimiento obtenido.

## 2.1. El proceso gestor de memoria

El programa dispondrá de una zona de memoria principal, una caché de datos independiente propia de cada proceso que participe en los cálculos y una caché común de nivel 2. La palabra que manejará la memoria será un entero de 32 bits. Todos los tipos de memoria (principal y cachés) serán simuladas por el proceso encargado del módulo de memoria. Los procesos calculadores, cada vez que necesiten acceder a memoria, tanto para lectura como para escritura, realizarán una petición al proceso de gestión de memoria acerca de una palabra de la memoria. Dicho proceso, siguiendo un esquema cliente-servidor, procesará la petición, llevará la gestión de las cachés, actualizará las estadísticas y devolverá el valor solicitado en el caso de que la petición hubiera sido de lectura de memoria. El sistema de memoria llevará cuenta del número de accesos a memoria para lectura y escritura, el número de aciertos de caché que se han producido y el tiempo de acceso a memoria efectivo, considerando la mejora debida a la presencia de la caché.

El esquema del gestor de memoria puede ser el siguiente:



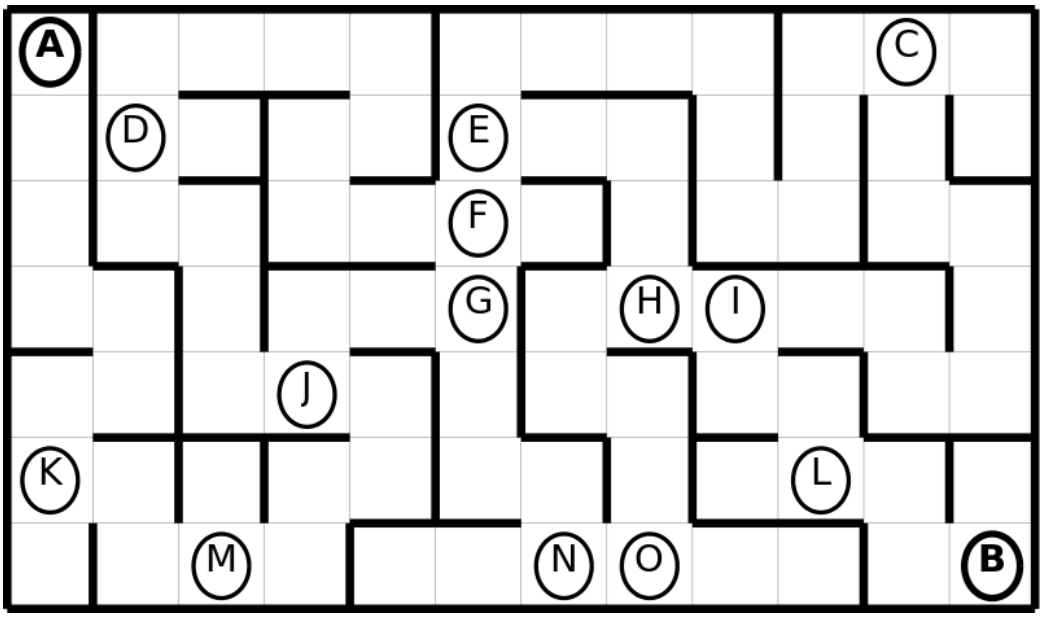

La solicitud constará de una dirección de memoria, el tipo de acceso (lectura o escritura) y el valor que se debe escribir (solo si se trata de una escritura). La respuesta consistirá en el valor leído de la memoria si de una lectura se trata. En el caso de las escrituras, si se opta por un modelo no confirmado, no será necesaria una respuesta del servidor. Si el modelo es confirmado, la respuesta, incluso vacía, sirve para confirmar que la escritura se ha realizado. En ningún momento tienen los procesos calculadores constancia de la existencia de las cachés ni podrán interferir en su funcionamiento. El mismo principio se aplica al proceso gestor de memoria respecto al funcionamiento de los procesos calculadores. Tan es así, que salvo porque no hemos especificado una interfaz de peticiones, cualquier módulo de memoria debería funcionar sin más con cualquier módulo de cálculo de los hechos en clase.

## 2.2. El cálculo que se debe realizar

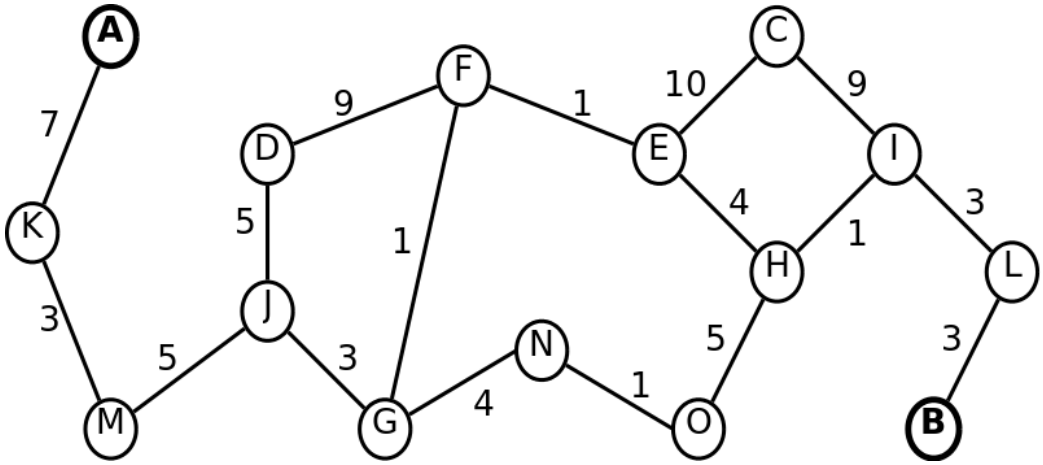
Los procesos calculadores se deben organizar para resolver de modo distribuido el problema del camino más corto mediante el algoritmo de Dijkstra.

Dadas un grafo no dirigido y dos de sus nodos, un nodo de partida y un nodo de destino, se debe encontrar el camino más corto que une dichos nodos. Cada arco del grafo tendrá asociada una distancia positiva. Se entiende por camino más corto aquel cuya suma de distancias de sus arcos es mínima.

Partamos de un ejemplo de aplicación. Para el siguiente laberinto encuéntrese el camino más corto desde la entrada (A) a la salida (B):

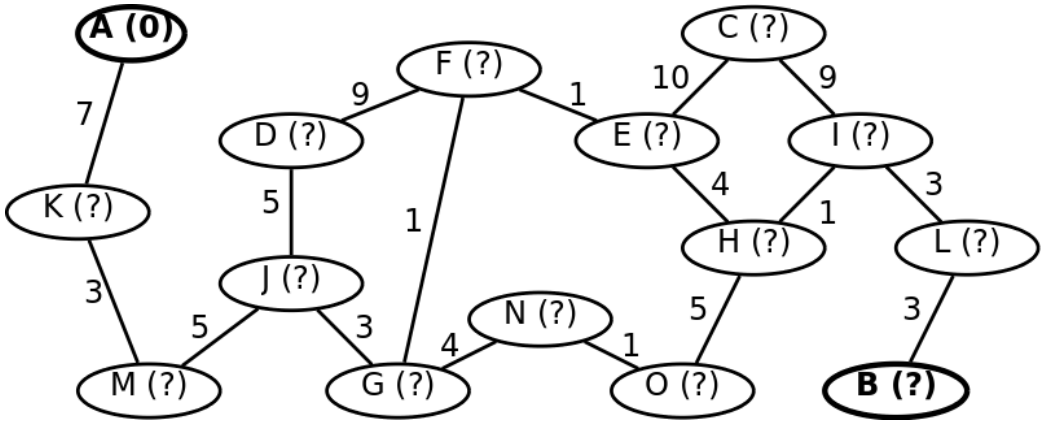



Primero debemos construir el grafo. Para simplificar hemos colocado nodos solo en aquellas casillas en las que efectivamente se puede tomar una decisión (tienen tres o cuatro salidas, o sea, una o ninguna pared). Son las casillas marcadas con las letras de la C a la O en la figura anterior. Este es el grafo resultante:

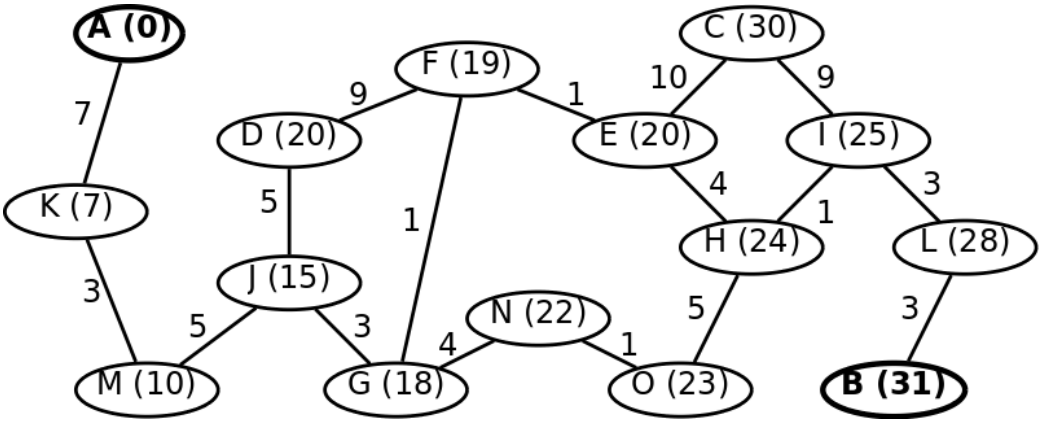



En el algoritmo de Dijkstra, los nodos pueden estar en estado "desconocido", "abierto" o "cerrado". Un nodo abierto es aquel del cual conocemos una distancia al nodo de partida, pero no conocemos una distancia para todos los que conectan con él. Un nodo cerrado será aquel del cual conocemos su distancia mínima al nodo de partida. Cualquier nodo que no esté ni abierto ni cerrado es que está en estado desconocido. Se procede del siguiente modo:

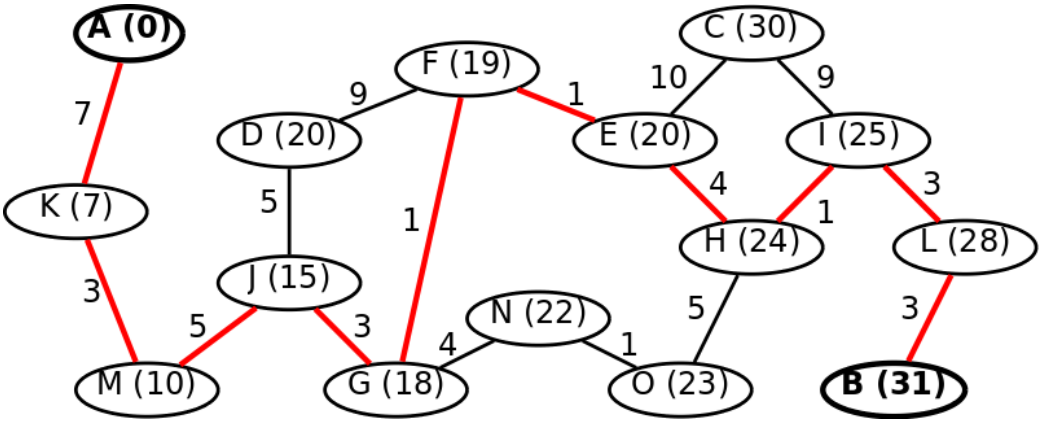
1) Se marcan todos los nodos como desconocidos
2) Se abre el nodo de partida y se le asigna una distancia de 0
3) Mientras no se cierre el nodo de destino, de todos los nodos abiertos, se elige el nodo $A$ de menor distancia. Para todos los nodos $B_i$ que conectan con $A$, si $B_i$ es desconocido, se abre y se le asigna como distancia la de $A$ más la distancia de $A$ a $B_i$. Si el nodo $B_i$ ya estuviera abierto, se le asigna como distancia la distancia de $A$ más la distancia de $A$ a $B_i$ solo si esta distancia es menor que la que ya tiene el propio nodo. Se cierra el nodo $A$ y se repite.

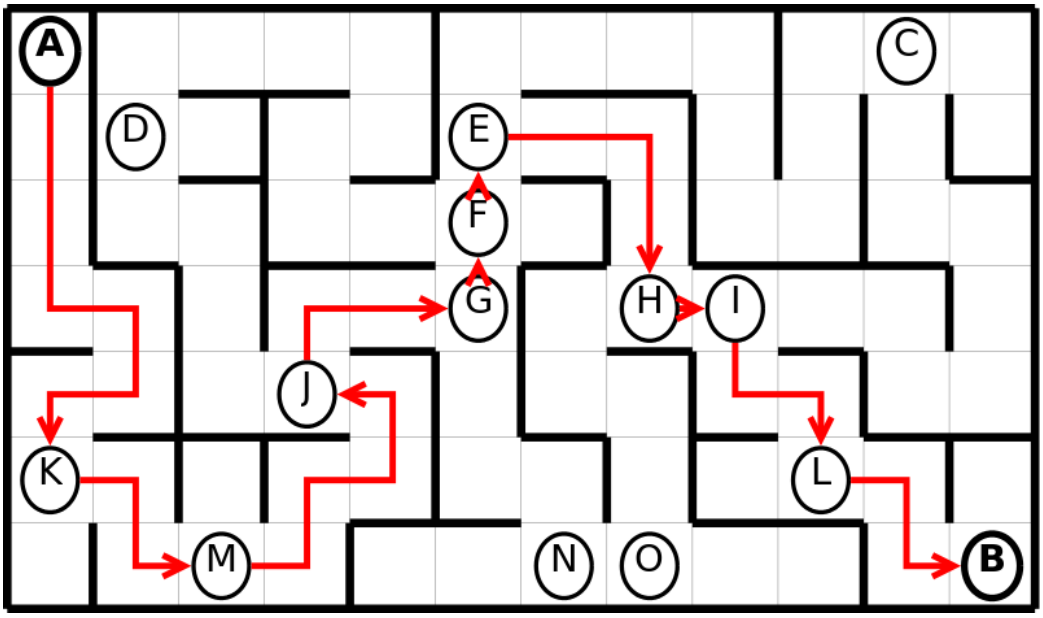


Después de cerrados todos los nodos necesarios, el grafo de ejemplo nos queda:




Para encontrar el camino más corto, se parte del nodo de destino hacia atrás hasta el nodo de partida eligiendo en cada paso la opción (o una de las opciones, si hay más de una) en la que la distancia del nodo actual menos la distancia marcada en el arco es igual a la distancia del nodo elegido. En nuestro caso:






Para plantear un problema el día de la defensa no se pondrá un laberinto, sino directamente un grafo. Por lo tanto, la entrada del problema será:

a) Grafo de distancias
b) El nodo de partida y el de llegada

El resultado del cálculo se dará por la pantalla y será el camino más corto entre el nodo de partida y el nodo de llegada y la distancia que hay entre ambos.

El grafo se proporcionará en un fichero de texto que el programa deberá leer. Primero vendrá una línea con el número de nodos. Luego, una matriz simétrica con diagonal nula con las distancias entre nodos conectados o 0 si es que no hay conexión entre ese par de nodos. Por ejemplo, el caso del grafo anterior sería:

```
15
 0  0  0  0  0  0  0  0  0  0  7  0  0  0  0
 0  0  0  0  0  0  0  0  0  0  0  3  0  0  0
 0  0  0  0 10  0  0  0  9  0  0  0  0  0  0
 0  0  0  0  0  9  0  0  0  5  0  0  0  0  0
 0  0 10  0  0  1  0  4  0  0  0  0  0  0  0
 0  0  0  9  1  0  1  0  0  0  0  0  0  0  0
 0  0  0  0  0  1  0  0  0  3  0  0  0  4  0
 0  0  0  0  4  0  0  0  1  0  0  0  0  0  5
 0  0  9  0  0  0  0  1  0  0  0  3  0  0  0
 0  0  0  5  0  0  3  0  0  0  0  0  5  0  0
 7  0  0  0  0  0  0  0  0  0  0  0  3  0  0
 0  3  0  0  0  0  0  0  3  0  0  0  0  0  0
 0  0  0  0  0  0  0  0  0  5  3  0  0  0  0
 0  0  0  0  0  0  4  0  0  0  0  0  0  0  1
 0  0  0  0  0  0  0  5  0  0  0  0  0  1  0
```

El nodo de partida será el primer nodo de la matriz y el de llegada, el segundo.

El proceso de rango uno leerá el fichero[1] y, mediante la memoria compartida simulada, pasará la información necesaria al resto de procesos.

# 3. Paralelización del algoritmo

La paralelización del algoritmo que haremos no es ni sencilla ni eficiente. Pero nos servirá al objetivo de la práctica.

Añadimos dos nuevos estados en los que puede estar el nodo: "calculándose" y "tentativa de cerrado". El procedimiento es similar al caso anterior salvo:

1) Mientras un proceso trabaja con un nodo está en el estado de "calculándose"
2) En cada momento solamente uno de los procesadores estará procesando el nodo de distancia mínima, que es el que se procesaría en el caso del algoritmo original. Si ese nodo está en "abierto", procede como el caso general, visitando los nodos vecinos no cerrados y lo cierra finalmente. Si está en "tentativa de cerrado",

[1]El proceso de rango cero se reservará para el gestor de memoria

lo cierra definitivamente.

3) El resto de procesadores puede estar con otros nodos que estaban abiertos. Procede cada uno a visitar los nodos con los que está conectado. Pasa ese nodo a "tentativa de cerrado" si la distancia del nodo no fue modificada por otro mientras estuvo en "calculándose". En caso contrario, permanece abierto.
4) Mientras se visitan los nodos conectados con un nodo $A$, en cualquiera de los casos anteriores, si al nodo visitado hay que cambiarle la distancia anotada, pasa a estar "abierto" si estaba en "tentativa de cerrado" o "desconocido". Si estaba en "calculándose" su estado se deja igual.

**Nota importante:** al tratarse de un cálculo distribuido hay que programarlo de modo que no se produzcan problemas de concurrencia (condiciones de carrera, etc.). Se cuenta con el auxilio de la técnica descrita en el apartado siguiente.

# 4. Restricciones

En general, los procesos únicamente pueden interactuar entre ellos mediante la memoria simulada. Podrán usar funciones estándar de C, pero sólo podrán hacer uso, para realizar la simulación, de dos variables enteras `a` y `b`, que serán globales y que hacen el papel de registros de la CPU. El resto de valores que necesiten habrán de residir en la memoria simulada. No podrán, por tanto usar directamente funciones MPI entre ellos.

Para sincronizarse, por ejemplo en el instante inicial al recoger los datos suministrados por el proceso de rango uno, se usará también solo la memoria simulada.

Para sincronizarse en la ejecución del algoritmo, no se podrá usar paso de mensajes. Se contará con una operación adicional sobre la memoria: un intercambio atómico. Se podrá enviar una petición al servidor para que, de un modo atómico lea lo que haya en una posición de memoria y escriba el valor que se le pasa. Se devolverá el valor leido previamente. Los tiempos en este caso serán los equivalentes a la suma de las dos operaciones por separado: la lectura y la escritura.

# 5. Especificaciones técnicas de la simulación de la memoria

- Las cachés de nivel 1 son de correspondencia directa
- Las cachés de nivel 1 usan *write back*
- El protocolo de coherencia será MESI
- El direccionamiento es de 32 bits y el tamaño de la palabra también es de 32 bits
- Se debe poder especificar en tiempo de compilación el tamaño de las cachés y de sus líneas
- La caché de nivel 2 usa *write through* y es completamente asociativa con algoritmo de reemplazo LRU.
- El tiempo de acceso a memoria L1 es 5ns, a la memoria L2 es 20 ns y a la memoria principal, 100ns. Considérese que un acierto de caché en escritura puede suponer accesos a otras memorias con el consiguiente sobrecoste en el tiempo de acceso.

# 6. Resultados de la simulación

- Para un determinado número de procesos y tamaño de las cachés, se obtendrá como resultado de la simulación el porcentaje de aciertos de cachés en lectura y escritura y el tiempo medio de acceso en lectura y escritura
- La parte creativa de la práctica consiste en hacer un pequeño estudio de viabilidad económica de una caché. Suponiendo unos costes relativos de memoria (100,50,1), evaluar para un problema concreto y variando el número de procesos un valor óptimo para el tamaño de la caché. Úsese para ello la función de evaluación tiempo·coste$^2$.

# 7. Consideraciones adicionales

Las prácticas se realizarán en parejas, salvo causa justificada para realizarlas individualmente (consultad primero).

Se abrirá un plazo en DIAWEB para formar grupos de prácticas y para depositar el código fuente. Permaneced atentos a la página web de la parte práctica de la asignatura:

`http://avellano.fis.usal.es/~gyermo/aeicc/`

Una vez superado el plazo de entrega, la práctica entregada no será modificada.

La práctica estará formada por dos ficheros fuente con extensión "`.c`": `memoria.c` y `calculo.c`. El primero se encargará de las funciones de gestión de memoria y generación de estadísticas. En el segundo se incluirá la función `main` y las funciones relacionadas con el cálculo. No se deberán usar más ficheros fuente, salvo los de cabeceras (.h) que, como es natural, no deben contener código.

Se entregará un listado del código fuente de la práctica junto con el estudio realizado. Se especificará el nombre de los autores de la práctica.

Para obtener una nota de 5, basta con que la práctica funcione correctamente en nogal o en el Linux de clase. Esta condición es inexcusable, incluso para aquellas personas que por razones de trabajo o de otra índole no pueden asistir a clase. A partir de ahí, y dependiendo de la calidad del trabajo realizado, la nota que se puede obtener es superior. Se valorará el trabajo adicional creativo de las prácticas.

Permaneced atentos a las versiones que pueden aparecer de este enunciado, para corregir posibles errores, por ejemplo.