SEPTIEMBRE 2013

Considere un sistema de ficheros con formato FAT32 y suponga que en ese sistema los ficheros a almacenar tienen una distribución uniforme con un tamaño medio de 100 KiB y tal que la agrupación es de x B. Conteste a las siguientes cuestiones sobre ese sistema:

a) Calcular, en función de x, el espacio de metainformación que necesita el fichero. A continuación,calcular, en función de x, el espacio perdido por fragmentación en el fichero.

b) Calcular el tamaño de agrupación que produce la menor perdida de espacio (metainformación mas fragmentación).

c) Repetir las preguntas a) y b) para el caso de un sistema UNIX clásico.

d) Supongamos que el sector es de 512 B, que el tiempo medio que se tarda en acceder a un sector es de x segundos y que el tiempo de leer cada sector es de y segundos. Calcular el tiempo que se tarda en leer completamente un fichero de 100 KiB totalmente fragmentado en los dos supuestos siguientes:agrupaciones de 1 KiB y de 4 KiB.

e) Repetir la pregunta anterior para el caso de que el fichero no tenga ninguna fragmentación.

f) ¿Se reduciría el tiempo de las dos preguntas anteriores si se utiliza un disco con cache integrada, que siga una política adecuada? Y ¿en el caso de utilizar cache el sistema operativo?

g) Al diseñar la gestión del espacio libre de un sistema de ficheros uno de los factores considerado es el espacio de disco necesario para soportar esta función. Para un disco con x B útiles, bloques de y B y agrupaciones de z B, calcular el espacio de disco necesario para esta función en los dos casos de utilizar mapa de bits y lista de recursos.

h) ¿Que otras ventajas o desventajas presenta cada una de estas alternativas?

SOLUCIÓN

a) Numero de entradas por fichero en la FAT: (100 KiB / x B).

Espacio perdido por cada fichero por la FAT: (100 KiB / x B)·4. Como el sistema mantiene dos FAT hay que multiplicar por dos, quedando $800 \cdot 2^{10} / x$

Espacio del directorio: 32 B.

Espacio perdido por por término medio por cada fichero es de 1/2 agrupación = x/2 B.

b) Espacio perdido total $z = 800 \cdot 2^{10} / x + x/2$.

Derivamos e igualamos a 0:

$z' = 0 = - 800 \cdot 2^{10} / x^2 + 1/2$.

$x^2 / 2 = 800 \cdot 2^{10}$.

$x^2 = 1600 \cdot 2^{10}$.

$x = 1280$ B.

La agrupación tiene que ser un número entero de sectores (se suele tomar una potencia de dos), por tanto, podremos elegir entre 1 y 2 KiB. Como 1 KB está más cerca del tamaño óptimo, lo seleccionaremos.

c) En el caso de UNIX por cada fichero tendremos:

- El nodo_i que ocupa 128 B, dado que es un sistema clásico.
- Un bit en el mapa de bits por cada agrupación. Como tendremos 100 KiB / x B agrupaciones, necesitamos $100 \cdot 2^{10} / x$ b.
- Los punteros de agrupaciones: Tendremos 100 KiB / x B agrupaciones. Suponiendo punteros de 4 B, dado que es un sistema clásico. Cada agrupación permite almacenar x / 4 B punteros de agrupación.

  Una primera observación es que la agrupación no puede ser menor que un sector del disco. Por tanto, para un sector típico de 512 B la agrupación mínima es de 512 B y, como máximo, necesitamos 100 KiB / 512 B = 200 punteros de agrupación. Además, en cada agrupación de 512 B caben 512 B/ 4B = 128 punteros de agrupación, por lo que como máximo vamos a necesitar dos agrupaciones de punteros.

  - Si la agrupación es muy grande (16 KiB o más) nos bastaría los 10 punteros de agrupación del nodo_i.

- Si la agrupación está entre 1 KiB y <16 KiB estamos en el caso de indirecto simple. El espacio necesario será 128 B + x B. Pudiendo tener hasta x / 4 punteros, que en el caso de x = 1 KiB son 256 punteros.
- Si la agrupación es de 512 B estamos en el caso de indirecto doble. Necesitamos dos agrupaciones de punteros (que nos da para 256 punteros, de los que necesitamos 200) más la agrupación de indirectos, por tanto se necesita: 128 B + 512 · 3 B.

◙ Por otro lado, como se ha visto en la pregunta a), cada fichero pierde por fragmentación x / 2 B.

En resumen tenemos los tres casos siguientes:

- Si la agrupación es 16 KiB o más la metainformación ocupa 128 B + $100 \cdot 2^{10}$ / x b + x / 2 B.
- Si la agrupación está entre 1 KiB y >16 KiB la metainformación ocupa 128 B + $100 \cdot 2^{10}$ / x b + x B + x / 2 B.
- Si la agrupación es de 512 B la metainformación ocupa 128 B + $100 \cdot 2^{10}$ / x b + 512 · 3 B + x / 2 B = 128 B + 200 b + 512 ·3 B + 256 B = 1945 B.

En los dos primeros casos derivamos e igualamos a 0.

- - $100 \cdot 2^{10}$ / x^2+ 1/2 = 0; x = 452,5 B. Este valor se sale del rango de aplicación de la expresión, por lo que el mínimo será el menor valor posible de x, es decir 16 KiB, por lo que el espacio ocupado será: 128 B + 7 b + 8 KiB = 8321 B.
- - $100 \cdot 2^{10}$ / x^2 + 3/2 = 0; x = 261,3 B. Este valor se sale del rango de aplicación de la expresión, por lo que el mínimo será el menor valor posible de x, es decir 1 KiB, por lo que el espacio ocupado será: 128 B + 100 b+ 1 KiB + 512 B = 1677 B.

Luego, la mejor solución, desde el punto de vista del espacio dedicado a metainformación, es utilizar agrupaciones de 1 KiB, que da una ocupación de 1677 B.

d) No vamos a considerar el tiempo invertido en acceder a la metainformación. Para un sistema UNIX, podría ser necesario llevar a memoria el o los bloques que contienen los punteros a las agrupaciones (para agrupaciones de 1 y 4 KiB sería un acceso según se ha visto en la pregunta anterior). Para el caso de la FAT serían los elementos de la misma que no estén ya en memoria.

- Agrupación de 1 KiB: hay 100 accesos y cada acceso lee 2 sectores consecutivos, puesto que una agrupación siempre es contigua, por lo que el tiempo es 100(x + 2y).
- Agrupación de 4 KiB: hay 100/4 = 25 accesos y cada acceso lee 2·4 sectores consecutivos, por lo que el tiempo es de 25(x + 8y).

Dado que un disco x es mucho mayor que y, el tiempo de lectura secuencial con agrupaciones de 4 KiB es prácticamente la cuarta parte que para agrupaciones de 1 KiB, debido a la menor fragmentación que existe para el caso de 4 KiB.

e) En el supuesto de que el SO optimice los accesos al disco, como todos los sectores son contiguos, lazará una sola operación de lectura con todos los sectores. El tiempo es igual para los dos casos, puesto que tenemos un solo acceso y la lectura de 200 sectores consecutivos, es decir: x+ 200y.

f) La utilidad de las caches solamente aparecería si la operación se repitiese. Como no se nos dice que la operación se repita, analizaremos el caso de un único acceso. Si el fichero está totalmente fragmentado, ninguna política de cache será efectiva, algunas podrían ser contraproducentes pues incluirían operaciones adicionales totalmente inútiles. Si el fichero está sin fragmentar y dado que se accede de forma secuencial podría ser de utilidad una técnica de lectura anticipada, pero si, como se indica en la pregunta anterior, el SO optimiza la lectura lanzando una sola operación de lectura de todo el fichero, tampoco serviría de nada, pudiendo ser contraproducente.

En el caso de utilizar cache el SO aparece otro efecto negativo, puesto que se utilizaría memoria principal con iformación que no se va a utilizar nunca más.

g) Para el mapa de bits hacen falta x / z bits = x / 8z B. Para la lista se pueden considerar dos implementaciones. Que los elementos de la lista se incluyan cada uno en la propia agrupación libre, por lo que solamente hace falta el puntero inicial. O que se reserve un espacio para dicha lista, espacio que deber ser capaz de albergar una lista con todas las agrupaciones, lo que ocurre cuando el disco está vacío. Por ello, haría falta un puntero por cada agrupación del disco es decir x / z punteros = 4x / z B. Vemos que se emplearía 32 veces más información que con el mapa de bits.

h) Además del espacio ocupado, tenemos las siguientes características:

Si el disco está muy lleno, para encontrar una agrupación libre hay que recorrer una gran parte del mapa de bits, sin embargo, en el caso de la lista es una operación inmediata, tomando el primero de la lista.

Tanto en el caso del mapa de bits como el de la lista encadenada, estas estructuras de información pueden estar copiadas en memoria, permitiendo una mayor rapidez en su gestión. Sin embargo la lista encadenada exigirá más accesos a disco para mantener la coherencia, especialmente si se construye en las propias agrupaciones libres.

El mapa de bits permite fácilmente seleccionar un conjunto de agrupaciones contiguas. En el caso de la lista habría o bien que mantener la lista ordenada o bien hacer una búsqueda hasta encontrar las agrupaciones contiguas.

En los sistemas tipo FAT. Dado que la FAT es ya una lista encadenada, para controlar el espacio libre se utiliza un fichero que contiene agrupaciones libres, sin necesitarse otros mecanismos adicionales.