**Problema 3.16** (febrero 2011)

*Sea un sistema con bloques de 4kB y direcciones de bloque físico de 48 bits. Consideraremos dos situaciones, la clásica y la basada en extents.*

*En la solución clásica el i_nodo dedica 78 B para referenciar los bloques del fichero, contemplando 10 directos, 1 indirecto, 1 doble indirecto y 1 triple indirecto.*

*La solución con extents tienen las siguientes características:*

- *El i_nodo incluye una cabecera que indica si contiene índices o extents. Pudiendo tener o 12 índices o 6 extents, según indique la cabecera.*
- *Los bloques indirectos dedicados a describir la estructura física del fichero tienen una cabecera de 48 bits que indica si contiene índices o extents.*
- *Se utilizan extents tipo ext4, compuesto por:*
  - *Bloque físico (48 bits)*
  - *Bloque lógico (32 bits)*
  - *Tamaño (15 bits)*
  - *En uso (1 bit)*

*a) Para el caso clásico, indicar cuantas direcciones de bloque puede contener un bloque indirecto. Para el caso con extents indicar cuantos índices puede contener un bloque indirecto y cuantos extents caben en un bloque.*

*b) Indicar cuál debería ser el tamaño mínimo de extent para garantizar que el espacio ocupado por la metainformación necesaria para describir la estructura física de los fichero en el caso de los extents sea igual o menor que para la solución clásica.*

*c) Sea un fichero está formado por los siguientes bloques físicos, en el orden que se indica:*

| |
|---|
| *34590* |
| *del 123.412 al 123.892* |
| *487873* |
| *del 73.625 al 74.589* |
| *3749* |
| *del 37.562 al 39.983* |
| *34865* |
| *234456* |

*Calcular el número de bloques de metainformación ocupados para describir la estructura física del fichero, indicando el tipo de contenido de cada uno de ellos para la solución clásica y para los extents.*

*d) Como sabemos las memorias flash tienen un número limitado de operaciones de escritura (que puede ser del orden del millón de escrituras), además, la operación de borrado es relativamente lenta y afecta a una cantidad importante de información (del orden del kB).*

*Estamos planteando el diseño de un dispositivo portátil que tiene las siguientes características.*

- *SO Linux.*
- *Alimentación por batería con aviso de batería con baja carga.*
- *512 MB de memoria RAM y 128 GB de memoria flash.*
- *Alta volatilidad de los ficheros almacenados (borrado y carga de ficheros).*
- *Acceso secuencial a grandes ficheros (cientos de MB) y acceso mixto secuencial y aleatorio al resto de los ficheros.*

*Suponiendo que se plantea usar un de sistema de ficheros clásico, indicar los problemas que se pueden encontrar.*

*Plantear el diseño de un sistema de ficheros adecuado para esta situación que trate de resolver los problemas anteriores.*

## SOLUCIÓN

**a**) Solución clásica

4 KB / 6 B = 682,66. Un bloque puede albergar 682 direcciones de bloque físico.

Solución con extents

Índices: como hay una cabecera de 6 B, caben 681 índices (uno menos que en el caso anterior)

Extents: Un extent requiere 12 B, como también hay una cabecera de 6 B, en un bloque caben (4 KB – 6 B) / 12 B = 340,83. Por lo que caben 340.

**b**) Para referenciar un extent hacen falta 12 B, mientras que para referenciar un bloque en un sistema clásico hacen falta 6 B. Si los extents son como mínimo de 2 bloques, la metainformación para el caso de extents ocupará lo mismo o menos que el caso clásico.

**c**) En total el fichero utiliza 1 + 481 + 1 + 965 +1 +2.422 + 1 +1 = 3.873 bloques.

La figura 3.3 muestra los 7 bloques de direcciones de bloque que son necesarios para la solución clásica. Nótese que del último bloque de indirectos solamente se utilizan 453 posiciones.

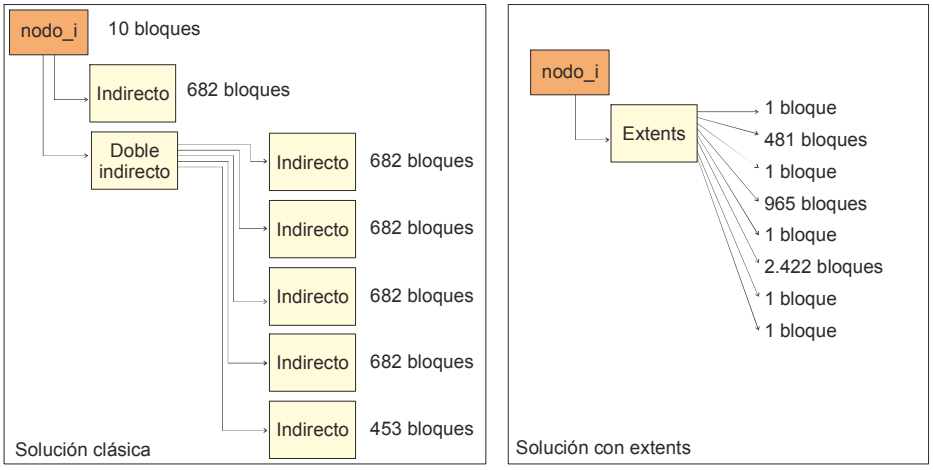


**Figura 3.3**

Por otro lado, solamente son necesarios 8 extents. Como en el nodo_i solamente caben 6, hay que utilizar un bloque de extents, del que solamente se utilizan 8 de las 340 posiciones.

Por otro lado, en ambos casos el nodo_i comparte un bloque con otros nodos_i y el espacio ocupado en el mapa de bits de bloques es casi igual. Observamos que la solución con extents requiere mucho menos espacio de metainformación.

**d**) El principal problema es el limitado número de escrituras que permite una memoria flash. Funciones como el soporte a la memoria virtual, la escritura inmediata (write-through) de la metainformación o incluso el volcado a disco los datos cada cierto tiempo producen una gran carga de escrituras, especialmente en los bloques que almacenan los mapas de bits y los nodos_i. Adicionalmente, los bloques que contienen los mapas de bits y los nodos-_i sufren un elevadísimo número de escrituras en comparación con el resto de los bloques.

El diseño de un sistema de ficheros para esta aplicación debe intentar dos cosas: reducir el número total de escrituras y repartir las escrituras de forma uniforme a lo largo de toda la memoria, evitando así que se "queme" pronto una parte de la misma.

El sistema se puede diseñar con técnicas de escritura diferida (write-back), tanto para los datos como los metadatos, maximizando el tiempo que la información se mantiene y modifica en la memoria RAM. Hay que destacar que esto no debe plantar ningún problema de fiabilidad, puesto que al contar con alarma de batería baja de carga se puede diseñar el sistema de forma que se haga dicha escritura diferida al pulsar la tecla de apagado o al generarse la mencionada alarma.

Además, se podría utilizar un esquema de tipo journaling cuyas escrituras se vayan distribuyendo con una política circular a lo largo de todo el espacio de la memoria flash.

Finalmente, el sistema no debería tener memoria virtual, lo cual, para un dispositivo portátil de uso específico no debería ser ningún problema.