Julio 2011 – Ficheros y entrada/salida

En el dimensionamiento de un sistema de ficheros tipo UNIX siempre se tiene que ajustar el tamaño que se asigna a los i-nodos en proporción a los bloques de datos. Es por ello que se propone un modelo de sistema de ficheros que permite asignar, cada bloque dinámicamente, al vector de i-nodos o al de bloques de datos.

Adicionalmente, otra de las estrategias habituales en la gestión de los recursos de almacenamiento es usar la parte del i-nodo que está asignada a los punteros (directos e indirectos) como espacio para almacenar el contenido del fichero para los casos en los cuales el fichero ocupa los pocos bytes que representa ese espacio. Dicha estrategia es la usada por el sistema de ficheros XFS, por ejemplo.

A lo largo de este ejercicio se propone analizar las cuatro combinaciones posibles derivadas de estas estrategias: UNIX-Estándar, UNIX-BloquesDinámicos, XFS-Estándar y XFS-BloquesDinámicos.

Datos comunes de los sistemas de ficheros:

- Tamaño del i-nodo (sólo información, sin punteros): 144 bytes
- Tamaño del puntero (a i-nodo o a bloque): 8 bytes
- Punteros dentro del i-nodo: directos 43, indirectos 3 (simple, doble y triple, respectivamente).
- Tamaño de bloque 4KB.
- Tamaño del sistema de ficheros: 2TB.

a) Calcule los siguientes parámetros:
   - Número de i-nodos por bloque.
   - Para el caso de un sistema de ficheros UNIX estándar, y considerando un ratio 1 a 1 (bloques/i-nodos), tamaño del bitmap de bloques y el de i-nodos.
   - Número de direcciones de bloque por bloque de indirección.
   - Para el caso de un sistema UNIX estándar, el tamaño del vector de i-nodos.
   - Tamaño máximo del fichero direccionable (de acuerdo con los punteros directos/indirectos del i-nodo).

b) Indique cuál es la limitación sobre el número máximo de ficheros en cada uno de las combinaciones de sistemas de ficheros y para los siguientes tamaños de fichero (rellene la tabla a continuación y **justifique los resultados**):

Combinaciones de Sistemas de Ficheros	Tamaño de los ficheros			
	0 bytes	64 bytes	3 Kbytes	2 Mbytes
UNIX-Estándar				
UNIX-BloquesDinámicos				
XFS-Estándar				
XFS-BloquesDinámicos				

c) Para el caso de los sistemas de ficheros XFS, si se usa, para los nombres de las entradas, un formato del tipo [1 byte para indicar el número de caracteres y tantos caracteres como indique dicho valor, hasta un máximo de 256], ¿hasta cuántos ficheros podría haber dentro de un directorio para que no se consumiese un bloque de datos (entrase todo dentro del i-nodo)? (Considere las entradas por defecto que hay en todo directorio).

d) ¿De qué forma en el caso de un sistema de ficheros XFS se sabe si los datos están almacenados dentro del i-nodo o se tienen que usar los punteros para acceder a bloques de datos? Indique la solución que requiera menos estructuras de información adicionales o que reutilice información ya disponible sobre el fichero.

e) En los modelos de Bloques Dinámicos, ¿de qué forma se podría gestionar que un bloque determinado del disco se encuentre asignado al vector de i-nodos o al de bloques y si está libre o no? Indique el tamaño en KB ocupado por las estructuras que necesite.

SOLUCIÓN

a) 1,5 puntos

En primer lugar es necesario determinar el tamaño del i-nodo (contando los punteros):
i-nodo (sin punteros) 144 bytes + (8 bytes/puntero x 43+3 punteros) = 512 bytes

- Número de i-nodos por bloque: 4KB/bloque / 512 bytes/i-nodo = 8 i-nodos/bloque

Si consideramos el ratio 1-a-1 entre bloques de datos e i-nodos, debemos computar cual es (aproximadamente) el número de bloques del disco: 2 TB / 4KB/bloque = 2^{41}bytes / 2^{12}byes/bloque = 2^{29} bloques (de datos).
Con el ratio 1-a-1, debe haber también 2^{29} i-nodos

- El tamaño de los bitmap de i-nodos (el de bloque de datos será exactamente igual), es decir un bit por elemento: 2^{29} bits / 8 bits/byte=2^{29} bits / 2^3 bits/byte=2^{26}bytes=64MB

- Número de direcciones de bloque por bloque de indirección: 4KB/bloque / 8 bytes/dirección = 512 direcciones/bloque

- El tamaño del vector de i-nodos es 2^{29} i-nodos x 512 bytes/i-nodo = 2^{29} i-nodos x 2^9 bytes/i-nodo=2^{38} bytes = 256GB.

Para determinar el tamaño máximo del fichero direccionable debemos considerar lo direccionable:

- Por los punteros directos: 43 punteros x 4KB/puntero = 172KB
- Por un puntero indirecto simple: 512 direcciones/bloque x 4KB/dirección=2^9 direcciones/Bloque x 2^{12}bytes/dirección=2^{21}bytes=2MB
- Por un puntero indirecto doble: 512x512 direcciones/bloque x 4KB/dirección=2^{18} direcciones/Bloque x 2^{12}bytes/dirección=2^{30}bytes=1GB
- Por un puntero indirecto triple: 512x512x512 direcciones/bloque x 4KB/dirección=2^{27} direcciones/Bloque x 2^{12}bytes/dirección=2^{39}bytes=512GB
    - Fichero máximo direccionable=172KB+2MB+1GB+512GB

b) 3,5 puntos

Vamos a comenzar considerando el tamaño (en bloques de disco y otras estructuras de cada uno de los tamaños de los ficheros):

- Fichero de 0 bytes: Consume 1 i-nodo y ningún bloque de datos.
- Fichero de 64 bytes: Consume 1 i-nodo y 1 bloque de datos (para el caso de los sistemas de ficheros UNIX estándar). En el caso de XFS, como los 64 bytes entran en el i-nodo (el espacio ocupado por los punteros es de 46 direcciones x 8 bytes=268 bytes) todo entra en el i-nodo (no se ocupa ningún bloque de datos).
- Fichero de 3KB: Consume 1 i-nodo y 1 bloque de datos (en todos los casos: UNIX y XFS).
- Fichero de 2MB: Consume 1 i-nodo y 2MB / 4 KB/bloque=512 bloque; a los que hay que sumarle 1 bloque más porque se usa un bloque indirecto.

La segunda de las consideraciones importantes es que, en el caso de los bloques no dinámicos, tenemos 229 elementos (bloques de datos y de i-nodos) y nuestra limitación es cuando se terminen cualquiera de estas estructuras. En el caso de los bloques dinámicos podemos asumir que si se necesitan más bloques de datos, o más i-nodos se asignarán dinámicamente a la estructura que corresponda.

Ficheros de 0 bytes:

- UNIX-Estándar=2^{29} ficheros (hasta que se terminan los i-nodos).
- UNIX-BloquesDinámicos=2TB / 512bytes/i-nodo= 2^{41} bytes / 2^9 bytes/inodo=2^{32} i-nodos → es decir 2^{32} ficheros (a 1 por i-nodos, representaría que el disco entero está ocupado por un enorme vector de i-nodos).
- XFS-Estándar=2^{29} ficheros (igual que el caso UNIX).

- XFS-BloquesDinámicos=2^{32} ficheros (igual que el caso UNIX).

Ficheros de 64 bytes:

- UNIX-Estándar=2^{29} ficheros (hasta que se terminan los i-nodos y los bloques de datos, que será a la vez).
- UNIX-BloquesDinámicos=2^{29} ficheros (es igual que el caso anterior, porque si se van ocupando 1 i-nodo y 1 bloque a la vez, al final se cumple el ratio 1-a-1 que es el caso del reparto no dinámico).
- XFS-Estándar=2^{29} ficheros (igual que el caso análogo de 0 bytes, ya que al ser el fichero tan pequeño entra en el i-nodo y no se consumen bloques de datos).
- XFS-BloquesDinámicos=2^{32} ficheros (igual que el caso análogo de 0 bytes, ya que al ser el fichero tan pequeño entra en el i-nodo y no se consumen bloques de datos).

Ficheros de 3KB:

- UNIX-Estándar=2^{29} ficheros (igual que el caso análogo de 64 bytes, hasta que se terminan los i-nodos y los bloques de datos, que será a la vez).
- UNIX-BloquesDinámicos=2^{29} ficheros (igual que el caso análogo de 64 bytes, que es igual que el caso anterior, porque si se van ocupando 1 i-nodo y 1 bloque a la vez, al final se cumple el ratio 1-a-1 que es el caso del reparto no dinámico).
- XFS-Estándar=2^{29} ficheros (igual que el caso de UNIX-Estándar, se ocupa 1 i-nodo y 1 bloque).
- XFS-BloquesDinámicos=2^{29} ficheros igual que el caso de UNIX-BloquesDinámicos, se ocupa 1 i-nodo y 1 bloque).

Ficheros de 2MB:

- UNIX-Estándar=2^{29} bloques de datos / 513 bloques de datos/fichero ≈ 2^{20} ficheros (aquí el límite es el número de bloques de datos disponibles).
- UNIX-BloquesDinámicos: Aquí el cálculo se puede hacer de una forma diferente, cada fichero ocupa 513 bloques de datos + 1 i-nodo. Como hay 8 i-nodos por bloque, 1 i-nodo representa 1/8 bloques=0,125 bloques. En su ocupación óptima un sistema de bloques dinámicos podrá incluir tantos ficheros como el número de bloques disponibles dividido entre 513,125, es decir=2^{29} bloques/513,125≈ 2^{20}.
  Aunque el número está en el mismo orden de magnitud del anterior, y auqnue parezca que es algo menor (al dividir por un número ligeramente mayor) no sería así. El cálculo del apartado anterior es una aproximación, ya que tomamos todos los bloques del disco, sin contar los 256GB del vector de i-nodos (que en el caso de ficheros de 2MB estaría parcialmente vacío).
  Para hacernos una idea de la diferencia, podemos estimar cuánto de esos 256GB estarían ocupados, que sería (aproximadamente) 1/512 del espacio, es decir menos del 0,2%. Por lo tanto, en el caso de un sistema de ficheros de bloques dinámicos, esos 256GB de vector de i-nodos estarían ocupados por ficheros útiles, es decir unos 128.000 ficheros más. Puede parecer mucho, pero teniendo en cuenta que estamos hablando de órdenes de magnitud de 2^{20} (millones de ficheros) no es mucho más.
- XFS-Estándar ≈ 2^{20} ficheros (igual que en el caso de UNIX-Estándar).
- XFS-BloquesDinámicos ≈ 2^{20} ficheros (igual que en el caso de UNIX-BloquesDinámicos).

Combinaciones de Sistemas de Ficheros	Tamaño de los ficheros			
	0 bytes	64 bytes	3 Kbytes	2 Mbytes
UNIX-Estándar	2^{29}	2^{29}	2^{29}	$\approx 2^{20}$ (A)
UNIX-BloquesDinámicos	2^{32}	2^{29}	2^{29}	$\approx 2^{20}$ (B)
XFS-Estándar	2^{29}	2^{29}	2^{29}	$\approx 2^{20}$ (A)
XFS-BloquesDinámicos	2^{32}	2^{32}	2^{29}	$\approx 2^{20}$ (B)

(B) será mayor que (A) en unos 128.000 ficheros.

c) 1,5 puntos

En XFS no se consume el bloque de datos, si el contenido del fichero (en este caso del directorio) es menor de 46x8 bytes = 368 bytes.

El espacio mínimo que ocuparía una entrada completa de directorio sería:

- 1 byte para indicar el número de caracteres de la entrada.
- 1 carácter (byte) con el nombre (del tipo "a").
- 8 bytes de dirección (puntero al i-nodo correspondiente).

Número de entradas 368 bytes / 10 bytes por entrada=36 entradas. Si quitamos las dos entradas estándar del directorio (la entrada "." y la ".."), entonces tenemos 34 entradas. Entre letras mayúsculas y minúsculas, hay más de 34 caracteres, por lo tanto la suposición de que con 1 carácter vale para todos los nombres de entradas, sería correcta.

d) 1,5 puntos

La solución más eficiente es la de utilizar el campo de tamaño del fichero, que ya aparece como información del i-nodo. Si dicho valor es menor que 368 bytes, entonces entra todo dentro del i-nodo, en otro caso, se empiezan a consumir bloques.

e) 2 puntos

Hay varias alternativas de diseño y algunas serán mejores que otras dependiendo de cuáles sean las operaciones más habituales, ya que funcionarán mejor buscando entradas libres o liberando el espacio. Aquí vamos a comentar una de las posibles soluciones:

Podemos definir 3 bitmaps diferentes:

- El primero que indica si un bloque está asignado al vector de i-nodos o al de bloques. Este bitmap debe tener un tamaño suficiente para tener un bit por bloque: 2^{29} bits / 8 bits/byte=2^{29} bits / 2^3 bits/byte=2^{26}bytes=64MB
- El segundo será el de i-nodos (que indica libres u ocupados), que a diferencia del caso del sistema de bloques no dinámicos se tiene que dimensionar para que pueda representar todos los i-nodos que podría haber en el disco (si sólo tuviese i-nodos), que son: 2TB / 512 bytes/i-nodo= 2^{41} bytes / 2^3 bytes/i-nodo=2^{32} . Por lo tanto: 2^{32} bits / 8 bits/byte=2^{32} bits / 2^3 bits/byte=2^{29}bytes=512MB.
- El tercero que será el de bloques de datos (que indica libres u ocupados): 2^{29} bits / 8 bits/byte=2^{29} bits / 2^3 bits/byte=2^{26}bytes=64MB

Características de este diseño:

- Las estructuras son de un tamaño fije y son indexables directamente. Una solución basada en listas enlazadas puede ocupar menos espacio pero no estaría acotado. Además, dado un sistema de ficheros grande pueden ser estructuras de un grandísimo tamaño y con muchos saltos.
- Los bloques se asignan de partida a i-nodos o a bloques de datos, si en un momento se necesita un bloque más de alguna de las dos estructuras, sería necesario buscar un bloque asignado a las estructuras opuestas, verificar si esta libre (en el caso de los bloques de datos) o si están libres (todos los i-nodos en el otro caso) y se le cambiaría la asignación en el primer bitmap.