Sea un disco de 1TiB y un tiempo medio de acceso de A ms. más B ms. por bloque transferido. El disco acaba de ser desfragmentado y contiene un sistema de ficheros de UNIX clásico con nodos_i.

El gestor de ficheros utiliza bloques de 4KiB y agrupaciones de 1 bloque. Consideraremos las siguientes alternativas:

Alternativa	***Cache de bloques***	***Prefeching***
A1	*NO*	*NO*
A2	*Tipo write-through*	*NO*
A3	*Tipo write-through*	*2 bloques adicionales*
A4	*Tipo write-back*	*2 bloques adicionales*

***NOTA**: El prefeching se realiza únicamente cuando se accede un bloque que no está en la cache, indicando el número de bloques adicionales que se traen del disco a cache. La modificación de metadatos se realiza siempre con política de write-through.*

Considere un programa que ejecuta un bucle que contiene la siguiente línea `n = read(fd, buf, 1);` *y que recorre de forma completa el fichero fd de 1MiB. Suponer que la cache está inicialmente vacía y que tiene una capacidad de 4MiB. Asimismo, considere que el tiempo de último acceso solamente se actualiza al cerrar el fichero y que el código del bucle es muy simple.*

***a)** Calcular, para cada una de las alternativas, el número de accesos a disco que se generan, indicando los que son de datos y los que son de metadatos así como su tipo.*

***b)** Calcular, para cada una de las alternativas, el tiempo que se tardaría en ejecutar el bucle.*

***c)** ¿Serían diferentes los tiempos si se cambiase el read por* `n = read(fd, buf, 4096);`*?*

A continuación, considere un programa que ejecuta un bucle que contiene las siguientes líneas `n1 = read(fd1, buf, 1); n2 = write(fd2, buf, n1);` *y que recorre de forma completa el fichero fd1 de 1MiB. fd2 corresponde a un fichero vacío inicialmente. Suponer que la cache está inicialmente vacía y que tiene una capacidad de 4MiB. Asimismo, considere que el tiempo de último acceso solamente se actualiza al cerrar el fichero y que el código del bucle es muy simple.*

***d)** Calcular, para cada una de las alternativas, el número de accesos a disco que se generan, indicando los que son de datos y los que son de metadatos así como su tipo.*

***e)** Calcular, para cada una de las alternativas, el tiempo que se tardaría en ejecutar el bucle.*

***f)** ¿Serían diferentes los tiempos si se cambiase el read por* `n1 = read(fd1, buf, 4096);`*?*

Solución

a) Al estar el fichero abierto el nodo_i se encuentra en memoria. Por otro lado, un fichero de 1MiB ocupa 1MiB/4KiB = 256 bloques, lo que significa que se necesita un bloque indirecto (un bloque indirecto almacena 1024 direcciones de bloque o 512 según que las direcciones de bloque sean de 4 B o de 8 B). Supondremos que, aun el el caso de no tener cache, el gestor de ficheros mantiene en memoria el bloque indirecto, por lo que supone un solo acceso. Es de notar que el acceso al disco para modificar el nodo_i, cambiando el instante del último acceso, se produce fuera del bucle, al cerrar el fichero, por lo que no hay que considerarlo.

A1	Datos	Un acceso a disco por cada read = 1 Mi = 2^{20} accesos a disco .
	Metadatos	Un acceso a disco para el bloque de indirectos.
A2	Datos	Un acceso a disco por cada bloque del fichero = 256 accesos a disco.
	Metadatos	Igual que en A1.
A3	Datos	Al estar el fichero desfragmentado se pude acceder a los tres bloques en un solo acceso al

		disco. Un acceso a disco por cada tres bloques del fichero = 256/3 = 85,3. Luego son 86 accesos a disco de tres bloques. Del último acceso solamente se aprovechará el primer bloque, pero el acceso será también de tres bloques.
	Metadatos	Igual que en A1.
A4	Datos	Igual que en A3.
	Metadatos	Igual que en A1.

b) Nos dicen que el bucle es muy simple, por lo vamos a despreciar el tiempo de ejecución y el de las llamadas al SO y calcular solamente el tiempo que se tarda en acceder al disco. Supondremos que con los metadados no se hace prefeching.

A1	T1read = $(2^{20} + 1)\cdot(A + B)$
A2	T2read = $(256 + 1)\cdot(A + B)$
A3	T3read = $86\cdot(A + 3B) + 1\cdot(A + B)$
A4	T4read = T3read = $86\cdot(A + 3B) + 1\cdot(A + B)$

c) El cambio afectaría a la alternativa A1, puesto que, en vez de 1 Mi accesos para datos, se tendrían 256 , por lo que el tiempo sería igual que en el caso de la alternativa A2, es decir, T2read. Es de destacar que al tener menos llamadas al SO, se reduce aún más el tiempo de ejecución.

d) Los accesos a disco para el read son los mismos que en el caso anterior, por lo que nos centraremos en los del write. Para el fichero fd2 nos hará falta asignar un bloque de indirectos, además de los 256 bloques de datos. Para seleccionar estos bloques es necesario acceder al mapa de bits. Dicho mapa de bits requiere 1 TiB / 4 KiB = 256 Kib = 32 KiB = 8 bloques. Supondremos que el mapa de bits se encuentra en memoria, en caso contrario habría que tener en cuenta los accesos de lectura necesarios para leer la zona del mapa de bits afectada (no hay datos suficientes para conocer qué bloque del mapa de bits se verían afectados, pues depende de qué bloques del disco estén libres). Finalmente, consideraremos que el nodo_i tiene capacidad para direccionar 10 bloques de datos. Como los metadatos siempre utilizan write-through, los accesos que causan son los mismos para los 4 casos.

A1	Datos	Un acceso a disco por cada write= 1 Mi = 2^{20} accesos a disco. Cada write que se hace sobre un bloque con contenido requiere leer primero el bloque del disco, luego $2^{20} – 256$ accesos a disco. Lo que da un total de $2^{20} + 2^{20} – 256 = 2^{21} – 256$
	Metadatos	Cada escritura modifica el nodo_i, puesto que modifica el tamaño del fichero. Cuando la escritura requiere un nuevo bloque de datos, en el caso de 10 primeros bloques también se modifica el nodo_i. Esto no supone un nuevo acceso, puesto que se hace al mismo tiempo que al modificar el tamaño del fichero. Un acceso a disco para modificar el mapa de bits por cada uno de los 10 nuevos bloque de datos = 10 accesos. Hay que modificar el nodo_i para guardar la dirección del bloque de indirectos. Esto no supone un nuevo acceso, puesto que se hace al mismo tiempo que al modificar el tamaño del fichero. Un acceso al disco para modificar el mapa de bits con bit correspondiente al bloque de indirectos seleccionado. Un acceso a disco para modificar el bloque de indirectos por cada nuevo bloque de datos = 246 accesos. Un acceso a disco para modificar el mapa de bits por cada uno de los 246 nuevos bloque de datos = 246 accesos. En total son $2^{20} + 10 + 1 + 246 + 246 = 2^{20} + 503$ accesos a disco
A2	Datos	Un acceso a disco por cada write= 1 Mi = 2^{20} accesos a disco.
	Metadatos	Igual que en A1, es decir, $2^{20} + 503$ accesos a disco
A3	Datos	Igual a A2, es decir, 2^{20} accesos a disco
	Metadatos	Igual que en A1, es decir, $2^{20} + 503$ accesos a disco
A4	Datos	Los datos se escriben en la cache. Al hacer el flush se pasan al disco los 256 bloques. El número de escrituras sería 256, a menos que el flush se haga de forma inteligente agrupando bloques contiguos, en cuyo caso el número de accesos sería menor. En todo caso esto se hace

		fuera del bucle, por lo que no lo consideraremos.
	Metadatos	Igual que en A1, es decir, $2^{20} + 503$ accesos a disco

e) El tiempo es la suma del tiempo de los read calculados anteriormente y los tiempos de los write.

A1	T1read + $(2^{21} - 256 + 2^{20} + 503)\cdot(A + B)$
A2	T2read + $(2^{20} + 2^{20} + 503)\cdot(A + B)$
A3	T3read + $(2^{20} + 2^{20} + 503)\cdot(A + B)$
A4	T3read + $(2^{20} + 503)\cdot(A + B)$

f) Además del cambio en los read, según se ha visto anteriormente, los write de datos también cambiarían al hacerse de 4 KiB, pasando a 256 en los casos A1, A2 y A3. Adicionalmente, las escrituras de metadatos cambian profundamente, pasando a un acceso por write para actualizar el nodo_i, más los accesos al bloque de indirectos y al mapa de bits, en total 256 + 10 + 1 + 246 + 246 = 759

A1	T2read + $(256 + 759)\cdot(A + B)$
A2	T2read + $(256 + 759)\cdot(A + B)$
A3	T3read + $(256 + 759)\cdot(A + B)$
A4	T3read + $(759)\cdot(A + B)$