Problema 2.12 (septiembre 2010)

Se desea diseñar una aplicación servidora que hace continuos accesos a un fichero de 8 TB organizado en registros de 512 B. Cada acceso de cliente supone la lectura y modificación de uno de los registros del fichero. La distribución de accesos a lo largo del fichero es aleatoria y sigue una distribución uniforme.

La aplicación ejecutará en una máquina con procesador de 64 b, 16 GB de memoria principal y sistema operativo con memoria virtual y servidor de bloques común para memoria y E/S. El tamaño del bloque y de la página es de 8 KB y el sistema de ficheros es de tipo Unix.

Se desea decidir si la aplicación debe plantearse con operaciones de E/S o con proyección del fichero en memoria. Para ello, se desea calcular los siguientes valores:

a) Tamaño aproximado ocupado por la metainformación del fichero.

b) Tamaño aproximado ocupado por la tabla de páginas del fichero proyectado.

c) Tasa aproximada de fallos en los accesos a la cache del sistema de ficheros

d) Tasa aproximada de fallos de página en los accesos al fichero proyectado

e) Número de veces que, por término medio, debe ejecutar el sistema operativo por cada 100 accesos de cliente en el caso de solución con E/S. Explicar qué funciones realiza el SO en cada ocasión.

f) Número de veces que, por término medio, debe ejecutar el sistema operativo por cada 100 accesos de cliente en el caso de solución con proyección de ficheros. Explicar qué funciones realiza el SO en cada ocasión.

g) Número de accesos a disco que, por término medio, se realizan por cada 100 accesos de cliente en el caso de solución con E/S.

h) Número de accesos a disco que, por término medio, se realizan por cada 100 accesos de cliente en el caso de solución con proyección de ficheros.

i) Comentar, en vista de los datos anteriores, qué solución le parece más adecuada.

Solución

El enunciado no nos indica el tamaño de los identificadores de bloque en el sistema de ficheros. Suponiendo palabras de 32 b = 4 B podemos direccionar hasta 2^{32}= 4 G bloques = 4 G x 8 KB = 32 TB. Por tanto, es posible utilizar un sistema de ficheros conidentificadores de 32 b, porque nuestro fichero es de 8 TB. Sin embargo, el sistema de ficheros podría tener identificadores de bloque de 8 B, por lo que analizaremos los dos supuestos.

a) La metainformación del fichero es el nodo_i de ese fichero (el mapa de bits es metainformación del sistema de ficheros para conocer qué bloque o nodos-i están libres, pero no forma parte de la metainformación del fichero).

El fichero tiene 8 TB / 8KB = 1 G bloques, por lo que se necesitan 1 G identificadores de bloque. El espacio ocupado por estos identificadores supone:

Identificadores de 4 B: 1 G identificador de bloque x 4 B = 4 GB

Identificadores de 8 B: 1 G identificador de bloque x 8 B = 8 GB

A esto habría que añadir el espacio ocupado por los bloques indirectos. Dado que en un bloque caben 1 K identificadores de 8 B, en total habrá uno doble, uno triple y ($2^{20} – 2^{10}$ - 1) /2^{10} hijos del triple. Aproximadamente 2^{10} bloques, lo que supone 8 MB adicionales, que es un 0,1% de los 8 GB.

b) El espacio ocupado en memoria es de 8 TB / 8 KB = 1 G páginas. Por tanto, la tabla de páginas tendrá 1 G entradas en sus tablas de último nivel. El espacio ocupado por estas tablas es de 1 G páginas x 8 B = 8 GB.

Dado que una tabla real tendrá varios niveles, por ejemplo cuatro, al valor anterior habría que añadir el espacio ocupado por las tablas de nivel 1, 2 y 3. Igual que en el apartado a) esta será una pequeña cantidad comparada con los 8 GB.

Dado que se dispone de 16 GB de memoria principal la tabla de páginas cabe, pero ocupará más de la mitad de la misma.

c) Para calcular la tasa de fallos de cache es necesario conocer el tamaño que puede tener la cache de del sistema de ficheros. También es importante tener en cuenta que dicha cache contendrá tanto bloques de datos del fichero como de bloques de identificadores, por lo que se producirán fallos de cache para los datos y para los identificadores.

Para obtener un valor óptimo de la tasa de fallos, vamos a suponer que se dedica la mitad de la memoria principal a cache de ficheros. Dado que bloque de identificadores determina 1 K bloques de datos, es de suponer que en la cache se tenga un bloque de identificador por cada 1 K bloques de datos.

En este caso tenemos en cache (8 GB – 1 MB) / 512 B ≈ 16 M registros. Como hay un total de de 8 TB / 512 B = 16 G registros y la distribución de accesos está uniformemente distribuida, la probabilidad de encontrar el registro en cache es de 16 M registros / 16 G registros = 1.024, por lo que la tasa de fallos es de 1 – 1/1.024 ≈ 0,999 = 99,9 %.

En la práctica el tamaño de la cache se establece dinámicamente, puesto que depende de las peticiones que los procesos realicen al servidor de bloques y raramente llegará a ocupar la mitad de la memoria principal, por lo que la tasa de fallos sería mayor.

Es interesante constatar que la cache de sistema de ficheros no está produciendo ningún beneficio apreciable (lo que es lógico dado que los accesos están uniformemente distribuidos y, además, se modifica cada registro accedido), por lo que se obtendría un resultado muy similar con una cache mucho menor.

d) Para calcular la tasa de fallos de cache es necesario conocer el conjunto residente de la región del proceso donde está el fichero proyectado. Primero hemos de tener en cuenta que la tabla de páginas reside en memoria principal, por lo que nos quedarán libres algo menos de 8 GB.

Suponiendo que de estos 8 GB se asignan 4GB como conjunto residente del fichero proyectado, lo que supone que se tienen en memoria 4 GB / 512 B = 8 M registros. Luego, la tasa de fallos de página será de 1 – 1/2048 = 99,95 %.

Con un conjunto residente menor se obtiene un valor muy similar. Este es un fenómeno parecido al que se observa en el caso anterior.

e) Cada acceso de cliente exige leer un registro y escribirlo, por lo que son necesarios los siguientes servicios del sistema operativo:

- lseek para posicionarse en el registro a leer.
- read del bloque en el que esté el registro.
- lseek para volver a posicionarse en el registro a escribir.
- write delayed-write.

Como las operaciones que se realizan dependen de la política de escritura, consideraremos el delayed-write, por ser la política más corriente en sistemas de ficheros UNIX.

El fallo de cache se puede producir por el bloque de identificadores o por el bloque de datos. Dado que la antigüedad de uso de ambos será casi siempre la misma, consideraremos que cuando hay fallo de cache hay fallo en ambos.

Cuando no hay acierto en cache, el read supone dos lecturas al disco, una para el bloque de identificadores y otro para el de datos. Además, habrá que limpiar los boques de la cache en los que se desea escribir (la probabilidad de encontrar un bloque limpio es muy pequeña, puesto que todos los bloques de datos se modifican. Se podría tener en cuenta que cada 30 s se limpia la cache, pero para eso deberíamos saber cuantos accesos de cliente se producen por segundo). El SO entra a ejecutar las siguientes veces:

- Ejecución del servicio lseek.

- Ejecución de servicio read. Es necesario leer el bloque de identificadores, para lo cual hay que limpiar un bloque de la cache. Se lanza la escritura del blque seleccionado.
- Interrupción del disco de que la escritura ha completado. Se lanza la lectura del bloque de identificadores.
- Interrupción del disco de que la lectura ha completado. Hay que leer el bloque de datos, para lo cual hay que limpiar un bloque de la cache. Se lanza la escritura del blque seleccionado.
- Interrupción del disco indicando que la escritura ha terminado. Se lanza la lectura del bloque de datos.
- Interrupción del disco para indicar que ya está el bloque de datos leído.
- Ejecución del servicio lseek.
- Ejecución del servicio write. Siempre tiene acierto en la cache, puesto que se hace después de la lectura. Dado que se utiliza delayed-write, el bloque no se escribe en dico..

Cuando hay acierto cada servicio representa una única ejecución del SO, por lo que son 4 ejecuciones.

Por tanto tendremos 99,9*8 + 0,1*4 ejecuciones del SO por cada 100 accesos de cliente.

f) En proyección de ficheros el SO entra en ejecución para atender las interrupciones de fallo de página y del disco. Si no hay fallo de página el SO no necesita ejecutar. Cuando hay fallo de página se produce la interrupción de fallo de página. El SO selecciona una página, que deberá limpiar, por lo que lanza la operación de escritura. El SO entra a ejecutar por la interrupción del disco y lanza la lectura del bloque. Finalmente, entra a ejecutar por la interrupción del disco

Por tanto tendremos 99,95*3 ejecuciones del SO por cada 100 accesos de cliente.

g) Veamos primero el caso de no acierto en cache. El read, según hemos visto, supone dos escrituras para limpiar bloques y dos lecturas. Por su lado, el write no implica ningún acceso al disco.

Para el caso de acierto en cache no es necesaria ninguna operación en el disco.

Por tanto el número de accesos a disco es: 99,9*4 accesos al disco por cada 100 accesos de cliente.

Cuando se cierre el fichero se actualizarón los instantes de modificación y último acceso.

h) Si hay fallo de página la lectura del registro conlleva dos accesos a disco uno para limpiar un bloque y otro para escribirlo. Por su lado, la escritura del registro no implica operaciones de disco.

Por tanto el número de accesos a disco es: 99,95*2 accesos al disco por cada 100 accesos de cliente.

i) La conclusión final es que la solución por E/S supone más operaciones de disco y más actuaciones del SO, por lo que es menos eficiente. Por el contrario, la solución de proyección en memoria exige tener la tabla de páginas en memoria principal, por lo que no se podría utilizar en una máquina con poca memoria principal.