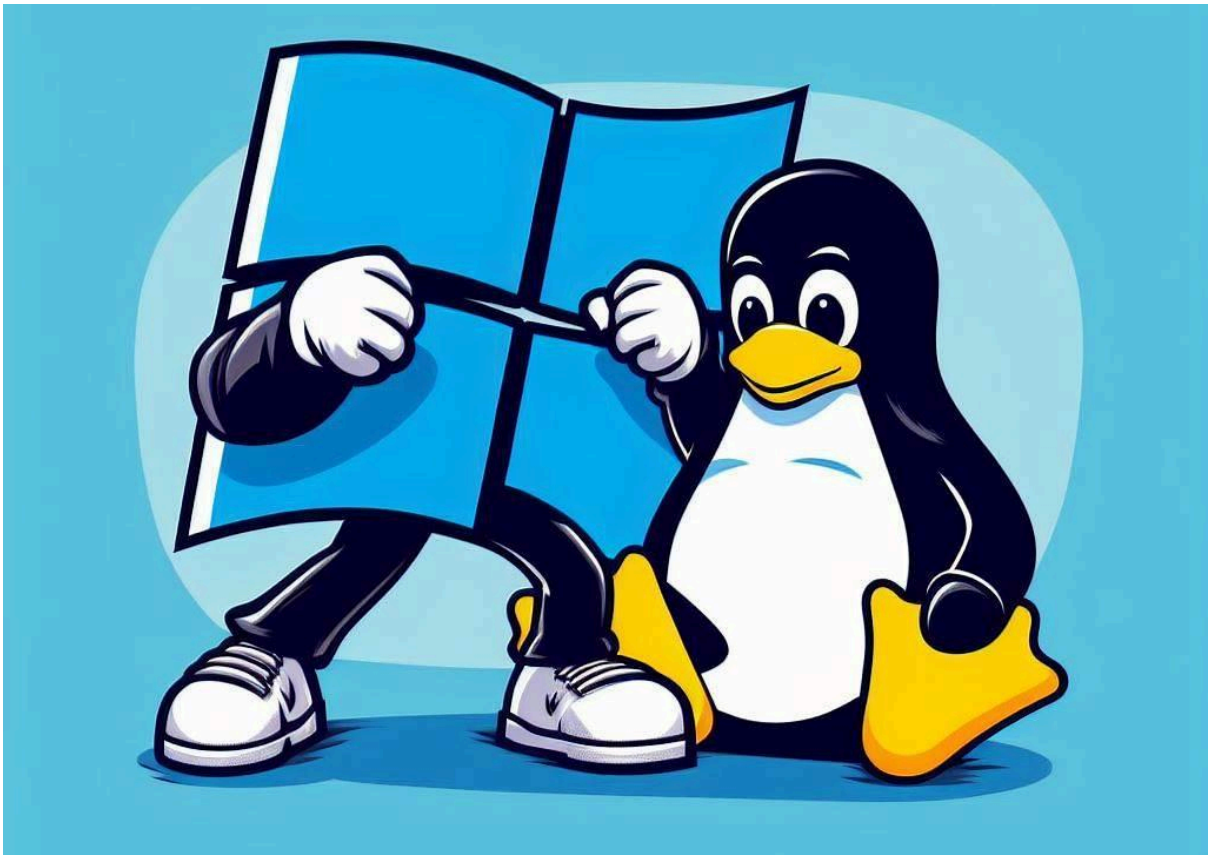
INTRODUCCIÓN A LOS SISTEMAS OPERATIVOS

BUFFER CACHÉ

Resumen teoría



Caporal Nicolás

Facultad de Informática

UNLP

2023

ÍNDICE

DISK CACHE

Es un mecanismo que implementa el SO de un conjunto de estructuras de datos para reducir la frecuencia de acceso a disco.

Son Buffers en memoria principal (RAM) para almacenamiento temporario de bloques de disco.

Objetivo: MINIMIZAR LA FRECUENCIA DE ACCESO AL DISCO

ALGUNAS OBSERVACIONES

Cuando un proceso quiere acceder a un bloque de la cache hay dos alternativas:

- Se copia el bloque al espacio de direcciones del usuario → no permitiría compartir el bloque con otro proceso
- **Se trabaja como memoria compartida → permite acceso a varios procesos**
- Esta área de memoria debe ser limitada, con lo cual debe existir un algoritmo de reemplazo

ESTRATEGIA DE REEMPLAZO

Cuando se necesita un buffer para cargar un nuevo bloque, se elige el que hace más tiempo que no es referenciado.

Para esto el sistema mantiene una lista de bloques, donde el último es el más recientemente usado (LRU, Least Recently Used)

Cuando un bloque se referencia o entra en la cache queda al final de la lista

No se mueven los bloques en la memoria: se asocian punteros.

Otra alternativa: Least Frecuently Used. Se reemplaza el que tenga menor número de referencias

Buffer Caché en Unix System V

OBJETIVO Y ESTRUCTURA

Minimizar la frecuencia de acceso a disco

Es una estructura formada por buffers

Un buffer tiene dos partes:

- Header: Estructura de datos que contiene información del bloque, número del bloque, estado, relación con otros buffers, etc.
- El buffer en sí: el lugar donde se almacena el bloque de disco traído a memoria

El módulo de buffer cache es independiente del sistema de archivos y de los dispositivos de hardware

Es un servicio del SO

Es independiente del File System

HEADER

- Identifica el nro. de dispositivo y nro. de bloque
- Estado
- Punteros a:
    - 2 punteros para la hash queue
    - 2 punteros para la free list
    - 1 puntero al bloque en memoria real

ESTADOS DE LOS BUFFERS

- **Free** o disponible (el bloque se puede utilizar, no necesariamente está vacío)
- **Busy** o no disponible (en uso por algún proceso)
- Se está **escribiendo** o **leyendo** del disco.
- **Delayed Write** (DW): buffers modificados en memoria, pero los cambios no han sido reflejados en el bloque original en disco

HASH QUEUES

Son colas para optimizar la búsqueda de un bloque en particular

Los headers de los buffers se organizan según una función de hash usando (dispositivo,#bloque)

Al número de bloque (dispositivo/bloque) se le aplica una función de hash que permite agrupar los buffers cuyo resultado dio igual para hacer que las búsquedas sean más eficientes

Se busca que la función de hash provea alta dispersión para lograr que las colas de bloques no sean tan extensas

FREE LIST

Organiza los **buffers disponibles** para ser utilizados para cargar nuevos bloque de disco.

No necesariamente los buffers están vacios (el proceso puede haber terminado, liberado el bloque pero sigue en estado delayed write)

Se ordena según LRU (least recent used).

La free List sigue el mismo esquema de la Hash queue pero confiene los headers de los buffers de aquellos procesos que ya han terminado.

El header de un buffer siempre está en la Hash Queue.

Si el proceso que lo referenciaba terminó, va a estar en la Hash Queue y en la Free List

FUNCIONAMIENTO DEL BUFFER CACHÉ

Cuando un proceso quiere acceder a un archivo, utiliza su inodo para localizar los bloques de datos donde se encuentra éste.

El requerimiento llega al buffer cache quien evalúa si puede satisfacer el requerimiento o si debe realizar la E/S.

Se pueden dar 5 escenarios:

1) El kernel encuentra el bloque en la hash queue y el buffer está libre (en la free list).

- Hashea el numero de Bloque, le da N.

- Entra a la Hash Queue N y recorre siguiendo los enlaces
- Encuentra el Bloque
- Verifica que tiene estado Free por tanto está en free list
- Lo remueve de la free list
- Pasa el bloque a estado BUSY modificando la Header
- El proceso usa el Bloque
- Reacomoda los punteros de la free list

2) El kernel no encuentra el bloque en la hash queue y utiliza un buffer libre.

- Hashea el numero de Bloque, le da N.
- Entra a la Hash Queue N y recorre siguiendo los enlaces
- Llega hasta el final, no está
- Debe buscar un bloque libre para cargar el Bloque que se necesita
- Toma el primer buffer de la free list (es el LRU)
- Lee del disco el bloque deseado
- Se ubica el Bloque en la hash queue correspondiente (solo se cambian punteros, **NO** se intercambian posiciones de memoria)

3) El kernel no encuentra el bloque en la hash queue y utiliza un buffer libre pero el bloque libre esta marcado como DW.

- Hashea el numero de Bloque, le da N.
- Entra a la Hash Queue N y recorre siguiendo los enlaces
- Llega hasta el final, no está
- Debe buscar un bloque libre para cargar el Bloque que se necesita
- Toma el primer buffer de la free list (es el LRU) pero está como DW
- Se escribe en disco el bloque que estaba en ese buffer y sigue buscando hasta encontrar uno libre
- Mientras se escriben en disco los que estaban marcados como DW (es asincronico) el proceso usa el bloque deseado.
- Una vez escritos a disco los bloques DW, son ubicados al **principio** de la Free List

4) El kernel no encuentra el bloque en la hash queue y la free list está vacía.

- Hashea el numero de Bloque, le da N.
- Entra a la Hash Queue N y recorre siguiendo los enlaces
- Llega hasta el final, no está
- Debe buscar un bloque libre para cargar el Bloque que se necesita pero la free list está vacía
- El proceso queda bloqueado a la espera de que se libere algun buffer

- Cuando el proceso despierta se debe verificar nuevamente que el bloque no este en la hash queue (algún proceso pudo haberlo pedido mientras éste dormía)
- Luego idém 2 o idém 3

5) El kernel encuentra el bloque en la hash queue pero está BUSY.

- Hashea el numero de Bloque, le da N.
- Entra a la Hash Queue N y recorre siguiendo los enlaces
- Encuentra el Bloque pero el buffer está marcado como BUSY
- El proceso se bloquea a la espera de que el buffer se libere
- Eventualmente el proceso que tenia el Buffer lo libera
- Se despiertan todos los procesos en espera de algún buffer
- El proceso que buscaba ese Buffer debe buscarlo nuevamente en la hashqueue y en la freelist