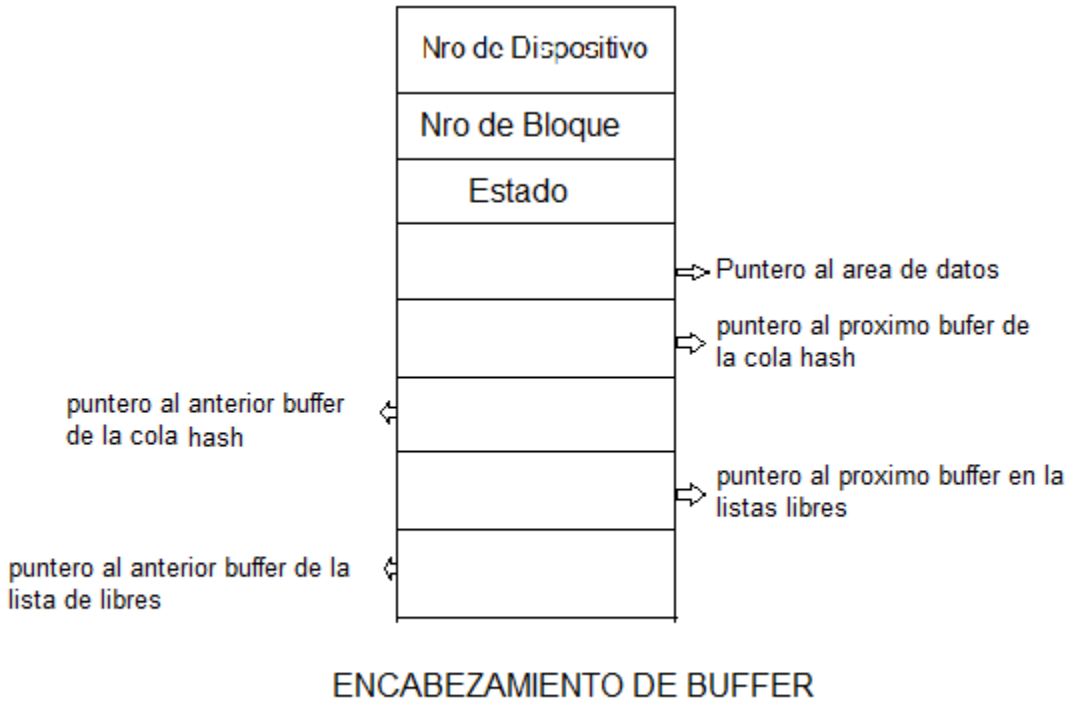
Nombre: http://www.network-uagrm.blogspot.com/

TEMA NRO 3

SISTEMA OPERATIVO UNIX

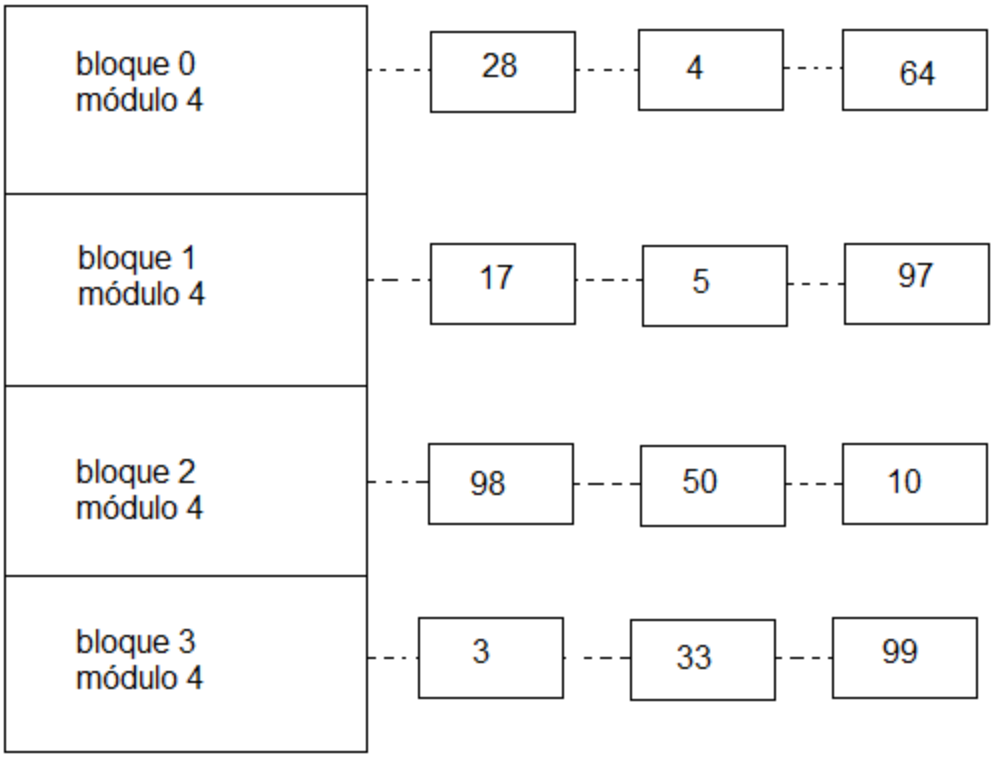


ENCABEZAMIENTO DE BUFFER

Estructura del equipo (pool) de buffer.- El kernel deposita datos en el equipo de buffer de acuerdo a los algoritmos LRU (menos usado recientemente) ósea que elige al buffer LRU para colocar un bloque de disco, el kernel mantiene una lista de buffer libres en orden menos recientemente utilizado y es una lista circular doblemente encadenada, todo buffer se coloca en la lista libre cuando se carga el sistema y el kernel toma un buffer de la cabeza de la lista libre cuando lo necesita pero también puede tomar un buffer de la mitad de la lista libre si es que se encuentra allí el bloque requerido, cuando el kernel devuelve un buffer generalmente lo coloca en la parte de atrás de la lista libre.

Cuando el kernel accede a un bloque de disco, busca un bloque del buffer con la combinación apropiada de número. Bloque dispositivo y en su lugar de hacer la búsqueda en todo el equipo del buffer organiza organiza a estos mediante colas separadas que resultan como función hosh de numero bloque dispositivo.

El kernel encausa los buffer sobre las colas hash como una lista circular doblemente encadenada similar a la estructura de lista libres, el kernel debe usar una función hashing para distribuir los bloques uniformemente a través del conjunto de colas, no obstante, la función hosh deberá ser tan simple como para no interferir el desempeño, el administrador de sistemas configura el número de colas cuando genera el sistema operativo.

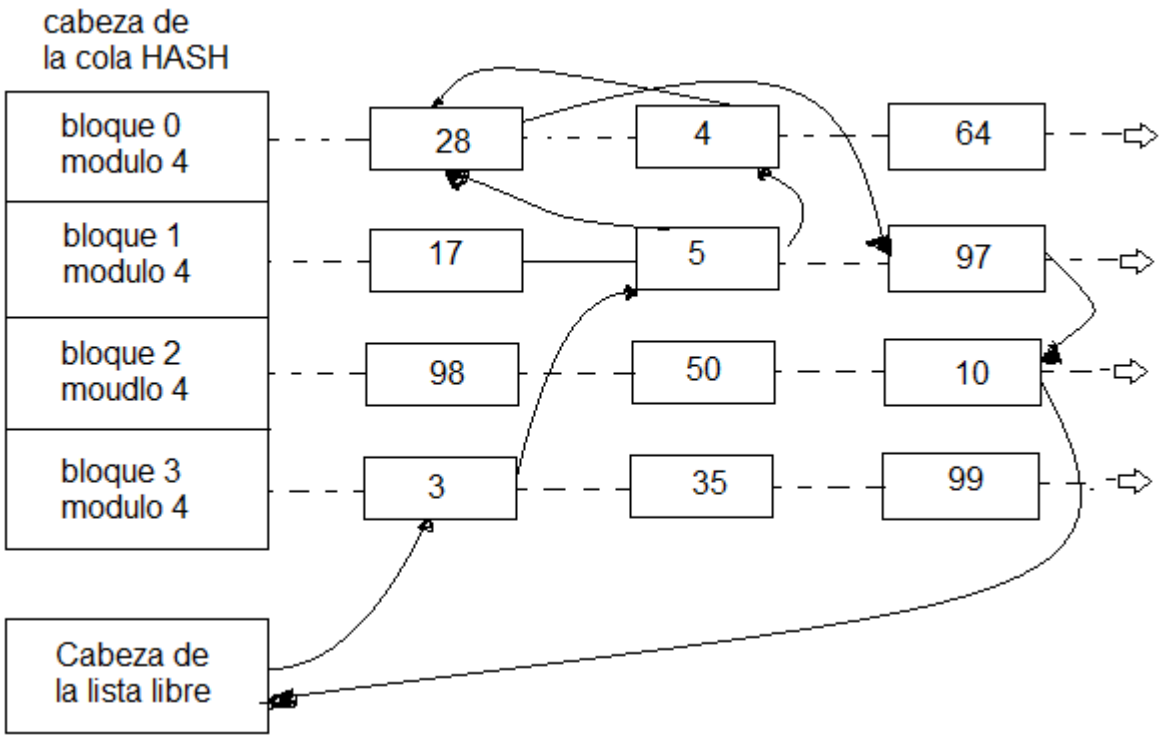


Los encabezamientos de las colas hash se muestran a la izquierda de la figura y los cuadrados de cada fila son los buffer de la cola hash de esta manera los cuadrados marcados 25, 4, 64, son los buffer de la cola hash para el bloque cero y las líneas punteadas representan los punteros hacia delante y hacia atrás para la cola hash. Todo buffer pertenece a una cola hash dos buffer no pueden tener simultáneamente un mismo bloque de disco, por lo tanto cada bloque de disco en el pool de buffer solo se encuentra una vez en la cola hash, de cualquier modo un buffer puede pertenecer a una lista libre y por el hecho de que un buffer puede estar simultáneamente en una cola hash y una lista libre el kernel tiene dos manera de hallarlo, es decir puede buscarlo en una cola hash si es que requiere un buffer en particular o puede sacar el buffer de la lista libre, en resumen un buffer pertenece siempre a una cola hash y podría estar en la lista libre.

AMBIENTES PARA EL MANEJO DE BUFFERS

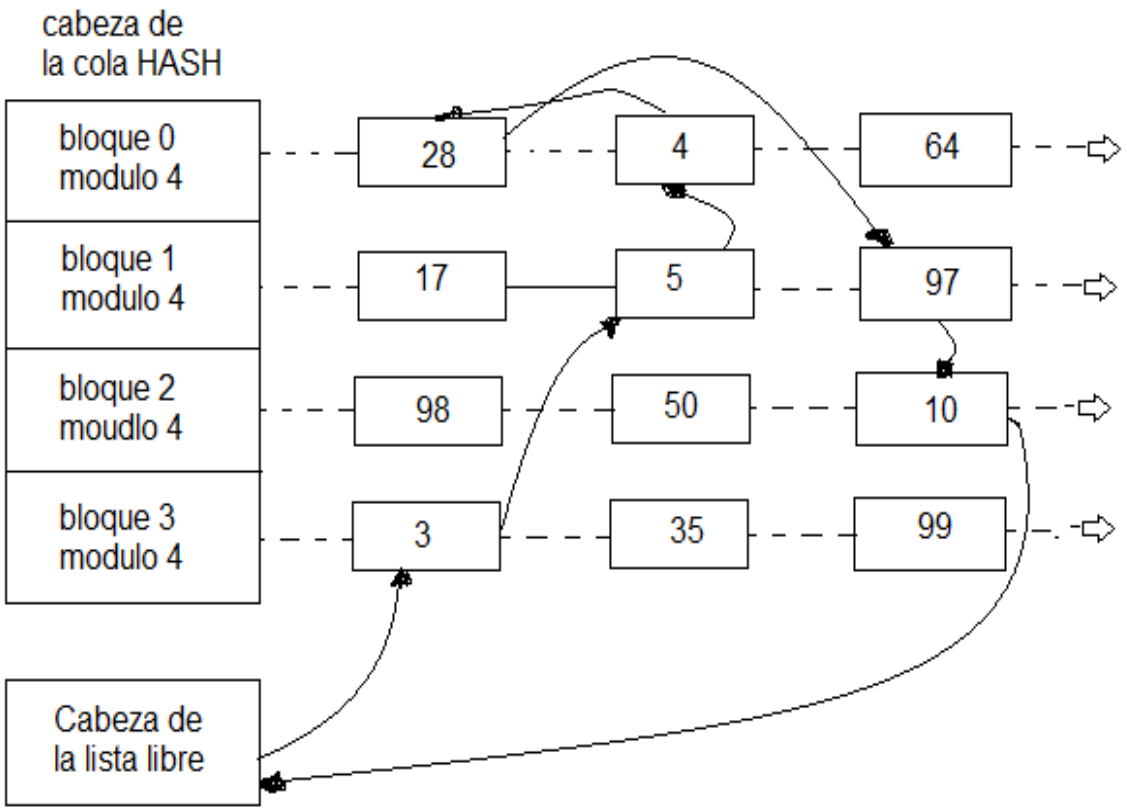
Los algoritmos de alto nivel determina el # de dispositivos lógicos y el # de bloque cuando requiera recuperar el bloque por ejemplo: un proceso, un dato de algún archivo el KERNEL determina el archivo y lo cual es el bloque que contiene ese dato cuando se trata de leer un dato de algún bloque particular del disco el Kernel averigua si este bloque esta en el buffer de Pool y cuando no lo es asi le asigna un buffer libre cuando se trata de escribir un bloque especifico del disco el kernel también averigua si ese bloque esta en Pool de buffers de lo contrario le asigna un buffer libe para ese bloque. Los algoritmos para lectura y escritura de los bloques de discos usan el algoritmo GETBLK para asignar buffers del Pool a continuación se muestra 5 ambientes típicos donde el kernel puede ejecutar el algoritmo GETBLK para asignar un buffer para el bloque de discos.

1.- el kernel encuentra el bloque en la cola HASH y su buffer esta libre:

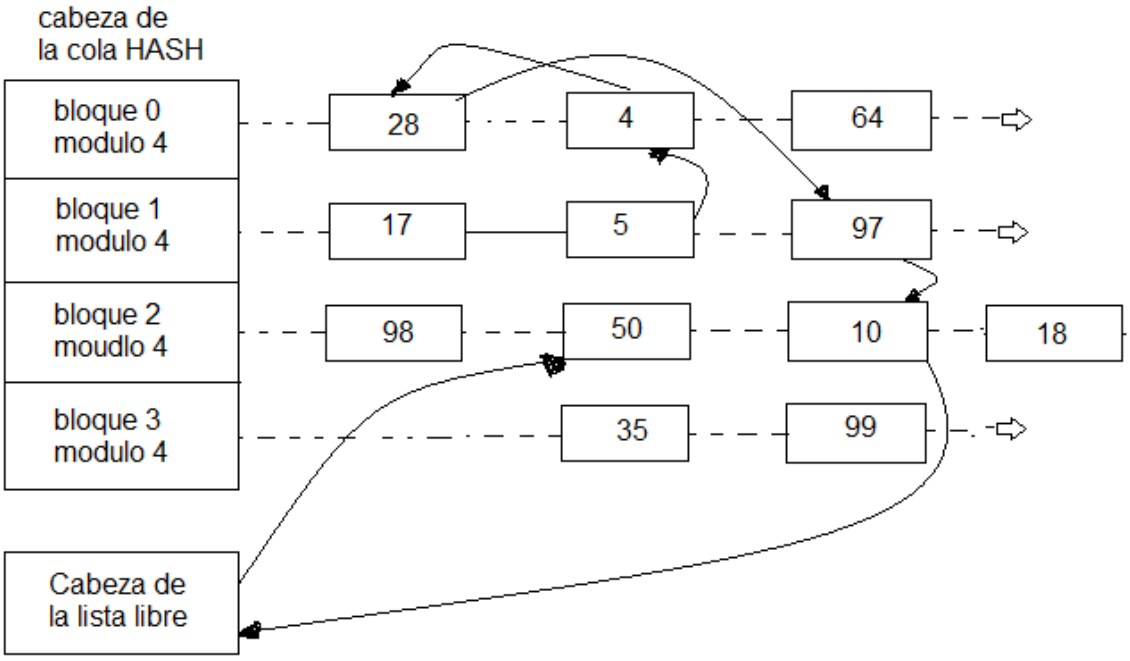


a) Búsqueda del bloque4 en la primera cola HASH.
b) Retira el bloque 4 de la lista libre

2.- El kernell no puede hallar el bloque en la cola HASH por lo que asigna un buffer de la lista libre.



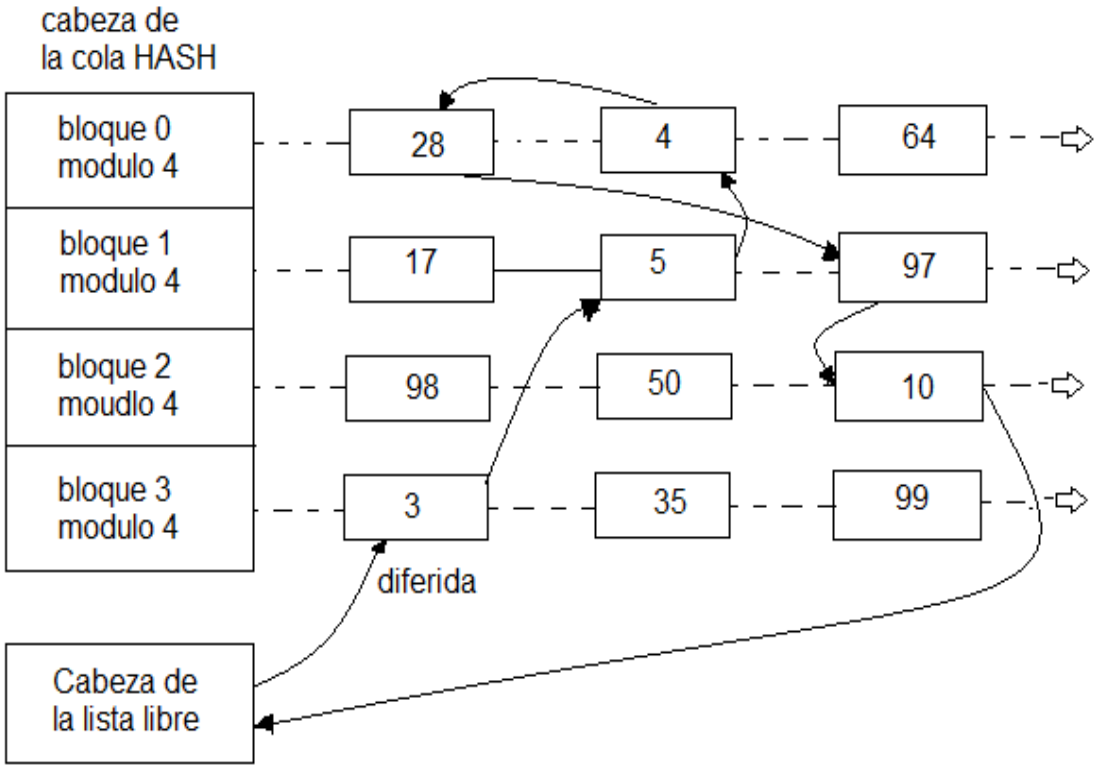
c) Búsqueda del bloque 18, no está en la caché.
d) Retira el primer bloque de la lista libre lo asigna a 18.

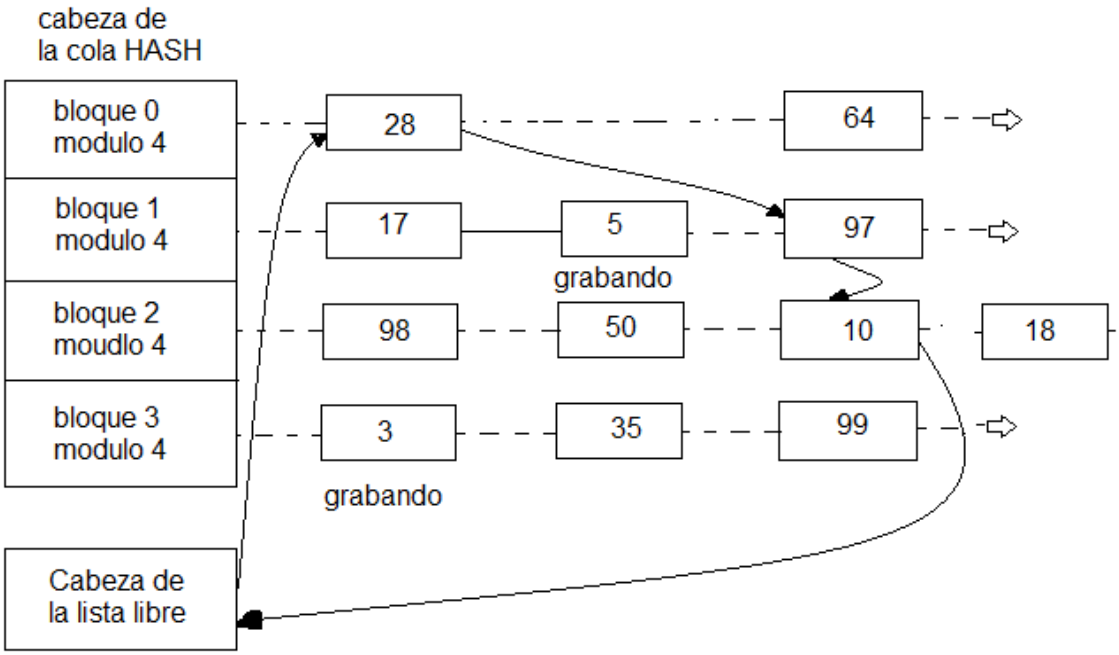


AMBIENTES PARA EL MANEJO DE BUFFER

3.- No halla el bloque en la cola HASH escritura diferida.

El kernell no puede hallar el bloque en la cola HASH y en el intento de asignar un buffer de la lista libre como lo hizo en el ambiente 2 descubre un buffer de la lista marcada con escritura diferida, el kernel deberá esperar escribir ese buffer en disco y asignara otro buffer.

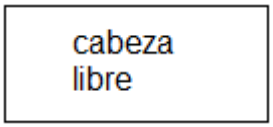




Cuando el kernel intenta asignar el buffer de la lista libre tiene la marca de escritura diferida de modo que debe escribir el contenido del buffer en el disco antes de poder utilizar ese buffer. El kernel inicia una escritura asíncrona a disco y trata de asignar otro buffer de la lista libre, cuando se completa la escritura asíncrona el kernel libera ese buffer y lo coloca a la cabeza de la lista libre, el buffer había empezado al final de la lista libre y realizó el recorrido hasta la cabeza de la lista libre.

ANBIENTE 4

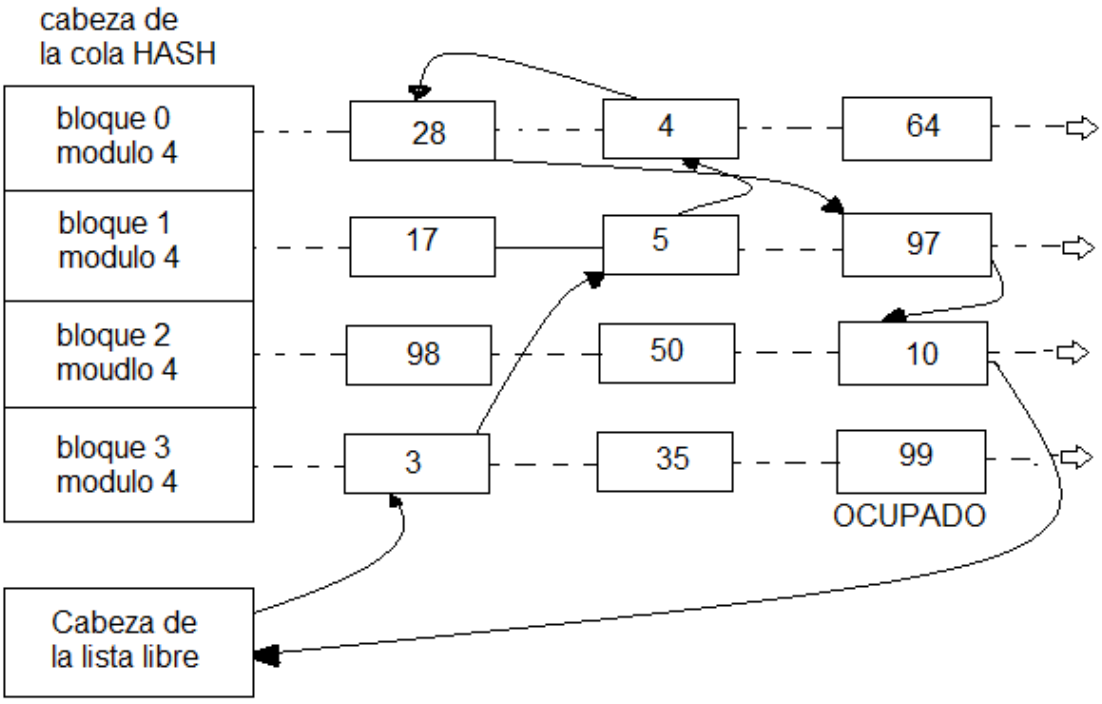
El kernel no puede hallar el bloque en la cola HASH y la lista libre del bloque está vacía



Búsqueda del bloque 18

El kernel en representación de un proceso no encuentra el bloque de disco en su cola HASH de modo que intenta asignar un nuevo buffer de la cola de la lista libre, similar que el ambiente 2, sin embargo la lista libre no tiene buffer disponible por lo que el proceso es puesto a dormir hasta que algún otro proceso libere el buffer.

AMBIENTE 5



Búsqueda del bloque 99, bloque ocupado.

En este caso enredado porque incluye relaciones complejas entre varios procesos suponiendo que el kernel realiza la búsqueda de un bloque y localiza un buffer como está ocupado el proceso es puesto a dormir hasta que el buffer sea liberado.

LECTURA Y ESCRUTURA DEL BLOQUE DE DISCO.

Una lectura y una escritura de un bloque de disco constan de las siguientes etapas:

a) Se hace la búsqueda en memoria cache del bloque de disco que se requiera, si se encuentra en memoria cache no se efectúa la lectura física del bloque de disco.

b) Si el bloque no está en memoria cache el Kernell planifica la lectura y pone a dormir el proceso hasta que la entrada salida sea empleada.

Escritura:

a) Una vez informado por el Kernell del requerimiento para escribir el contenido del Buffer, se planifica el bloque para la escritura.

- ***Sincronizado.*** Si la escritura es sincronizada el proceso que hizo la solicitud es puesto a dormir hasta que se complete la entrada y salida.

- ***Asíncrono.*** El Kernell inicia la escritura pero no espera hasta que finaliza, el Kernell libera el buffer cuando acaba la entrada y salida.

Una escritura demorada es diferente.

- Escritura asíncrona, El Kernell inicia la operación de entrada y salida inmediatamente pero no espera hasta su finalización.

- Escritura demorada, El Kernell posterga la escritura física tanto como sea posible, cuando inicia la escritura señala al Buffer con una marca y escribe el bloque a disco a forma asíncrona, cuando finaliza la escritura se libera el Buffer y es colocado a la cabeza de la lista libre.

Ventajas y desventajas:

- Es uso de Buffers establece un acceso uniforme a disco, porque los Buffers son utilizados independientemente en el sentido que los datos son parte de un archivo de un Inodo o de de un súper bloque, esto permite la comunicación más modular y el diseño del sistema es más simple.

- El sistema no tiene restricciones de orden para poner los datos con la entrada salida del proceso usuario, porque el Kernell ordena los datos internamente, si no hubiera este mecanismo los programadores tendrían que hacer ordenamiento en el Buffer.

- El uso del Buffer cache puede reducir la cantidad de trafico de disco e incrementando el rendimiento del sistema, los procesos de lectura de sistema de archivos evitan una operación de entrada y salida del disco, si encuentran los bloques de datos en memoria cache, las oportunidades de hallar bloques de memoria cache son mayores para sistemas con mayor cantidad de Buffers, sin embargo el número de Buffers en un sistema se obtiene restringiendo la cantidad de memoria para los procesos, esto puede ocasionar un excesivo swapping o páginamiento.

- Los algoritmos de Buffer garantiza la integridad del sistema porque en memoria cache se obtiene una sola copia de los bloques de disco, los algotirmo de Buffers se realizan los accesos evitando la corrupción de los datos.

Desventaja:

- En escritura demorada el Kernell no escribe inmediatamente los datos a disco si el sistema es vulnerable a las caídas deja al disco en forma incorrecta.

- El buffers cache requiere una copia original de los datos cuando se hace la lectura y escritura para un proceso.

REPRESENTACION INTERNA DE ARCHIVOS.

iNodos (PCB) el lugar permanente del iNodo es el disco y es allí que lo lee el kernel para luego llevarlo a memoria principal y luego proceder al trabajo correspondiente los campos que tiene el iNodo son:

Diseño	ABC
Grupo	
Tipo de archivo	regular
Permiso	R,W,R X,R,X
Acceso	Novienbre 3 1999 0:30 pm
Modificado	Octubre 31 1999 11:00 pm
iNodo	
Tamaño	
Direccion del disco	

El contenido de un iNodo cambia cuando se cambia en el contenido del archivo o cuando se cambia su diseño, los permisos a los punteros, el cambio del contenido del archivo implica el cambio automático en el hilo pero no viceversa.

Estructura de un archivo regular.- El iNodo contiene una tabla para ubicar los datos del archivo en el disco y como cada bloque en disco es direccionable por un número entonces el contenido de la referida tabla es un conjunto de numero de bloques de disco, si el almacenamiento fuera contiguo en el disco entonces sería suficiente el número del bloque inicial y el tamaño del archivo, sin embargo tal estrategia de asignación no permitiría una simple expansión o contracción del archivo sin causar el riesgo de fragmentación en el disco.

Asignación de iNodos a nuevos archivos.- cuando un proceso requiere un nuevo iNodo el kernel tendrá que buscar en la lista de iNodos libres pero esa búsqueda podría resultar costosa puesto que requiere por lo menos una operación de lectura por cada iNodo, para mejorar el desempeño en el superbloque del sistema de archivos contiene un arreglo de los números de iNodos libres en el sistema de archivos (similar a una memoria cache) la lista de iNodos libres del superbloque está vacía, el kernel examina el disco y coloca la mayor cantidad de números de iNodos libres en el superbloque.

Asignación de bloques de disco.- Toda vez que un proceso escribe un archivo el kernel debe asignar bloques de disco, el superbloque tiene un arreglo similar a la memoria caché que es utilizado los numero de bloques libres de disco, los bloques de datos se organizan mediante una lista encadenada de manera que cada eslabón es un bloque de disco que están libres.

Si un proceso escribe una gran cantidad de datos de un archivo sus requerimiento de datos serán frecuentes, el kernel procura organizar la lista de iNodos que la distribución de bloques o archivos restantes próximos entre sí. Esos mejora el desempeño porque reduce el tiempo de búsqueda y el tiempo de latencia del disco, cuando un proceso lee de forma secuencial, los algoritmos para asignación y liberación de iNodos y de bloque de disco son similares en el sentido que el kernel usa el super-bloque con una memoria caché conteniendo índice de recursos libres, numero de bloques y números de iNodos, mantiene una lista encadenada de números de bloques de manera que todo número de bloque libre se encuentra en esa lista encadenada, pero tal cosa nos sucede en la lista de iNodos libres debido a tres razones:

1.- el kernel puede determinar si un nodo está libre mediante infección, si el campo tipo de archivo está libre, sin embargo no se puede seguir el mismo método con un bloque

2.- los bloques de un disco son apropiados para usar listas encadenada de bloque de disco puede contener sin dificultad una lista grande de numero de bloques que están libres pero los iNodos no tienen el espacio suficiente para contener listas grandes de números de iNodos libres.

3.- los usuarios consumen en mayor proporción bloques de discos que iNodos de manera que la aparente disminución de desempeño cuando se examina el disco buscando iNodos libres no es tan crítico como una búsqueda de bloques libres.