UNIVERSIDAD DE COSTA RICA

FACULTAD DE INGENIERÍA

ESCUELA DE CIENCIAS DE LA

COMPUTACION E INFORMÁTICA

CI-0123 Proyecto Integrador de Redes y Sistemas operativos

Prof. Adrián Lara

Prof. Tracy Hernández

Proyecto Final.
Sistema distribuido de almacenamiento y consulta de expedientes médicos.

Elaborado por:

Ayales León Eduardo B90833

Castillo Campos Angie Sofia B91750

Quesada Quesada Esteban B96157

Ureña Torres Jose David B88044

1. Interoperabilidad

La interfaz realizará una serie de solicitudes para ejecutar procedimientos, como lectura/escritura de datos, permitiendo de esta forma un intercambio de información entre los nodos y la interfaz, es decir, un conocimiento entre ambas partes. Asimismo, los nodos que conformarán una pareja pasivo-activo se conocen entre ellos, debido a que deben realizar acciones como replicado de información. Para permitir esta interoperabilidad los nodos que almacenarán la información seguirán el protocolo elegido de red para lograr encontrar a su pareja y conectarse a la interfaz.

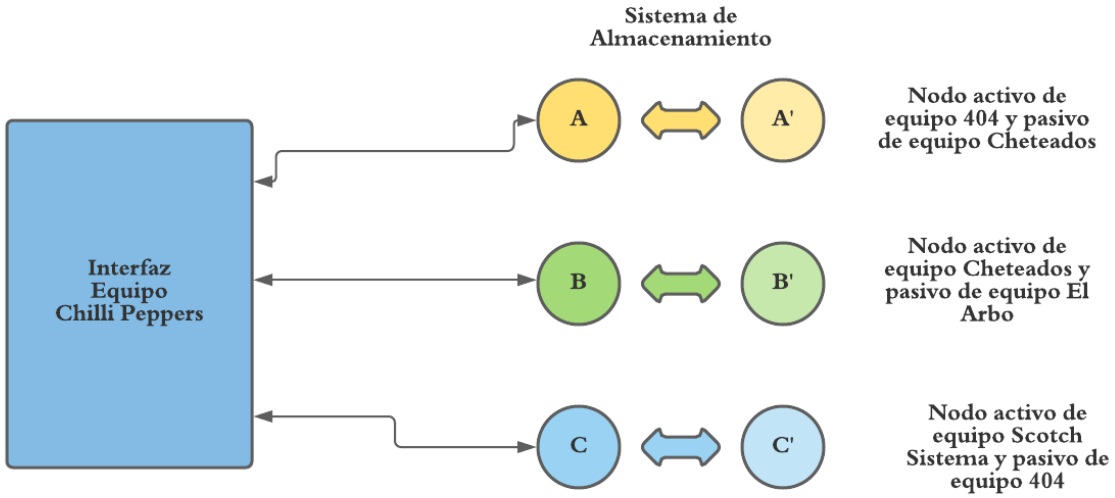

Figura 1. Ejemplo de interoperabilidad entre la interfaz de un equipo y el sistema de almacenamiento de diferentes equipos.

2. Incorporación dinámica de nodos de almacenamiento

Emparejamiento de nodos

En el momento que un nodo está buscando pareja, see deben tomar en cuenta diversas operaciones, por lo que utilizaremos los siguientes códigos:

- 0 → El nodo busca pareja sin importar el rol.
- 1 → El nodo activo se encuentra buscando un nodo pasivo.
- 2 → El nodo pasivo notifica que formó pareja con X nodo activo.
- 3 → El nodo activo confirma que su pareja final es Y.

Asimismo, se debe tener en cuenta que el formato de envío y recepción de mensajes será el siguiente:

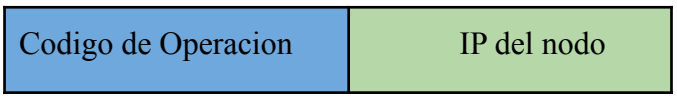

Figura 2. Código emparejamiento de nodos

Procedimientos:

1. Todos los nodos envían broadcast UDP por los puertos acordados grupalmente con código 0 o 1.
2. Todos los nodos escuchan broadcast por puertos acordados.
3. Se usa la comparación de IP para decidir quien es activo y quien es pasivo; el nodo activo será el de mayor IP y el pasivo el de menor.
4. El nodo pasivo elige al azar (a menos de que exista un mensaje con código 1, ya que este toma prioridad) y envía un mensaje con código 2 y la IP de su posible nodo activo. Ejemplo: 2172.16.1.1
5. Todos los nodos monitorean el broadcast de parejas propuestas.
6. El nodo activo confirma su pareja (nodo pasivo): envía un mensaje con código 3 y la ip de su nodo pasivo confirmado. Ejemplo: 3173.134.1.2
7. Todos los nodos monitorean el broadcast de parejas confirmadas.
8. Los nodos activos hacen listen TCP y los nodos pasivos hacen connect TCP.
9. Inicia la conexión con la Interfaz.

Manejo de errores de emparejamiento

- Los nodos que ya tienen pareja seguirán escuchando broadcast UDP. Sin embargo, no van a procesar más broadcast UDP mientras se esté configurando la pareja; los ignora.
- Cuando una pareja esté configurada estos no emitirán broadcast UDP a otros nodos.
- Si algún nodo intenta establecer una conexión TCP con algún nodo que ya está emparejado, la conexión TCP fallará por lo que el nodo deberá seguir buscando pareja.

Comunicación interfaz-cliente, interfaz-nodo(procesar solicitud del cliente read-write)

Cuando el cliente desea hacer una solicitud hacia la interfaz, ya sea de lectura o escritura, deberá utilizar el siguiente formato para enviar solicitudes:

- La letra r para hacer una solicitud de lectura.
- La letra w para hacer una solicitud de escritura.

Asimismo, la interfaz responderá con los siguientes estados la solicitud:

- 0 → La operación fue exitosa.
- 1 → Error.

Por lo que el formato del protocolo para la emisión y recepción de solicitudes será el siguiente:

- Solicitud de lectura Cliente → Interfaz: usuarioΣcontraseñaΣrΣcedula
- Respuesta de solicitud de lectura Interfaz → Cliente: estadoΣcedulaΣnombreΣdatos
- Solicitud de escritura Cliente → Interfaz: usuarioΣcontraseñaΣwΣcedulaΣnombreΣdatos
- Respuesta de solicitud de escritura Interfaz → Cliente: estado.

Comunicación nodos de almacenamiento-interfaz

Para poder entender la estructura de los mensajes intercambiados se deben tomar en cuenta los siguientes códigos:

- Código 0 indica nueva pareja.
- Código 1 indica nuevo nodo pasivo.

Asimismo, se deben tener en mente algunas anotaciones:

1. La interfaz debe estar escuchando mensajes broadcast UDP de nuevos nodos activos que ya tienen pareja.
2. Tanto el nodo activo como el pasivo comienzan a escuchar por TCP, por los puertos acordados grupalmente.
3. La interfaz se conecta al nodo pasivo via TCP, por puertos acordados grupalmente.
4. La interfaz guarda en una estructura de datos la IP del nodo activo, la del nodo pasivo y la cantidad de bloques disponibles.

   Nota: El almacenamiento es el mismo para ambos nodos. La ip del activo se pasa por simpleza.
5. La interfaz tendrá una estructura de datos cómo la siguiente:

Lista de parejas registradas				
IP activo	Socket TCP (apuntador)	IP pasivo	Espacio disponible (bloques)	Cédulas de pacientes almacenados
172.168.10.4	Object	172.168.10.15	32	11112222, 3333444, 5555666
172.168.10.20	Object	172.168.10.24	32	77778888, 99990000
172.168.10.2	Object	172.168.10.3	32	19581648, 19284846

Figura 3. Lista de parejas registradas

6. Para indicar una nueva pareja de nodos, el nodo activo debe enviar un mensaje a la interfaz vía UDP con el siguiente formato:

0	IP nodo activo	IP nodo pasivo	Bloques disponibles

Figura 4. Protocolo notificar interfaz de nueva pareja.

7. Para indicar un nuevo nodo pasivo, este debe enviar un mensaje a la interfaz vía UDP con el siguiente formato:

1	IP nodo pasivo	Bloques disponibles

Figura 5. Protocolo notificar interfaz de nueva pareja.

Emisión y envío de solicitudes de lectura y escritura (interfaz-nodo)

Se usarán los siguientes códigos para para enviar solicitudes:

- La letra r para hacer una solicitud de lectura.
- La letra w para hacer una solicitud de escritura.

Asimismo, se responderá con los siguientes estados la solicitud:

- 0 → La operación fue exitosa.
- 1 → Error.

Por lo que el formato del protocolo para la emisión y recepción de solicitudes será el siguiente:

- Lectura Interfaz → Nodo: operaciónΣcédula
- Lectura Nodo → Interfaz: estadoΣcédulaΣnombreΣdatos
- Escritura Interfaz → Nodo: operacionΣcédulaΣnombreΣdatos
- Escritura Nodo → Interfaz: estadoΣcantidadbloquesrestantes

3. Duplicación de información (Nodo espejo)

En el protocolo destinado a la duplicación de información de un nodo de almacenamiento se utilizarán los siguientes códigos para enviar solicitudes:

- w para escribir
- c para duplicar la totalidad de la información de un nodo

A la hora de responder solicitudes los códigos serán:

- 0 para denotar que todo salió bien
- 1 ocurrió un error

A la hora de duplicar información existen dos casos:

1. Duplicación normal: llega una solicitud al nodo activo y este se la envía al nodo pasivo para que este la guarde. El nodo activo le enviará la información al nodo pasivo de la siguiente manera: opΣcedulaΣnombreΣdata

   Ejemplo: wΣ2288ΣPabloΣenfermo

   El nodo pasivo responderá: statusΣcantidad-de-bloques-restantes

2. Duplicación especial: el nodo pasivo se acaba de unir y se le debe pasar toda la información del nodo activo. Donde el nodo activo le envía la información de todos los pacientes al nodo pasivo de la siguiente forma:

   Se utiliza **Σ** como separador entre los campos op, cedula, nombre y data. Se usa σ como separador entre cada paciente. Un ejemplo sobre como ocurriría dados 3 pacientes sería el siguiente:

   opΣcedula-1Σnombre-1Σdata-1σcedula-2Σnombre-2Σdata-2σcedula-3Σnombre-3Σdata-3

   Observándose en un ejemplo:

   cΣ1234ΣPabloΣEnfermoσ5678ΣMariaΣCansadaσ9012ΣAnaΣDurmiendo

   Asimismo, el nodo pasivo le responderá con el siguiente formato:

   estadoΣcantidad-de-bloques-restantes

4. Manejo de fallos

Fallo nodo activo

1. Hay dos formas en las que se va a detectar cuando un nodo muere:
    a. El nodo activo no responde, pero no está muerto ya que la conexión sigue viva. La interfaz intenta enviar una solicitud de lectura o escritura al nodo activo por TCP. Esta operación falla porque el nodo activo está muerto. Se maneja la excepción lanzada.
    b. Se detiene la conexión TCP entre la interfaz y el nodo activo. Debe manejarse la excepción lanzada.
2. De manera que la interfaz intenta establecer una conexión TCP con el nodo pasivo (connect). Recordemos que el nodo pasivo ha estado escuchando a la interfaz desde que se emparejó, por si el nodo activo falla.
3. La interfaz procede a enviar la solicitud de lectura o escritura al nuevo nodo activo.
4. El nuevo nodo activo comienza a enviar los broadcast UDP con mensaje 1 para encontrar un nuevo nodo pasivo.
5. Se realiza el proceso de emparejamiento.
6. El pasivo le envía un mensaje con el formato indicado en *Comunicación nodos de almacenamiento - interfaz*.
7. La interfaz recibe este mensaje y actualiza la tabla de parejas.
8. Una vez que se establece la conexión, deberá pasarle toda la información almacenada que tiene al pasivo.

Fallo nodo pasivo

1. Hay dos formas en las que un nodo pasivo falla:
    a. El nodo pasivo no responde, pero no está muerto porque la conexión TCP no se ha cerrado. El nodo activo procesa exitosamente una operación de escritura que le encargó la interfaz. Procede a reflejar la información al nodo pasivo. Esta comunicación falla.
    b. La conexión se cierra inesperadamente. La excepción lanzada debe manejarse.

2. El activo empieza a mandar broadcast UDP para buscar un nuevo nodo pasivo. El contenido de este mensaje debe ser un 1.
3. Se realiza el proceso de emparejamiento de nodos.
4. El pasivo le envía un mensaje con el formato indicado en *Comunicación nodos de almacenamiento - interfaz.*
5. La interfaz recibe este mensaje y actualiza la tabla de parejas.
6. Una vez que se establece la conexión, deberá pasarle toda la información almacenada que tiene al pasivo.

5. Seguridad

La interfaz deberá estar en una subred distinta a la del back end, es decir, los nodos de almacenamiento tendrán una IP distinta a la de la interfaz, asimismo, la interfaz va a contar con un listado de todas las IP de los nodos de almacenamiento. La interfaz consultará en la estructura de datos que posee para saber en cuál nodo se encuentra la cédula.

A la hora de hacer el emparejamiento de nodos, cada nodo tomará su rol ya sea como activo o como pasivo y lo mantendrá hasta que muera o que sea forzado a cambiarlo (caso de que nodo activo muera). Cada nodo estará en una subred distinta. Por otra parte, a nivel de grupo se decidieron ciertos puertos para que sean utilizados por todos los distintos equipos.

Para UDP:

Nodo ↔ Nodo: 69420 e Interfaz ↔ Nodos: 8085

Para TCP:

Nodo ↔ Nodo: 11337

Interfaz ↔ Nodos: 12345

6. Sistema de almacenamiento en bloques y diseño del sistema de archivos

File System

Consideraciones

- Cada nodo tiene su propio sistema de archivos.
- El almacenamiento en disco se simulará en memoria principal.
- No habrá persistencia de datos.

Atributos de archivos

Un archivo está constituido por los siguientes elementos:

Tamaño actual del texto (en bytes): Req. 1 byte	Tamaño máximo del texto (en bytes). Req. 1 byte	Texto. Req. Tamaño del bloque- 2

Figura 6. Atributos de archivo

Ejemplo de archivo:

29	100	11112222-juan perez-gastritis

Figura 7. Ejemplo de archivo

Operaciones del Sistema de Archivos

En este sistema de archivos se podrá realizar operaciones de lectura y escritura de los usuarios almacenados. Se usa como llave la cédula. Adicionalmente se podrá añadir más información a un usuario ya existente.

Implementación del espacio de almacenamiento

Se agruparán los bytes en conjuntos llamados bloques. De manera que el espacio de almacenamiento es un vector de bloques alojado en la memoria principal.
Para este proyecto cada bloque es de 16 bytes y hay 32 bloques en total. Sin embargo, por simplicidad este ejemplo tiene 8 bloques en total. El primer bloque es el superbloque que contiene información de la unidad de almacenamiento:

- Total de bloques
- Bloques disponibles
- Tamaño del bloque (en bytes)

El espacio de almacenamiento comienza con el bloque 2. Esto se ejemplifica en la siguiente imagen (cada fila es un bloque y cada celda son 4 bytes):

Total de bloques k = 8	Bloques disponibles 7	Tamaño del bloque p = 4 bytes	Prohibido
disponible	disponible	disponible	disponible
disponible	disponible	disponible	disponible
disponible	disponible	disponible	disponible
disponible	disponible	disponible	disponible
disponible	disponible	disponible	disponible
disponible	disponible	disponible	disponible
disponible	disponible	disponible	disponible

Figura 8. Almacenamiento en disco

Nota: Dado que el superbloque ocupa 1 bloque, el tamaño de cada bloque es 4 bytes.

Implementación de archivos

Se utilizará una implementación de lista enlazada utilizando una tabla en memoria. Esta tabla de memoria se carga en la memoria RAM. La tabla de memoria es un vector de tamaño k, es decir el total de bloques.
El número de bloque se usa como índice.

Tabla de bloque inicial

Cuando se agrega un nuevo usuario (archivo) se guarda la dirección del primer bloque correspondiente. Esta información se guarda en una estructura de llave-valor cuya llave es el id del archivo, en este caso, la cédula del usuario a guardar; el valor es la dirección del primer bloque a leer.
De manera que si tenemos un usuario con cédula 11112222 al guardarlo ocupa los bloques 3, 5 y 7, un usuario con cédula 33334444 que ocupa los bloques 4,6 y 8 y un usuario con cédula 55556666 que ocupa los bloques 9 y 10, entonces esta estructura de datos lucirá de esta manera:

Llave (cédula)	Valor (dirección de bloque)
11112222	3
33334444	4
55556666	9

Figura 9. Tabla de bloque inicial

De manera que si se quiere leer alguno de esos archivos solo hay que ver la dirección del bloque inicial.

Lectura de archivo

Consiste en leer la dirección inicial en la tabla de bloque inicial. Posteriormente buscar los bloques que corresponden a ese archivo en la tabla de memoria.

Suponga que hay dos archivos llamados Archivo A y Archivo B. Se sabe que el archivo A empieza en el bloque 1 y el Archivo B empieza en el bloque 3. Entonces se lee ese bloque de memoria. Una vez leído, se vuelve a la tabla de memoria y se lee la posición del vector con el índice del bloque.

De manera que:

- El Archivo A está compuesto por los bloques 1, 2 y 6.
- El Archivo B está compuesto por los bloques 3 y 4.

La siguiente tabla resume el ejemplo anterior:

Posiciones	Siguiente bloque a leer
0	Prohibido
1	2
2	6
3	4
4	-1
5	
6	-1
7	

Figura 10. Tabla de direcciones de archivos

Inserción de un nuevo archivo

Se busca en la tabla la primera posición disponible y se almacena en su respectivo bloque en el disco duro. Si se necesita escribir en más bloques se repite el proceso y se busca el siguiente índice en la tabla de memoria disponible para escribir.

A partir de la tabla de la sección anterior, suponga que se quiere escribir un Archivo C con un tamaño de 2 bloques. La tabla de memoria quedaría de la siguiente manera.

Posición	Siguiente bloque a leer
0	Prohibido
1	2
2	6
3	4
4	-1
5	7
6	-1
7	-1

Figura 11. Inserción de archivo en tabla de direcciones de archivos

Esto debido a que la primera posición disponible era la 5 y la siguiente es la 7.

Consideraciones:

- Si no hay suficientes bloques disponibles para escribir un archivo, no se realiza la operación.
- Añadir más información a un usuario ya existente (append) consiste en sobrescribir el -1 al final del último bloque de ese archivo con una dirección de un nuevo bloque disponible y continuar insertando en el siguiente bloque disponible.

Almacenamiento físico

Un usuario (archivo) no necesariamente va a medir bloques exactos. De manera que si quedan bytes disponibles al guardar un archivo (usuario), se utiliza la diferencia de atributos de archivo de tamaño actual y tamaño máximo para determinar si podemos aprovechar esos bytes disponibles. Una vez lleno este bloque faltante, se continúa con otro bloque vacío.

Suponga que el estado del almacenamiento es el siguiente:

Total de bloques k = 8	Bloques disponibles 4	Tamaño del bloque p = 4 bytes	Prohibido
Archivo A	Archivo A	Archivo A	Archivo A
Archivo A	disponible	disponible	disponible
Archivo B	Archivo B	Archivo B	Archivo B
disponible	disponible	disponible	disponible
disponible	disponible	disponible	disponible
disponible	disponible	disponible	disponible
disponible	disponible	disponible	disponible

Figura 12. Estado inicial de almacenamiento en disco

Como se puede observar hay un Archivo A ubicado en los bloques 1 y 2 y un Archivo B ubicado en el bloque 3.

Si se quiere hacer un append a ese usuario del archivo A, la tabla de memoria dirá que debe terminar el bloque 3 que aún tiene bytes disponibles y luego escribir en el bloque 4. De manera que queda así:

Total de bloques k = 8	Bloques disponibles 3	Tamaño del bloque p = 4 bytes	Prohibido
Archivo A	Archivo A	Archivo A	Archivo A
Archivo A	Archivo A	Archivo A	Archivo A
Archivo B	Archivo B	Archivo B	Archivo B
Archivo A	disponible	disponible	disponible
disponible	disponible	disponible	disponible
disponible	disponible	disponible	disponible
disponible	disponible	disponible	disponible

Figura 13. Estado final de almacenamiento en disco

Nota: Recordar que cada vez que se hace un append hay que actualizar los atributos del archivo ubicados en el primer bloque correspondiente.

7. Sistema de Bitácoras

Se guardarán registros de todas las solicitudes, así como el *status* de las respuestas. Cada registro llevará la fecha y hora en la que fue guardado. Por lo tanto, cada vez que accedemos al archivo creado mediante logging se observarán los distintos mensajes de registro con sus descripciones respectivas. En un inicio para configurar el archivo en el que se escribirán todos los registros, se recurre a una inicialización del mismo mediante la función *loggin.basicConfig,* la cual realiza una configuración básica para el sistema de logging creando una *StreamHandler* con un *Formatter* predeterminado y agregándolo al logger raíz. Las funciones debug, info,error llamarán automáticamente a basicConfig automáticamente si no se definen gestores para el logger raíz.

8. TLB

Memoria virtual y TLB

Consideraciones

- Cada nodo debe tener un sistema de memoria virtual y page table.
- Se debe implementar primero el File System.

Almacenamiento físico

Es el almacenamiento en disco. Se simulará el almacenamiento físico en la memoria RAM.
Para este proyecto se acordó en la clase usar:
Cada bloque: 16 bytes.
Total de bloques: 32
Tamaño de TLB: 4 entradas.
Página virtual: 2 bloques.
Marco de página: 2 bloques.

Por simplicidad a la hora de explicar se usarán los siguientes datos:
Cada bloque: 16 bytes
Total de bloques: 16
Tamaño de TLB: 4 páginas virtuales
Página virtual: 2 bloques.
Marco de página: 2 bloques.

Memoria Física en Bloques

Esta es una tabla con el propósito de ejemplificar, no representa los valores acordados en el grupo.

0
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15

Figura 14. Almacenamiento en disco (en bloques)

Páginas virtuales del almacenamiento

Son una abstracción del almacenamiento en disco. Consiste en dividirlo en cajitas de igual tamaño.
Cada página virtual: 2 bloques
Número de páginas virtuales: 8
Esta es una tabla con el propósito de ejemplificar, no representa los valores acordados en el grupo.

0: 0-1
1: 2-3
2: 4-5
3: 6-7
4: 8-9
5: 10-11
6: 12-13
7: 14-15

Figura 15. Almacenamiento en disco (en páginas virtuales)

Si leemos la página 0 estamos leyendo los bloques 0 y 1.
Si leemos la página 1 estamos leyendo los bloques 2 y 3.
Si leemos la página 2 estamos leyendo los bloques 3 y 4.

MMU

Es una función que nos traduce de direcciones virtuales a físicas.
Es decir si la entrada de la función es la página 1 la salida es la dirección del bloque 2 y 3 ya se sabe que debe leer dos bloques por que ese es el tamaño de una página (direcciones).

Marco de página

Se ubica en la memoria RAM. Es un conjunto de páginas. La diferencia es que son menos marcos de página que páginas virtuales.
Marcos de página: 4

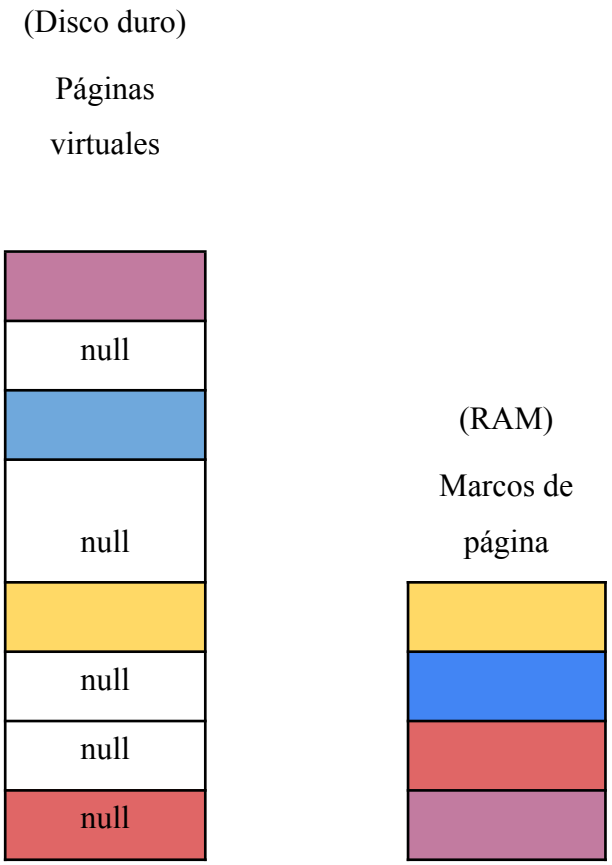

Figura 16. Relación páginas virtuales y marcos de página

Los marcos de página es el espacio en memoria principal que podemos utilizar para cargar páginas del disco. Como es más pequeña hay que compartir el espacio en la memoria principal (reemplazar marcos de página).

En este ejemplo los colores indican dónde está guardada esa página virtual en algún marco de página.

Page Table/TLB

Es un espacio para lista de entradas. Esta page table vive en memoria.

Número de Página Virtual	Bit Referenciada	Marco de página

Figura 17. Entrada de la TLB

Para este proyecto el TLB y la page table son lo mismo. En la vida real la TLB es un dispositivo de hardware. La TLB es una page table más pequeña.
Para este proyecto, se usará una TLB de 4 páginas virtuales. Nuestra TLB es la siguiente:

Número de página virtual	Marco de página	Bit de referencia
0	0	0
1	1	1
2	2	0
3	3	1

Figura 18. TLB con capacidad para 4 marcos de página.

Esta tabla nos permite saber fácilmente si una página tiene un marco de página referenciado en memoria.

La primera vez que se carga un marco de página no se marca el bit de referencia, sino hasta la segunda vez que se accede al marco de página.

Nota: Si no existe el usuario que hay que guardar, se procede a guardar directamente con el file system. De manera que los marcos de página cargan páginas virtuales que son para lectura.

Si sabemos que el usuario (archivo) que estamos buscando está en los bloques 0, 5, y 7. Entonces necesitaremos leer las páginas virtuales 0, 2 y 3. De manera que buscamos en la tabla si estas páginas virtuales tienen un marco de página referenciado. Si no, la TLB debe desalojar marcos de página para referenciar las páginas virtuales que necesitamos.

Reemplazo de Páginas

Algoritmo de segunda Oportunidad

Se va a utilizar el algoritmo de segunda oportunidad porque aprovecha la simpleza de FIFO pero corrige la principal desventaja de FIFO: el desechar tablas que son utilizadas frecuentemente. Esto se debe a que inspecciona el bit R (bit de referencia) de manera que estas no se borran en primera instancia si tienen el bit en 1, sino que se vuelven a colocar en la cola.

De manera que necesitamos una estructura de datos del tamaño de entradas del marco de páginas. Suponga que el marco de páginas tiene un tamaño de 4 páginas. La estructura luce de esta manera:

Estructura FIFO para algoritmo de segunda oportunidad

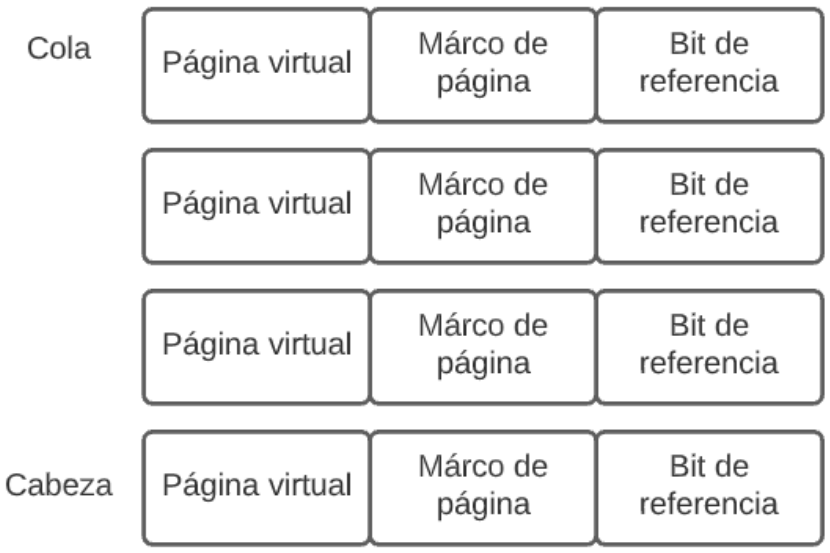

Figura 19. Estado inicial TLB

Esta estructura es necesaria para actualizar las direcciones cuando se deseche un marco de página.

Suponga que actualmente los marcos de páginas están llenos y la tabla es la siguiente:

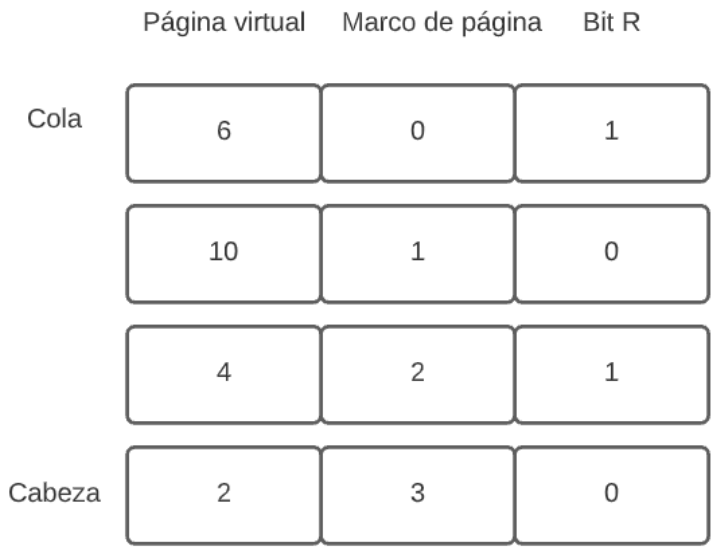

Figura 20. Estado TLB al cargar la página virtual 6

Si ahora se debe cargar la página 7 en algún marco se busca en esta estructura cuál es la más vieja. El marco más viejo es el que está en la cabeza de la cola. Como el bit de referencia es 0 significa que la página no ha sido usada, por lo que se elimina.
Ahora el marco 3 está disponible por lo que la página 7 se carga en el marco de la página 3. De manera que queda de la siguiente manera:

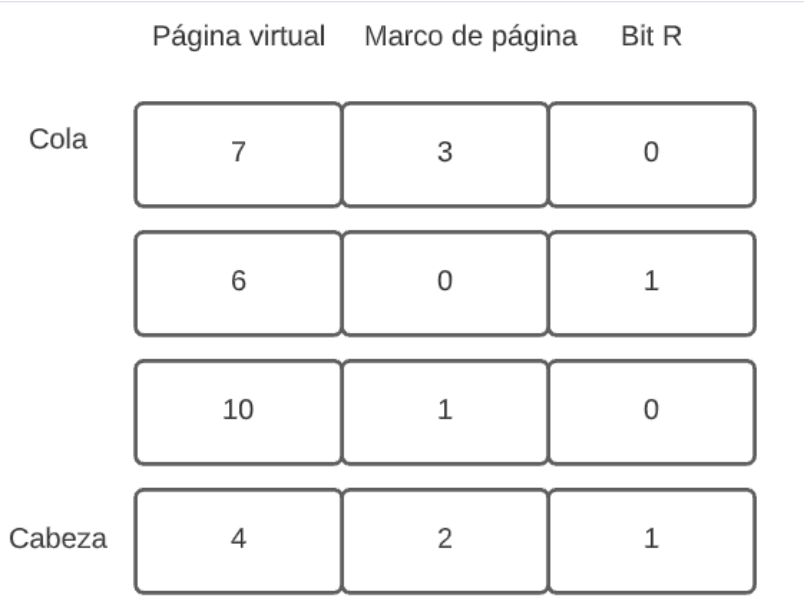

Figura 21. Estado TLB al cargar la página virtual 7

Si ahora se quiere cargar la página 9 se debe eliminar la más vieja que es la que está en la cabeza. El marco de página correspondiente sería el 2. Sin embargo, el bit de referencia es 1. Por lo que se debe sacar de la cabeza y desalojar la página 2. Ahora se puede cargar la página 9 en el marco 2.

La cola sería la siguiente:

Figura 22. Estado TLB al cargar la página virtual 9

Sin embargo, como el marco desalojado tenía el bit de referencia en 1, esta página debe volver al final de la cola. Para esto hay que desalojar la página que está en la cabeza, es decir la que está en el marco 1.

Cuando hay que volver a meter una página en la cola el bit de referencia se pone en 0. De manera que el estado de la cola es el siguiente:

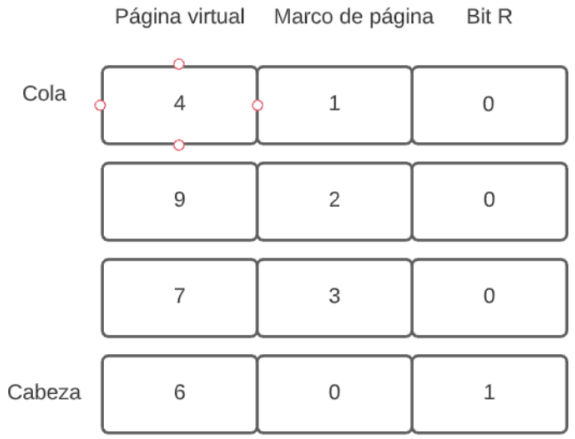

Figura 23. Estado TLB al cargar la página virtual 4

Consideraciones:

- Cuando se desaloja un marco de página se debe actualizar la tabla de páginas para que la próxima vez que se intente leer buscar la dirección de marco de página de esa página virtual se produzca un fallo. Esto ocasionará que se vuelva a cargar esa página y se ingrese al final de la cola.
- Cuando se carga una página virtual en un marco de página hay que actualizar la tabla de páginas, es decir cuando se coloca al final de la cola.
- Los marcos de página son solo regiones de memoria de tamaño de páginas que por simpleza estamos indexando del 0 a n - 1, con n como el número de marcos de página.

9. Direcciones de los bloques

Cada vez que la interfaz le envía una solicitud de escritura a los nodos, el nodo le responde con la cantidad de bloques disponibles que quedan, este dato es guardado por la interfaz en una estructura de datos para tomarlo en cuenta para futuras escrituras.

Referencias:

Sajip, V. (s. f.). *Cómo Hacer Registros (Logging) — documentación de Python - 3.9.5*. Python. Recuperado 17 de mayo de 2021, de https://docs.python.org/es/3/howto/logging.html