CASO 2 INFRAESTRUCTURA COMPUTACIONAL

Integrantes:

* Lindsay Pinto Morato 202023138
* Juan Camilo Gonzalez Cabrera 201911030
* Leonidas Villamil Pachon 202013910

INTRODUCCION

La intención de este informe es presentar la implementación de un proyecto que simula el sistema de paginación utilizando el algoritmo de envejecimiento de un sistema. Descrito en el libro Sistemas Operativos Modernos, capítulo 3 – sección 3.4.7. Se evidenciará su comportamiento con un simple programa que suma dos matrices en el que se almacenan las entradas de dichas matrices en memoria virtual. Un componente crítico de los sistemas operativos es el sistema de paginación, que administra la memoria de la computadora y garantiza que se use de manera eficiente. El sistema de paginación divide la memoria en bloques de tamaño fijo llamados páginas, que se asignan dinámicamente a los programas a medida que se ejecutan. Este proceso permite que los programas utilicen la memoria de manera más eficiente y evita que se produzcan errores de memoria.

El sistema de paginación también se encarga de realizar el intercambio de páginas entre la memoria principal y la memoria secundaria, lo que permite que los programas se ejecuten incluso si la memoria principal está llena. Esto es esencial para garantizar que los programas se ejecuten sin problemas y que la computadora tenga un rendimiento óptimo.

DESCRIPCIÓN DEL SISTEMA DE PAGINACIÓN

Cuando un proceso necesita acceder a una página de memoria que no se encuentra actualmente en la memoria principal, se produce un "fallo de página". En este caso, el sistema de paginación debe buscar la página en el almacenamiento secundario, como un disco duro, y cargarla en la memoria principal. Si no hay suficiente espacio libre en la memoria principal para acomodar la página solicitada, el sistema de paginación utiliza una técnica llamada "reemplazo de página" para liberar espacio y hacer espacio para la página solicitada.

PLAN DE IMPLEMENTACIÓN

Para poder simular el sistema de paginación implementamos dos Threads. El thread de actualización (implementado en la clase ThActualizacion) el cual se crea con el ArrayList de referencias, la memoria virtual, la tabla de paginación, el número de fallos inicializado en 0 y un Arraylist nuevo llamado referenciasRemplazo. Cuando este thead se corre ejecuta mediante el metodo run() el cual solo ejecuta el método actualizar(). En actualizar recorremos la el ArrayList de referencias una por una y para las cuales primero accederemos a la página de referencia en la memoria virtual y luego identificamos si hay un fallo en la tabla de paginación de la página de referencia, si hay un error entonces sumamos un fallo a numFallos y agregamos esta referencia al ArrayList de referenciasRemplazo, si no hay error no hacemos más. Este thread se ejecuta cada 2 milisegundos y cuando termina de recorrer todas las referencias deja vacío el ArrayList de referencias. Luego el otro thread el de envejecimiento (implementado en la clase ThEnvejecimiento) el cual se crea con el ArrayList de referencias, la memoria virtual, la tabla de paginación, memoria física y el Arraylist referenciasRemplazo. Cuando este thead se corre ejecuta mediante el metodo run() el cual solo ejecuta el metodo envejecimiento(). En envejecimiento mediante un while, que termina cuando tanto el ArrayList referencias y el ArrayList referenciasRemplazo quedan vacíos, si hay una referencia en el ArrayList referenciasRemplazo toma la primera referencia y obtiene un indice de remplazar la página de referencia en memoria real a la página de referencia en la memoria virtual, luego agrega la página de referencia a la tabla de paginación teniendo en cuenta el índice del remplazo, por ultimo retira esa referencia del ArrayList referenciasRemplazo y verifica si debe terminar el while. Este thread se ejecuta cada 2 milisegundos.

Se utilizo sincronización en la implementación de los Threads, justamente en los datos para los cuales estos comparten información a la hora de correrse. Así que para líneas de código que afectan a la memoria virtual, el ArrayList de referenciasRemplazo y el ArrayList de referencias se utilizó el método synchronized() para garantizar la exclusión mutua cuando se estén modificando.

Se decidió implementar el caso en Java, y asi aprovechar el concepto de clases para modelar los diferentes componentes del sistema. En la implementación se han creado cinco clases: Memoria Real, Memoria Virtual, Pagina, Principal y Tabla Pagina. A continuación, se da una breve descripción de la clase, sus atributos y métodos:

* Página: Esta clase simula una pagina o marco de pagina de las memorias virtual y principal. La pagina cuenta con tres atributos: id que identifica la página, un booleano que expresa si la pagina esta en memoria real o no. Y los bits, que representa el contador de cada pagina en el que se almacena la frecuencia en la que dicha página es usada. Los métodos de esta clase son getters y setters de los atributos discutidos anteriormente.
* Tabla Paginación: Esta clase simula la tabla de paginas del sistema de paginación. Dicha tabla tiene la misma cantidad de entradas que páginas en la memoria virtual. Dicho valor es ingresado por el usuario. El atributo de esta clase es un arrayList de enteros en la que el índice representa el id de la página y el valor si la página está en memoria principal su posición o no (-1). El método SetPagina(id, numPagina) actualiza la tabla de paginación, dado el id y la posición de la página en memoria principal.
* Memoria Virtual: esta clase simula la memoria virtual del sistema. El atributo de dicha clase es una arrayList de páginas. El método accederPagina(id) implementa el algoritmo de envejecimiento modificando los bits de las páginas por cada acceso a una página.
* Memoria Real: Esta clase simula la memoria principal de un computador. Su atributo es un arraylist de páginas. El número de páginas o marco de páginas es dado por el usuario. Inicialmente todas las posiciones del arreglo están vacías. Esta clase tiene dos métodos. mmarcoLibre() retorna el primer marco libre de la memoria, es decir la primera posición del arreglo en la que su valor sea nulo. El método RemplazarPagina(Pagina página) simula la acción de remplazo de página. Si existe un marco libre se dispone ahí la página dada por parámetro. Si no hay marcos libres ocurre un fallo de página. Utilizando el algoritmo de envejecimiento se encuentra la página menos usada dentro de la memoria y se remplaza quitando la página menos usada de la memoria y poniendo en su posición la página dada por parámetro.
* Principal: Esta es la clase en la que se implementan los dos modos del programa. Inicialmente los parámetros de tamaño de página, tamaño de memoria virtual, memoria principal y tamaño de entero se leen de un archivo. A continuación, se describe el funcionamiento de los modos:
  + Modo 1: Este modo genera todas las referencias entre las entradas de las matrices y la memoria virtual. Las matrices se almacenan por filas, es decir, en la memoria los elementos consecutivos de una fila están almacenados uno tras otro. Asi la posición de una entrada de una matriz depende el tamaño de la matriz el tamaño de pagina y el tamaño de entero. Asi la posición en memoria virtual de este dado por la formula:

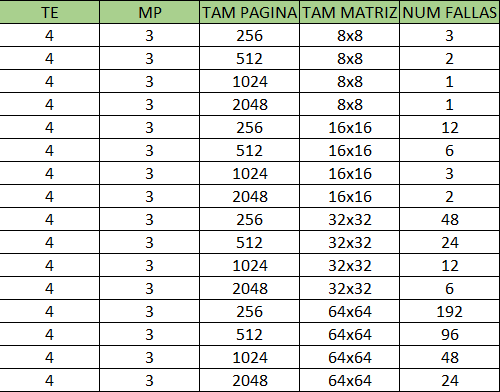
Note que la posición va a estar en la posición de . Es se debe a que las matrices son del mismo tamaño y la información de una matriz se guarda consecutivamente. Este número representa donde esta almacenada la entrada de la matriz, pero no en que página ni su posición en dicha página.

Para calcular la pagina a la que corresponde la entrada de la matriz se puede dividir su posición entre en tamaño de cada página. La parte entera representara entonces el numero de pagina y el residuo su posición en dicha página. La pagina a la que pertenece cada entrada es almacena en una lista con la que después se va a llenar la memoria virtual

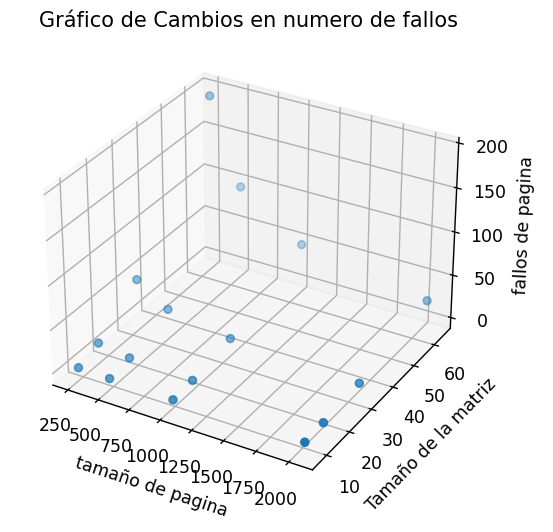
* + Modo 2: En este modo se simulan los fallos de página, para esto se recorre la lista generada en el modo 1. Note que el orden de dicha lista simula la suma de las matrices, primero se accede a la misma entrada en las dos matrices A y B y luego se accede a la misma entrada, pero de la matriz C en donde se almacena la suma. Lo primero que ocurre es un llamado al método accederPagina de la memoria virtual, en la que se actualizan los contadores de frecuencia de las páginas. Seguidamente se busca la pagina en la tabla de páginas, si el valor en la tabla es -1. Entonces ocurre el fallo de página. Por lo que se llama al método reemplazar pagina de la memoria principal y por ultimo se actualiza la información en la tabla de páginas.

RESULTADOS

Se ha utilizado los dos modos del programa para simular el numero de fallas con distintos parámetros. Se ha decidido dejar el tamaño del entero en 4 bits. Y el numero de marcos de pagina en 3. Los valores del tamaño de página varían entre 256 y 2048. Los tamaños de las matrices entre 8x8 y 64x64.



Para entender mejor la información de esta tabla se ha decidido graficar los datos:



Como se puede evidenciar por el grafico, los tamaños de pagina mas pequeños son los que generan el mayor numero de fallos de páginas. Esto se debe a que un menor tamaño implica una mayor cantidad de paginas para almacenar toda la información en la memoria virtual. Esto a su vez implica mayor numero de fallos ya que el porcentaje de páginas en memoria virtual relacionadas con un marco de pagina en memoria principal es más bajo. De la misma forma al aumentar el tamaño de la pagina se crean menos paginas y por lo tanto menos fallos, es por esto por lo que la cantidad de fallos con un tamaño de página de 2048 es la menor en todas las matrices.

Asi mismo entre mas grande es la matriz mas fallos se presentan, esto se debe a que el numero de entradas aumenta y por lo tanto se deben crear mas paginas lo que implica mas fallos como se menciono anteriormente.

Se evidencia que el numero de fallas y el tamaño de la página son inversamente proporcionales. En el sentido de que cuando se duplica el tamaño de pagina se reduce a la mitad el numero de fallas. Asi mismo el tamaño de la matriz es directamente proporcional a el numero de fallos. Note que cuando se duplica el numero de filas y columnas se esta cuadriplicando el numero de entradas en la matriz. Se puede evidenciar dicha cuadruplicación en el numero de fallas con un tamaño de pagina de 256. En la que se da una secuencia de 3, 12, 48, 192 fallos de página.

Se puede entonces concluir que tanto el tamaño de página y el tamaño de la matriz son intrínsecos al numero de fallos de página y esta relación es de carácter lineal. Ambas por la misma razón, al alterar estos valores se deben crear mas o menos paginas en memoria virtual lo cual lleva a mas fallos. Se podría suponer que dicho comportamiento es parecido si se altera el tamaño del entero y el numero de marco de páginas. Si el tamaño de entero cambia entonces menos entradas de la matriz van a entrar en la misma pagina y por lo tanto el numero de paginas aumenta y asi el numero de fallas. Por otro lado, si el número de marcos de pagina en memoria real aumenta mas paginas en la memoria virtual van a tener relación con un marco y por lo tanto menos fallos de paginas se deben presentar. Entonces el tamaño de entero tiene una relación directamente proporcional con el numero de fallo y el numero de marcos una relación inversamente proporcional.

CASOS PARA CONSIDERAR

* ¿Qué pasaría si las matrices fueran almacenadas siguiendo column-major order?

No debería haber cambio alguno en la cantidad de fallos, si se cambia la forma en la que se suman las matrices, iterando por fila y después columnas. Si no es asi posiblemente se crearían mas fallos de paginas porque dos elementos consecutivos de una fila pueden estar en distintas paginas con mayor frecuencia lo cual implica mas cambios de páginas y más fallas.

* ¿Cómo varía el número de fallas de página si el algoritmo estudiado es el de multiplicación de matrices?

El número de fallas aumentaría considerablemente ya que para calcular una entrada de la matriz resultante se deben acceder a varias entradas de las matrices multiplicadas. Para ser precisos 2n datos por entrada de la matriz de resultado. Donde n es la dimensión de la matriz siendo esta cuadrada. Puede que sea posible reducir los fallos si se almacena de forma más efectiva los datos. En un sentido en el que se tenga en cuenta que para multiplicar matrices se debe acceder tanto a la fila de la entrada como a la columna de la otra entrada.

CONCLUSIONES

El informe presentó la implementación de un proyecto que simula el sistema de paginación utilizando el algoritmo de envejecimiento. Se evidenció su comportamiento con un simple programa que suma dos matrices en el que se almacenan las entradas de dichas matrices en memoria virtual. Notamos que el sistema de paginación es esencial para garantizar que los programas se ejecuten sin problemas y que la computadora tenga un rendimiento óptimo. Se reforzaron los conceptos de memoria virtual, memoria física, tabla de paginación y pagina. También mediante pruebas se evidencio la relación inversamente proporcional entre el tamaño de la página y el número de fallos de paginación, además que el número de fallos de paginación es directamente proporcional al tamaño de la matriz. En resumen, este informe proporciona información valiosa sobre el sistema de paginación y su importancia en los sistemas operativos, además de una implementación sencilla.