Informe

Jonathan Obed Cabanzo Certuche - 201911749
Juan Manuel Rodriguez - 202013372
Simulador de memoria virtual
Universidad de los Andes, Bogotá, Colombia
{jo.cabanzo, j.rodriguezd}@uniandes.edu.co
Fecha de presentación: Abril 14 de 2023

Tabla de contenido

1 Descripción del modo 1

Para la creación de las referencias se debe tener en cuenta el tamaño que ocupan cada una de las matrices, que afecta en el numero de paginas necesarias para guardar todas las matrices en la tabla de páginas, por lo que este modo sigue el siguiente algoritmo:

- Se calcula el tamaño que ocupa una fila multiplicando el tamaño de entero por el numero de columnas que hay en una fila
- Se calcula el numero de paginas que caben en una página dividiendo el tamaño de la pagina por el tamaño de una fila anteriormente calculado
- Se calcula el número de páginas que ocupa una matriz dividiendo el numero de filas que tiene una matriz entre las filas que caben en una pagina

```
/*
    Ejemplo:
        Tamaño de pagina:  256 Bytes
        Tamaño de entero: 4 Bytes
        Filas = 8
        Columnas = 8
*/

// 8*4 = 32 Bytes por fila
int tamanoFila = numeroColumnas * tamanoEntero;
// 256 / 32 = 8 filas por pagina
int filasEnUnaPagina = tamanoPagina / tamanoFila;
 // 8 / 8 = 1 pagina por matriz
int numeroPaginasNecesariasPorMatriz = numeroFilas / filasEnUnaPagina;
```

Para el caso base donde todas las matrices caben en una sola página, se realizan dos ciclos anidados con el número de filas y el número de columnas respectivamente, se crean todas las referencias con cada una de ellas en la misma página:

```java
if (filasEnUnaPagina >= numeroFilas*3) {
    filasEnUnaPagina = numeroFilas;

    int desplazamiento = 0;

    for (int j = 0; j < numeroFilas; j++) {

        for (int k = 0; k < numeroColumnas; k++) {


            String refA =  "0" + "," + desplazamiento;
            String refB = "0" + "," + desplazamiento;
            String refC = "0" + "," + desplazamiento;

            String valor = "[A-" + j + "-" + k + "]"
                                 + "," + refA + "\n"
                                 + "[B-" + j + "-" + k + "]"
                                 + "," + refB + "\n"
                                 + "[C-" + j + "-" + k + "]"
                                 + "," + refC + "\n";

            referencias += valor;
            desplazamiento += tamanoEntero;

            numeroReferencias += 3;

        }
        filaActual++;
    }

    tablaPaginas = new TablaPaginas(numeroPaginas:2);

}
```

Para el caso base donde todas las matrices caben en dos páginas, se realizan dos ciclos anidados con el número de filas y el número de columnas respectivamente. Las referencias de la matriz A y la matriz B se crean con la página en 0 y las referencias de la matriz C se crean con la página 1:

```
// Cuanto dos matrices caben en una pagina
else if (filasEnUnaPagina >= numeroFilas*2) {
    numeroPaginasNecesariasPorMatriz = 1;
    filasEnUnaPagina = numeroFilas;

    int desplazamiento = 0;
    int paginaActual = 0;

    for (int j = 0; j < numeroFilas; j++) {

        for (int k = 0; k < numeroColumnas; k++) {


            String refA = paginaActual + "," + desplazamiento;
            String refB = paginaActual + "," + desplazamiento;
            String refC = paginaActual + 1 + "," + desplazamiento;

            String valor = "[A-" + j + "-" + k + "]"
                                + "," + refA + "\n"
                                + "[B-" + j + "-" + k + "]"
                                + "," + refB + "\n"
                                + "[C-" + j + "-" + k + "]"
                                + "," + refC + "\n";

            referencias += valor;
            desplazamiento += tamanoEntero;

            numeroReferencias += 3;

        }
        filaActual++;
    }

    tablaPaginas = new TablaPaginas(numeroPaginas:2);

}
```

En los demás casos dado que ya encontramos el número de páginas que ocupa una matriz, hacemos un ciclo que termine en este número (número de páginas por matriz) , y por cada ciclo creamos las tres referencias que corresponden a los 3 llamados a las diferentes matrices. Las referencias de la matriz A siempre apuntan a la página **i,** para las de la matriz B se apuntan a la página **i + numeroPaginasNecesariasPorMatriz,** dado que la matriz B debe dejar el espacio necesario (las páginas necesarias) que ocupa la matriz A, de manera similar con la matriz C que debe dejar espacio para las matrices A y B.

```
for (int i = 0; i < numeroPaginasNecesariasPorMatriz ; i++) {

    int desplazamiento = 0;

    for (int j = 0; j < filasEnUnaPagina; j++) {

        for (int k = 0; k < numeroColumnas; k++) {


            String refA = i + "," + desplazamiento;
            String refB = (i + numeroPaginasNecesariasPorMatriz) + "," + desplazamiento;
            String refC = (i + 2*numeroPaginasNecesariasPorMatriz) + "," + desplazamiento;

            String valor = "[A-" + filaActual + "-" + k + "]"
                                 + "," + refA + "\n"
                                 + "[B-" + filaActual + "-" + k + "]"
                                 + "," + refB + "\n"
                                 + "[C-" + filaActual + "-" + k + "]"
                                 + "," + refC + "\n";

            referencias += valor;
            desplazamiento += tamanoEntero;

            numeroReferencias += 3;

        }
        filaActual++;

    }
}
tablaPaginas = new TablaPaginas(numeroPaginasNecesariasPorMatriz * 3);
```

2 Descripción de las estructuras de datos

A continuación, se muestra el manejo de estructuras de datos necesarios para la simulación de memoria virtual.

2.1 Marcos de página (MarcosPagina)

Es una clase que tiene una lista que guarda enteros. Cada índice en la lista representa un marco de página, y su valor entero representa el numero de página de memoria virtual que esta guardado. Se inicializan sus valores en -1, dado que al principio ningún marco tiene guardada alguna página.

Los marcos se actualizan cada que se ingresa una nueva página, esto implica ingresar al marco (índice) la página que se quiera (valor entero), también se actualizan en caso de que se quiera retirar una pagina en memoria, en dado caso se registra en el marco (índice) el valor entero -1, para indicar que ahora está vacío o se ingresa el valor de la nueva página que se está ingresando. Cualquiera de estas actualizaciones implica que también se actualice la tabla de páginas.

2.2 Tabla de páginas (TablaPaginas)

Es una clase que tiene una lista que guarda enteros. Cada índice en la lista representa una página en memoria virtual, y su valor entero representa el marco de pagina donde se encuentra guardado. Se inicializan sus valores en -1, dado que al principio ninguna página esta guardada en los marcos.

Las paginas se actualizan cada que alguna de estas entra a los marcos, esto es ingresar a la página (índice) el número de marco en el que ahora está (valor entero). También se actualiza cuando la pagina ya no está en los marcos, en dado caso la página en cuestión (índice) se le ingresa el valor entero -1, para indicar que ya no está en memoria.

2.3 Tabla de tiempo (TablaTiempo)

Es la clase donde se maneja el algoritmo de envejecimiento para los marcos de página, tiene una lista de String que son los "Bytes" que manejan el tiempo de vejez de cada marco.

En caso de que ocurra un fallo de página se ejecuta el método **actualizarTiempo** que recibe por parámetro el marco que se acaba de actualizar, su función es agregar un 1 al el tiempo de vejez de este marco, para indicar que es el marco más "reciente". Así mismo cambio la variable **actualizaciónPendiente** a 1, para indicar que el tiempo de todos los demás marcos debe avanzar.

Para avanzar el tiempo se ejecuta la función **avanzarTiempo** cuyo propósito es agregar un 0 a la tabla de tiempos de todos los marcos. Esta función solo se ejecutará cuando ocurre un fallo de página (cuando **actualizaciónPendiente** es igual a 1), dado que si se ejecutara cuando no hay un fallo de página puede que la tabla de tiempos de todos los marcos se reinicie en "00000000", perdiendo así el valor marco mas viejo.

3 Esquema de sincronización

Para entender la sincronización se deben entender en primera instancia los Threads que se sincronizan, estos son:

- **Actualizador:** Esta es una clase que extiende de Thread cuya función es leer una por una todas las referencias. Por cada referencia debe verificar si la página en cuestión está en la tabla de páginas, si no es así, genera un fallo de página. En este caso se debe buscar en la tabla de tiempos cual es el marco mas "viejo", y con este actualizar la tabla de páginas, y los marcos de página. Este thread se ejecuta cada 2 milisegundos.
- **Envejecimiento:** Esta es una clase que extiende de Thread cura función es avanzar el tiempo de la tabla de tiempos (Agregar un 0 a todos los tiempos de la tabla). Este thread se ejecuta cada milisegundo.

La sincronización se da cuando ocurre un fallo de pagina en el Actualizador, dado que una vez ocurre el fallo este debe esperar a que el thread de Envejecimiento avance el tiempo de la tabla de tiempos para evitar inconsistencias. Asimismo, el thread de Envejecimiento debe esperar a que ocurra un fallo de pagina para poder actualizarse, para evitar que se tome la memoria y simplemente avance el tiempo (Agregue ceros a los marcos de página), sin dejar que el thread Actualizador, actualice la tabla de tiempos.

4 Datos Recopilados

MATRICES 8 X 8	
Tamaño de página: 256 Marcos de página: 4 Tamaño de entero: 4	Numero de Fallos: 3
Tamaño de página: 512 Marcos de página: 4 Tamaño de entero: 4	Numero de Fallos: 2

Tamaño de página: 1024 Marcos de página: 4 Tamaño de entero: 4	Numero de Fallos: 1
Tamaño de página: 2048 Marcos de página: 4 Tamaño de entero: 4	Numero de Fallos: 1

MATRICES 16 X 16	
Tamaño de página: 256 Marcos de página: 4 Tamaño de entero: 4	Numero de Fallos: 12
Tamaño de página: 512 Marcos de página: 4 Tamaño de entero: 4	Numero de Fallos: 6
Tamaño de página: 1024 Marcos de página: 4 Tamaño de entero: 4	Numero de Fallos: 3
Tamaño de página: 2048 Marcos de página: 4 Tamaño de entero: 4	Numero de Fallos: 2

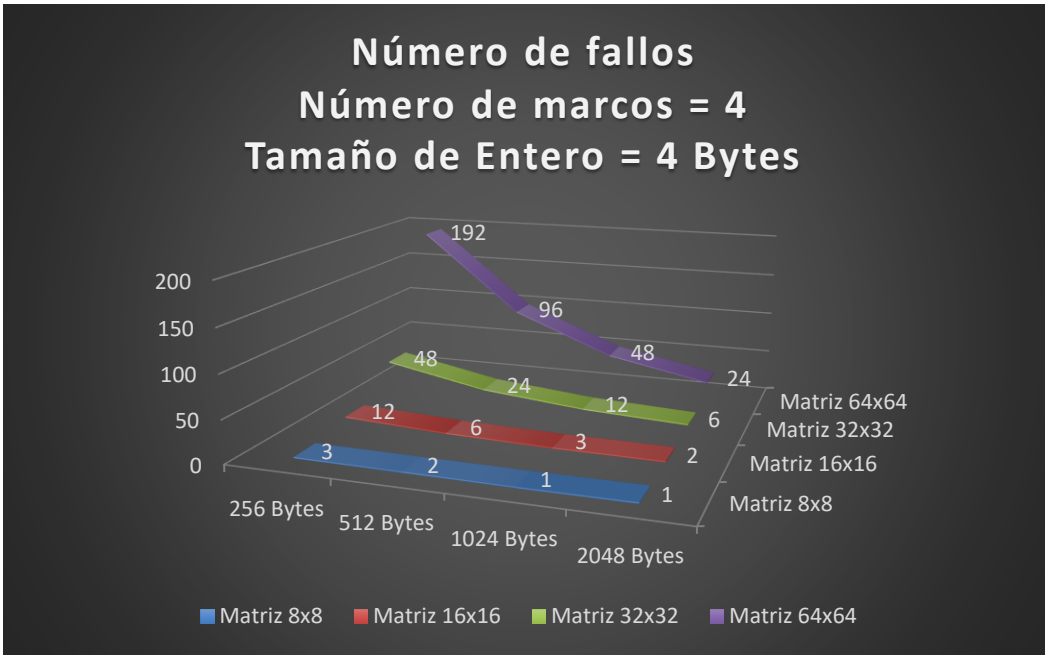
MATRICES 32 X 32	
Tamaño de página: 256 Marcos de página: 4 Tamaño de entero: 4	Numero de Fallos: 48
Tamaño de página: 512 Marcos de página: 4 Tamaño de entero: 4	Numero de Fallos: 24
Tamaño de página: 1024 Marcos de página: 4 Tamaño de entero: 4	Numero de Fallos: 12
Tamaño de página: 2048 Marcos de página: 4 Tamaño de entero: 4	Numero de Fallos: 6

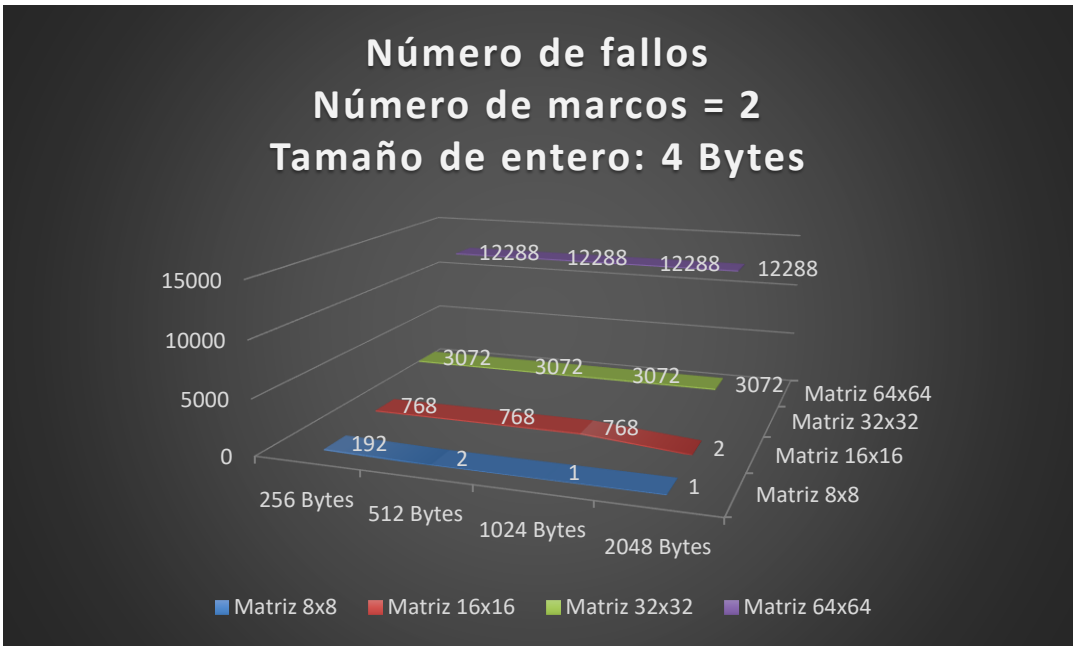
MATRICES 64 X 64

Tamaño de página: 256 Marcos de página: 4 Tamaño de entero: 4	Numero de Fallos: 192
Tamaño de página: 512 Marcos de página: 4 Tamaño de entero: 4	Numero de Fallos: 96
Tamaño de página: 1024 Marcos de página: 4 Tamaño de entero: 4	Numero de Fallos: 48
Tamaño de página: 2048 Marcos de página: 4 Tamaño de entero: 4	Numero de Fallos: 24

5 Graficas de comportamiento

A continuación, se presentan las graficas de variación en el numero de fallos dados ciertos parámetros





6 Análisis de resultados

En la primera grafica se puede observar que la cantidad de fallos aumenta en proporción a la cantidad de filas, columnas y tamaño de página. En un marco de referencia de 4 marcos de pagina y un tamaño de entero de 4 Bytes. Entre menor es el tamaño de página y entre mayor es el tamaño de las matrices, más aumentan la cantidad de fallos.

Se puede observar que, al reducir la cantidad de marcos, la cantidad de fallos tiende a ser el mismo numero de referencias que se generan, dado que constantemente hay que reemplazar los marcos.

7 Dudas

- ¿Qué pasaría si las matrices fueran almacenadas siguiendo column-major order?
  En dado caso la respuesta seria la misma, dado que se generan la misma cantidad de referencias.
- ¿Cómo varía el número de fallas de página si el algoritmo estudiado es el de multiplicación de matrices?
  La multiplicación de matrices requiere un mayor numero de referencias por su complejidad, y aunque el espacio requerido en memoria virtual es el mismo dado que el tamaño de las matrices no ha variado, es probable que se generen mayor cantidad de errores dado que se podrían requerir acceso a paginas que no están en memoria muy constantemente.