Caso 2 INFRACOMP - Informe

Integrantes:

- Pablo Méndez Morales 202210379
- Santiago Najar Gómez 202021647
- Luis Fernando José Ruiz Ortega 202211513

Descripción Algoritmo para crear referencias:

La forma de crear las referencias se describe a continuación. Primero se almacena el tamaño de la página, el número de filas y el número de columnas dados por el usuario. A partir de estos datos se calcula también el número de enteros por página, el número de páginas necesarias y el número de enteros en todas las matrices. Una vez se tienen estos datos se procede a calcular la posición y el desplazamiento en memoria de cada entero de la matriz.

Para hacer esto se llama la función matricesToPaginas(), esta función recorre cada página necesaria y cada posición en la página que es capaz de almacenar un entero y le asigna a una posición de una matriz esa página y desplazamiento. Por ejemplo, a la primera página en la primera posición en la que puede guardar un entero (desplazamiento=0) le asigna F[0][0], una vez hayan sido asignados todos los enteros de F, continua a la matriz M y por último a la matriz R.

Para saber cuándo continuar y cuantos enteros de cierta matriz han sido asignados se utilizan las variables locales listaTamanioMatriz[] y k. listaTamanioMatriz[] guarda la cantidad de enteros de cada matriz que todavía no hay sido asignados, por ejemplo, para la matriz filtro el valor en la lista iniciaría en 9 e iría descendiendo conforme se asignen páginas y desplazamientos. Por otra parte, el entero k indica que matriz está siendo asignada en esa iteración (Filtro, Datos o Resultado).

Entonces, mientras se recorren las páginas y posiciones tambien se recorren las listas usando estas dos variables. Cuando se está en una página y en un desplazamiento dado se utiliza el valor guardado en listaTamanioMatriz[k] para saber en qué entero de la matriz actual se va a asignar. Esta conversión, de número a posición i j de la matriz, se hace con la función numAPosMatriz(). Una vez se tiene la posición se le asigna la página y el desplazamiento en ese instante y se pasa a la siguiente iteración. Este resultado se guarda en la tabla de hash matrizPagina.

Por último, se implementa el funcionamiento del algoritmo que aplica el filtro, solo que en vez de solicitar una posición de una matriz se guarda el llamado en un array. De esta manera se tiene el orden de llamados a memoria y las páginas afectadas, en este plazo también se asignan los bits de lectura y escritura. Una vez se tiene esto se escribe toda la información en el archivo y se guarda.

Descripción de la simulación del sistema de paginación:

Tenemos la clase CalculoDatos, dentro de este archivo de código, métodos como leerArchivoInicial() y loadReferencia() son responsables de leer y manejar las referencias de datos.

Se utilizan varias estructuras de datos, como arrays y hashmaps (memoriaVirtual y memoriaReal), para simular el sistema de paginación. Estas estructuras las vamos actualizando durante la ejecución del programa para reflejar el estado actual de la memoria y las operaciones realizadas.

Se utiliza el algoritmo de "El algoritmo de reemplazo de páginas: no usadas recientemente" haciendo uso de las diferentes categorías (0,1,2,3) para conocer el estado en el que se encuentra la página y hacer los swaps a memoria, específicamente en el método de algoritmoRLU() donde llega al método sabiendo si fue un hit o una falla. En el caso de los hits revisamos en que se encuentra el bit de referencia (0,1,2,3) y luego de eso verificamos en que se encuentra el bit de modificación (R, W) para hacer el cambio. En el caso de la falla vamos buscando las páginas referenciadas en la memoria real hasta encontrar la del bit más bajo de referencia (0,1,2,3). Luego de identificar que página es la menos usada hacemos el "swap" que para nuestro caso con el HashMap consiste en agregar la nueva página a referenciar con el bit de referencia establecido en el valor necesario (teniendo en cuenta el bit de modificación y su valor anterior) y eliminar del HashMap la página anterior para seguir respetando las páginas permitidas según lo marcos ingresados.

El proceso de lectura de las referencias se maneja en un hilo separado LectorReferencias, el cual, en su método run(), llama continuamente a calculoDatos.leerReferencias() para simular la lectura de referencias en tiempo real.

Donde se usa sincronización:

```
public synchronized void leerReferencias() {
        String referencia = this.memoriaVirtual.get(0);
        String pagina = referencia.substring(8, 10).replace(",", "");
        if (this.memoriaReal.containsKey(pagina)) {
            this.hits++;
            algoritmoLRU(referencia, true);
        } else {
            this.miss++;
            algoritmoLRU(referencia, false);
        }
        this.memoriaVirtual.remove(0);

        if (this.memoriaVirtual.size() == 0) {
            System.out.println("Hits: " + this.hits);
            System.out.println("Fallas: " + this.miss);
            System.out.println("Numero referencias: " + this.numRegistros);
            this.porcentajeHits = (double) this.hits / (this.numRegistros) * 100;
            this.porcentajeHits = Math.round(porcentajeHits * 100.0) / 100.0;
            System.out.println("Porcentaje de hits: " + porcentajeHits + "%");
            System.exit(0);
        }

    }
```

```java
public synchronized void algoritmoLRU(String referencia, Boolean hit) {
        String pagina = referencia.substring(8, 10).replace(",", "");
        char operacion = referencia.charAt(referencia.length() - 3);

        if (hit) {
            String bit = this.memoriaReal.get(pagina).get(0);
            if (bit == "0") {
                if (operacion == 'W') {
                    this.memoriaReal.get(pagina).set(0, "1");
                } else {
                    this.memoriaReal.get(pagina).set(0, "2");
                }

            } else if (bit == "1") {
                if (operacion == 'W') {
                    this.memoriaReal.get(pagina).set(0, "3");
                } else {
                    this.memoriaReal.get(pagina).set(0, "3");

                }
            } else if (bit == "2") {
                if (operacion == 'W') {
                    this.memoriaReal.get(pagina).set(0, "3");
                } else {
                    this.memoriaReal.get(pagina).set(0, "2");

                }
            } else if (bit == "3") {
                if (operacion == 'W') {
                    this.memoriaReal.get(pagina).set(0, "3");
                } else {
                    this.memoriaReal.get(pagina).set(0, "3");

                }
            }

```

```
} else {
            Boolean cambio = false;
            for (int i = 0; i < 4; i++) {
                for (String keyReal : this.memoriaReal.keySet()) {
                    String bitAnterior = this.memoriaReal.get(keyReal).get(0);
                    String paginaNueva = referencia.substring(8, 10).replace(",", "");

                    ArrayList<String> lista = new ArrayList<String>();
                    if (bitAnterior == "0" && i == 0) {
                        if (operacion == 'W') {
                            lista.add("1");
                            this.memoriaReal.put(paginaNueva, lista);
                            this.memoriaReal.remove(keyReal);
                            cambio = true;
                            break;
                        } else {
                            lista.add("2");
                            this.memoriaReal.remove(keyReal);
                            cambio = true;
                            break;
                        }

                    } else if (bitAnterior == "1" && i == 1) {
                        if (operacion == 'W') {
                            lista.add("3");
                            this.memoriaReal.put(paginaNueva, lista);
                            this.memoriaReal.remove(keyReal);
                            cambio = true;
                            break;
                        } else {
                            lista.add("3");
                            this.memoriaReal.put(paginaNueva, lista);
                            this.memoriaReal.remove(keyReal);
                            cambio = true;
                            break;
                        }
                    } else if (bitAnterior == "2" && i == 2) {
                        if (operacion == 'W') {
                            lista.add("3");
                            this.memoriaReal.put(paginaNueva, lista);
                            this.memoriaReal.remove(keyReal);
                            cambio = true;
                            break;
                        } else {
                            lista.add("2");
                            this.memoriaReal.put(paginaNueva, lista);
                            this.memoriaReal.remove(keyReal);
                            cambio = true;
                            break;
                        }
                    } else if (bitAnterior == "3" && i == 3) {
                        if (operacion == 'W') {
                            lista.add("3");
                            this.memoriaReal.put(paginaNueva, lista);
                            this.memoriaReal.remove(keyReal);
                            cambio = true;
                            break;
                        } else {
                            lista.add("3");
                            this.memoriaReal.put(paginaNueva, lista);
                            this.memoriaReal.remove(keyReal);
                            cambio = true;
                            break;
                        }
                    }
                }
                if (cambio) {
                    break;
                }
            }

        }
    }
```

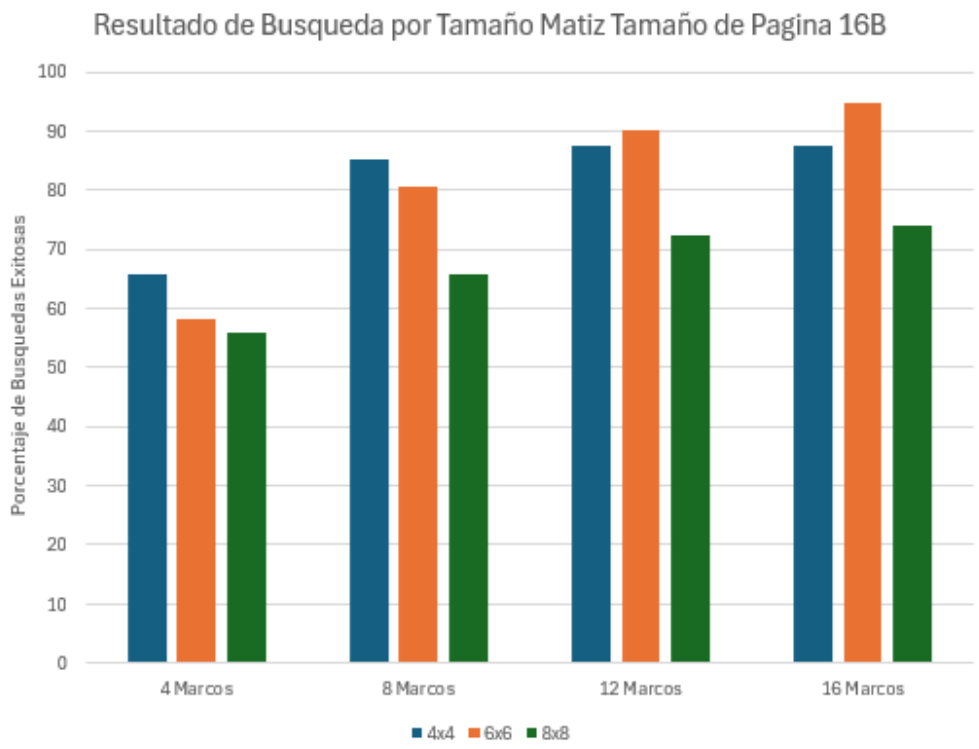
La sincronización se utiliza dentro de la clase CalculoDatos ya que en esta clase es donde se realizan los métodos para ir actualizando la memoria. Específicamente los métodos sincronizados son: leerReferencias(); algoritmoRLU() y restart(), esto debido a que en estos métodos se hace una actualización concurrente del Hashmap que representa la memoria real y de no ser manejado de forma correcta podría presentar problemas a la hora de la veracidad de los datos.

Tablas y gráficos:

Páginas de 16B, matriz 4x4									
						"+/-"			%
Marcos Asignados	Total referencias	Hits	Fallas	Hits	Fallas	Permitido	Desfase	OK	
4	88	61	27	58	30	6,1	-3	1	65,9
8	88	75	13	75	13	7,5	0	1	85,2
12	88	77	11	77	11	7,7	0	1	87,5
16	88	77	11	77	11	7,7	0	1	87,5

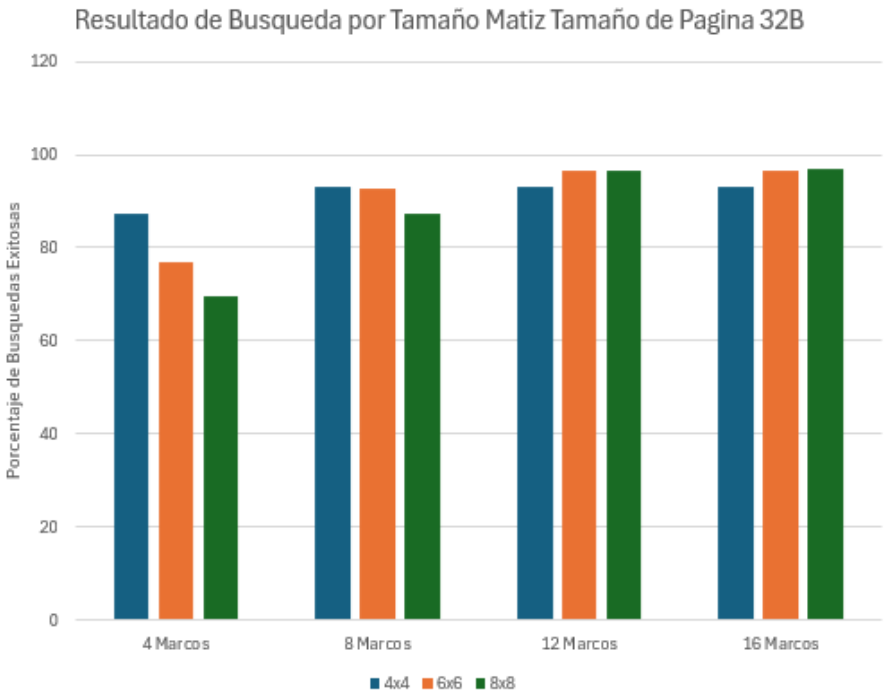
Páginas de 16B, matriz 6x6									
Marcos Asignados	Total referencias	Hits	Fallas	Hits	Fallas	Permitido	Desfase	OK	
4	324	168	156	188	136	16,8	20	0	58,0
8	324	264	60	261	63	26,4	-3	1	80,6
12	324	279	45	292	32	27,9	13	1	90,1
16	314	298	16	297	27	29,8	-1	1	94,6

Páginas de 16B, matriz 8x8									
Marcos Asignados	Total referencias	Hits	Fallas	Hits	Fallas	Permitido	Desfase	OK	
4	712	437	275	398	314	43,7	-39	1	55,9
8	712	531	181	469	243	53,1	-62	0	65,9
12	712	566	146	515	197	56,6	-51	1	72,3
16	712	584	128	526	186	58,4	-58	1	73,9



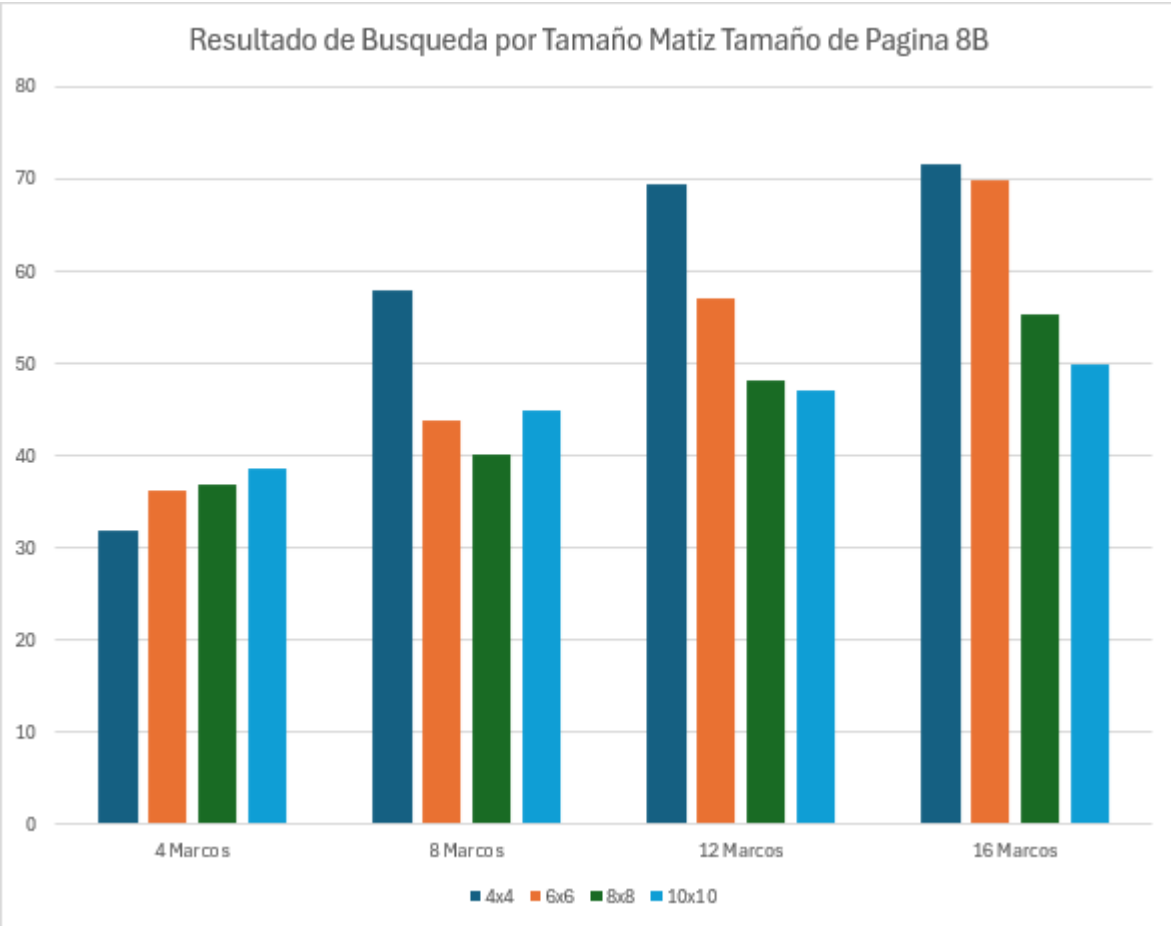
Se puede concluir de este grafico como con un tamaño de pagina de 16B a medida que se aumenta el tamaño de la matriz decrece el porcentaje de búsquedas exitosas. Probablemente se deba a que 16B es un espacio limitado para matrices grandes. En todos los casos a medida que se aumentaron los marcos, el porcentaje de busqueda mejora.

Páginas de 32B, matriz 4x4										
Marcos Asignados	Total referencias	Hits	Fallas			Hits	Fallas	Permitido	Desfase	OK	
4	88	80	8			77	11	8	-3	1	87,5
8	88	82	6			82	6	8,2	0	1	93,2
12	88	82	6			82	6	8,2	0	1	93,2
16	88	82	6			82	6	8,2	0	1	93,2
Páginas de 32B, matriz 6x6										
Marcos Asignados	Total referencias	Hits	Fallas			Hits	Fallas	Permitido	Desfase	OK	
4	324	271	53			249	75	27,1	-22	1	76,9
8	324	305	19			301	23	30,5	-4	1	92,9
12	324	313	11			313	11	31,3	0	1	96,6
16	324	313	11			313	11	31,3	0	1	96,6
Páginas de 32B, matriz 8x8										
Marcos Asignados	Total referencias	Hits	Fallas			Hits	Fallas	Permitido	Desfase	OK	
4	712	553	159			495	217	55,3	-58	0	69,5
8	712	664	48			623	89	66,4	-41	1	87,5
12	712	664	48			687	25	66,4	23	1	96,5
16	712	664	48			689	23	66,4	25	1	96,8



En el tamaño de pagina de 32B vemos como no hay cambios significativo en el porcentaje de busqueda exitosa con respecto de una matriz a otra. Sin embargo si exite una leve tendencia a incrementar el porcentaje con respecto a los marcos. Posiblemente al ser 32B un tamaño pertinente y capaz de manejar esta cantidad de datos vemos como no hay problema de una matriz a otra.

Páginas de 8B, matriz 4x4										
Marcos Asignados	Total referencias	Hits	Fallas				Hits	Fallas	Permitido	Desfase	OK	
4	88	28	60				28	60	2,8	0	1	31,8
8	88	51	37				51	37	5,1	0	1	58,0
12	88	61	27				61	27	6,1	0	1	69,3
16	88	63	25				63	25	6,3	0	1	71,6
Páginas de 8B, matriz 6x6												
Marcos Asignados	Total referencias	Hits	Fallas				Hits	Fallas	Permitido	Desfase	OK	
4	324	117	207				117	207	11,7	0	1	36,1
8	324	142	182				142	182	14,2	0	1	43,8
12	324	185	139				185	139	18,5	0	1	57,1
16	324	226	98				226	98	22,6	0	1	69,8
Páginas de 8B, matriz 8x8												
Marcos Asignados	Total referencias	Hits	Fallas				Hits	Fallas	Permitido	Desfase	OK	
4	712	263	449				263	449	26,3	0	1	36,9
8	712	285	427				285	427	28,5	0	1	40,0
12	712	342	370				342	370	34,2	0	1	48,0
16	712	393	319				393	319	39,3	0	1	55,2
Páginas de 8B, matriz 10x10												
Marcos Asignados	Total referencias	Hits	Fallas				Hits	Fallas	Permitido	Desfase	OK	
4	1252	482	770				482	770	48,2	0	1	38,5
8	1252	562	690				562	690	56,2	0	1	44,9
12	1252	590	662				590	662	59	0	1	47,1
16	1252	625	627				625	627	62.5	0	1	49.9



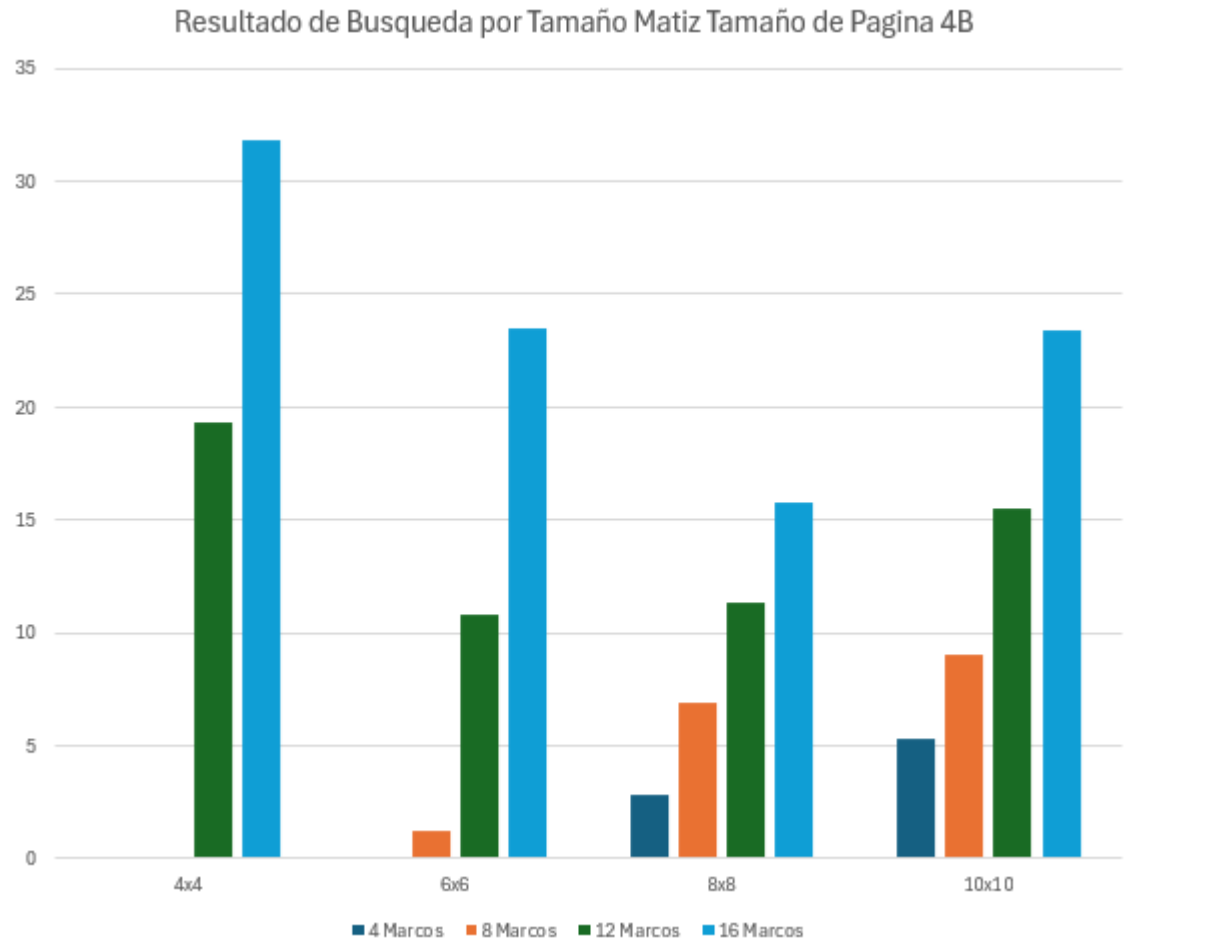
A medida que se aumentan los marcos el porcentaje de busquedas exitosas mejora para cualquier matriz.

Páginas de 4B, matriz 4x4									
Marcos Asignados	Total referencias	Hits	Fallas		Hits	Fallas	Permitido	Desfase	OK	
4	88	0	88		0	88	0	0	1	0,0
8	88	0	88		0	88	0	0	1	0,0
12	88	17	71		17	71	1,7	0	1	19,3
16	88	28	60		28	60	2,8	0	1	31,8

Páginas de 4B, matriz 6x6									
Marcos Asignados	Total referencias	Hits	Fallas		Hits	Fallas	Permitido	Desfase	OK	
4	324	0	324		0	324	0	0	1	0,0
8	324	4	320		4	320	0,4	0	1	1,2
12	324	35	289		35	289	3,5	0	1	10,8
16	324	76	248		76	248	7,6	0	1	23,5

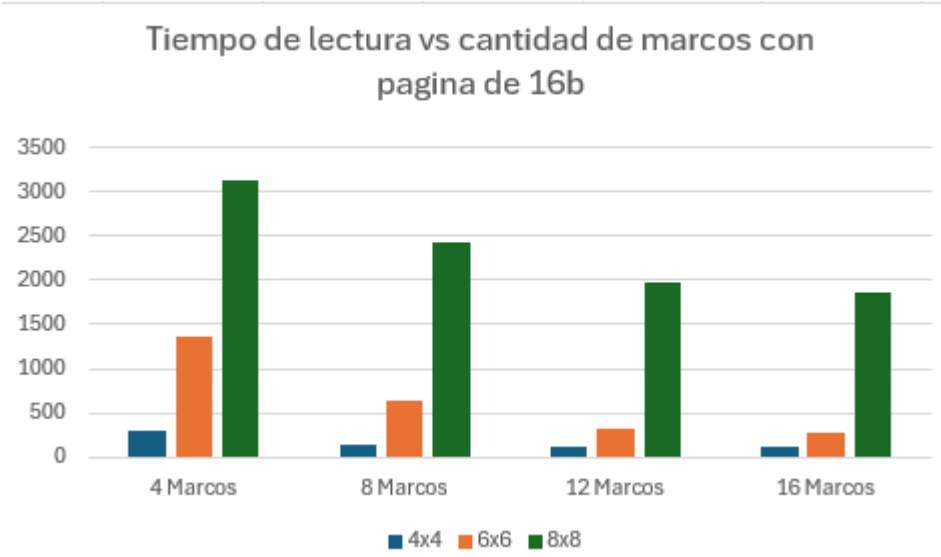
Páginas de 4B, matriz 8x8									
Marcos Asignados	Total referencias	Hits	Fallas		Hits	Fallas	Permitido	Desfase	OK	
4	712	20	629		20	629	2	0	1	2,8
8	712	49	663		49	663	4,9	0	1	6,9
12	712	81	631		81	631	8,1	0	1	11,4
16	712	112	600		112	600	11,2	0	1	15,7

Páginas de 4B, matriz 10x10									
Marcos Asignados	Total referencias	Hits	Fallas		Hits	Fallas	Permitido	Desfase	OK	
4	1252	66	1186		66	1186	6,6	0	1	5,3
8	1252	113	1119		113	1119	11,3	0	1	9,0
12	1252	194	1058		194	1058	19,4	0	1	15,5
16	1252	293	959		293	959	29,3	0	1	23,4

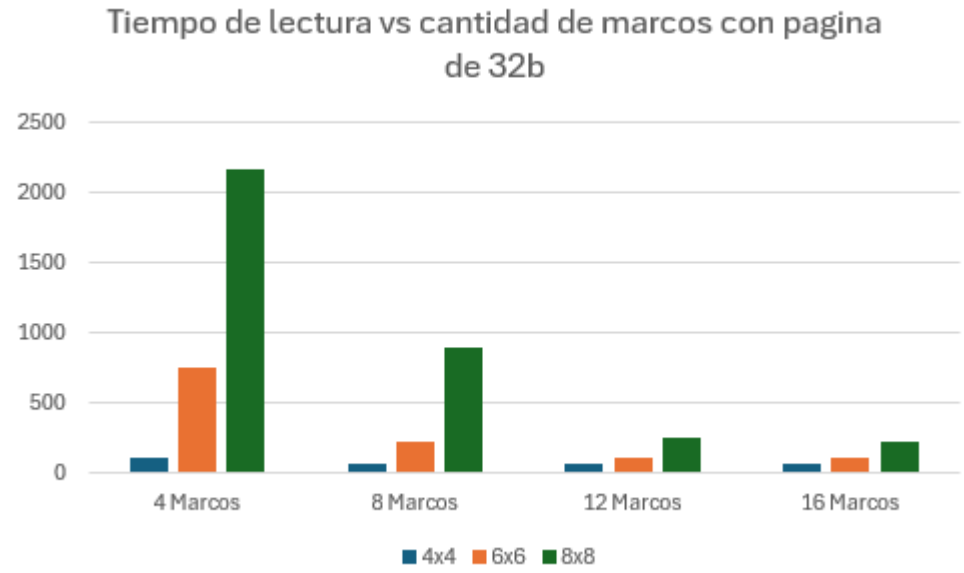


GRAFICOS DE TIEMPO:

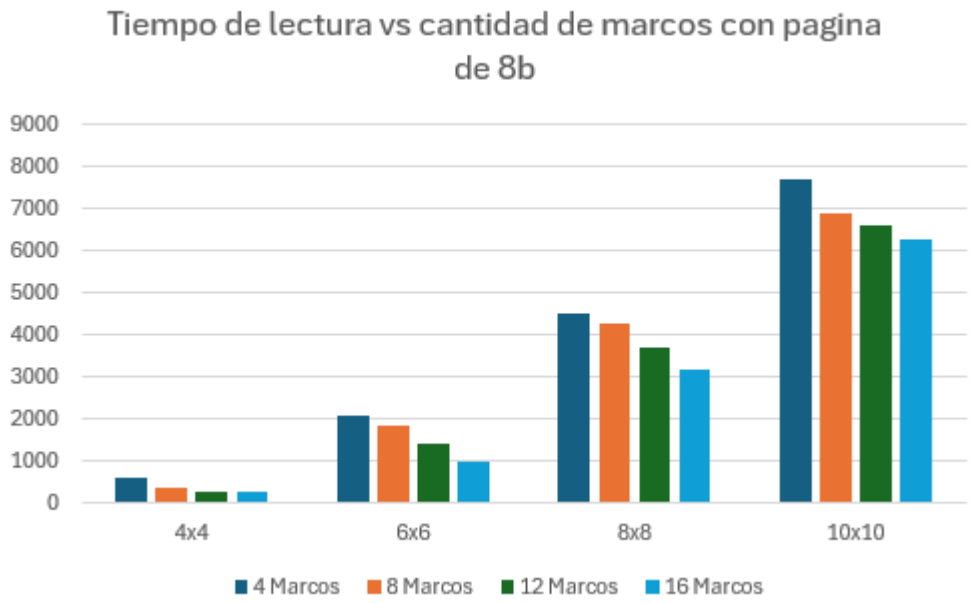
Tiempo de lectura vs cantidad de marcos con pagina de 16b				
Tamaño Matriz	4 Marcos	8 Marcos	12 Marcos	16 Marcos
4x4	300	130	110	110
6x6	1360	630	320	270
8x8	3140	2430	1970	1860



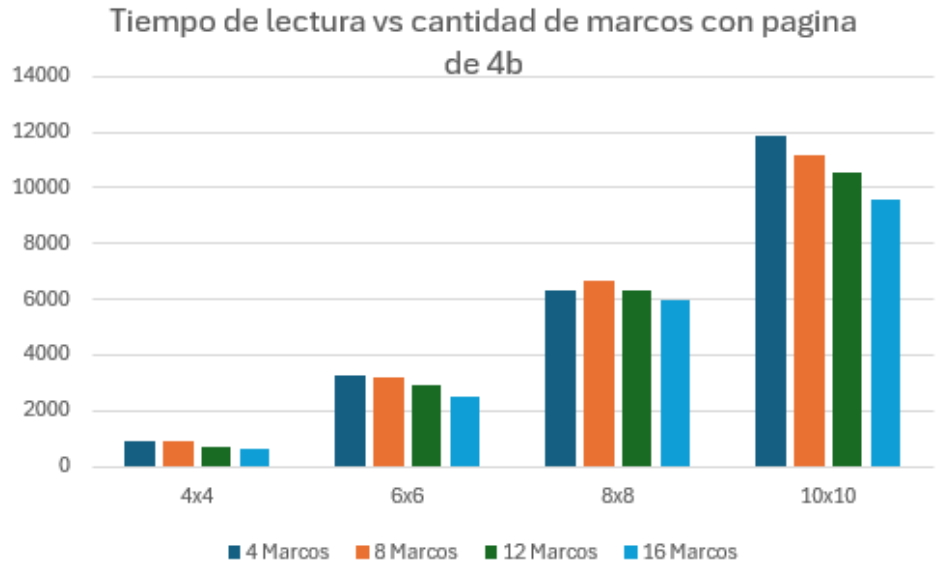
Tiempo de lectura vs cantidad de marcos con pagina de 32b				
Tamaño Matriz	4 Marcos	8 Marcos	12 Marcos	16 Marcos
4x4	110	60	60	60
6x6	750	230	110	110
8x8	2170	890	250	230



Tiempo de lectura vs cantidad de marcos con pagina de 8b				
Tamaño Matriz	4 Marcos	8 Marcos	12 Marcos	16 Marcos
4x4	600	370	270	250
6x6	2070	1820	1390	980
8x8	4490	4270	3700	3190
10x10	7700	6900	6620	6270



Tiempo de lectura vs cantidad de marcos con pagina de 4b				
Tamaño Matriz	4 Marcos	8 Marcos	12 Marcos	16 Marcos
4x4	880	880	710	600
6x6	3240	3200	2890	2480
8x8	6290	6630	6310	6000
10x10	11860	11190	10580	9590



(Se adjunta el Excel en los archivos del proyecto por si se quiere una visualización detallada.)

Conclusiones:

Vemos como dependiendo del tamaño de la pagina los resultados de la grafica varian una con respecto a las otras. Se presentan los principales insights:

1. A medida que se le aumenta el tamaño a la pagina esta es capaz de manejar las diferencias entre las matrices desde la 4x4 hasta la 10x10.
2. El aumento de los marcos manejando una misma matriz incrementa el porcentaje de búsquedas exitosas.
3. Un mayor tamaño de matriz representa un mayor tiempo de lectura. Al mismo tiempo un mayor uso de marcos representa un menor tiempo de lectura manejando la misma matriz.