INFORME CASO 2 INFRAESTRUCTURA COMPUTACIONAL

Daniel Felipe Díaz Moreno - 202210773
Juan Diego Ortega Medina - 202014624

Descripción del algoritmo usado para generar las referencias de página (modo uno)

Para crear referencias, utilizamos el método proceso, descrito en el documento, con la particularidad de que los accesos a memoria se realizan con los métodos leer y escribir. Al utilizar el método leer, obtenemos el valor de la posición especificada, mientras que al utilizar el método escribir, sobrescribimos dicho valor.

```
*/
public void proceso() {
    for (int i = 1; i < nf - 1; i++) {
        for (int j = 1; j < nc - 1; j++) {
            int acum = 0;
            for (int a = -1; a <= 1; a++) {
                for (int b = -1; b <= 1; b++) {
                    int i2 = i + a;
                    int j2 = j + b;
                    int i3 = 1 + a;
                    int j3 = 1 + b;
                    acum += (leer(mat1,i2,j2) * leer(mat2,i3,j3));
                }
            }
            if (acum >= 0 && acum <= 255) {
                escribir(mat3,i,j,acum);
            } else if (acum < 0) {
                escribir(mat3,i,j,0);
            } else {
                escribir(mat3,i,j,255);
            }
        }
    }
    for (int i = 0; i < nc; i++) {
        escribir(mat3,0,i,0);
        escribir(mat3,nf - 1,i,255);
    }
    for (int i = 1; i < nf - 1; i++) {
        escribir(mat3,i,0,0);
        escribir(mat3,i,nc - 1,255);
    }
```

Estos métodos también generan una referencia utilizando los índices y la matriz proporcionados como base.

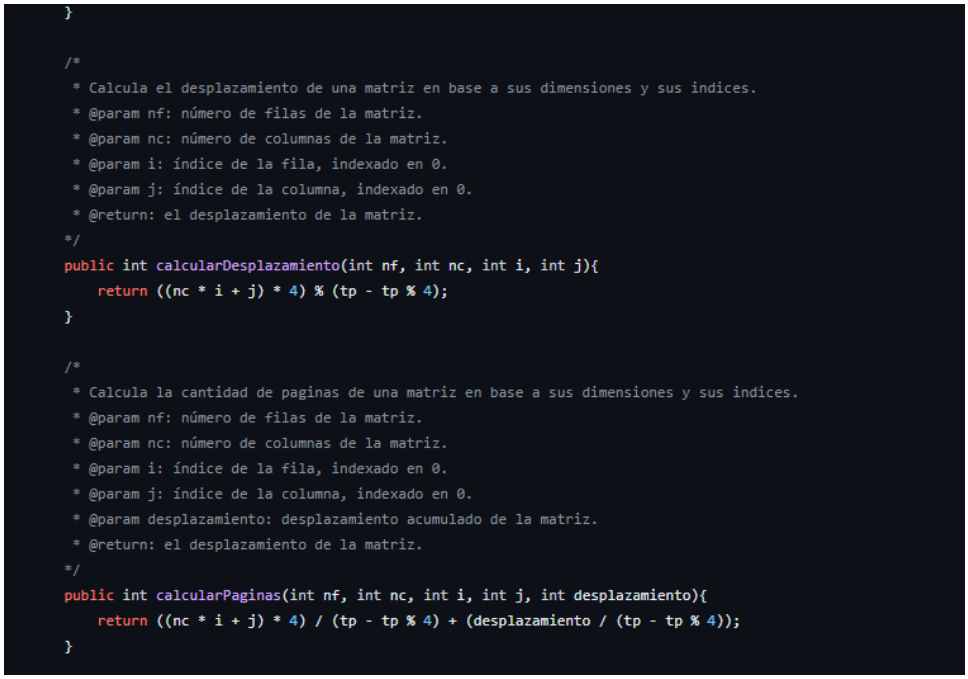
```java
    */
    public void escribir(int[][] mat, int i, int j, int valor){
        reportarReferencia(mat, i, j, 'W');
        mat[i][j] = valor;
    }

    /*
     * Lee un valor de una matriz y reporta la referencia en el archivo de salida.
     * @param mat: matriz de la que se leerá el valor.
     * @param i: indice de la fila, indexado en 0.
     * @param j: indice de la columna, indexado en 0.
     * @return: el valor leido de la matriz.
    */
    public int leer(int[][] mat, int i, int j){
        reportarReferencia(mat, i, j, 'R');
        return mat[i][j];
    }
```

La función ReportarReferencia tiene la responsabilidad de identificar a qué página y qué desplazamiento corresponde una referencia dada, utilizando como base la matriz y sus índices.

```java
    */
    public void reportarReferencia(int[][] mat,int i, int j, char tipo){
        char tipoMatriz = 'M';
        if (mat == mat2) tipoMatriz = 'F';
        else if (mat == mat3) tipoMatriz = 'R';

        // Las siguientes 4 líneas de código calculan el desplazamiento y paginas del filtro y la matriz para realizar calculos acumulativos.
        // Los indices de una matriz estan indexados desde 0, por lo que se le resta 1 a los indices de la matriz.
        // Sin embargo, necesitamos la referencia al final de la matriz, por lo que se le suma 1 a j.
        int desplazamientoDelFiltroTotal = calcularDesplazamiento(3,3,(3-1),(3));
        int paginasDelFiltroTotal = calcularPaginas(3,3,(3-1),(3) ,0);
        int desplazamientoDeMatrizTotal = calcularDesplazamiento(nf,nc,(nf-1),(nc));
        int paginasDeMatrizTotal = calcularPaginas(nf,nc,(nf-1),(nc) ,0);

        // Calcula el desplazamiento y la página de la referencia según el tipo de matriz.
        int pagina, desplazamiento;
        if (tipoMatriz == 'M') {
            desplazamiento = desplazamientoDelFiltroTotal + calcularDesplazamiento(nf,nc,i,j);
            pagina = paginasDelFiltroTotal + calcularPaginas(nf,nc,i,j,desplazamiento);
        } else if (tipoMatriz == 'R') {
            desplazamiento = desplazamientoDelFiltroTotal + desplazamientoDeMatrizTotal + calcularDesplazamiento(nf,nc,i,j);
            pagina = paginasDelFiltroTotal + paginasDeMatrizTotal + calcularPaginas(nf,nc,i,j,desplazamiento);
        } else {
            desplazamiento = calcularDesplazamiento(3,3,i,j);
            pagina = calcularPaginas(3,3,i,j,desplazamiento);
        }
        desplazamiento = desplazamiento % (tp - tp % 4);

        // Escribe la referencia en el archivo de salida.
        try {
            escritorArchivo.write(tipoMatriz + "["+ i +"][" + j + "]," + pagina + "," + desplazamiento +"," + tipo +"\n");
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
```

Los métodos calcularDesplazamiento y calcularPaginas son subrutinas que emplean operaciones matemáticas para determinar el desplazamiento y la página correspondientes. En la fórmula, la única peculiaridad es (tp-tp%4), que se utiliza cuando el tamaño de la página no es múltiplo de 4. En

este caso, se redondea al múltiplo de 4 más cercano, hacia abajo.



```
}

/*
 * Calcula el desplazamiento de una matriz en base a sus dimensiones y sus indices.
 * @param nf: número de filas de la matriz.
 * @param nc: número de columnas de la matriz.
 * @param i: índice de la fila, indexado en 0.
 * @param j: índice de la columna, indexado en 0.
 * @return: el desplazamiento de la matriz.
*/
public int calcularDesplazamiento(int nf, int nc, int i, int j){
    return ((nc * i + j) * 4) % (tp - tp % 4);
}

/*
 * Calcula la cantidad de paginas de una matriz en base a sus dimensiones y sus indices.
 * @param nf: número de filas de la matriz.
 * @param nc: número de columnas de la matriz.
 * @param i: índice de la fila, indexado en 0.
 * @param j: índice de la columna, indexado en 0.
 * @param desplazamiento: desplazamiento acumulado de la matriz.
 * @return: el desplazamiento de la matriz.
*/
public int calcularPaginas(int nf, int nc, int i, int j, int desplazamiento){
    return ((nc * i + j) * 4) / (tp - tp % 4) + (desplazamiento / (tp - tp % 4));
}
```

Se cuentan con otras funciones, pero estas no contribuyen enormemente al entendimiento del algoritmo al ser funciones para inicializar variables o para recibir o imprimir datos por consola.

Descripción del algoritmo y estructuras de datos usadas

El algoritmo utilizado es el NRU, y se encuentra principalmente en la función realizarCalculos del archivo Java denominado Calculo.java.

```java
public void realizarCalculos(){
    actualizador.start();

    int i = 0;
    while (i < this.nr) {
        try {
            Thread.sleep(1);
        } catch (InterruptedException e) {
            e.printStackTrace();
        }
        referencia = this.referencias[i].split(",");

        pagina = Integer.parseInt(referencia[1]);
        escritura = referencia[3].equals("W");

        if ((i+1)%4 == 0) conteo.pedirActualizar();

        if (this.tablaPaginas[pagina] == -1) {
            misses++;
            int marco = 0;
            while (marco < this.NMP && this.marcosPagina[marco]) {
                marco++;
            }
            if (marco < this.NMP) {
                this.tablaPaginas[pagina] = marco;
                this.marcosPagina[marco] = true;
                this.conteo.referenciarPagina(pagina, escritura);
            } else {
                int idx  = this.conteo.obtenerPaginaAEliminar(this.tablaPaginas);
                this.tablaPaginas[pagina] = this.tablaPaginas[idx];
                this.tablaPaginas[idx] = -1;
                this.conteo.referenciarPagina(pagina, escritura);
            }
        } else {
            hits++;
            this.conteo.referenciarPagina(pagina, escritura);
        }
        i++;
    }
    while (i%4 != 0) {
        try {
            Thread.sleep(1);
        } catch (InterruptedException e) {
            e.printStackTrace();
        }
        i++;
    }
    actualizador.detener();
    conteo.pedirActualizar();
    imprimirResultados(hits, misses);
}
```

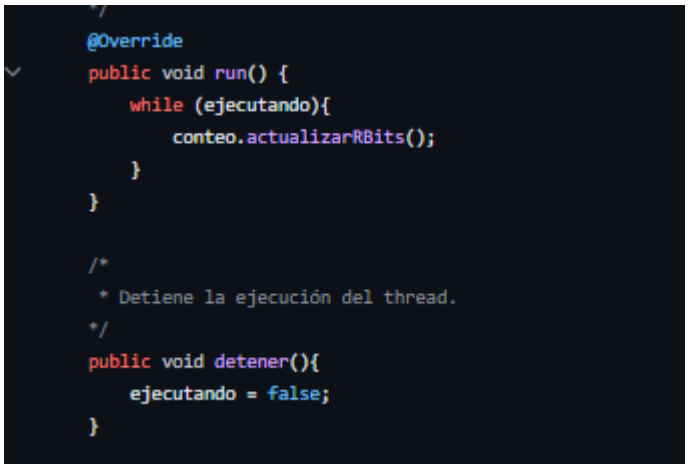
El programa procesa cada referencia generada. Inicia simulando que el proceso corre cada milisegundo mediante una pausa de un milisegundo. Luego, obtiene la información necesaria de la referencia, que incluye la página y el modo de acceso (escritura o lectura). Posteriormente, revisa si ha transcurrido un múltiplo de cuatro segundos para solicitar al otro hilo que realice ciertas operaciones.

La lógica de acceso a memoria implica verificar si la página está cargada. Si lo está, se incrementa el contador de "hits" y se continúa con la siguiente iteración. Si no está cargada, se produce un fallo de página y se incrementa el contador correspondiente.

Dentro del bucle interno se busca determinar si hay algún marco de página sin utilizar. Si existe, se utiliza; de lo contrario, se aplica el algoritmo de NRU para determinar qué página debe ser reemplazada.

Finalmente, si el otro hilo no ha terminado su ejecución se espera a que pase un múltiplo de 4 segundos para que acabe, lo cual se ve reflejado en el ultimo bucle.

El otro hilo tiene una lógica simple que intenta ejecutar la actualización de bits a menos que se especifique lo contrario.



La clase Conteo contiene estructuras y métodos sincronizados. Estos métodos incluyen referenciarPagina, obtenerPaginaAEliminar, actualizarRBits y pedirActualizar.

- pedirActualizar se encarga de notificar al otro hilo cuándo debe comenzar su trabajo.
- Los otros métodos implementan el algoritmo estándar del NRU. Si dos páginas tienen igual capacidad para modificarse, se selecciona la que tenga el índice más bajo.

rBits y mBits son arreglos que representan los bits de referencia y modificación para cada página en la memoria, hacen parte de la clase Conteo, la cual es la estructura compartida.

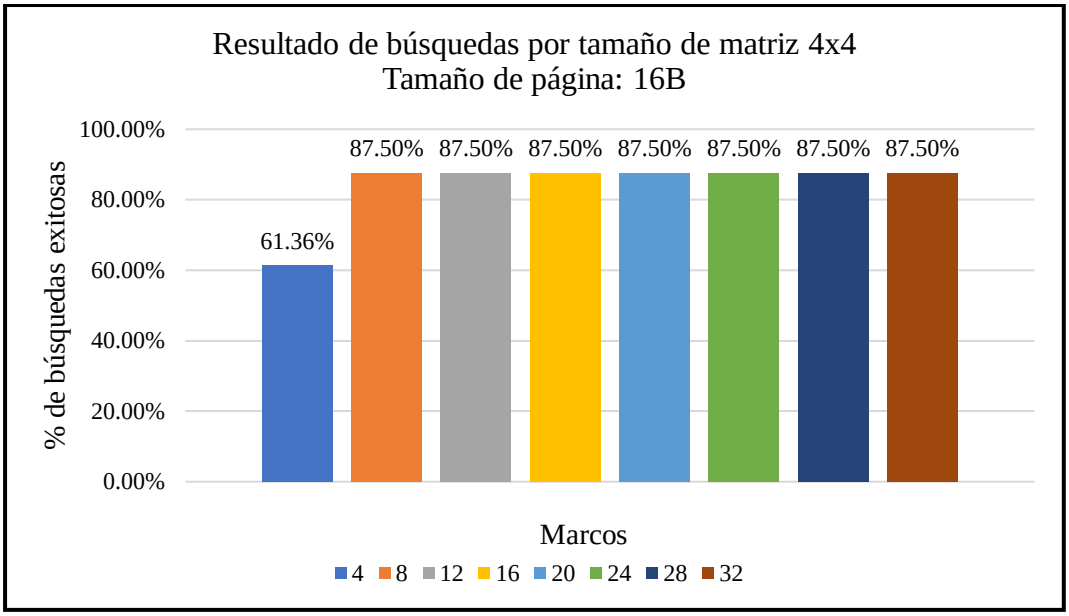
Esquema de sincronización usado

Dentro de las clases implementadas, el esquema de sincronización está codificado en la clase Conteo, la cual se encarga de la referenciación y modificación de las páginas. Esta sincronización se basa en la exclusión mutua, lo que significa que solo un hilo puede acceder a los métodos sincronizados de la clase Conteo en un momento dado. Aunque esta exclusión puede llevar a imprecisiones en los cálculos, que los procesos se suspendan de un milisegundo ayuda a que las diferencias en los cálculos sean mínimas.

Casos de prueba tabulados y graficados

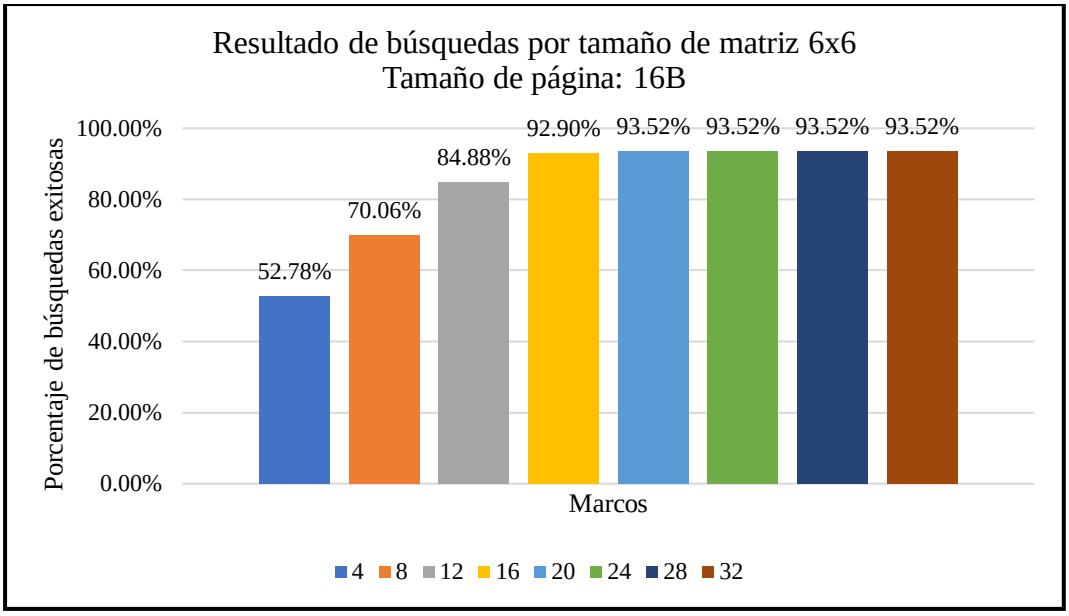
Páginas de 16B, matriz de 4x4

Marcos	Referencias	Hits	% Hits	Fallas
4	88	54	61,36%	34
8	88	77	87,50%	11
12	88	77	87,50%	11
16	88	77	87,50%	11
20	88	77	87,50%	11
24	88	77	87,50%	11
28	88	77	87,50%	11
32	88	77	87,50%	11



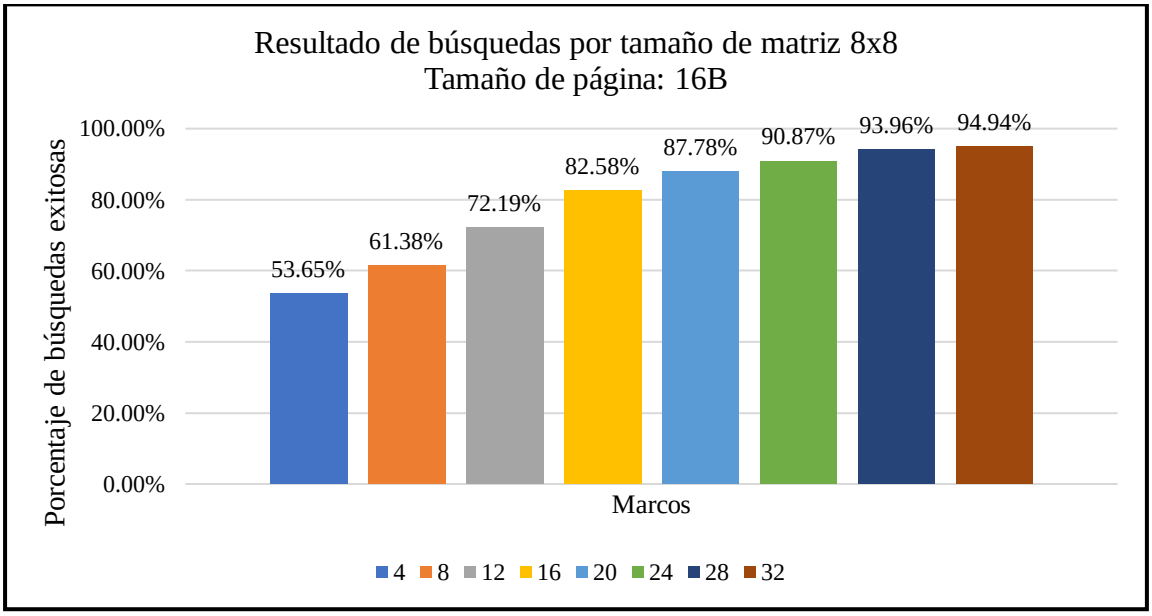
Páginas de 16B, matriz de 6x6

Marcos	Referencias	Hits	% Hits	Fallas
4	324	171	52,78%	153
8	324	227	70,06%	97
12	324	275	84,88%	49
16	324	301	92,90%	23
20	324	303	93,52%	21
24	324	303	93,52%	21
28	324	303	93,52%	21
32	324	303	93,52%	21



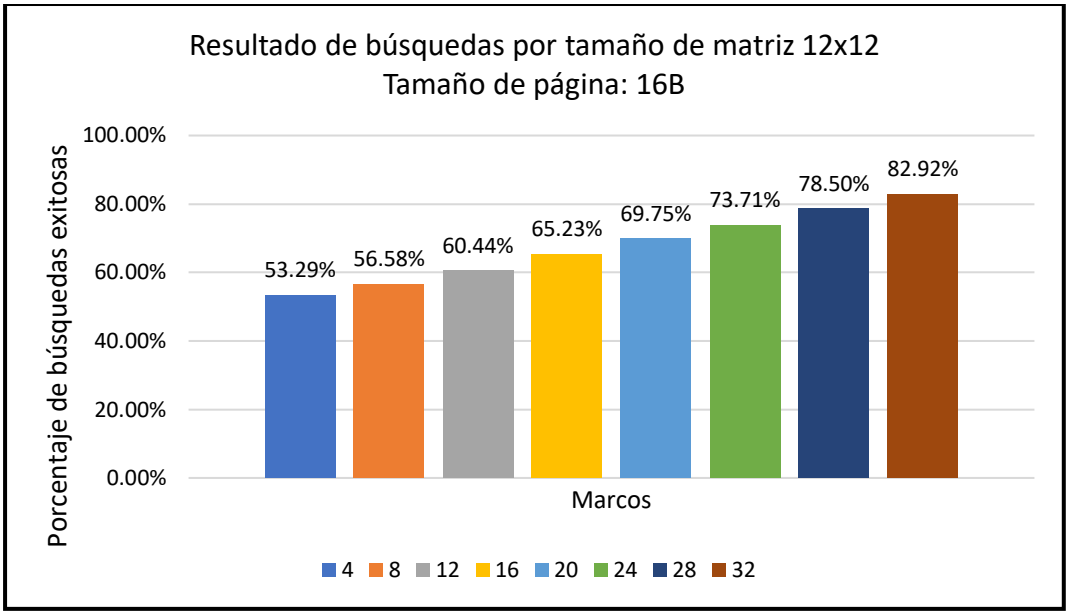
Páginas de 16B. marcos de 8x8

Marcos	Referencias	Hits	% Hits	Fallas
4	712	382	53,65%	330
8	712	437	61,38%	275
12	712	514	72,19%	198
16	712	588	82,58%	124
20	712	625	87,78%	87
24	712	647	90,87%	65
28	712	669	93,96%	43
32	712	676	94,94%	36



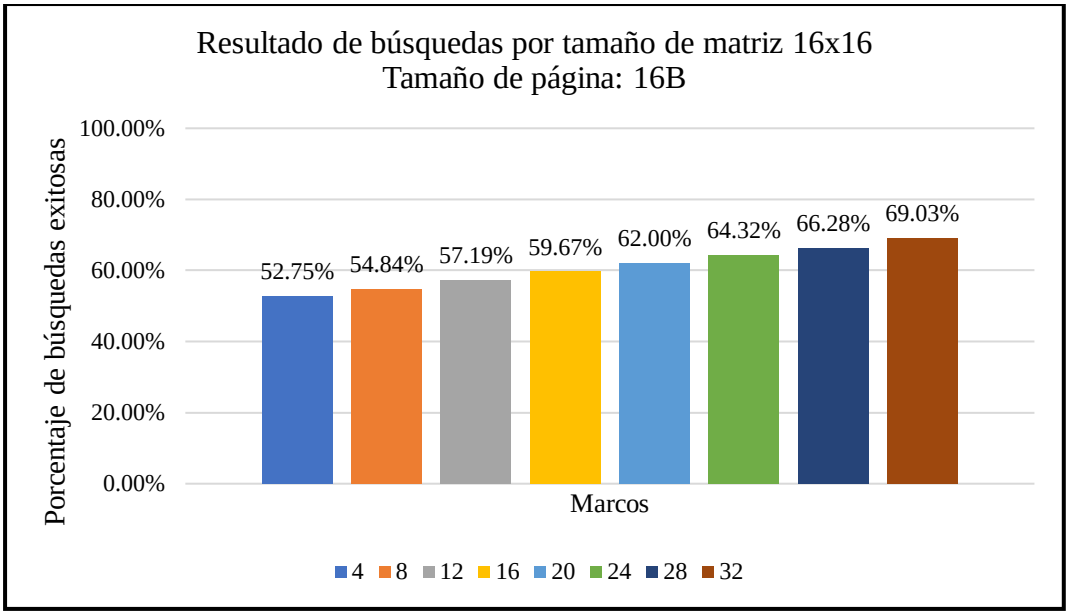
Páginas de 16B, matriz de 12x12

Marcos	Referencias	Hits	% Hits	Fallas
4	1944	1036	53,29%	908
8	1944	1100	56,58%	844
12	1944	1175	60,44%	769
16	1944	1268	65,23%	676
20	1944	1356	69,75%	588
24	1944	1433	73,71%	511
28	1944	1526	78,50%	418
32	1944	1612	82,92%	332



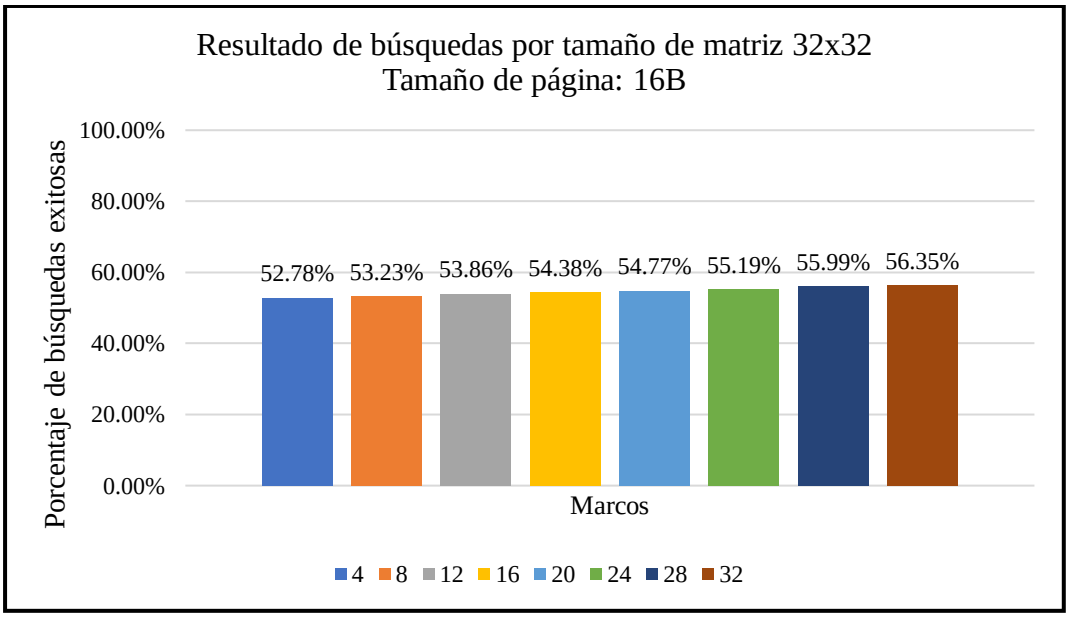
Páginas de 16B, matriz de 16x16

Marcos	Referencias	Hits	% Hits	Fallas
4	3784	1996	52,75%	1788
8	3784	2075	54,84%	1709
12	3784	2164	57,19%	1620
16	3784	2258	59,67%	1526
20	3784	2346	62,00%	1438
24	3784	2434	64,32%	1350
28	3784	2508	66,28%	1276
32	3784	2612	69,03%	1172



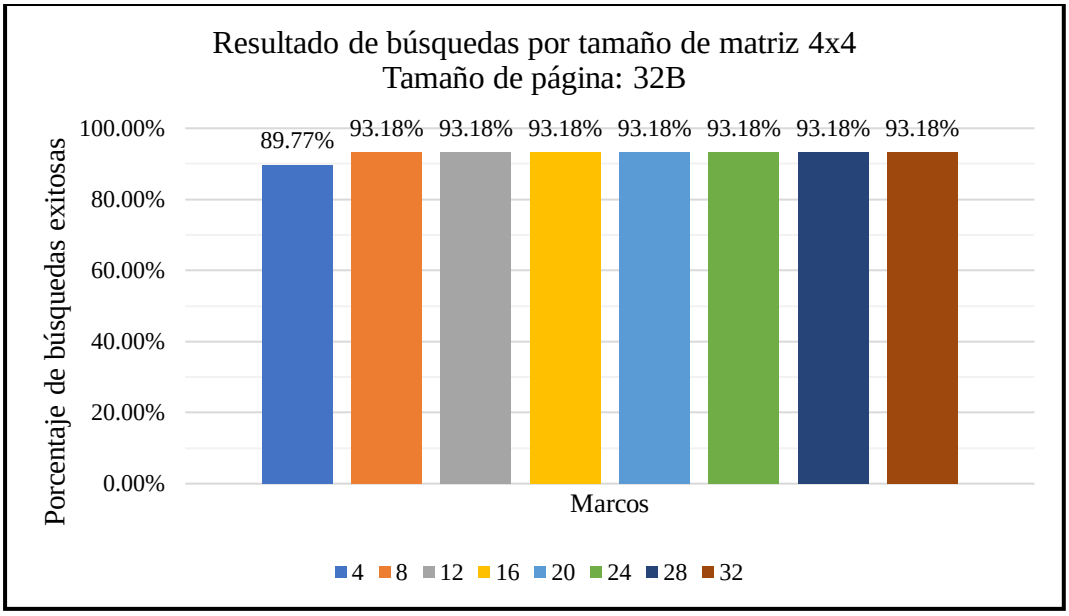
Páginas de 16B, matriz de 32x32

Marcos	Referencias	Hits	% Hits	Fallas
4	17224	9090	52,78%	8134
8	17224	9169	53,23%	8055
12	17224	9276	53,86%	7948
16	17224	9366	54,38%	7858
20	17224	9433	54,77%	7791
24	17224	9506	55,19%	7718
28	17224	9643	55,99%	7581
32	17224	9706	56,35%	7518



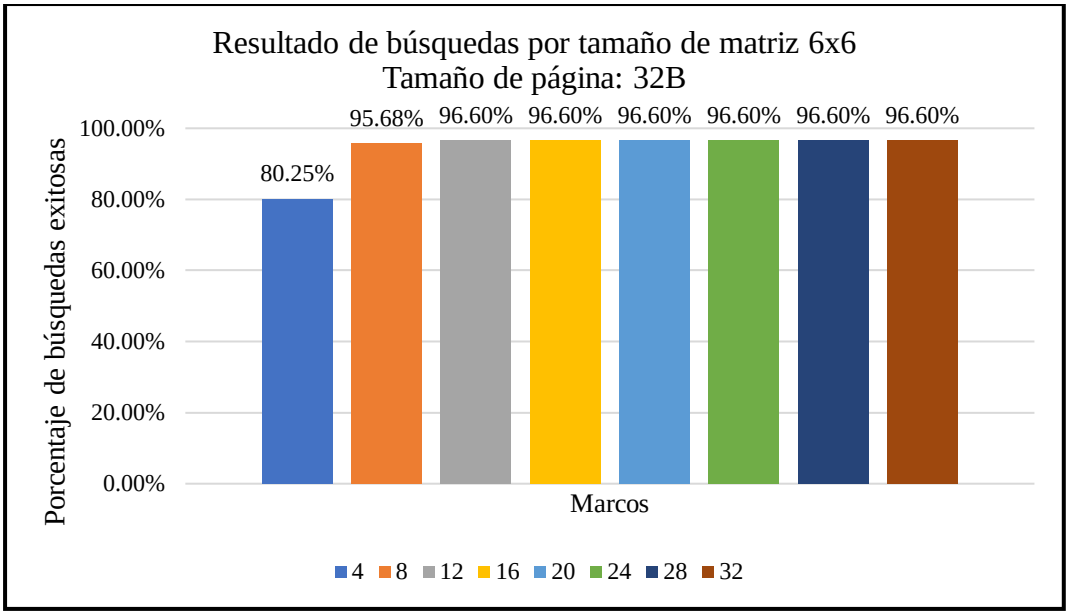
Páginas de 32B, matriz de 4x4

Marcos	Referencias	Hits	% Hits	Fallas
4	88	79	89,77%	9
8	88	82	93,18%	6
12	88	82	93,18%	6
16	88	82	93,18%	6
20	88	82	93,18%	6
24	88	82	93,18%	6
28	88	82	93,18%	6
32	88	82	93,18%	6



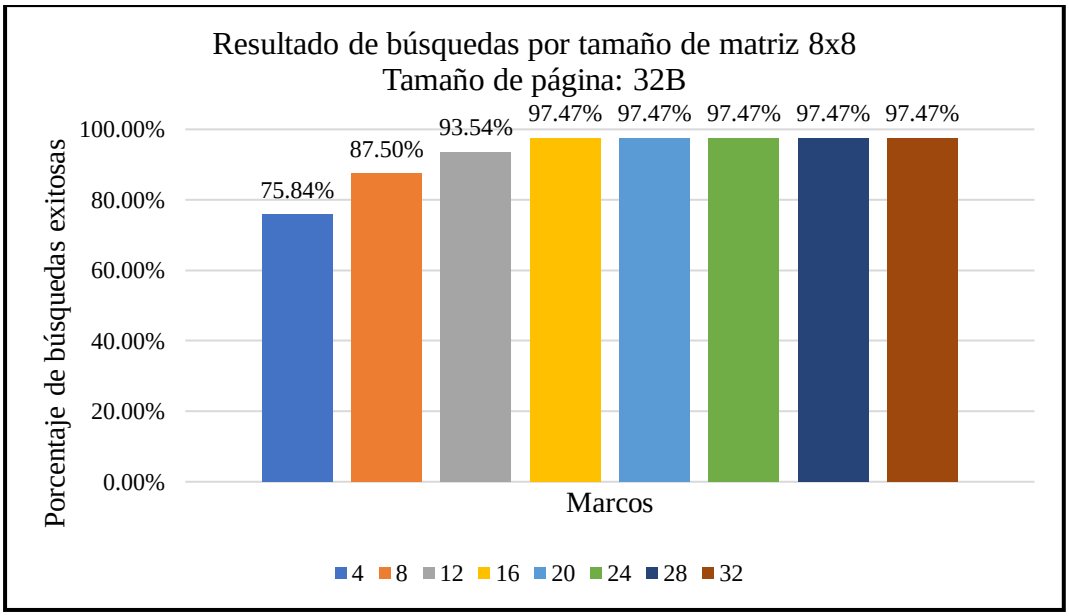
Páginas de 32B, matriz de 6x6

Marcos	Referencias	Hits	% Hits	Fallas
4	324	260	80,25%	64
8	324	310	95,68%	14
12	324	313	96,60%	11
16	324	313	96,60%	11
20	324	313	96,60%	11
24	324	313	96,60%	11
28	324	313	96,60%	11
32	324	313	96,60%	11



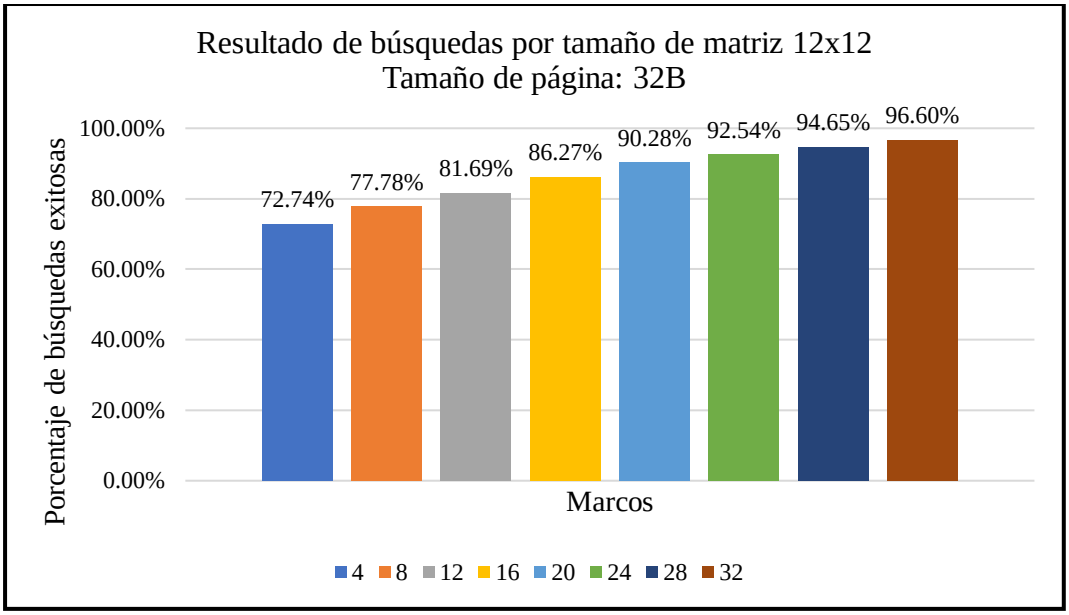
Páginas de 32B, matriz de 8x8

Marcos	Referencias	Hits	% Hits	Fallas
4	712	540	75,84%	172
8	712	623	87,50%	89
12	712	666	93,54%	46
16	712	694	97,47%	18
20	712	694	97,47%	18
24	712	694	97,47%	18
28	712	694	97,47%	18
32	712	694	97,47%	18



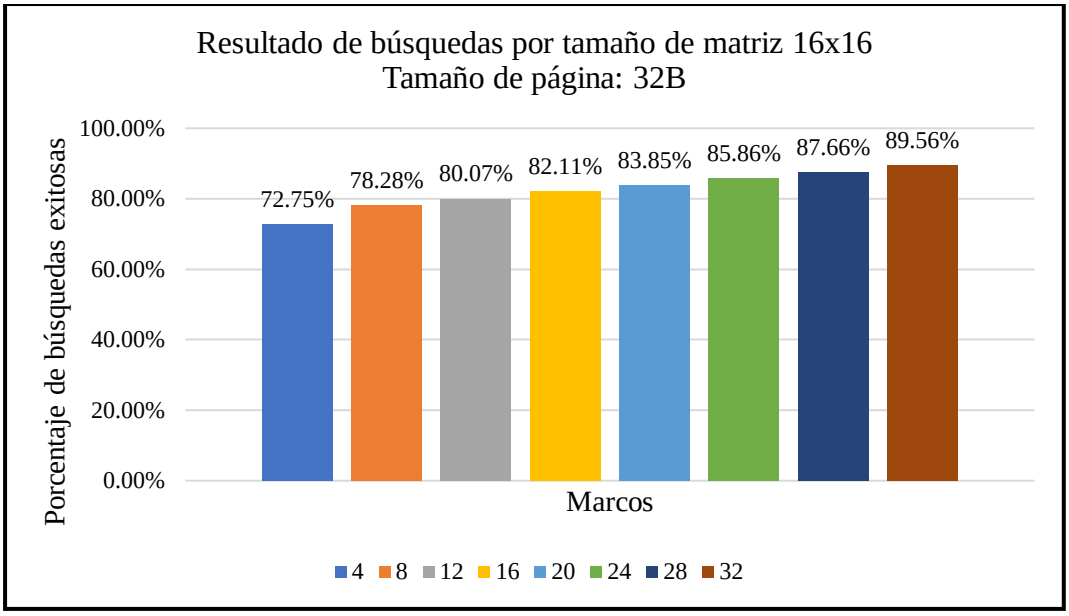
Páginas de 32B, matriz de 12x12

Marcos	Referencias	Hits	% Hits	Fallas
4	1944	1414	72,74%	530
8	1944	1512	77,78%	432
12	1944	1588	81,69%	356
16	1944	1677	86,27%	267
20	1944	1755	90,28%	189
24	1944	1799	92,54%	145
28	1944	1840	94,65%	104
32	1944	1878	96,60%	66



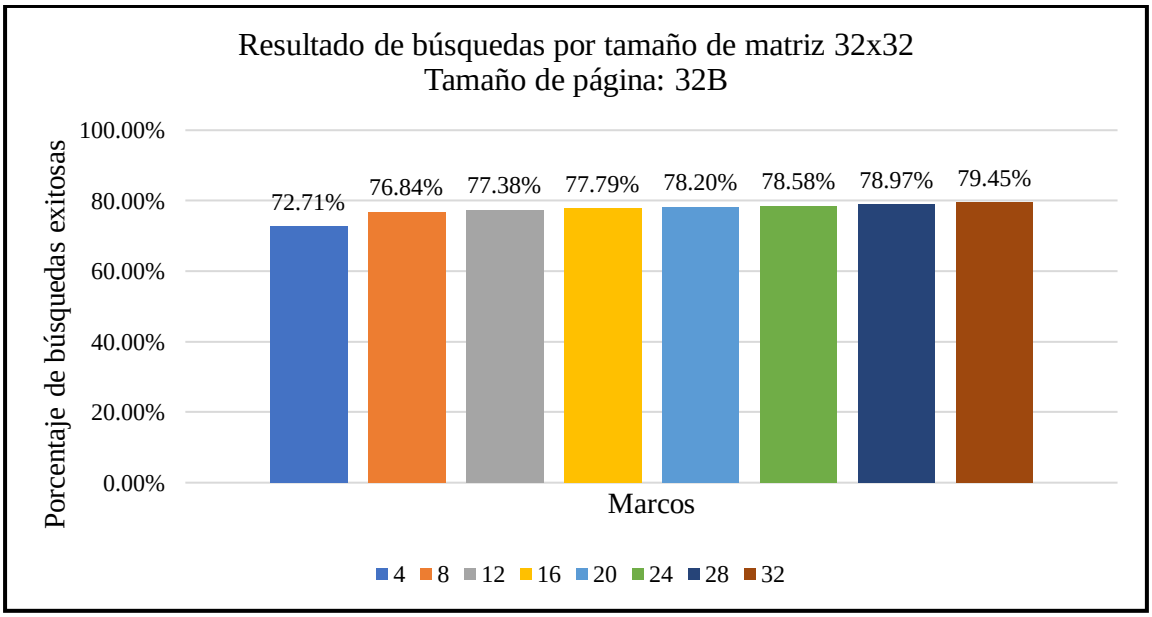
Páginas de 32B, matriz de 16x16

Marcos	Referencias	Hits	% Hits	Fallas
4	3784	2753	72,75%	1031
8	3784	2962	78,28%	822
12	3784	3030	80,07%	754
16	3784	3107	82,11%	677
20	3784	3173	83,85%	611
24	3784	3249	85,86%	535
28	3784	3317	87,66%	467
32	3784	3389	89,56%	395



Páginas de 32B, matriz de 32x32

Marcos	Referencias	Hits	% Hits	Fallas
4	17224	12524	72,71%	4700
8	17224	13235	76,84%	3989
12	17224	13328	77,38%	3896
16	17224	13399	77,79%	3825
20	17224	13470	78,20%	3754
24	17224	13535	78,58%	3689
28	17224	13601	78,97%	3623
32	17224	13685	79,45%	3539



Como se pudo apreciar en las gráficas, se hicieron pruebas en matrices de 4x4, 6x6, 8x8, 12x12, 16x16 y 32x32, así como con 4, 8, 12, 16, 20, 24, 28 y 32 marcos de página. La intención de esto es para poder observar bien las tendencias en la variación de los hits y los fallos conforme aumentábamos los marcos y las matrices, así como también el tiempo de ejecución de cada una de estas pruebas. Las pruebas se ejecutaron en un computador con un procesador Intel Core i7 de 10ma generación con una velocidad de 1.8 GHz y una memoria RAM de 16GB.

Se puede evidenciar una clara relación entre el número de marcos y la cantidad de hits. Conforme aumentan los marcos de página, aumentan los hits y por tanto disminuyen los fallos. Cuando se trata de matrices pequeñas, el numero hits se acerca cada vez más al número de referencias, al punto qué desde cierta cantidad de marcos, la cantidad de hits ya no aumenta. Conforme se va agrandando la matriz, por tanto, empiezan a existir más referencias, el número de fallos empieza a aumentar, aunque cada vez que se añaden marcos, el número de hits también va aumentando.

Al aumentar el tamaño de la página, se observa un gran aumento en la cantidad de hits en todas las pruebas. En cuanto a tiempos de ejecución, sobra mencionar que entre más marcos le damos, más tiempo se demora en calcular los hits respecto a los casos con menos marcos. Se hace la salvedad en los primeros casos, ya que, al tener ya todos los hits calculados, se demora menos. Esto se ve principalmente en los casos de las matrices de 4x4 (16B y 32B), 6x6 (16B y 32B) y 8x8 (32B), en los que al tener todo ya calculado, se “ahorra” tiempos recurriendo a la caché.

Interpretación de los resultados

Con los resultados obtenidos de las pruebas ejecutadas, podemos concluir que:

- Si aumentamos el tamaño de la página, es probable que más datos estén disponibles en la memoria, lo que aumenta los hits
- Al aumentar el número de marcos de página en la memoria, se puede almacenar más páginas en la memoria al mismo tiempo. Esto puede aumentar la probabilidad de que una página requerida ya esté en la memoria, lo que resultaría en un aumento de los hits.
- A medida que aumenta el número de marcos de página y el tamaño de la matriz, también aumenta la cantidad de datos que el algoritmo necesita procesar. Esto resulta en un mayor tiempo de ejecución debido a la necesidad de gestionar más marcos de página y de procesar una matriz más grande.