Simulación de memoria virtual y manejo de paginación

Caso de Estudio 02

Mateo Jurado - 202210996
Santiago Quiroz - 202216453
Camilo Inoue - 202225593

Profesor Magistral

Sandra Rueda

Universidad De Los Andes, Bogotá
Departamento de Ingeniería de Sistemas y Computación

Introducción

Con el fin de administrar de manera apropiada la memoria de un equipo, teniendo en cuenta su infraestructura, cantidad de procesos en ejecución, demanda de recursos por proceso y demás consideraciones especificadas en la descripción del problema, y construir un prototipo a escala del sistema de administración de memoria virtual que permite simular y evaluar el comportamiento de un proceso de acuerdo con los recursos disponibles, presentaremos a continuación la solución del Caso 2.

Opción 1:

La primera opción de nuestro sistema trata de un sistema de generación de referencias. Este recibe como parámetros el tamaño de cada página, y el nombre del archivo con la imagen que se procesa, todo esto mediante la consola.

El programa se estructura en la clase principal SimuladorReferencias, que contiene la clase interna CrearReferencias. El método clave es crearReferencia, el cual realiza la mayor parte del procesamiento. La entrada de datos se obtiene a través de un Scanner, solicitando al usuario el tamaño de la página de memoria y el nombre de un archivo BMP.

El proceso inicia con la carga de la imagen desde el archivo BMP, obteniendo sus dimensiones. Luego, se valida que la imagen tenga al menos 3x3 píxeles y no supere los 300x500 píxeles; si no cumple con estos requisitos, se muestra un mensaje y el programa finaliza. A continuación, se calculan los tamaños de la imagen, los filtros Sobel (X e Y) y la imagen de salida, además de determinar la cantidad de páginas necesarias para su almacenamiento.

Posteriormente, se asignan direcciones de memoria virtual a la imagen de entrada, los filtros Sobel y la imagen de salida. Se simula el acceso a memoria para aplicar el filtro Sobel recorriendo la imagen (excluyendo los bordes) en dos bucles anidados. Para cada píxel, se generan referencias de acceso a memoria para los componentes RGB y los filtros Sobel, formateándose y almacenándose en una lista.

Finalmente, el programa escribe un archivo referencias.txt, que contiene detalles como el tamaño de página, dimensiones de la imagen, cantidad de referencias generadas y número de páginas necesarias. Cada referencia especifica la posición del píxel, la dirección de memoria de sus componentes de color, la dirección de los filtros Sobel y si la operación es de lectura (R) o escritura (W).

El programa incluye un menú simple que permite al usuario seleccionar opciones, aunque solo la opción 1 está implementada en este fragmento. También maneja la entrada del usuario y gestiona posibles errores. En conclusión, este simulador genera referencias de

acceso a memoria para procesar imágenes con el filtro Sobel, facilitando la comprensión de la gestión de memoria virtual y su impacto en el procesamiento de imágenes.

Opción 2:

La segunda opción de nuestro sistema consiste en un sistema que calcula hits y misses a partir de un archivo de referencias (el cual recibe por consola, al igual que el tamaño de el número de marcos de página) y la simulación de un sistema de paginación.

El programa utiliza una estructura llamada GestorMemoria, que es una lista ordenada donde se almacenan las páginas en memoria. Cada página tiene un identificador y un objeto llamado PaginaInfo, que guarda información sobre su estado. Esto permite saber qué páginas han sido usadas recientemente y facilita decidir cuáles reemplazar cuando la memoria se llena.

Cada página tiene dos atributos clave: si ha sido accedida recientemente y si ha sido modificada (escrita). Cuando una página es utilizada, se marca como "accedida", y si se le hace un cambio, también se marca como "modificada".

Cuando se accede a una página, el sistema verifica si ya está en memoria. Si es así, simplemente la marca como usada. Si no, ocurre un "fallo de página", lo que significa que hay que traerla desde otra parte y, si no hay espacio disponible, reemplazar una de las páginas existentes. Para decidir cuál eliminar, se busca una página que no haya sido utilizada recientemente.

Para evitar que todas las páginas parezcan usadas todo el tiempo, hay un proceso que periódicamente reinicia estos estados. Esto permite que el sistema pueda identificar con precisión cuáles páginas llevan más tiempo sin ser accedidas y gestionarlas mejor.

Usamos un sistema de sincronización con bloques syncronized sobre el objeto de GestorMemoria de la manera synchronized (gestorMemoria) en donde ningúna de las dos clases de ProcesadorReferencias y ActualizadorEstado entran en condiciones de carrera ya que la sincronización utilizada permite el uso del recurso de manera no compartida evitando así errores a la hora de ller referencias e ir actualizando estados (opción 2). Cada vez, que ActualizadoEstado desea cambiar los bits de memoria (NRU) ocurre la sincronización, y cuando ProcesadorReferencias desea acceder para actualizar o ingresar otra página ocurre lo mismo.

Las clases ProcesadorReferencias y ActualizadorEstado implementan la interfaz Runnable, permitiendo que trabajen en paralelo (pero sin caer en condiciones de carrera). ProcesadorReferencias gestiona las referencias de acceso a memoria y usa el synchronized sobre GestorMemoria para asegurarse de que solo un hilo pueda modificar GestorMemoria a la vez. ActualizadorEstado restablece los bits de acceso de las páginas y también usa la sincronización del objeto para evitar interferencias.

La sincronización es clave para evitar problemas cuando varios hilos intentan acceder a la memoria al mismo tiempo. Sin ella, podrían ocurrir errores en la actualización del estado de las páginas, haciendo que los datos sean inconsistentes. También es importante en la gestión de fallos de página, ya que el proceso de eliminar y agregar páginas debe ser seguro y sin interrupciones.

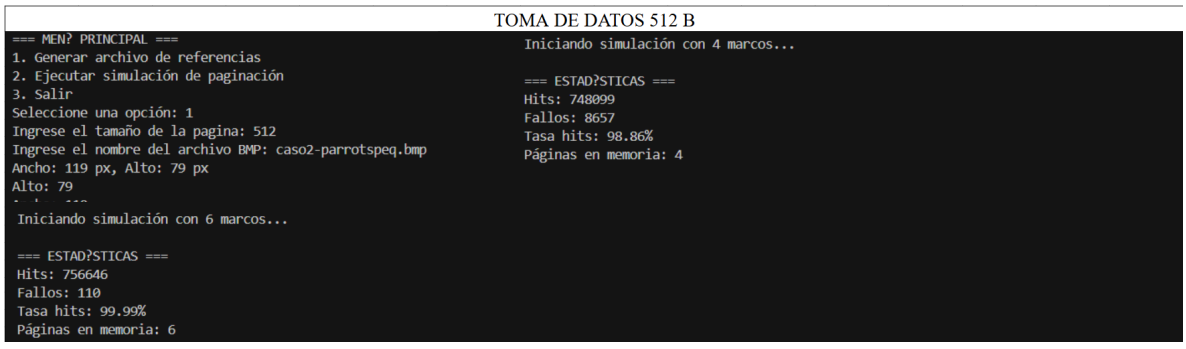
Finalmente, el restablecimiento de bits de acceso se ejecuta en un hilo aparte y necesita acceso a la misma memoria que los demás procesos. Si no estuviera sincronizado, podría causar problemas en la toma de decisiones sobre qué páginas reemplazar.

Datos recopilados de los distintos escenarios de pruebas: En caso de que se necesite se pueden visualizar de mejor manera las tablas en el siguiente link (existen 2 hojas cada una con la información correspondiente por imagen): [Gráficas Auxiliares Documento de Análisis caso 2.xlsx]

Imagen: caso2-parrotspeq.bmp

Página de 512B, parrotspeq.bmp			
Marrcos Asignados	Total referencias	Hits	Fallos de páginas
4	**756756**	**742145**	**14611**
6	**756756**	**756646**	**110**
Página de 1024B, parrotspeq.bmp			
Marrcos Asignados	Total referencias	Hits	Fallos de páginas
4	**756756**	**756700**	**56**
6	**756756**	**756701**	**55**
Página de 2048B, parrotspeq.bmp			
Marrcos Asignados	Total referencias	Hits	Fallos de páginas
4	**756756**	**756726**	**30**
6	**756756**	**756727**	**28**

Escenario 1 (Página de 512B, 4 y 6 marcos de páginas).



Escenario 2 (Página de 1024B, 4 y 6 marcos de páginas).

TOMA DE DATOS 1024 B

```
Seleccione una opción: 1
Ingrese el tamaño de la pagina: 1024
Ingrese el nombre del archivo BMP: caso2-parrotspeq.bmp
Ancho: 119 px, Alto: 79 px
Alto: 79
Ancho: 119
Número obtenido de referencias: 756756
Iniciando simulación con 6 marcos...

=== ESTAD?STICAS ===
Hits: 756701
Fallos: 55
Tasa hits: 99.99%
Páginas en memoria: 4
```

```
Iniciando simulación con 4 marcos...

=== ESTAD?STICAS ===
Hits: 756700
Fallos: 56
Tasa hits: 99.99%
Páginas en memoria: 5
```

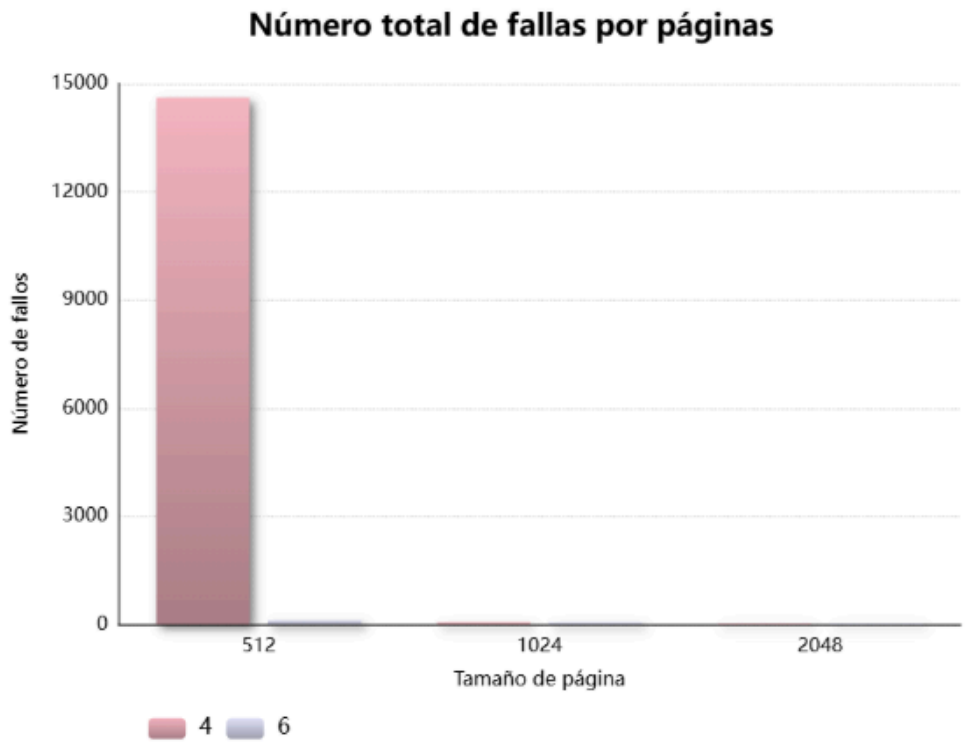
Escenario 3 (Página de 2048B, 4 y 6 marcos de páginas).

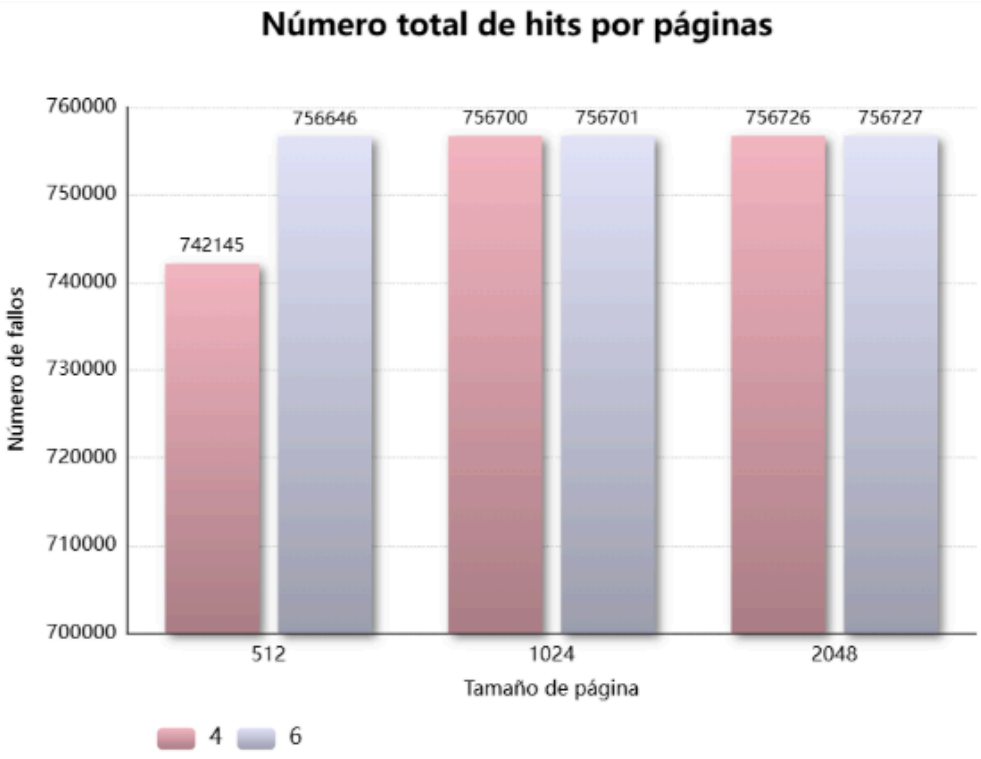
TOMA DE DATOS 2048B

```
Iniciando simulación con 4 marcos...

=== ESTAD?STICAS ===
Hits: 756726
Fallos: 30
Tasa hits: 100.00%
Páginas en memoria: 4


=== ESTAD?STICAS ===
Hits: 756728
Fallos: 28
Tasa hits: 100.00%
Páginas en memoria: 6
```

```
rrotspeq.bm
Iniciando simulación con 4 marcos...

=== ESTAD?STICAS ===
Hits: 756726
Fallos: 30
Tasa hits: 100.00%
Páginas en memoria: 4
```

Gráficas:


Número total de fallas por páginas
15000
12000
9000
6000
3000
0
Número de fallos
512
1024
2048
Tamaño de página
4
6


Número total de hits por páginas
760000
750000
740000
730000
720000
710000
700000
Número de fallos
742145
756646
756700
756701
756726
756727
512
1024
2048
Tamaño de página
4
6

El análisis de las gráficas muestra una clara relación entre el tamaño de página y el rendimiento del sistema de memoria virtual. En la primera gráfica, que representa el número total de fallos de página, podemos observar que cuando el tamaño de página es de 512, los fallos alcanzan casi los 15,000. Sin embargo, si aumentamos el tamaño de página a 1024 y 2048, la cantidad de fallos se reduce prácticamente a cero. Esto indica que el uso de páginas más grandes disminuye significativamente la cantidad de fallos, lo que puede traducirse en un mejor rendimiento del sistema al minimizar las interrupciones por falta de datos en memoria.

En la segunda gráfica, que muestra el número total de hits, podemos notar que con un tamaño de página de 512, el número de aciertos es el más bajo, con 742,145 hits. En cambio, cuando el tamaño de página aumenta a 1024 y 2048, los aciertos se incrementan a aproximadamente 756,700, mostrando una mejora en la eficiencia del uso de la memoria. Esto significa que con tamaños de página más grandes, los datos requeridos permanecen en la memoria por más tiempo, reduciendo la necesidad de acceder al disco, lo que optimiza el rendimiento del sistema.

Imagen: caso2-parrotspeq_sal.bmp

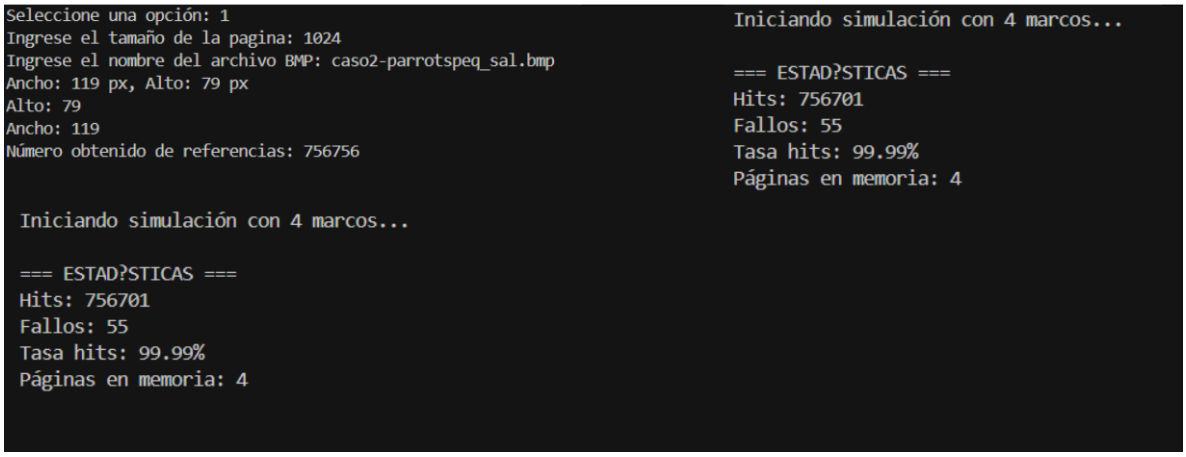
Página de 512B, parrotspeq_sal.bmp			
Marrcos Asignados	Total referencias	Hits	Fallos de páginas
4	**756756**	**742142**	**14614**
6	**756756**	**756646**	**110**
Página de 1024B, parrotspeq_sal.bmp			
Marrcos Asignados	Total referencias	Hits	Fallos de páginas
4	**756756**	**756701**	**55**
6	**756756**	**756701**	**55**
Página de 2048B, parrotspeq_sal.bmp			
Marrcos Asignados	Total referencias	Hits	Fallos de páginas
4	**756756**	**756728**	**28**
6	**756756**	**756728**	**28**

Escenario 1 (Página de 512B, 4 y 6 marcos de páginas).

TOMA DE DATOS 512 B

Escenario 2 (Página de 1024B, 4 y 6 marcos de páginas).

TOMA DE DATOS 1024 B



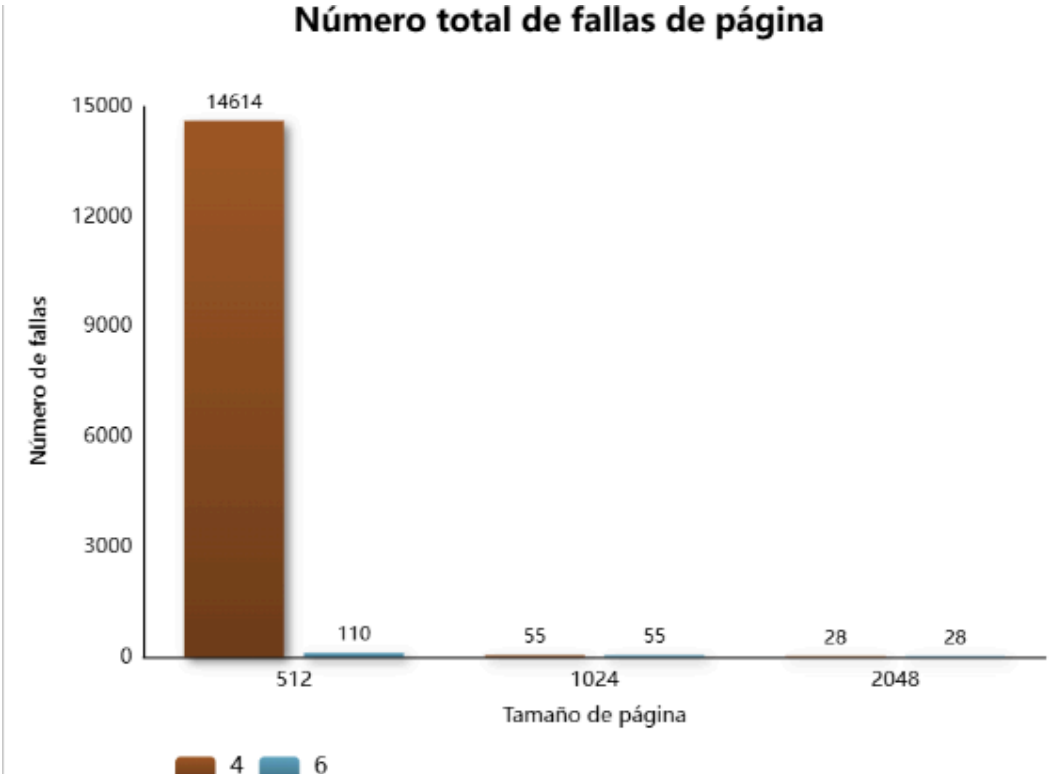
Escenario 3 (Página de 2048B, 4 y 6 marcos de páginas).

TOMA DE DATOS 2048B

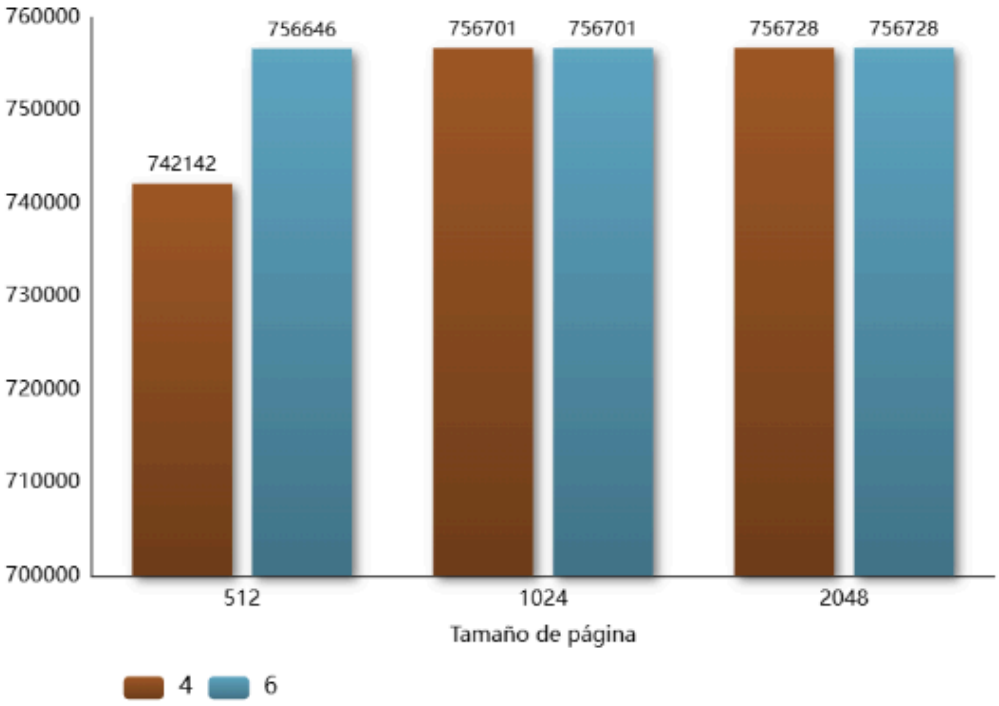
```
Seleccione una opción: 1
Ingrese el tamaño de la pagina: 2048
Ingrese el nombre del archivo BMP: caso2-parrotspeq_sal.bmp
Ancho: 119 px, Alto: 79 px
Alto: 79
Ancho: 119
Número obtenido de referencias: 756756


Iniciando simulación con 6 marcos...

=== ESTAD?STICAS ===
Hits: 756728
Fallos: 28
Tasa hits: 100.00%
Páginas en memoria: 5
```

```
Iniciando simulación con 4 marcos...

=== ESTAD?STICAS ===
Hits: 756728
Fallos: 28
Tasa hits: 100.00%
Páginas en memoria: 4
```

Gráficas asociadas:



Número de hits asociado


760000
750000
740000
730000
720000
710000
700000
742142
756646
756701
756701
756728
756728
512
1024
2048
Tamaño de página
4
6

Casos extra para poder entender de mejor manera:

Caso 1

Los datos ingresados en esta simulación fueron de 1024B de tamaño de página y de 8 marcos.

```
Iniciando simulación con 8 marcos...

=== ESTAD?STICAS ===
Hits: 756701
Fallos: 55
Tasa hits: 99.99%
Páginas en memoria: 8
```

Cómo se logra evidenciar la cantidad de marcos en este caso, no afectó la cantidad de misses ocurridos lo que puede dar idea sobre que la cantidad de misses se deben a que siempre se cargan nueva información en la página lo que causa el fallo inicial en la página. Adicionalmente comparando con los valores iniciales de 6 y 4

```
=== ESTAD?STICAS ===
Hits: 756701
Fallos: 55
Tasa hits: 99.99%
Páginas en memoria: 6
```

```
=== ESTAD?STICAS ===
Hits: 756701
Fallos: 55
Tasa hits: 99.99%
Páginas en memoria: 4
```

Pero si ocurre una reducción en los marcos como se muestra a continuación la cantidad de misses aumenta significativamente lo cual es congruente con la información enunciada y la información teórica (se generaliza a lo largo de los distintos tamaños ya que los resultados fueron los mismos solo que no se alcanzaron a documentar en el documento de análisis).

```
=== ESTAD?STICAS ===
Hits: 597519
Fallos: 159237
Tasa hits: 78.96%
Páginas en memoria: 2
```

Continuando, se sabe que las gráficas son congruentes con la información esperada y debido a la sincronización implementada (bloques de synchronize sobre el objeto gestor de memoria). Mostrando valores cercanos a los valores esperados reiterando que los valores de marco más grandes permiten mejores manejos de errores reduciendo así la cantidad de misses y por ende de SWAP en el disco y aumentando la cantidad de hits volviéndose así más eficiente.

Caso 2: La simulación fue hecha con los siguientes valores iniciales y con el archivo ahí presente.

```
=== MEN? PRINCIPAL ===
1. Generar archivo de referencias
2. Ejecutar simulación de paginación
3. Salir
Seleccione una opción: 1
Ingrese el tamaño de la pagina: 512
Ingrese el nombre del archivo BMP: caso2-parrotspeq.bmp
Ancho: 119 px, Alto: 79 px
Alto: 79
Ancho: 119
```

A su vez, los valores intermedios entre 4-6 no mostraron grandes diferencias como se muestra a continuación:

```
=== ESTAD?STICAS ===
Hits: 756639
Fallos: 117
Tasa hits: 99.98%
Páginas en memoria: 5
```

Por otro lado, los valores obtenidos cuando se reduce aún mas los marcos de página (<4) los valores crecen de manera exponencial (los fallos de página) aumentando directamente los valores de tiempo y el funcionamiento de la simulación

```
Iniciando simulación con 2 marcos...

=== ESTAD?STICAS ===
Hits: 595542
Fallos: 161214
Tasa hits: 78.70%
Páginas en memoria: 2
```

Siendo estos valores casi 10 veces mas grandes que los iniciales (valores con 4 marcos).

Caso 3: Los valores tomados para el caso 3 de simulación fueron los siguientes

```
=== MEN? PRINCIPAL ===
1. Generar archivo de referencias
2. Ejecutar simulación de paginación
3. Salir
Seleccione una opción: 1
Ingrese el tamaño de la pagina: 2048
Ingrese el nombre del archivo BMP: caso2-parrotspeq.bmp
Ancho: 119 px, Alto: 79 px
Alto: 79
Ancho: 119
```

Los valores intermedios no mostraron algún cambio significativo por ende se decide aumentar la cantidad de marcos y disminuir estos valores para lograr ver una diferencia.

```
Iniciando simulación con 3 marcos...

=== ESTAD?STICAS ===
Hits: 750191
Fallos: 6565
Tasa hits: 99.13%
Páginas en memoria: 3
```

```
=== ESTAD?STICAS ===
Hits: 756728
Fallos: 28
Tasa hits: 100.00%
Páginas en memoria: 12
```

Los valores en este caso permiten concluir que la mínima cantidad de fallos que puede haber en la simulación son 28. Ya que, se mantiene a lo largo de las páginas (desde el 4 en adelante) lo cual, habla sobre la eficiencia necesaria en este caso para aplicar el filtro sobel sobre la imagen y la importancia de los recurso en páginas de 2048 B con 4 marcos minimizando los impactos de consumo de recursos con 4 marcos siendo este el más efectivo ya que con 3 se dispararon los misses con el NRU que se tiene.

Conclusiones:

Para concluir, se sabe así que los valores obtenidos entran en el margen del 5%+- previsto lo cual deja inferir que la simulación es correcta. Adicionalmente, la cantidad y el formato de referencias es congruente al teórico o esperado inicialmente indicando así que los valores teóricos sobre los cuales se trabajaron fueron correctos.

Continuando, observando los resultados obtenidos a lo largo de las simulaciones (Valores de las tablas 512, 1024 y 2048) se permite concluir que aumentando los valores del tamaño de las páginas mejora significativamente el rendimiento de los programas (teóricamente

hablando) ya que, los valores de fallos de páginas disminuyen en casi un 1000% lo cual es satisfactorio para cualquier programa en ejecución hablando de manera teórica.

En cuanto a los casos propuestos, se logra ver y comprobar el como la eficiencia de los recursos a lo largo de las simulaciones no aumenta desde cierto punto, y de ahí en adelante se muestra un desperdicio de recursos (como es el caso de aumentar la cantidad de marcos) consiguiendo valores similares pese aumentar el consumo de espacio (desde la creación de páginas más grandes hasta aumento en los marcos de página.

A partir de estos resultados, se puede concluir que el uso de tamaños de página más grandes, como 1024 o 2048, es más eficiente en términos de reducción de fallos y aumento de aciertos. No obstante, debemos considerar que páginas más grandes también pueden generar desperdicio de memoria si los procesos no utilizan completamente el espacio asignado. Por ello, dependiendo del contexto de uso, un tamaño de página de 1024 podría ser un punto medio adecuado, equilibrando rendimiento y uso eficiente de la memoria.

Finalmente, la localidad. Entendiendo localidad como probabilidad de que si se accede a una posición de memoria en el futuro se accedan a posiciones cercanas, al aplicar el filtro sobel ocurre un problema de localidad alta, sobre todo en espacio. Esto debido a que como trabajamos con una ventana de píxeles 3x3 (dada la implementación del filtro sobel), es probable que se vuelva a necesitar de manera próxima en la siguiente operación. Además, los datos se descartan al avanzar en la imagen debido al uso del NRU. Ya que el procesamiento se hace en pequeñas regiones de la imagen y se avanza de manera sistemática, el filtro Sobel se beneficia del almacenamiento en caché, reduciendo la cantidad de accesos a memoria principal. Por esto, se considera una operación con alta localidad en comparación con otros algoritmos que acceden a datos de manera dispersa.