Caso 2 – Memoria Virtual

Integrantes del grupo

- Giuliana Volpi Muñoz 202123986
- Tomás Camilo Ostos Medina - 202123214
- Sebastián Martinez Arias - 202312210

Contrato de trabajo del grupo

Nombre del equipo: Grupo Caso 2 – Memoria Virtual

Objetivo del contrato: Establecer los lineamientos de trabajo colaborativo y compromiso entre los integrantes del grupo para la entrega exitosa y puntual del proyecto asignado en el marco del curso ISIS 2203.

Responsabilidades compartidas:

- Todos los miembros se comprometen a asistir (virtualmente) a las reuniones establecidas para planear, desarrollar y revisar el proyecto.
- Las decisiones técnicas y de diseño serán tomadas de manera consensuada, es decir, por mayoría de votos.
- Cada integrante se responsabiliza por conocer y comprender el funcionamiento global del proyecto, sin importar la parte específica que haya desarrollado.
- Todos los integrantes deben contribuir a al diseño del proyecto y resolver los problemas que se presenten en conjunto.
- Cada miembro debe documentar adecuadamente su trabajo, asegurando que otros integrantes puedan entender y continuar su desarrollo si es necesario.
- Todos los integrantes se comprometen a realizar revisiones cruzadas del código y los análisis, para garantizar la calidad y coherencia del resultado final.
- En caso de ausencia o dificultad, cada miembro debe notificar con anticipación al equipo y buscar un mecanismo de apoyo para cumplir con su parte del trabajo.
- Todos deben participar activamente en la elaboración del informe final y en la preparación de la presentación del proyecto, sin delegar completamente estas tareas.

- Se espera que cada integrante mantenga una comunicación constante y clara con el resto del equipo, utilizando los canales acordados y respetando los tiempos de respuesta.

Distribución de tareas (flexible y adaptativa):

- **Giuliana**: Elaboración de la Opción 1 del caso y participación en la elaboración del documento de análisis.
- **Tomás**: Elaboración de la Opción 2 y participación en la elaboración del documento de análisis.
- **Sebastián**: Elaboración de la Opción 2 y participación en la elaboración del documento de análisis

Cada miembro podrá intercambiar tareas con otro en caso de inconvenientes, manteniendo la comunicación abierta y continua.

Metodología de trabajo:

- Herramientas: Visual Studio Code, GitHub y Google Drive para documentación.
- Comunicación: grupo de WhatsApp y reuniones por Zoom según cronograma.
- Se usarán versiones compartidas y controladas del código para evitar conflictos o pérdidas de avances.

Cronograma interno estimado:

- 21 - 25 marzo: Desarrollo opción 1 (generador de referencias)
- 25 - 30 marzo: Desarrollo opción 2 (simulador con hilos y algoritmo NRU)
- 31 marzo - 2 abril: Redacción del informe, gráficas y revisión final

Compromiso ético:

- El trabajo será completamente original y realizado por los miembros del grupo, sin copiar soluciones externas.
- Cada integrante acepta que podría ser evaluado individualmente sobre cualquier parte del proyecto.
- Nos comprometemos a reportar cualquier incumplimiento de trabajo por parte de algún miembro en caso de que ocurra.
- En caso de presentar algún inconveniente y que no se pueda resolver entre los miembros del equipo se procederá a tratarlo con los monitores y posteriormente con el asistente graduado.

Aprobación: Este contrato ha sido discutido y aprobado por todos los integrantes del grupo el día 21 de marzo de 2025. Será incluido como parte del informe entregado.

Modo de uso del programa

Dirigido para la persona que ejecute el programa solución y tenga una breve guía de cómo usarlo correctamente.

Paso1

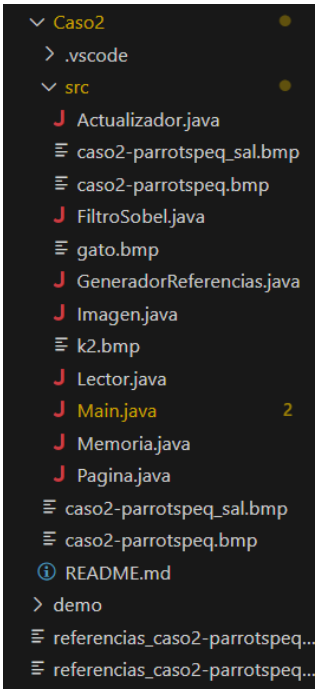
Descargue el archivo .zip y extráigalo en la ubicación de preferencia.

Paso2

Abra el directorio en su editor de preferencia (Visual estudio Code recomendado)

Paso3

Al abrir el directorio encontrara los archivos .java, .bmp, .txt pertinentes para el caso organizados de la siguiente manera:

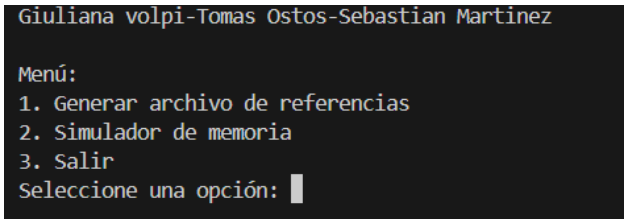


Estas clases y archivos son la escencia del proyecto, aquellos proporcionados por el caso son la clase Imagen, la clase FiltroSobel, el archivo tipo bmp caso2-parrotspeq y caso2_parrotspeq_sal. Las otras clases creadas por los estudiantes para la simulacion de memoria virtual son la clase Actualizador, GeneradorReferencias (clase que genera la lista de referencias para la memoria virtual), Lector, Memoria, Pagina y los archivos bmp con los

que se crean nuevos casos de prueba son k2 y gato. La clase principal main es donde usted deberia correr el programa.

Paso 4:

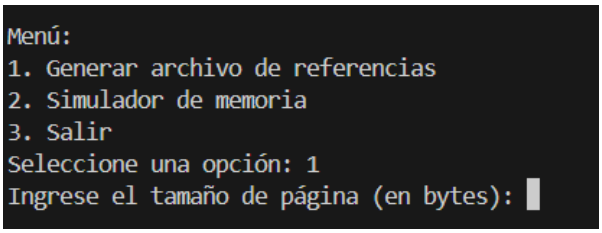
Diríjase al main y ejecute el programa. Al ejecutarlo deberia aparecerle lo siguiente:



Este es el menú que le permitirá interactuar con el programa. Los nombres de la parte superior pertenecen a los integrantes del grupo que realizaron el proyecto y en la parte inferior de Menú se encuentran las opciones que podrá elegir del simulador de memoria virtual.

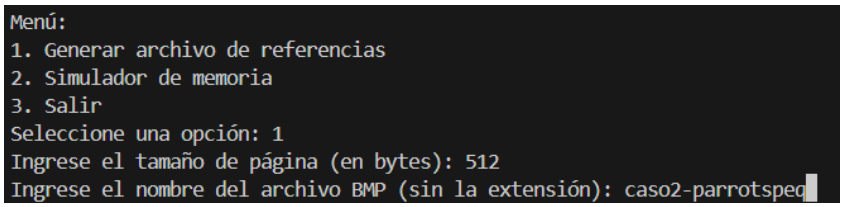
Paso 5:

Seleccione la opción 1, le aparecerá lo siguiente:

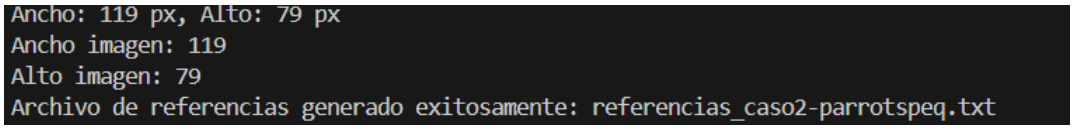


Para este caso ingrese los 512 bytes para el tamaño de página, esto con el fin de simular el ejemplo propuesto en el caso

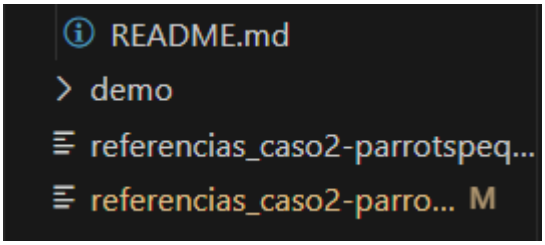
Luego de ingresar el tamaño de página le pedira el nombre del archivo bmp SIN LA EXTENSION, introduzca caso2-parrotspeq para este caso.



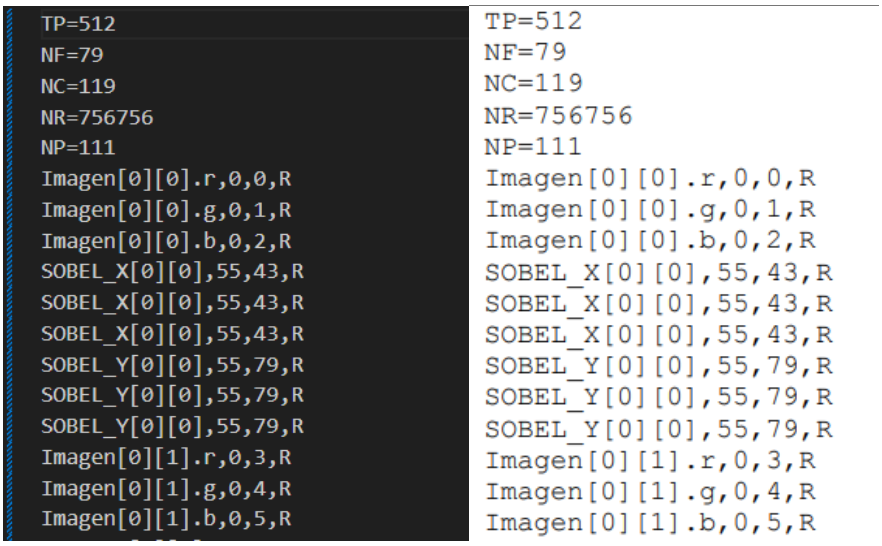
Al ejecutarlo, espere unos segundos y le saldra en la consola lo siguiente:



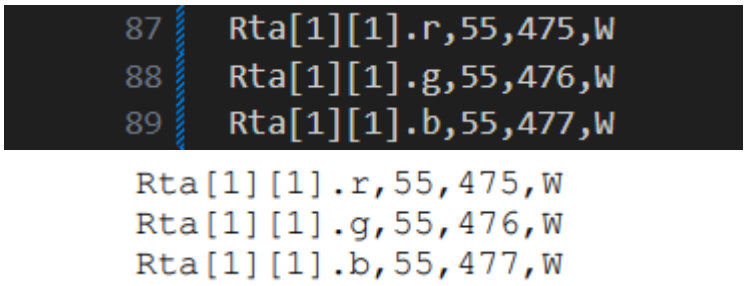
Note usted que se ha creado el archivo de referencias de la imagen propuesta, podra observar este archivo en la carpeta demo de la siguiente manera:



Podrá observar cómo se creó el archivo exitosamente con los datos esperados para el caso:



Además, note usted que la secuencia de las respuestas cumple con las iteraciones esperadas antes de su generación (81, descartando las primeras líneas que no son referencias), podrá observar las primeras respuestas en las líneas 87,88 y 89 respectivamente.



Paso 6:

Seleccione la opción 2 del caso el número de marcos en memoria que desee e ingrese el nombre del archivo de referencias que se generó sin la extensión:

```
Menú:
1. Generar archivo de referencias
2. Simulador de memoria
3. Salir
Seleccione una opción: 2
Ingrese el número de marcos en memoria: 16
Ingrese el nombre del archivo de referencias (sin el txt): referencias_caso2-parrotspeq
```

Espere unos segundos y al ingresar lo anterior le aparecerá lo siguiente:

```
Menú:
1. Generar archivo de referencias
2. Simulador de memoria
3. Salir
Seleccione una opción: Total de hits: 756646
Total de fallos de página: 110
Tiempo en segundos: 0.8432456
```

¡Ha utilizado correctamente el programa! Si desea probar con las otras imagenes en el proyecto y entenderlo

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

Para generar el archivo de referencias de página, simulamos el comportamiento del filtro de Sobel aplicado sobre una imagen BMP de 24 bits, que accede a memoria virtual. El objetivo principal de este algoritmo es construir un archivo de texto que registre, siguiendo el formato esperado, todas las referencias de memoria (lectura y escritura) asociadas al procesamiento de la imagen, teniendo en cuenta un tamaño de página definido por el usuario. Para cumplir con dicho objetivo, inicia definiendo un esquema de asignación de memoria virtual que separa el espacio en bloques contiguos de bytes para distintos propósitos: primero se reserva el espacio necesario para almacenar los datos de la imagen original en formato RGB (3 bytes por píxel), seguido por el espacio requerido para almacenar las matrices del filtro de Sobel en dirección X y Y (cada una de 3x3 enteros de 4 bytes), y por último el espacio necesario para almacenar la imagen resultante (también en formato RGB). Estos bloques están representados a través de variables de offset (offsetImagen, offsetFiltroX, offsetFiltroY, offsetRespuesta) que marcan el inicio de cada segmento en la memoria simulada. A partir del

tamaño total de memoria requerido, se calcula también el número total de páginas necesarias, considerando el tamaño de página ingresado.

La parte central del algoritmo consiste en un recorrido por todos los píxeles interiores de la imagen. Para cada píxel (i, j), se simula la lectura de los valores RGB de cada uno de los nueve píxeles que componen su vecindario. Por cada color, se calcula la posición virtual en memoria, la página correspondiente y el desplazamiento dentro de esa página. Adicionalmente, se simulan también las lecturas necesarias de los valores de los filtros Sobel en X e Y, accediendo a las posiciones específicas dentro de sus respectivas matrices de 3x3. Por cada operación de lectura, se construye una referencia en formato textual que incluye la dirección (página y desplazamiento), el tipo de acceso (R para lectura) y la ubicación del dato referenciado, por ejemplo, Imagen[x][y].r o SOBEL_X[1][2]. Finalmente, para cada píxel procesado, se generan tres accesos de escritura sobre la imagen de salida (uno por cada color RGB), los cuales también se representan como referencias de memoria, esta vez con tipo W (escritura). Todo esto se almacena en una lista secuencial que guarda el orden de la simulación de accesos a memoria.

Estructuras de datos usadas para simular el comportamiento del sistema de paginación

El proyecto actual aborda este desafío mediante la implementación de un simulador de memoria virtual, el cual se apoya en una serie de estructuras de datos e hilos concurrentes que simulan el proceso de paginación y reemplazo utilizando el algoritmo NRU (Not Recently Used). Ademas,El código emula cómo el hardware real manejaría la memoria durante el procesamiento de la imagen con el filtro Sobel. Las estructuras serán mencionadas a continuación:

Opcion 1 – Generacion de archivo de referencias.

Clase Main:

La clase main sirve como entrada del usuario para el sistema del caso. Mediante un menú que se muestra en consola, el usuario final puede interactuar con las opciones del programa. El usuario puede elegir entre generar referencias o simular la memoria virtual.

Clase FiltroSobel:

La clase FiltroSobel se encarga principalmente de detectar los bordes de la imagen que el usuario desea. Luego de detectarlos los reemplaza con transiciones claras y oscuras para detectarlas correctamente.

Clase GeneradorReferencias:
La clase se encargar de simular el sistema de paginación durante la aplicación del filtro Sobel. Al utilizarse genera el archivo con:
Tamaño de página (en bytes)

1. NF y NC: Número de filas y columnas de la imagen

2. NR: Número de referencias (en el archivo)
3. NP: Número de páginas virtuales (las páginas necesarias para almacenar la matriz imagen, las matrices de los dos filtros y la matriz de la imagen resultante)
4. Lista de todas las referencias del archivo

Los atributos y estructuras usadas en esta clase son:

- **tamPagina:** Tamaño de cada página de memoria en bytes
- **referencias:** Lista que almacena todas las operaciones de memoria
- **OffsetImagen:** Inicio de los datos de la imagen (3 bytes por píxel)
- **OffsetFiltroX:** Inicio del kernel Sobel X (9 enteros × 4 bytes)
- **OffsetFiltroY:** Inicio del kernel Sobel Y
- **OffsetRespuesta:** Inicio de la zona para resultados
- **ImagenIn:** Objeto que contiene los datos de la imagen original
    - alto (NF): Número de filas de píxeles
    - ancho (NC): Número de columnas de píxeles
    - Matriz de píxeles con sus canales RGB
- **NombreArchivo:** Almacena el nombre del archivo BMP a procesar (sin extensión), para construir las rutas de entrada/salida
- **TotalBytes:** Memoria total necesaria para todos los datos
- **numPaginas:** Número de páginas requeridas (redondeo hacia arriba)

Cabe resaltar que la opcion que genera el archivo de referencias solo funciona con imágenes tipo BMP, y la opcion 2 con el archivo generado .txt.

Opción 2 – Simulación de Memoria

Dentro del escenario de la opción 2 tenemos un apartado que hace la función de simular la memoria del computador, para que se tenga en cuenta cómo funciona el entrar al apartado de las páginas o el solo buscarlos, pero antes de eso, necesitamos algo que nos ayude a simular las páginas antes de entrar al apartado de memoria, por eso creamos una clase llamada Pagina la cual tiene un id (String) y dos boolean, uno para representar que ha sido modificada o no una página y otro que indica si ha sido referenciada o no.

Esta opción fue representada bajo la clase de Memoria, esta clase posee 5 atributos importantes para el funcionamiento de la simulación:

Clase Memoria:

- hits: Cuenta la cantidad de accesos a páginas que ya están en memoria y que son encontradas
- fallas: Cuenta la cantidad de fallos de página (cuando se accede a una página que no está en memoria).
- numMarcos: Número máximo de marcos de página disponibles en la memoria.

- memoria: Mapa (HashMap<String, Pagina>) que almacena las páginas actualmente en memoria, utilizando su identificador como clave.
- colaPaginas: Cola (Queue<String>) que mantiene el orden de llegada de las páginas a la memoria, para facilitar su eliminación en caso de reemplazo.

Ahora con respecto a la implementación tenemos funciones las cuales definen el uso de estos atributos, desglosaremos cada una de estas para saber cómo funcionan:

- reemplazarPagina(String id, boolean esEscritura): Con respecto a esta función, se encarga de reemplazar una página dentro del mapa de memoria cuando no hay espacio disponible dentro de la memoria, esto sigue el algoritmo NRU (Not Recently Used) el cual como podemos ver dentro de la implementación del codigo divide las páginas en 4 clases distintas para seleccionar según la importancia de cada una de estas
- agregarNuevaPagina(String id, boolean esEscritura): Esta función se encarga de agregar una página dentro de la memoria y la marca como referenciada, adicionalmente en dado caso de que sea de escritura también la marca.
- buscarPagina(String id, boolean esEscritura): Se encarga de buscar una página dentro de la memoria, en dado caso de que la encuentre aumenta los hits, se marca la página como referenciada y si tiene apartado de escritura se marca también como modificada; En dado caso de que la página no esté en memoria, se aumentan los hits y dependiendo si la memoria está llena o no se ejecuta una de las dos funciones anteriores, en caso de que este llena se usa reemplazarPagina, de lo contrario usamos agregarNuevaPagina
- resetearBitsReferencia(): Reestablece los bits de referencia de todas las páginas que se encuentran dentro de memoria
- mostrarEstadisticas(): Muestra la cantidad de hits y fallas de páginas.

Finalmente, ya con esto en mente pasamos a hablar de las últimas dos clases adicionales creadas, siendo esta la de Lector y la de Actualizador.

Clase Lector:

La clase Lector es un hilo que simula el acceso a memoria en un sistema de paginación. Lee el archivo de referencias generado por el GeneradorReferencias, ignorando las primeras 6 líneas (las cuales son TP, NF, NC, NR, NP), y por cada referencia posterior extrae el número de página y tipo de operación, ya sea W o R para enviarlo a una memoria simulada. Mide el tiempo de ejecución total y cada 10,000 accesos hace una pausa mínima para evitar sobrecarga, mostrando al final el rendimiento en estadisticas como fallos de página y tiempo transcurrido.

Clase Actualizador:

Esta clase solo tiene un atributo siendo la memoria, que debe ser la misma usada para la clase del lector, la función de esta clase es la de actualizar los bits de referencia de todas las páginas.

Esquema de Sincronización Usado

Con lo que respecta al esquema de sincronización hacemos uso de exclusión mutua, esto es necesario para que en la memoria no se acceda al mismo recurso al tiempo, de tal forma logramos hacer que los cambios se efectúen sin ninguna clase de error extraño, cada función que es usada tanto por el lector como por el actualizador tienen la propiedad synchronized lo cual permite hacer esta sincronización y evitar errores inesperados.

Tabla de datos recopilados

Se realizaron los experimentos para las dos imágenes dadas y se incluyeron dos imágenes adicionales:

- gato.bmp
- k2.bmp

A continuación, se adjuntan las tablas de datos recopilados para las 4 imágenes con las siguientes especificaciones:

Número de página: 128B, 256B, 512B y 1024B

Marcos de página asignados: 4, 8, 16, 32

Tabla datos experimento Parrots.bmp:

Parrots					
Num Página	Marcos Asignados	Total Referencias	Hits	Fallas	Tiempo (seg)
128	4	756756	685922	70834	0.3121
	8	756756	752375	4381	0.1801
	16	756756	756259	497	0.1642
	32	756756	756283	473	0.1869
Num Página	Marcos Asignados	Total Referencias	Hits	Fallas	Tiempo (seg)
256	4	756756	716276	40480	0.3032
	8	756756	756513	243	0.1912
	16	756756	756536	220	0.1867
	32	756756	756536	220	0.1667
Num Página	Marcos Asignados	Total Referencias	Hits	Fallas	Tiempo (seg)
512	4	756756	738417	18339	0.2106
	8	756756	756646	110	0.1638
	16	756756	756646	110	0.2087
	32	756756	756646	110	0.1751
Num Página	Marcos Asignados	Total Referencias	Hits	Fallas	Tiempo (seg)
1024	4	756756	756245	511	0.15
	8	756756	756701	55	0.2004
	16	756756	756701	55	0.1784
	32	756756	756701	55	0.1845

Tabla datos experimento Parrots_sal.bmp:

Parrots_sal					
Num Página	Marcos Asignados	Total Referencias	Hits	Fallas	Tiempo (seg)
128	4	756756	700709	56047	0.2496
	8	756756	754966	1790	0.2092
	16	756756	756208	548	0.2171
	32	756756	756207	549	0.1931
Num Página	Marcos Asignados	Total Referencias	Hits	Fallas	Tiempo (seg)
256	4	756756	711771	44985	0.1782
	8	756756	756530	226	0.20001
	16	756756	756536	220	0.1562
	32	756756	756536	220	0.1613
Num Página	Marcos Asignados	Total Referencias	Hits	Fallas	Tiempo (seg)
512	4	756756	741020	15736	0.1348
	8	756756	756646	110	0.1399
	16	756756	756646	110	0.1745
	32	756756	756646	110	0.1322
Num Página	Marcos Asignados	Total Referencias	Hits	Fallas	Tiempo (seg)
1024	4	756756	756363	393	0.1981
	8	756756	756701	55	0.1616
	16	756756	756701	55	0.1676
	32	756756	756701	55	0.1648

Tabla datos experimento Gato.bmp:

Gato					
Num Página	Marcos Asignados	Total Referencias	Hits	Fallas	Tiempo (seg)
128	4	1482852	1403858	78994	0.9029
	8	1482852	1480695	2157	0.6826
	16	1482852	1481375	1477	0.5308
	32	1482852	1481473	1379	0.4971
Num Página	Marcos Asignados	Total Referencias	Hits	Fallas	Tiempo (seg)
256	4	1482852	1405235	77617	0.6343
	8	1482852	1482367	485	0.5138
	16	1482852	1482368	484	0.5650
	32	1482852	1482328	524	0.5253
Num Página	Marcos Asignados	Total Referencias	Hits	Fallas	Tiempo (seg)
512	4	1482852	1438312	44540	0.2640
	8	1482852	1482638	214	0.3046
	16	1482852	1482638	214	0.2732
	32	1482852	1482639	213	0.2291
Num Página	Marcos Asignados	Total Referencias	Hits	Fallas	Tiempo (seg)
1024	4	1482852	1475639	7213	0.3015
	8	1482852	1482745	107	0.2819
	16	1482852	1482745	107	0.2763
	32	1482852	1482745	107	0.2721

Tabla datos experimento K2:

K2					
Num Página	Marcos Asignados	Total Referencias	Hits	Fallas	Tiempo (seg)
128	4	1129632	904357	225275	0.2493
	8	1129632	1122281	7351	0.3188
	16	1129632	1128246	1386	0.2171
	32	1129632	1128685	947	0.2254
Num Página	Marcos Asignados	Total Referencias	Hits	Fallas	Tiempo (seg)
256	4	1129632	1058943	70689	0.2151
	8	1129632	1128936	696	0.2499
	16	1129632	1129299	333	0.2341
	32	1129632	1129305	327	0.2299
Num Página	Marcos Asignados	Total Referencias	Hits	Fallas	Tiempo (seg)
512	4	1129632	1076265	53367	0.2148
	8	1129632	1129469	163	0.2466
	16	1129632	1129469	163	0.2581
	32	1129632	1129469	163	0.2897
Num Página	Marcos Asignados	Total Referencias	Hits	Fallas	Tiempo (seg)
1024	4	1129632	1124897	4735	0.3156
	8	1129632	1129550	82	0.2866
	16	1129632	1129550	82	0.2362
	32	1129632	1129550	82	0.2923

Gráficas del comportamiento del sistema e Interpretación de los resultados

A partir de las tablas de datos resultantes de los experimentos para las 4 imágenes se realizaron 3 tipos de gráficos:

- Marcos Asignados vs Número de Fallos de Página
- Marcos Asignados vs Número de Hits
- Marcos Asignados vs Tiempo de ejecución

Gráficas Parrots:

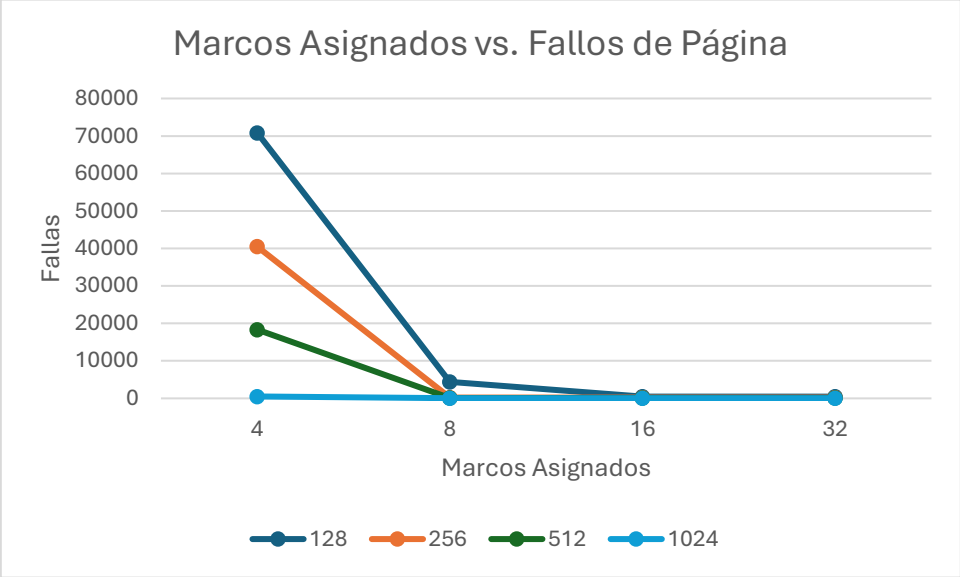


Ilustración 1: Marcos Asignados vs. Fallos de Página

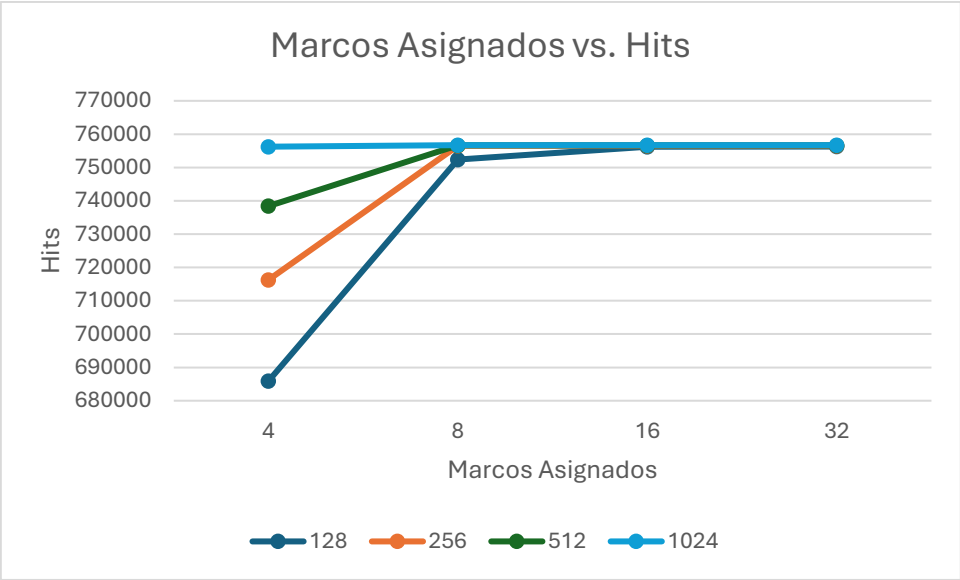


Ilustración 2: Marcos Asignados vs. Hits

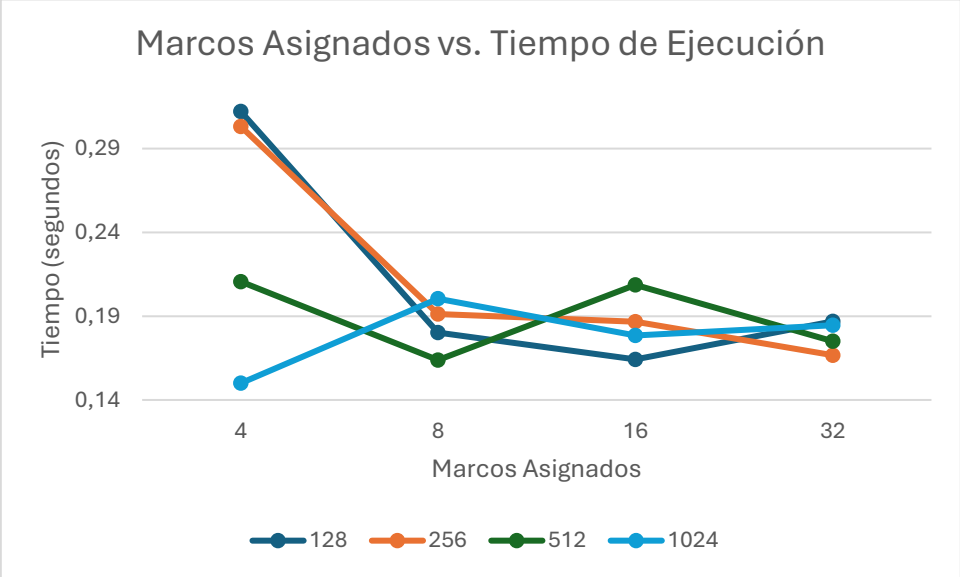


Ilustración 3: Marcos Asignados vs. Tiempo de Ejecución

Para el archivo *parrot.bmp*, las gráficas reflejan un comportamiento consistente con la teoría de la gestión de memoria virtual mediante paginación. En primer lugar, la gráfica de fallos de página muestra una reducción abrupta del número de fallos al pasar de 4 a 8 marcos asignados, independientemente del tamaño de página utilizado. Esta tendencia se mantiene con valores mínimos a partir de los 8 marcos, lo que evidencia que una asignación mínima adecuada de marcos puede contener efectivamente el conjunto de trabajo del proceso, reduciendo la frecuencia de reemplazos.

La gráfica de hits refuerza esta observación: a partir de los 8 marcos, el número de aciertos se estabiliza en un valor cercano al total de referencias, especialmente con tamaños de página de 512 y 1024 bytes. Esto demuestra que a mayor tamaño de página, la cantidad de datos que pueden residir en memoria por cada marco aumenta, mejorando así el rendimiento del sistema al reducir el número de accesos a disco.

Respecto a la gráfica de tiempo de ejecución, se observa una notable disminución del tiempo al pasar de 4 a 8 marcos, particularmente en tamaños de página pequeños (128 y 256 bytes). A partir de ese punto, el tiempo tiende a estabilizarse, aunque se presentan ligeras oscilaciones debido posiblemente al comportamiento interno de la gestión del buffer o del sistema operativo. En conjunto, estos resultados muestran que el rendimiento óptimo en este caso se logra con una configuración intermedia de marcos (8 o 16) y tamaños de página moderados, minimizando los fallos de página sin comprometer el uso de memoria.

Gráficas Parrots Sal:

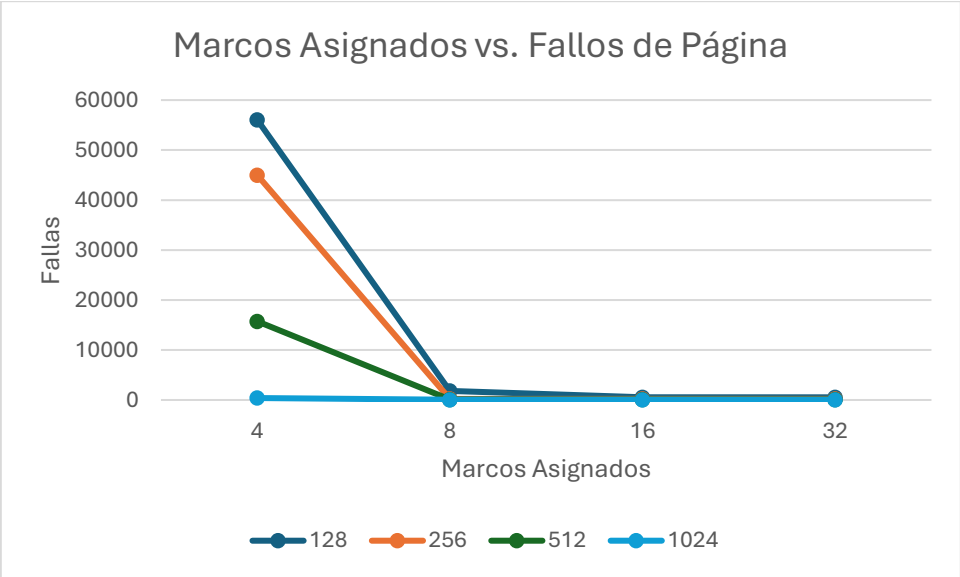


Ilustración 4: Marcos Asignados vs. Fallos de Página

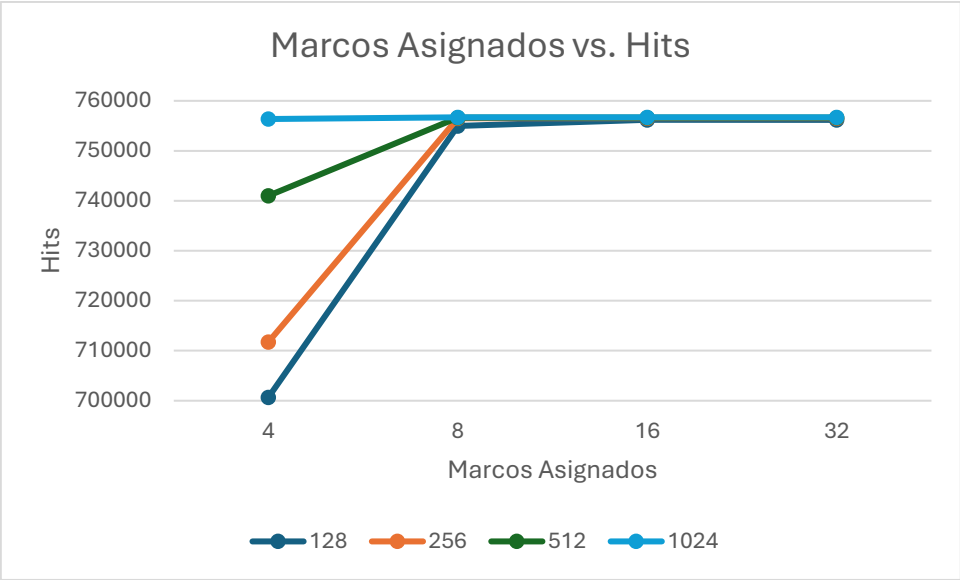


Ilustración 5: Marcos Asignados vs. Hits

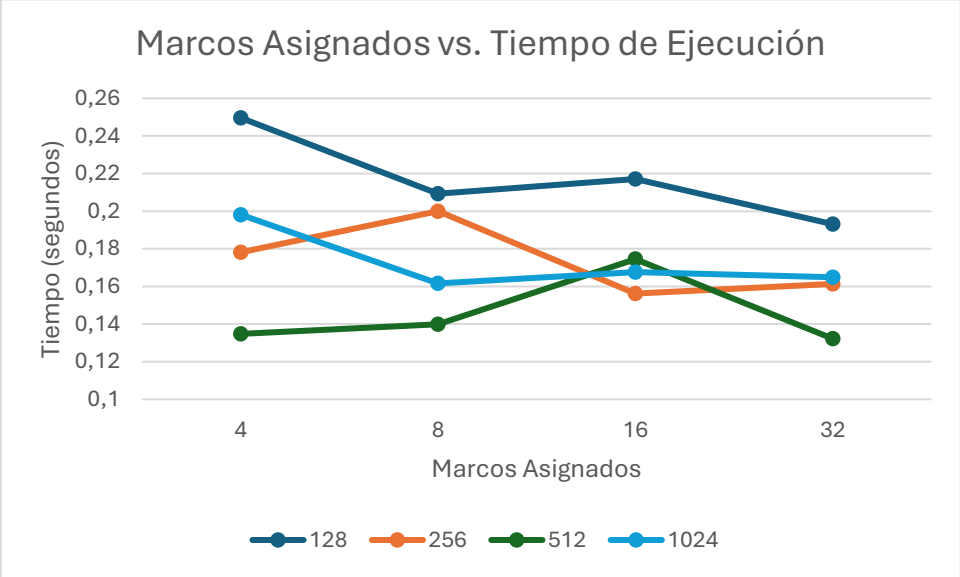


Ilustración 6: Marcos Asignados vs. Tiempo de Ejecución

Las gráficas correspondientes a la tabla *Parrots_sal* muestran un comportamiento esperado y coherente con los principios de la gestión de memoria virtual basada en paginación. En la gráfica de fallos de página, se evidencia una caída drástica en el número de fallos al aumentar los marcos asignados, especialmente para tamaños de página pequeños. A partir de 8 marcos, las fallas se reducen significativamente y se estabilizan en valores muy bajos a partir de 16 marcos, lo que sugiere una mayor eficiencia de la caché de páginas.

En cuanto a los hits, estos aumentan proporcionalmente con el número de marcos, alcanzando valores cercanos al total de referencias cuando se asignan 16 o más marcos, especialmente para tamaños de página de 512 y 1024 bytes. Esto indica que, con una mayor disponibilidad de marcos y tamaños de página más grandes, el sistema es capaz de mantener en memoria un mayor conjunto de páginas activas, reduciendo el reemplazo constante.

Finalmente, la gráfica de tiempo de ejecución refuerza esta interpretación: a medida que se incrementan los marcos asignados, el tiempo tiende a disminuir, en especial cuando se pasa de 4 a 8 marcos. Sin embargo, se observa cierta estabilización en el tiempo a partir de los 16 marcos, lo que sugiere un punto de rendimiento óptimo. En general, se concluye que una configuración equilibrada entre tamaño de página y número de marcos puede reducir considerablemente la latencia por fallos de página y mejorar la eficiencia del programa.

Gráficas Gato:

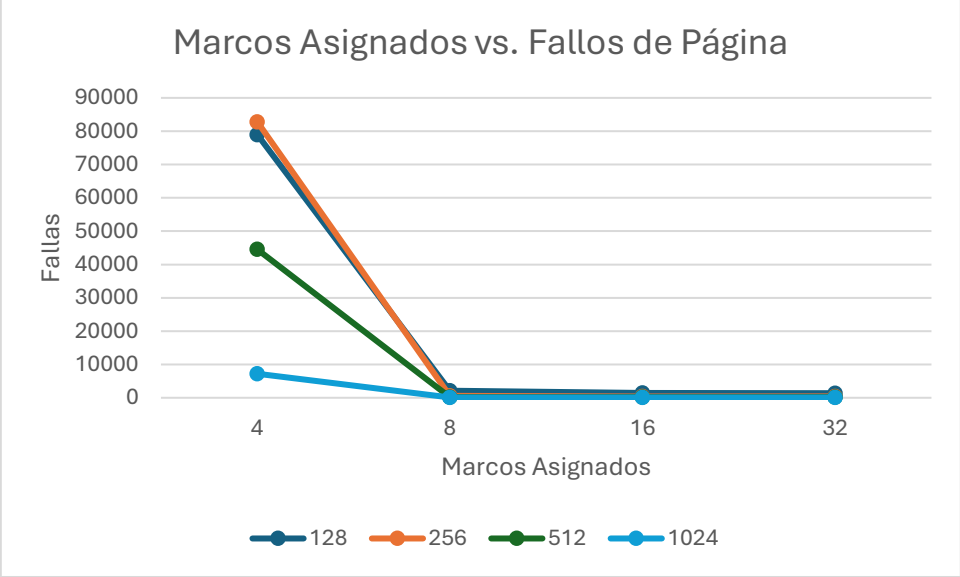


Ilustración 7: Marcos Asignados vs. Fallos de Página

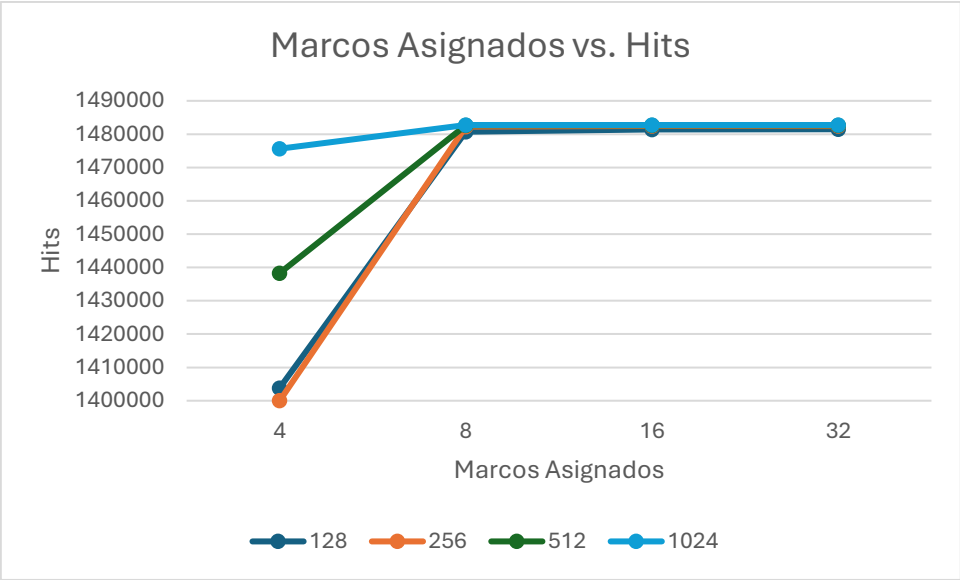


Ilustración 8: Marcos Asignados vs. Hits

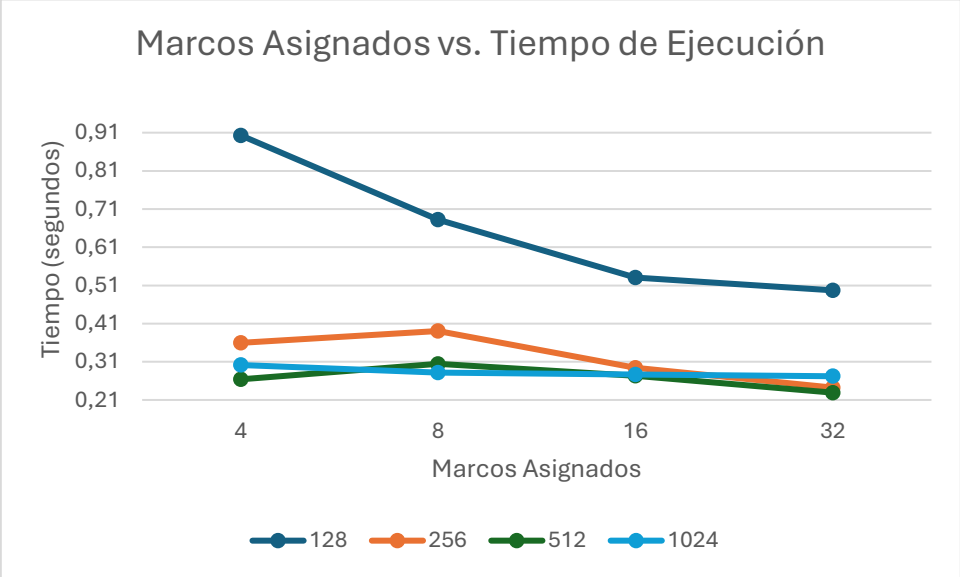


Ilustración 9: Marcos Asignados vs. Tiempo de Ejecución

Los resultados obtenidos para la imagen *gato.bmp* presentan un comportamiento aún más marcado en términos de eficiencia de paginación. La gráfica de fallos de página evidencia una altísima cantidad de fallos cuando se trabaja con solo 4 marcos asignados, en particular para tamaños de página pequeños, superando los 80.000 fallos. Esta cifra disminuye de manera drástica al asignar 8 marcos, estabilizándose en niveles prácticamente nulos a partir de ese punto, lo que demuestra que la presión de memoria causada por el conjunto de trabajo inicial es muy fuerte si la asignación es demasiado limitada.

La gráfica de hits complementa la interpretación, con 4 marcos, los aciertos son considerablemente bajos (alrededor de 1.400.000), pero con solo duplicar la cantidad de marcos a 8, se alcanza casi el máximo número de aciertos posibles. A partir de los 8 marcos, los valores se mantienen constantes, lo que indica que el sistema logra mantener en memoria las páginas activas sin necesidad de intercambios constantes.

Por otro lado, la gráfica de tiempo de ejecución refleja cómo estos fallos afectan directamente el rendimiento: con solo 4 marcos y tamaño de página 128, el tiempo supera los 0.9 segundos, siendo el más alto de todos los escenarios evaluados. Este tiempo disminuye de forma sostenida al incrementar los marcos, con una mejora significativa entre 4 y 8. A partir de 16 marcos, los tiempos se estabilizan, especialmente para tamaños de página mayores.

En conclusión, estos resultados demuestran que la imagen *gato.bmp* requiere una asignación mínima de 8 marcos para operar eficientemente, y que tamaños de página mayores permiten un desempeño más estable y rápido.

Gráficas K2:

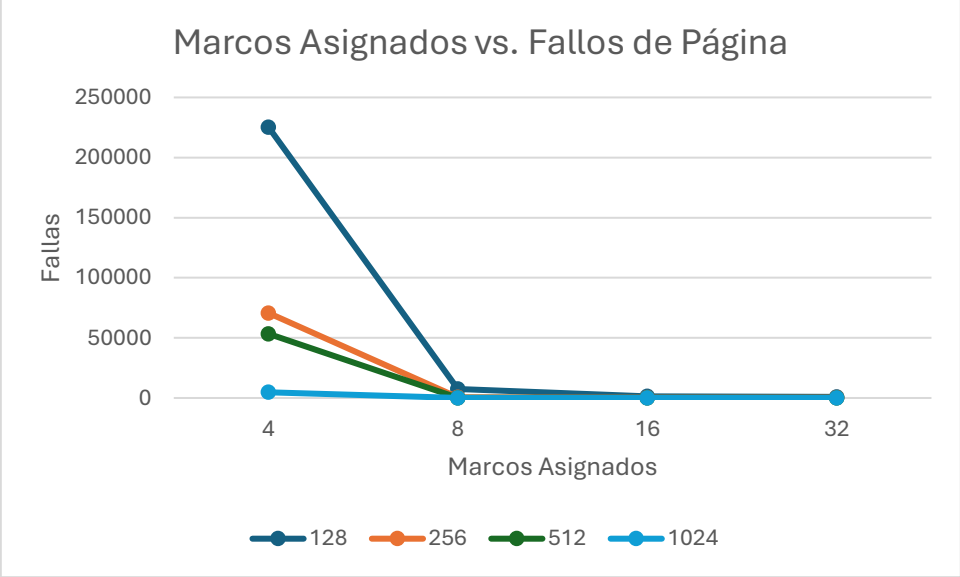


Ilustración 10: Marcos Asignados vs. Fallos de Página

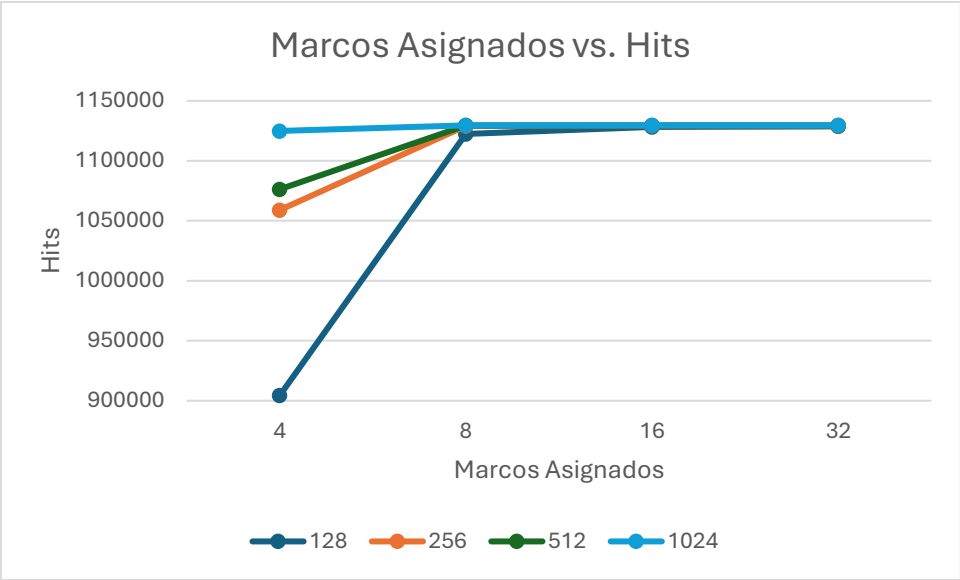


Ilustración 11: Marcos Asignados vs. Hits

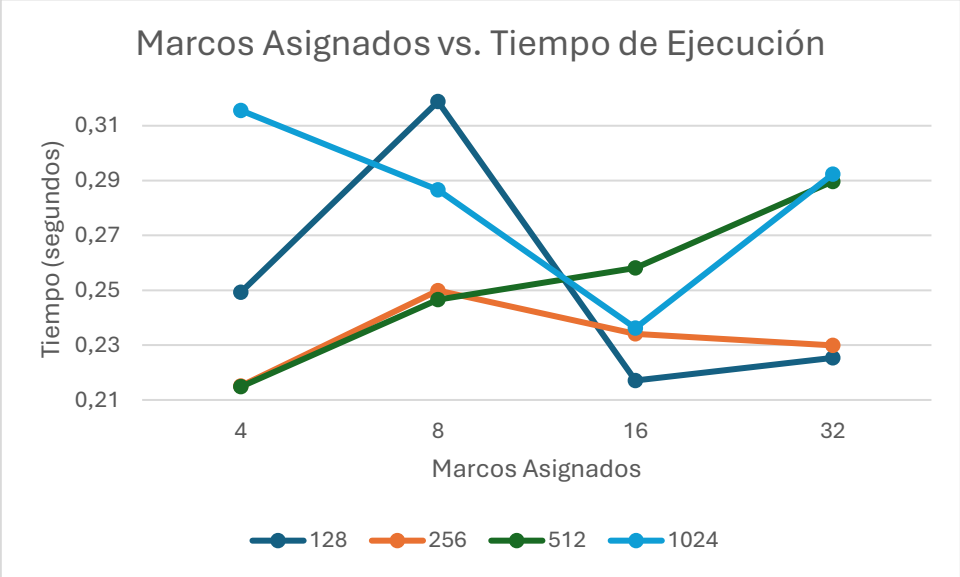


Ilustración 12: Marcos Asignados vs. Tiempo de Ejecución

Los resultados obtenidos para la imagen *k2.bmp* evidencian un escenario en el que la presión sobre la memoria es considerablemente más alta, en especial al trabajar con un número reducido de marcos asignados. La gráfica de fallos de página muestra que, al asignar solo 4 marcos, el número de fallos aumenta muchísimo, especialmente con tamaños de página pequeños, alcanzando más de 200.000 fallo. Esta cifra desciende drásticamente al pasar a 8 marcos, después, los valores se estabilizan, manteniéndose cercanos a cero para todos los tamaños de página.

Con 4 marcos los aciertos son significativamente bajos, con 8 marcos se alcanza una zona de estabilidad en la que los valores se mantienen por encima de los 1.120.000 hits, lo cual evidencia que a partir de esta configuración el conjunto de trabajo puede mantenerse en memoria principal sin necesidad de reemplazo continuo de páginas.

Sin embargo, la gráfica de tiempo de ejecución presenta un comportamiento más irregular que en los casos anteriores. Aunque se observa una reducción de tiempo general a medida que se incrementan los marcos, hay oscilaciones que podrían deberse a factores como la sobrecarga de gestión de páginas grandes o interferencias del sistema operativo. Por ejemplo, el tiempo para tamaño de página 128 alcanza un pico con 8 marcos antes de disminuir, lo que indica que no siempre más marcos implican un tiempo menor, y que el rendimiento puede depender también del patrón de acceso de la imagen y la alineación de páginas.

En conclusión, el conjunto de experimentos realizados permite identificar patrones consistentes sobre el comportamiento del sistema ante diferentes configuraciones de memoria virtual durante el procesamiento de imágenes. A través del análisis de las imágenes *Parrots_sal*, *parrot*, *gato* y *k2*, se observa una correlación directa entre el número de marcos asignados, el tamaño de página y el rendimiento general del sistema medido en términos de fallos de página, aciertos (hits) y tiempo de ejecución.

¿Aplicar el filtro Sobel representa un problema de localidad alta, media o baja?

Aplicar el filtro de Sobel representa un problema de localidad alta, espacial y temporal. La localidad espacial se evidencia porque para calcular el valor de un solo píxel en la imagen resultante, se accede a una vecindad de 3x3 píxeles en la imagen original, lo que implica accesos consecutivos o cercanos en memoria. Además, como la imagen se recorre de forma fila-columna, los accesos a memoria tienden a estar agrupados, favoreciendo el uso eficiente del caché. También existe localidad temporal, ya que los píxeles vecinos utilizados en una convolución suelen reutilizarse inmediatamente en las siguientes operaciones. Por tanto, el patrón de acceso en el filtro de Sobel maximiza la reutilización de datos en memoria cercana, lo que caracteriza un acceso con alta localidad.

Referencias

Sistemas Operativos Modernos. Andrew Tanenbaum. Editorial Pearson. Edición 3, año 2009