Contrato de equipo

Grupo: 1

Miembros del grupo:

Alberto Pertuz – a.pertuz@uniandes.edu.co – 202025856

Alejandro Pardo – a.pardos2@uniandes.edu.co – 202223709

Andrés Bolívar – a.bolivarc@uniandes.edu.co – 202214834

Procedimientos del equipo

1. **Reuniones periódicas**: Se realizó una única reunión el Domingo 30 de marzo de 2025, en la que se revisó el enunciado, se resolvieron dudas generales y se acordó la forma de trabajo.
2. **Método de comunicación**: Utilizamos un grupo de WhatsApp como canal principal para coordinar tareas, hacer preguntas, compartir archivos y decisiones importantes.
3. **Política de toma de decisiones**: Las decisiones se tomaron por consenso entre los tres integrantes en el grupo de WhatsApp.
4. **Agenda de reuniones**: Dado que solo hubo una reunión, no se llevó una agenda formal. En el grupo de WhatsApp se discutieron los temas en el momento en que fueron necesarios.
5. **Mantenimiento de registros**: No se llevaron actas formales. Los mensajes del grupo de WhatsApp sirvieron como registro de lo acordado.
6. **Cuidado social**: El enfoque fue exclusivamente académico, aunque se mantuvo un ambiente amable y de colaboración.

Expectativas del equipo

Calidad de trabajo

1. **Estándares**: El objetivo fue entregar un código funcional y documentado, con un informe claro que respondiera a lo solicitado en el enunciado.
2. **Estrategias**: Se dividió el trabajo según disponibilidad. Cada integrante era responsable de su parte, y todo se compartió y revisó por WhatsApp.

Participación del equipo

1. **Distribución de tareas**: La asignación de tareas fue equitativa y se basó en lo que cada uno podía aportar.
2. **Inclusión de ideas**: Se fomentó que todos opinaran. Cualquier sugerencia se discutía y se tomaban decisiones de grupo.
3. **Mantenerse enfocados**: Se trabajó con fechas internas para tener avances parciales antes de la entrega.
4. **Liderazgo**: No hubo un líder formal. Pero definitivamente Andrés fue el líder interino en el proyecto que más aportó para completar el proyecto

Responsabilidad personal

1. **Asistencia y participación**: Los tres estuvimos presentes en la reunión del viernes y activos en WhatsApp.
2. **Cumplimiento de tareas y plazos**: Todos entregaron su parte a tiempo y se coordinaron para consolidar el trabajo.
3. **Comunicación**: La comunicación fue fluida y constante por WhatsApp.
4. **Compromiso**: Cada integrante se comprometió con el trabajo y cumplió su parte.

Consecuencias por no cumplir

1. **Faltas ocasionales**: Se cubrían entre los demás compañeros, siempre que hubiera aviso.
2. **Faltas repetidas**: Si alguien no cumplía sin avisar ni participar, se consideraba no incluirlo en la entrega.

Reconocimiento y celebración

1. **Reconocimiento**: Se agradecieron los aportes dentro del grupo, especialmente cuando alguien avanzaba más rápido o solucionaba problemas.
2. **Celebración**: No se planificó celebración, pero se valoró el hecho de haber terminado bien el trabajo.

Firmas

Yo, como miembro del equipo, declaro que participé en la formulación de este acuerdo, entiendo sus términos y me comprometo a cumplirlos:

- Alberto Mario Pertuz Noriega – Fecha: [_02/04/2025_]
- Alejandro Pardo – Fecha: [_02/04/2025_]
- Andrés Bolívar – Fecha: [_02/04/2025_]

Informe Caso 2

Alberto Pertuz – a.pertuz@uniandes.edu.co – 202025856

Alejandro Pardo – a.pardos2@uniandes.edu.co – 202223709

Andrés Bolívar – a.bolivarc@uniandes.edu.co – 202214834

1. Diseño de la solución:

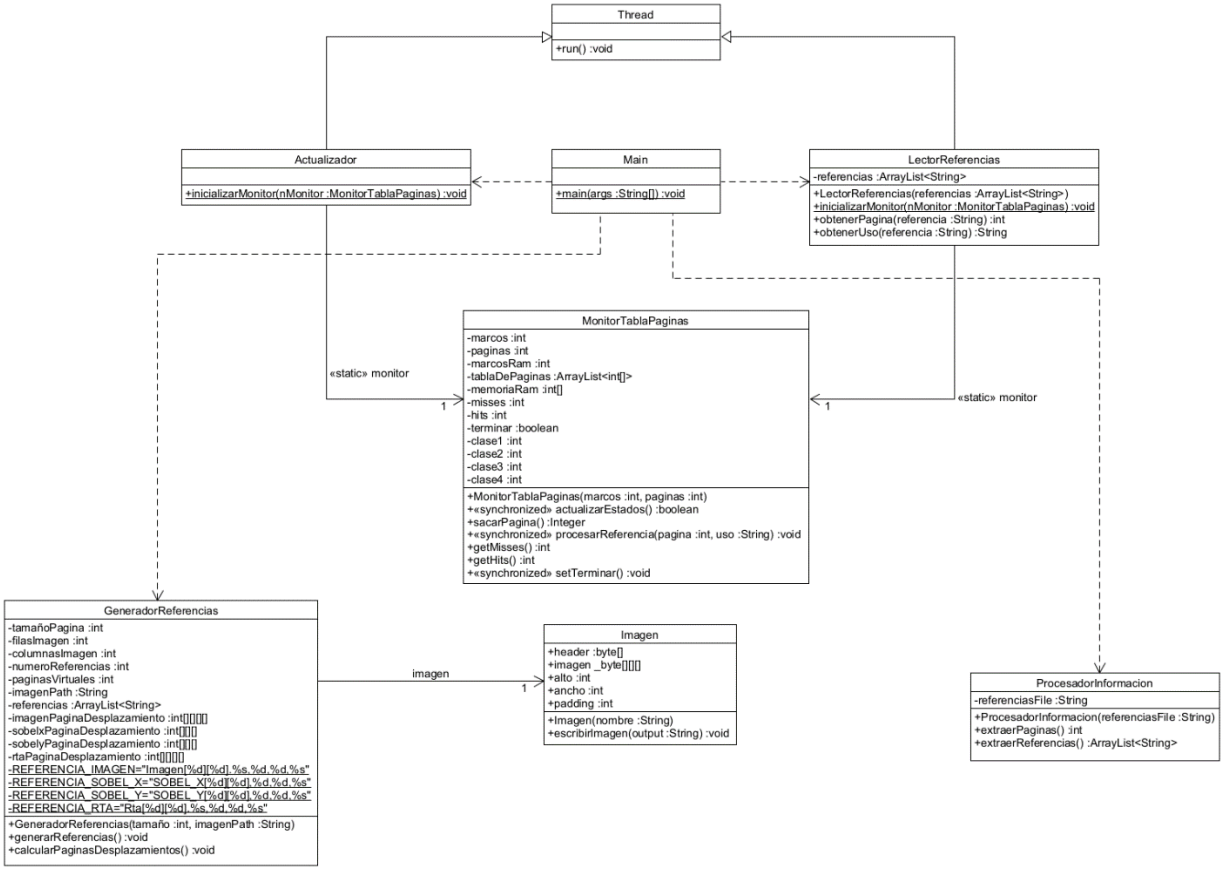


Figura 1: Diagrama de clases de la solución

Clases construidas:

- Main: Clase encargada de controlar la consola desplegada para el usuario y ejecutar las opciones pertinentes de acuerdo con la información brindada por el usuario. Asimismo, crea las instancias iniciales de todas las clases necesarias para cada solución.

- Actualizador: Clase que extiende la clase Thread y cuyas instancias corren de manera concurrente con las instancias de la clase LectorReferencias. La clase está encargada de actualizar la información de las páginas cargadas en RAM de manera sincronizada. En concreto, cada milisegundo actualiza el bit de lectura R de todas las páginas en RAM a 0.

- LectorReferencias: Clase que extiende la clase Threa y cuyas instancias corren en concurrencia con las instancias de la clase Actualizador. La clase de encarga de procesar secuencialmente el listado de referencias del archivo de referencias. En su procesamiento de referencias, el LectorReferencias simula el procedimiento de consultar si una página está en la RAM y, en caso de que no, de manera sincronizada carga la página necesaria en la RAM con el método pertinente dependiendo de si hay o no espacio en la RAM.

- MonitorTablaPaginas: Clase que funciona como monitor para la sincronización en exclusión mutua de los hilos de las clases Actualizador y LectorReferencias. En concreto, la clase contiene las estructuras de datos críticas que deben ser manejadas por ambas clases que corren concurrentemente para simular el manejo de la tabla de páginas y la memoria RAM. Asimismo, la clase contiene los métodos necesarios que se deben ejecutar de manera sincronizada para consultar y alterar la información de las estructuras de datos que representan la tabla de páginas y la memoria RAM.

- GeneradorReferencias: Clase que se encarga de ejecutar los algoritmos necesarios para la opción 1 de la solución solicitada, es decir, generar el archivo de referencias.

- ProcesadorInformacion: Clase auxiliar que sirve de apoyo para separar la información del archivo de referencias y, de este modo, brindarle la información segmentada del archivo de referencias al LectorReferencias y al MonitorTablaPaginas para facilitarles sus funcionalidades.

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

La generación de referencias de página se hace simulando cómo el método applySobel() accede a memoria durante la ejecución, esto con el fin de que dichas referencias de memoria virtual sean posteriormente procesadas. Solo se consideran las matrices imagen, filtroX, filtroY y respuesta, ignorando cualquier otro uso de memoria (heap, stack, código), utilizando estas matrices, se divide la imagen en páginas de memoria virtual y se calcula el desplazamiento dentro de cada página. La organización de la memoria es en row-major, y cada acceso a un dato genera una referencia que incluye la página virtual, el desplazamiento dentro de la página, y si se trata de lectura o escritura. Todo esto queda registrado en un archivo para ser usado en la simulación posterior.

En detalle, el algoritmo de generación de referencias toma como base la estructura de ciclos anidados (los for) que usa el algoritmo de Sobel (el método applySobel()). A partir de dicha estructura de los ciclos, se generan las referencias de acuerdo con el uso de las 4 matrices en el algoritmo de Sobel, donde si solo se accede a la información de la matriz, se genera una referencia de lectura “R” y, por su parte, si se modifica la información de la matriz, se genera una referencia de escritura “W”. Cabe agregar que, previo a la generación de referencias, se calcula la página y el desplazamiento de cada referencia generada haciendo un conteo secuencial de las referencias y, además, en dicho conteo se respeta el orden supuesto de las matrices (primero imagen, luego sobelx, luego sobely y luego respuesta). Adicionalmente, se generan 3 referencias de la imagen, de sobelx y de sobely dentro de las instrucciones de los 4 for anidados del algoritmo de Sobel. Por su parte, se generan 3 referencias a la matriz respuesta en las instrucciones de los primeros 2 for anidados del algoritmo de Sobel. En conclusión, la generación de referencias se basa en la estructura del algoritmo de Sobel (método applySobel()) y en la forma y orden en que este algoritmo accede y modifica la información de las 4 matrices usadas.

3. Estructuras de datos usadas en la simulación del sistema de paginación.

Para modelar el sistema de paginación se usaron varias estructuras clave:

- **Tabla de páginas**: guarda el estado de cada página virtual, incluyendo si está en RAM, los bits R (referencia) y M (modificación), y el tiempo desde su último uso.
- **Lista de marcos en RAM**: representa los espacios disponibles para cargar páginas, donde se hace seguimiento del contenido y orden de uso.

Estas estructuras se van actualizando durante la simulación. Cada vez que se accede a una página, se marca el bit R, y si se escribe, se marca solo el bit M. Cada cierto tiempo (simulado con milisegundos), se hace una limpieza de estos bits para determinar qué páginas se consideran “no usadas recientemente”, tal como lo sugiere Tanenbaum.

4. Sincronización entre hilos.

Como el programa corre con dos hilos de manera concurrente, es necesario sincronizar algunos accesos a estructuras compartidas. El primer hilo procesa las referencias, cargando páginas a RAM si es necesario y actualizando los bits R y M. El segundo hilo, que corre cada milisegundo, se encarga de actualizar el estado de las páginas para mantener el criterio del algoritmo NRU.

Se identificaron métodos críticos donde puede haber condiciones de carrera, especialmente al acceder o modificar la tabla de páginas o la lista de marcos. Para evitar errores, se usaron métodos sincronizados en Java para proteger esas secciones del código. En concreto, se hizo uso de la exclusión mutua como tipo de sincronización para estos dos Threads, con el fin de que cada uno

de ellos accediera y modificara la información de las 2 estructuras claves críticas que se han mencionado uno a la vez.

5. Tabla con datos recopilados.

Página de 512B, caso2-parrotspeq.bmp				
Marcos	Total	Hits	Misses	%Hits
4	756756	749306	7450	99.02%
6	756756	756646	110	99.99%

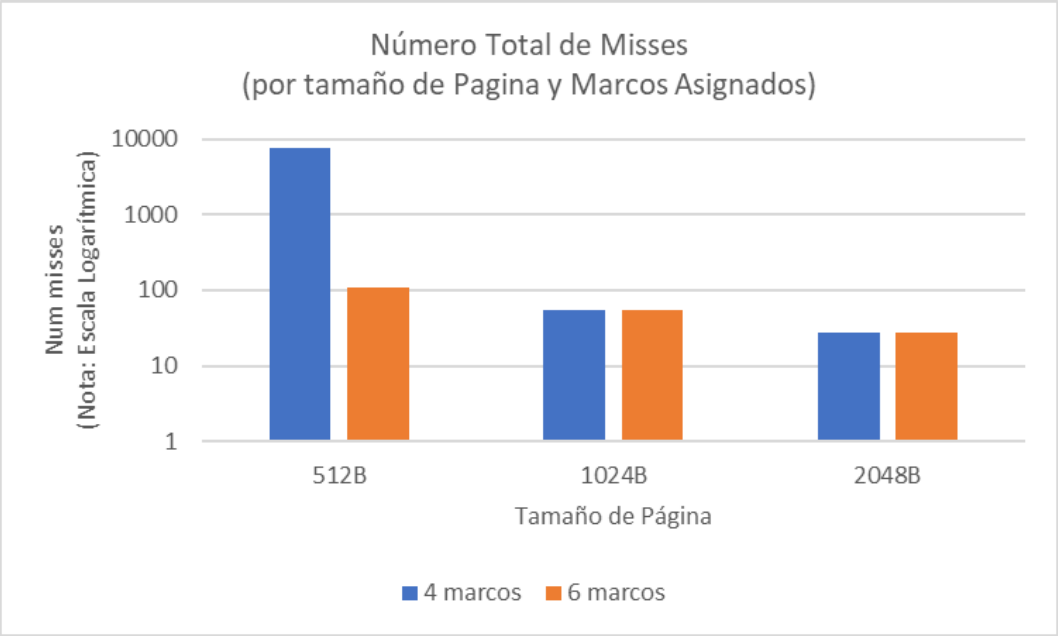
Página de 1024B, caso2-parrotspeq.bmp				
Marcos	Total	Hits	Misses	%Hits
4	756756	756701	55	99.99%
6	756756	756701	55	99.99%

Página de 2048B, caso2-parrotspeq.bmp				
Marcos	Total	Hits	Misses	%Hits
4	756756	756728	28	99.996%
6	756756	756728	28	99.996%

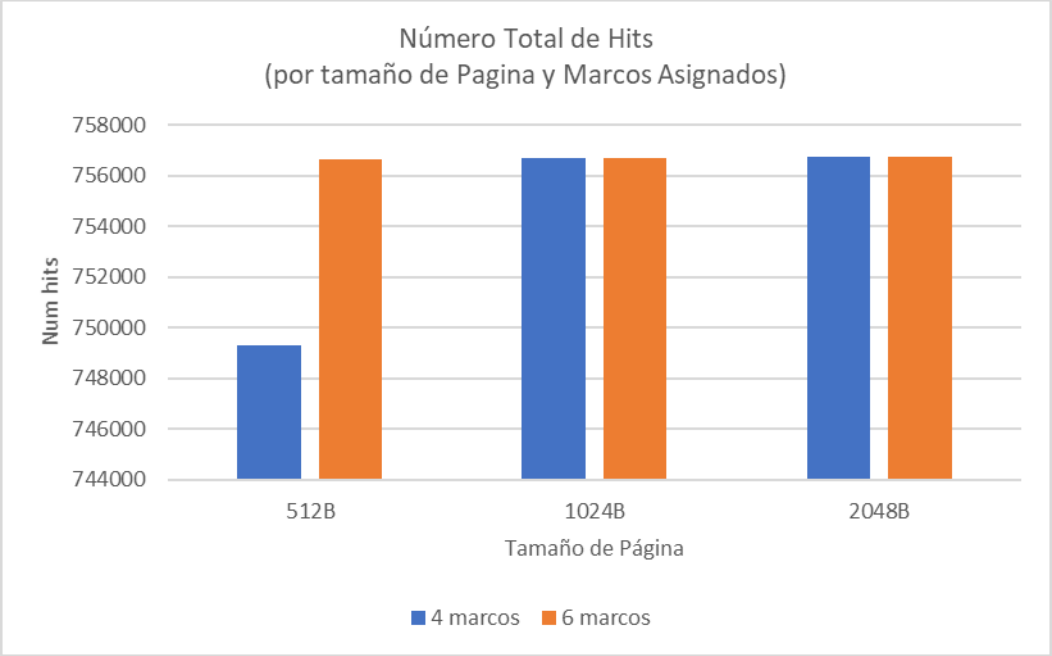
6. Gráficas de comportamiento

Se deben generar varias gráficas para visualizar el comportamiento del sistema. Las principales son:

- Marcos asignados vs. número de misses



- Marcos asignados vs. número de hits



Estas gráficas ayudan a ver cómo se mejora el desempeño al aumentar los recursos asignados y qué tan sensible es el algoritmo al tamaño de página.

7. Exploración de otras configuraciones

Además de los escenarios básicos, probamos con imágenes más grandes y pequeñas, tamaños de página de 256, 2048 y 4096 bytes, y también variamos el número de marcos entre 2 y 12. Esto

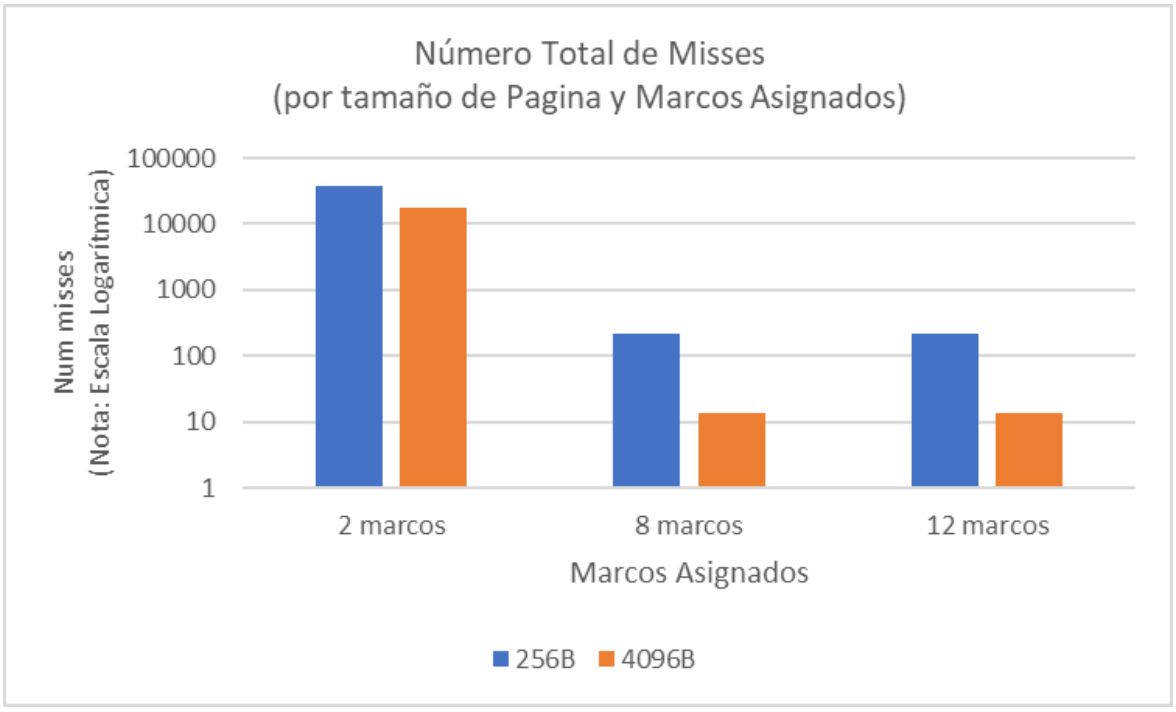
permitió ver en qué punto ya no mejora el rendimiento al asignar más marcos, lo cual es importante para entender el uso eficiente de la memoria virtual.

A continuación, se presentan los resultados variando los tamaños de página con 256 y 4096 Bytes y los marcos utilizados sobre la imagen llamada caso2-parrotspeq.bmp

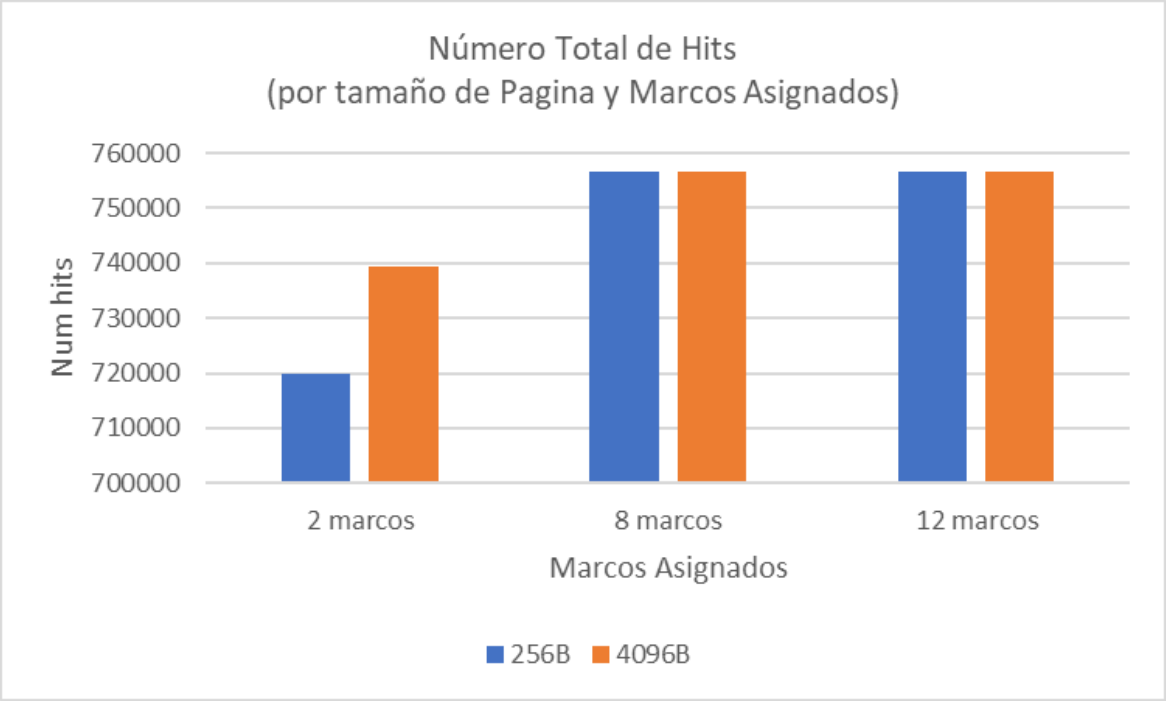
Página de 256B, caso2-parrotspeq.bmp					
Marcos	Total	Hits	Misses	%Hits	Tiempo total (ms)
2	756756	719803	36953	95.12%	369565.99
8	756756	756535	221	99.97%	2247.83
12	756756	756536	220	99.97%	2237.83

Página de 4096B, caso2-parrotspeq.bmp					
Marcos	Total	Hits	Misses	%Hits	Tiempo total (ms)
2	756756	739467	17289	97.715%	172926.97
8	756756	756742	14	99.998%	177.84
12	756756	756742	14	99.998%	177.84

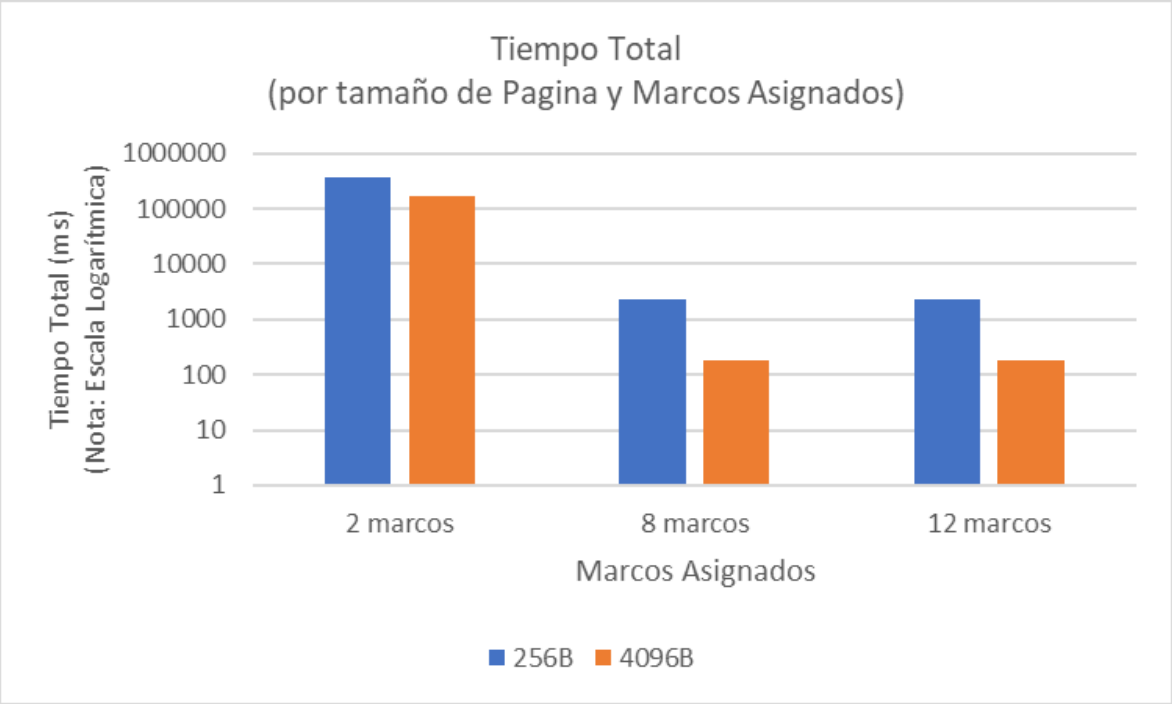
- Marcos asignados vs. número de misses



- Marcos asignados vs. número de hits



- Tiempo total vs. cantidad de marcos



Posteriormente se probó con imágenes más grandes (caso2-prueba4.bmp), se presentan los resultados a continuación:

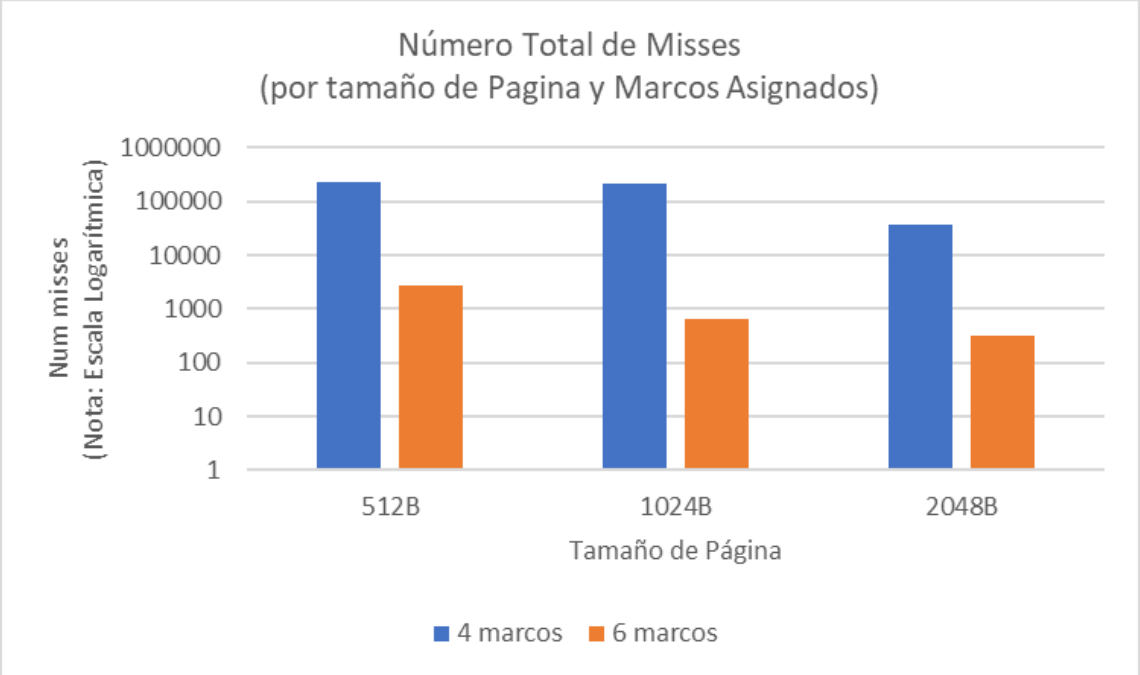
Página de 512B, caso2-prueba4.bmp					
Marcos	Total	Hits	Misses	%Hits	Tiempo total (ms)

4	8826048	8597676	228372	97.41%	2284149.88
6	8826048	8823389	2659	99.97%	27031.17

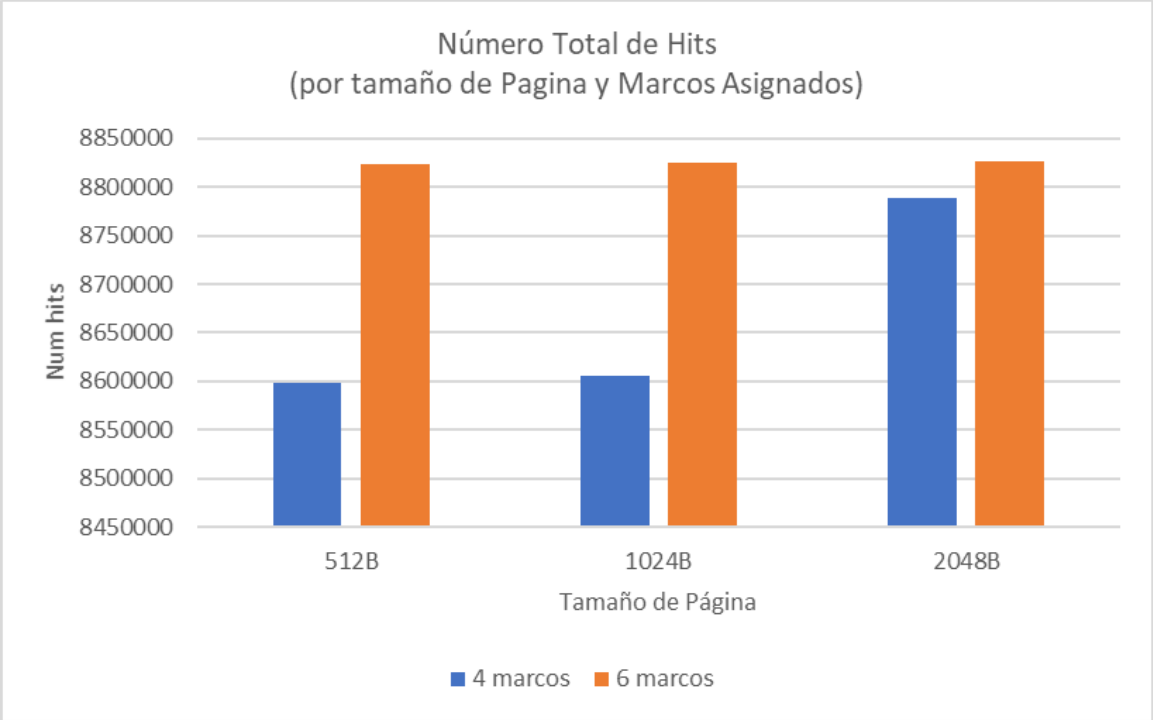
Página de 1024B, caso2-prueba4.bmp					
Marcos	Total	Hits	Misses	%Hits	Tiempo total (ms)
4	8826048	8606032	220016	97.51%	2200590.3
6	8826048	8825418	630	99.99%	6741.27

Página de 2048B, caso2-prueba4.bmp					
Marcos	Total	Hits	Misses	%Hits	Tiempo total (ms)
4	8826048	8788937	37111	99.580%	371549.45
6	8826048	8825732	316	99.996%	3601.29

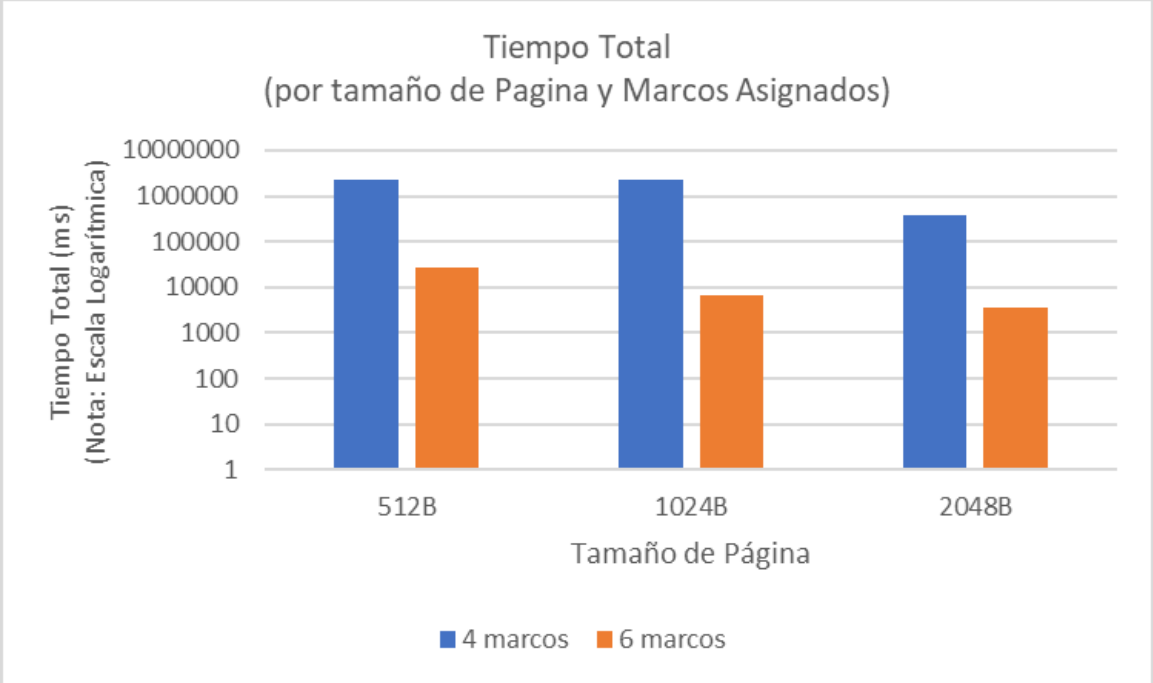
- Marcos asignados vs. número de misses



- Marcos asignados vs. número de hits



- Tiempo total vs. cantidad de marcos



8. Gráficas de tiempo

Se calcularon los tiempos aproximados considerando 50 nanosegundos por hit y 10 milisegundos por miss:

Página de 512B, caso2-parrotspeq.bmp					
Marcos	Total	Hits	Misses	%Hits	Tiempo total (ms)
4	756756	749306	7450	99.02%	74537.47
6	756756	756646	110	99.99%	1137.83

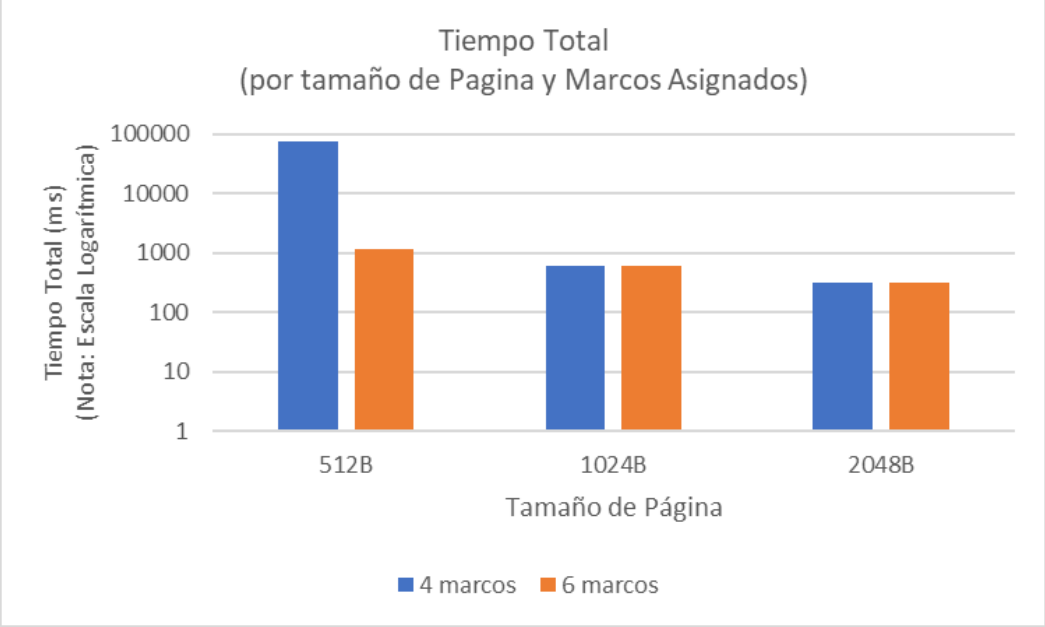
Página de 1024B, caso2-parrotspeq.bmp					
Marcos	Total	Hits	Misses	%Hits	Tiempo total (ms)
4	756756	756701	55	99.99%	587.84
6	756756	756701	55	99.99%	587.84

Página de 2048B, caso2-parrotspeq.bmp					
Marcos	Total	Hits	Misses	%Hits	Tiempo total (ms)
4	756756	756728	28	99.996%	317.84
6	756756	756728	28	99.996%	317.84

Nota: Se asume 50ns = 0.00000005 ms, y 10ms por cada fallo de página.

A partir de los datos recopilados, se graficaron los resultados por cada tamaño de página de acuerdo con los marcos utilizados

- Tiempo total vs. cantidad de marcos



9. Interpretación de resultados

Los resultados tienen mucho sentido con lo que se esperaba: cuando se asignan más marcos, hay menos fallas de página, y el sistema funciona más eficientemente. Sin embargo, no siempre más marcos implican mejoras notables. Llega un punto donde agregar más marcos ya no hace mucha diferencia. También se ve que, si el tamaño de página es adecuado para los accesos del algoritmo de Sobel, se logran buenos resultados incluso con menos marcos. En otras palabras, con un mayor tamaño de páginas se consiguen mejores resultados donde se aumentan los hits y se disminuyen los misses generados por el procesamiento de las referencias. Este hecho se basa en que, con un mayor tamaño de páginas, se consigue tener menos páginas virtuales para administrar y, además, cada vez que se carga una página en la memoria RAM, se carga una mayor cantidad de información que va a ser usada en el futuro con alta probabilidad.

10. Localidad del filtro de Sobel

El filtro de Sobel representa un problema de **localidad alta**. El hecho de que el filtro de Sobel represente un problema de localidad alta se puede justificar a partir de la relación del número de hits conseguidos en los escenarios y el algoritmo de reemplazo usado. De este modo, el algoritmo de reemplazo de “Páginas no usadas recientemente” está enfocado en aprovecharse del principio de localidad, donde se le da prioridad a mantener en la memoria RAM aquellas páginas usadas recientemente porque tienen mayor probabilidad de volver a ser usadas en el futuro. Así, con base en los resultados de los distintos escenarios donde la cantidad de hits es siempre superior al menos al 95% de las referencias, se puede concluir que el algoritmo de reemplazo usado es efectivo para el problema, es decir, el algoritmo sí está aprovechando una localidad alta del problema, lo cual se refleja en el alto porcentaje de hits conseguido en cada escenario.