Alejandro Lancheros - 202122797

Mateo López - 202220119

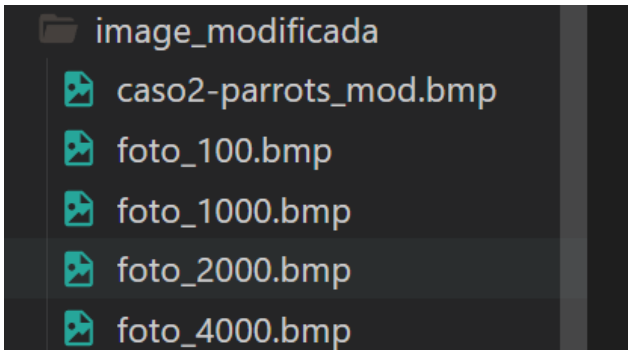
Sofia Losada - 202221008

Infraestructura Computacional

Caso 2

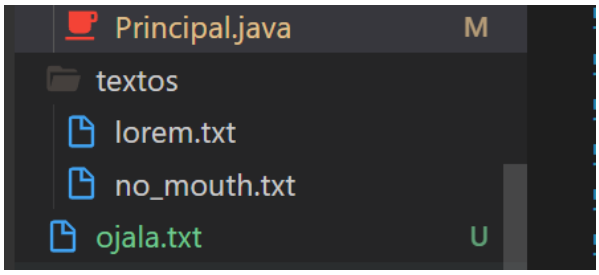
Reglas para ejecutar el programa:

- Para ejecutar la opción 1, se debe ingresar el nombre de la imagen, la cual **debe** existir en la carpeta /image_modificada/. Se debe escribir el nombre con el que se desea guardar el archivo de referencias, el cual se guardará en la carpeta base del proyecto. En este caso se usó foto_4000.bmp, la cual ya existe en la carpeta:




```
Ingrese el tamaño de cada página (en bytes): 9
Ingrese la ruta de la imagen: foto_4000.bmp
Ingrese el nombre del archivo de referencias (incluya el .txt): ojala.txt
```

Y se genera el archivo de referencias en la carpeta del proyecto:



- Para ejecutar el caso 2, se debe escribir el nombre del archivo de referencias, el cual debe existir en la carpeta principal del proyecto (misma ubicación donde se guardan al crearse por la opción 1. Este es un ejemplo de cómo se vería al ejecutarse la opción:

```
2
Ingrese el número de marcos de página: 4
Ingrese la ruta del archivo de referencias: ojala.txt
Número de hits: 62620
Número de fallas: 5396
Porcentaje de éxito (hits): 92.06657257115972%
Tiempo total: 53961.5655 ms
```

Descripción del algoritmo usado para generar las referencias de página

Se implementa un algoritmo que genera referencias de páginas para plasmar una simulación del acceso a los datos de una imagen y un mensaje cifrado dentro de esta. Para esto, la función ejecutar_Opcion en la clase Op1 genera un archivo de texto con las referencias de las páginas de la imagen y del mensaje cifrado que se utiliza para hacer el proceso de lectura y escritura de bits.

Primero los datos iniciales de la imagen se cargan utilizando la clase Imagen, cuyos atributos son alto y ancho, para obtener las dimensiones de esta. De esta manera, se utilizan para calcular la cantidad de pixeles (filas y columnas). Se multiplica por tres, pues de esta forma se representan los tres canales de color RGB. Del mismo modo, se calcula la longitud del mensaje y el valor se procesa en bits.

Después se calcula el número total de referencias, para esto se toman 16 referencias que son los accesos a los primeros 16 bytes de la imagen para obtener la longitud del mensaje secreto. Posterior a esto, se calcula que por cada carácter van a haber 17 referencias, ya que primero hay una referencia de escritura para inicializar el carácter en la página y desplazamiento asignados, y luego, por cada bit se tiene que hacer una referencia de lectura de la imagen y una de escritura. Esto nos lleva a 2*8 +1 = 17 referencias por carácter, entonces nuestro total de referencias llega a ser (17*longitud del mensaje) + 16.

Después de esto, se calcula el número de páginas que se necesitan para cargar toda la información de la imagen y el mensaje. Esto se hace sumando la cantidad de páginas necesarias para la imagen más la cantidad de páginas necesarias para cargar el mensaje. La forma de calcular la cantidad de páginas necesarias para la imagen es conocer el número de bytes que tiene la imagen y dividirla en la cantidad de bytes por página, lo cual se consigue obteniendo la cantidad de pixeles (filas*columnas) y a ese valor multiplicarle 3, ya que hay 3 bytes por píxel. A esto se le suma la cantidad de páginas para el mensaje, calculado a partir de dividir la longitud (en bytes) del mensaje y dividirlo sobre el número de bytes por página.

Descripción de las estructuras de datos usadas para simular el comportamiento del sistema de pagina

Se implementó el algoritmo Not Recently Used" (NRU) en un sistema de paginación de memoria. Este código permite leer un archivo de referencias de páginas, cargar las referencias, simular el comportamiento de la memoria virtual, registrar fallas de página y medir el tiempo total de acceso considerando el tiempo de acceso a la RAM y el swap.

Para la simulación del comportamiento del sistema de página se utilizaron diferentes estructuras. Para esto, lo primero fue declarar las siguientes variables:

- `Pagina`: objeto utilizado para simular la información guardada en los marcos de página. Incluye el número de la página, el bit R y el bit M.
- `numMarcos`: número de marcos de página (frames) disponibles en la memoria.
- marcos: ArrayList de objetos Pagina que no puede añadir más Paginas que numMarcos
- hits: contador de aciertos de páginas (cuando la página requerida ya está en la memoria).
- fallas: contador de fallas de página (cuando se necesita cargar una página en memoria).
- tiempoAcessRAM: tiempo de acceso a la RAM en milisegundos.
- tiempoAcessSwap: tiempo de acceso al espacio de intercambio (swap) en milisegundos.
- Correrт2: variable booleana que le indica al segundo thread que el primero ya terminó su ejecución

El comportamiento de paginación se realizó a través de un Thread, el cual utiliza un ArrayList llamado marcos, el cual simula la cantidad de marcos de página que se le asignó a este proceso. Como se mencionó anteriormente, el arraylist recibe objetos tipo Pagina, y solo puede añadir objetos hasta que la longitud alcance el número de marcos asignados.
Cada vez que se genera una nueva referencia, se busca si la página de esta referencia ya está cargada en los marcos de página, si se encuentra la página ya cargada en la estructura de marcos, se le suma 1 a la cantidad de hits, y se hace una modificación de los bits R y M correspondiente: Si era una referencia de lectura, únicamente se le asigna 1 al bit R, mientras que si era una referencia de escritura, se le asignan 1 a ambos el bit R como al bit M.
En caso de que no encuentre la página que la referencia necesita, pero aún hay marcos de página libres, se le suma 1 a la cantidad de fallas y se añade la página necesaria a los marcos, con los bits R:1 y M:0 en caso de que sea una referencia de lectura o R:1 y M:1 en caso de que sea uno de escritura.
El último caso posible es si no se encuentra la página necesitada y ya no hay marcos de página libres, todos están ocupados por alguna página. Para esto se realiza el algoritmo de NRU, el cual busca primero si hay alguna página en caso 0 (R:0, M:0). Si no se encuentra, busca si hay alguna página del caso 1 (R:0, M:1). Si no se encuentra, busca si hay alguna página del caso 2 (R:1, M:0). Finalmente, si no ha encontrado ninguna página que pertenezca a los casos anteriores, busca alguna del caso 3 (R:1,M:1). Esto se realiza hasta que se hayan cargado todas las referencias generadas por el archivo de la opción 1, mostrando la totalidad de hits, misses, tiempo que tarda en cargar las referencias con los valores teóricos de demora de acceso a memoria y a swap, y el porcentaje de aciertos

El segundo thread se inicializa al mismo tiempo que el primero, el cual trabaja realizando constantemente el apagado del bit R para simular que “el bit R se borra de forma periódica”. Su funcionamiento es acceder a cada página del ArrayList y asignarle un 0 al bit R, ejecutandose hasta que el thread 1 le avise que ya terminó de cargar las referencias, y el thread 2 ya puede detenerse. Para poder evitar problemas de sincronización entre los threads, ya que ambos acceden al mismo recurso del ArrayList, se implementó una exclusión mútua para el acceso a este ArrayList, permitiendo que únicamente uno de los threads pueda acceder en un dado momento a la lista, así que cuando ya libere este recurso, el otro thread puede modificarlo sin riesgos a que se presenten problemas de sincronización.

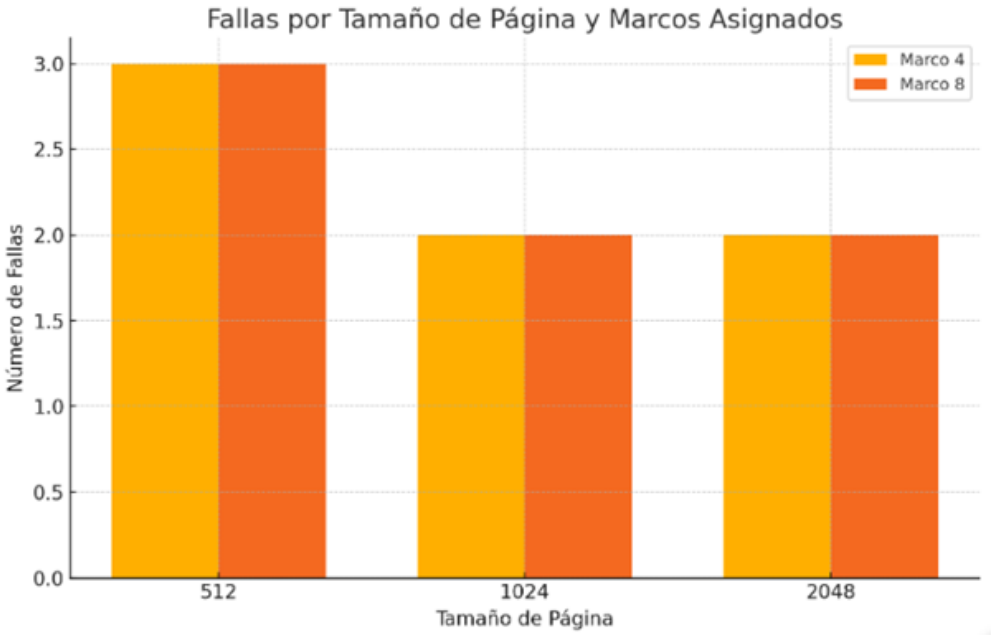
tabla con los datos recopilados, porcentaje de hits y misses por cada escenario simulado.
gráficas que ilustren el comportamiento del sistema:

- Tamaño de imagen vs Marcos asignados vs Porcentaje de hits

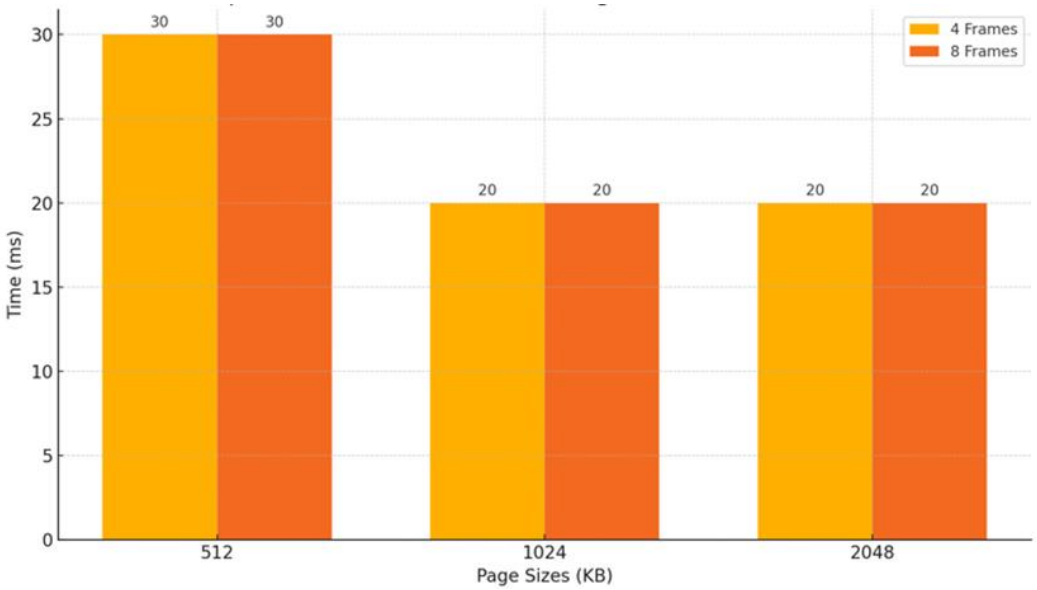
Graficas escenarios de prueba
Foto.bmp

- Archivo foto_100.bmp: 100 caracteres

Páginas de 512, foto_100.bmp: 100 Caracteres					
Marcos Asignados	Total Referencias	Hits	Fallas	%Hits	Tiempo(ms)
4	1716	1713	3	99,82	30
8	1716	1713	3	99,82	30
Páginas de 1024, foto_100.bmp: 100 Caracteres					
Marcos Asignados	Total Referencias	Hits	Fallas	%Hits	Tiempo(ms)
4	1716	1714	2	99,88	20
8	1716	1714	2	99,88	20
Páginas de 2048, foto_100.bmp: 100 Caracteres					
Marcos Asignados	Total Referencias	Hits	Fallas	%Hits	Tiempo(ms)
4	1716	1714	2	99,88	20
8	1716	1714	2	99,88	20

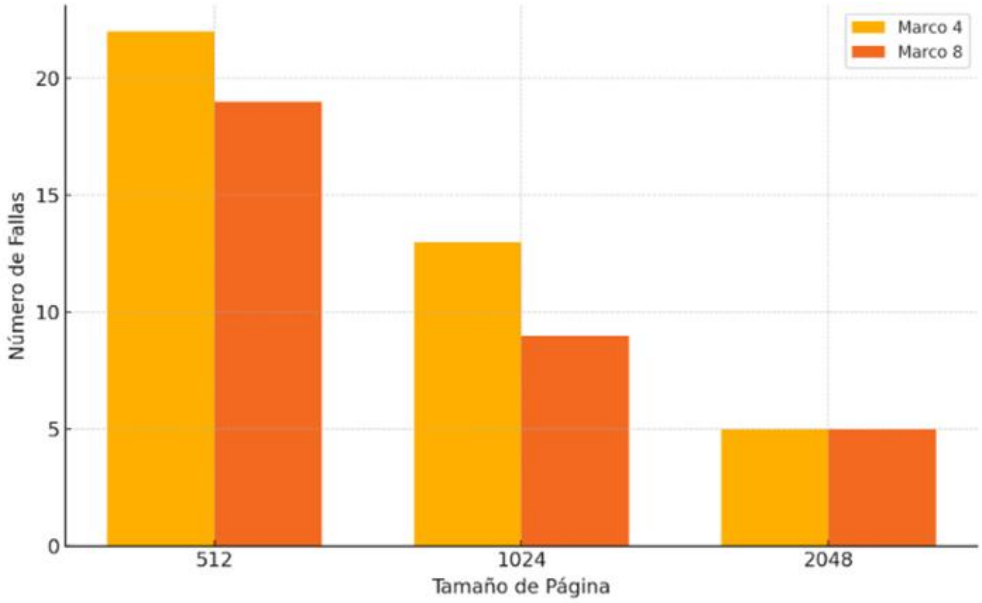


Tiempo:

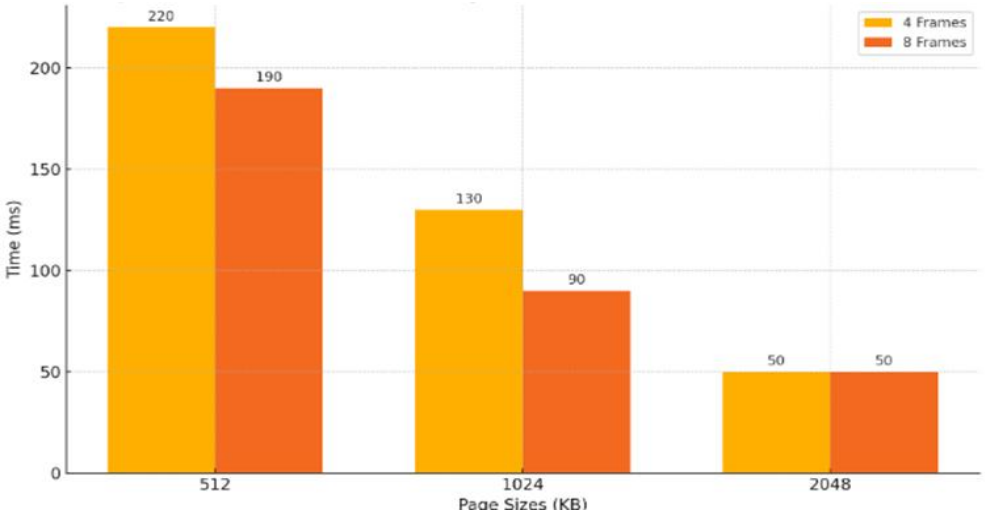


- Archivo foto_1000.bmp: 1000 caracteres

Páginas de 512, foto_1000.bmp: 1000 Caracteres					
Marcos Asignados	Total Referencias	Hits	Fallas	%Hits	Tiempo(ms)
4	17016	16994	22	99,87	220
8	17016	16997	19	99,88	190
Páginas de 1024, foto_1000.bmp: 1000 Caracteres					
Marcos Asignados	Total Referencias	Hits	Fallas	%Hits	Tiempo(ms)
4	17016	17003	13	99,92	130
8	17016	17007	9	99,95	90
Páginas de 2048, foto_1000.bmp: 1000 Caracteres					
Marcos Asignados	Total Referencias	Hits	Fallas	%Hits	Tiempo(ms)
4	17016	17011	5	99,97	50
8	17016	17011	5	99,97	50



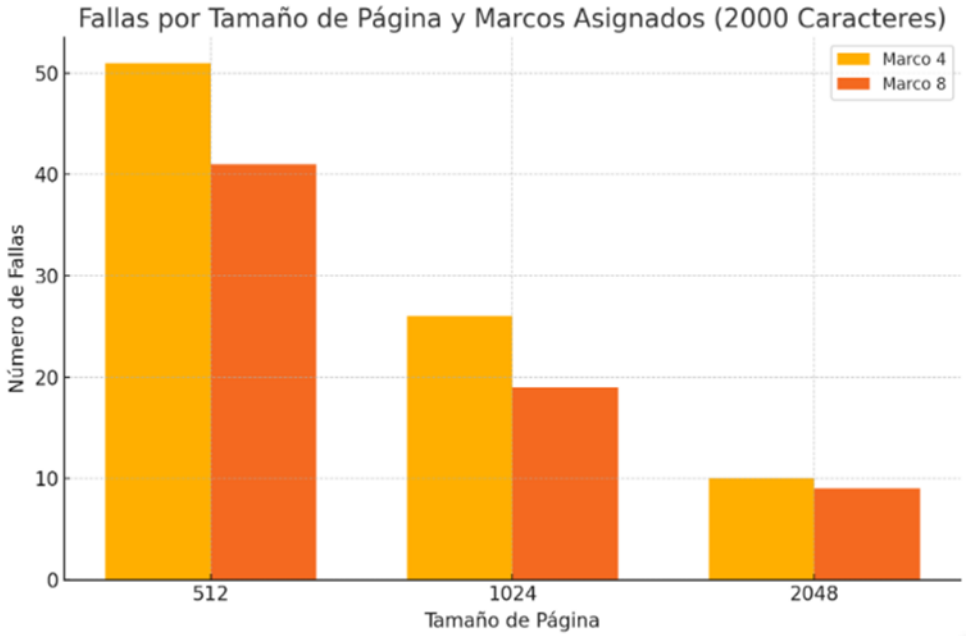
Tiempo:



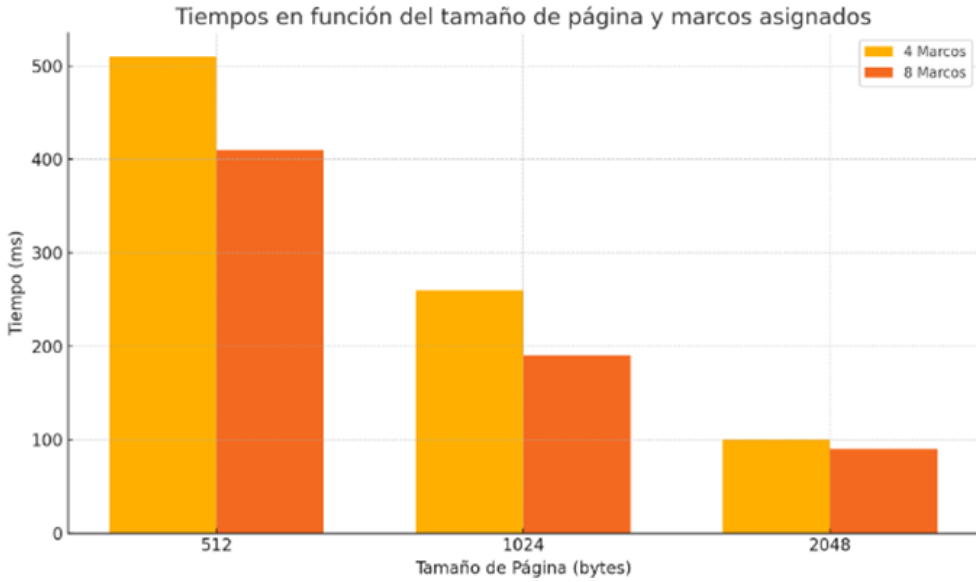
- Archivo foto_2000.bmp: 2000 caracteres

Páginas de 512, foto_2000.bmp: 2000 Caracteres					
Marcos Asignados	Total Referencias	Hits	Fallas	%Hits	Tiempo(ms)
4	34016	33965	51	99,85	510
8	34016	33975	41	99,88	410
Páginas de 1024, foto_2000.bmp: 2000 Caracteres					
Marcos Asignados	Total Referencias	Hits	Fallas	%Hits	Tiempo(ms)
4	34016	33990	26	99,92	260
8	34016	33997	19	99,94	190
Páginas de 2048, foto_2000.bmp: 2000 Caracteres					

Marcos Asignados	Total Referencias	Hits	Fallas	%Hits	Tiempo(ms)
4	34016	34006	10	99,97	100
8	34016	34007	9	90,97	90



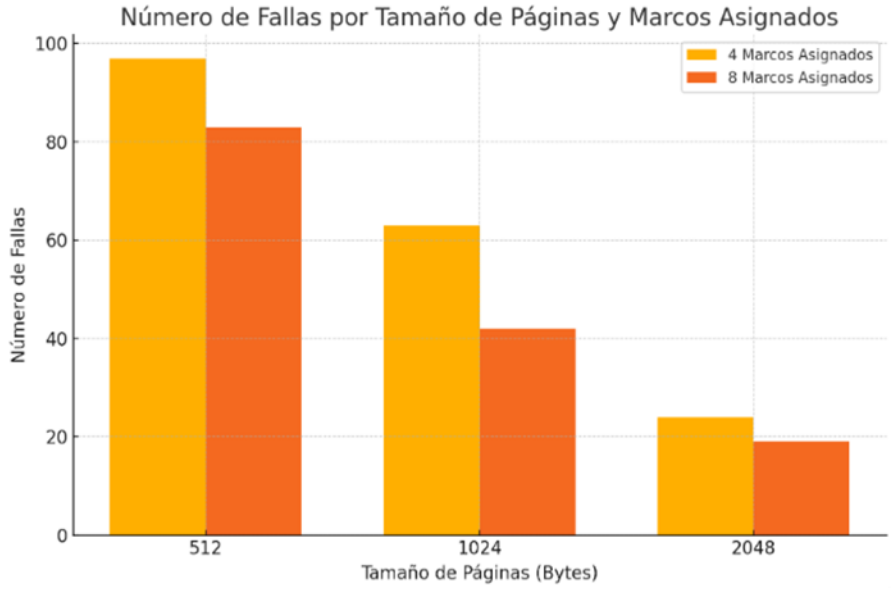
Tiempo:



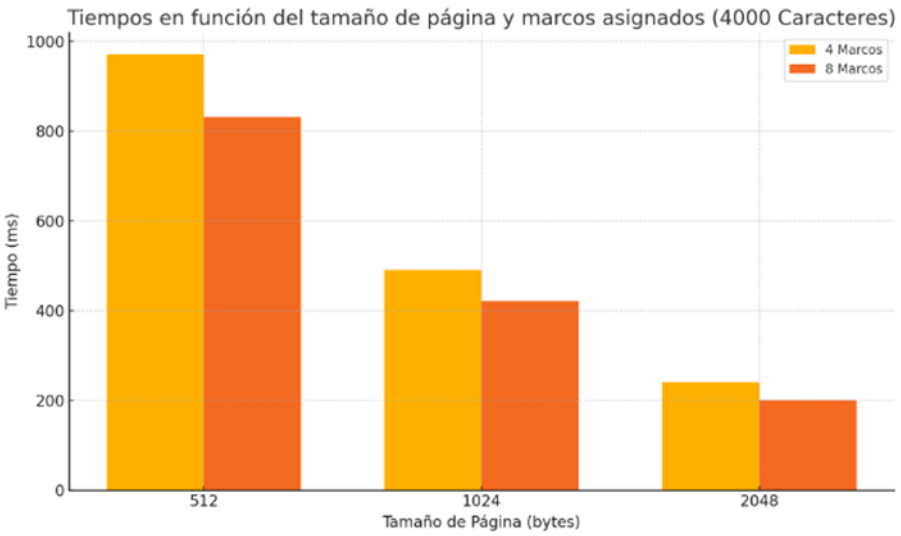
- Archivo foto_4000.bmp: 4000 caracteres

Páginas de 512, foto_4000.bmp: 4000 Caracteres					
Marcos Asignados	Total Referencias	Hits	Fallas	%Hits	Tiempo(ms)
4	68016	67919	97	99,85	971
8	68016	67933	83	99,88	831
Páginas de 1024, foto_4000.bmp: 4000 Caracteres					

Marcos Asignados	Total Referencias	Hits	Fallas	%Hits	Tiempo(ms)
4	68016	67967	49	99,92	491
8	68016	67974	42	99,94	421
Páginas de 2048, foto_4000.bmp: 4000 Caracteres					
Marcos Asignados	Total Referencias	Hits	Fallas	%Hits	Tiempo(ms)
4	68016	67992	24	99,96	241
8	68016	67996	20	99,97	201



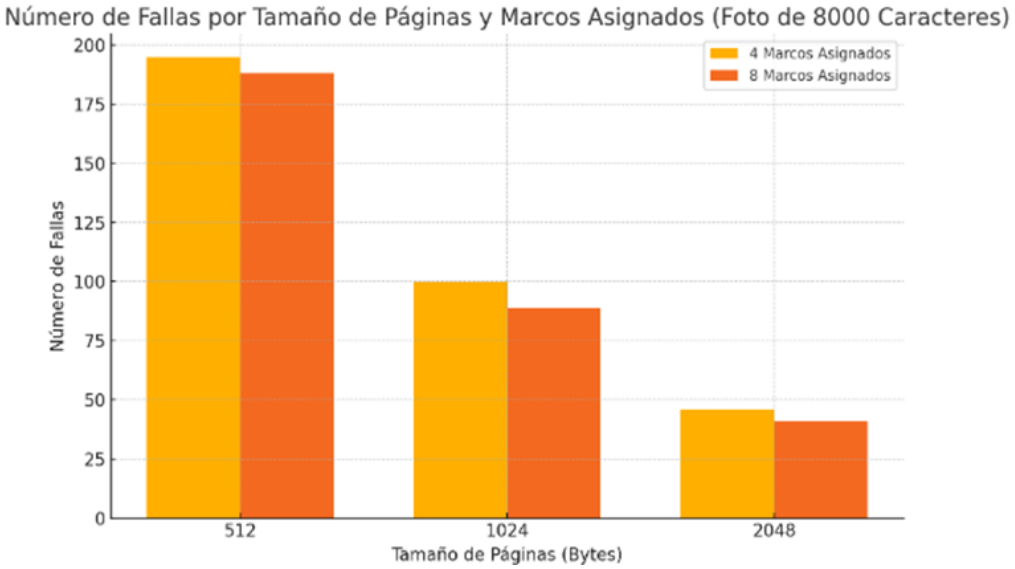
Tiempo:



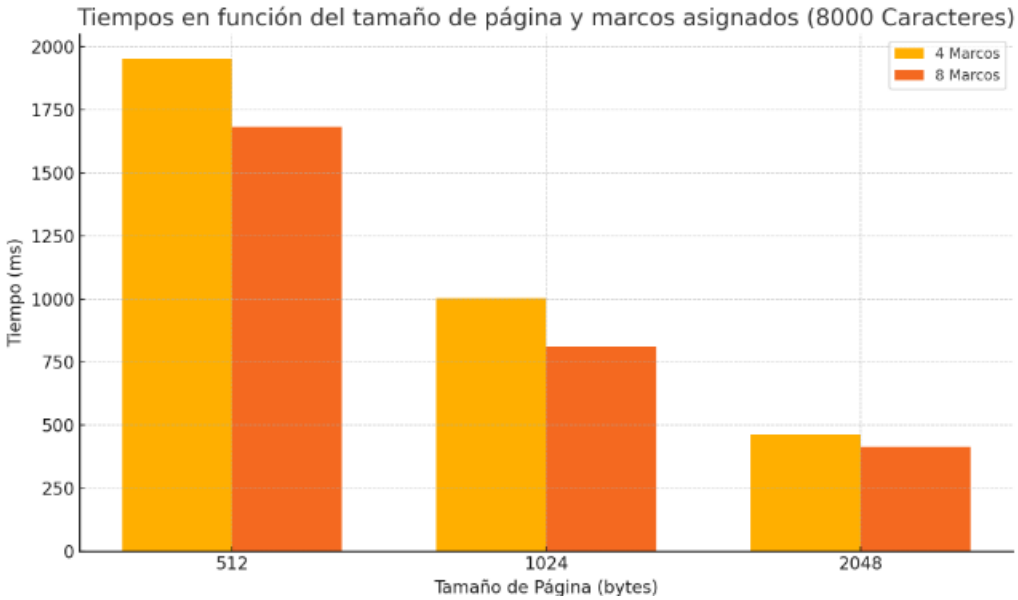
- Archivo foto_8000.bmp: 8000 caracteres

Páginas de 512, foto_8000.bmp: 8000 Caracteres					
Marcos Asignados	Total Referencias	Hits	Fallas	%Hits	Tiempo(ms)
4	136016	135821	195	99,86	1.953

8	136016	135848	168	99,88	1.683
Páginas de 1024, foto_8000.bmp: 8000 Caracteres					
Marcos Asignados	Total Referencias	Hits	Fallas	%Hits	Tiempo(ms)
4	136016	135916	100	99,93	1.003
8	136016	135935	81	99,94	813
Páginas de 2048, foto_8000.bmp: 8000 Caracteres					
Marcos Asignados	Total Referencias	Hits	Fallas	%Hits	Tiempo(ms)
4	136016	135970	46	99,97	463
8	136016	135975	41	99,97	413



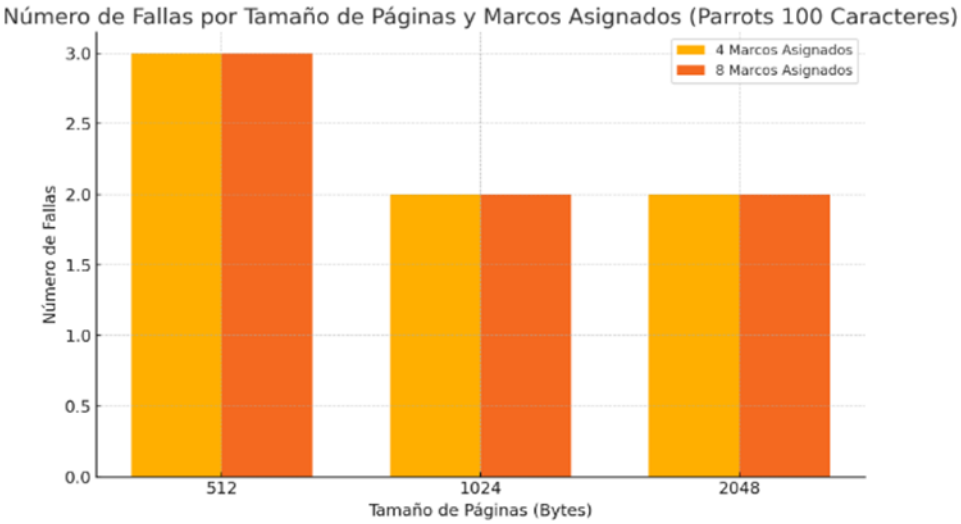
Tiempo:



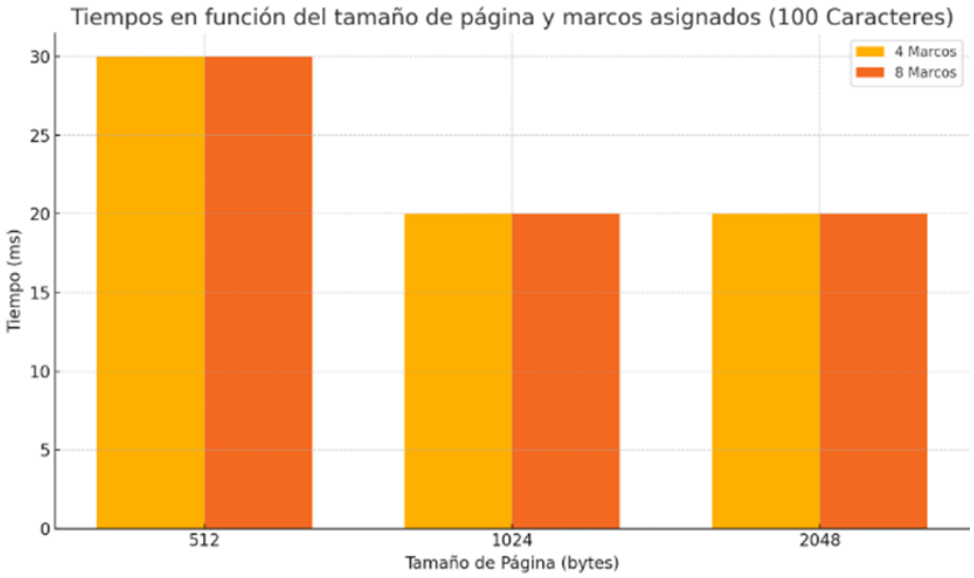
Parrots.bmp

- Archivo parrots_100.bmp: 100 caracteres

Páginas de 512, parrots_100.bmp: 100 Caracteres					
Marcos Asignados	Total Referencias	Hits	Fallas	%Hits	Tiempo(ms)
4	1716	1713	3	99,82	30
8	1716	1713	3	99,82	30
Páginas de 1024, parrots_100.bmp: 100 Caracteres					
Marcos Asignados	Total Referencias	Hits	Fallas	%Hits	Tiempo(ms)
4	1716	1714	2	99,88	20
8	1716	1714	2	99,88	20
Páginas de 2048, parrots_100.bmp: 100 Caracteres					
Marcos Asignados	Total Referencias	Hits	Fallas	%Hits	Tiempo(ms)
4	1716	1714	2	99,88	20
8	1716	1714	2	99,88	20

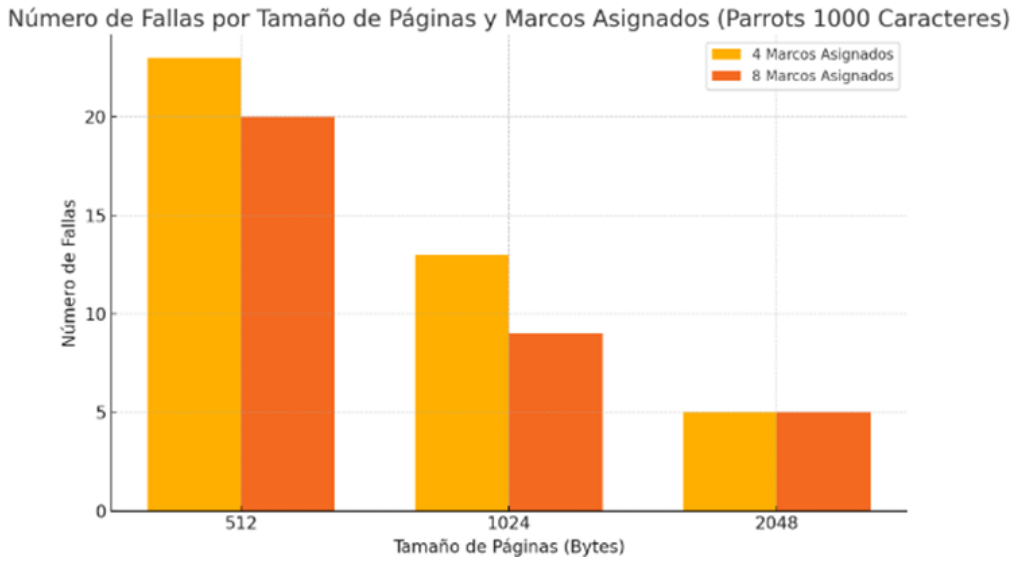


Tiempo:

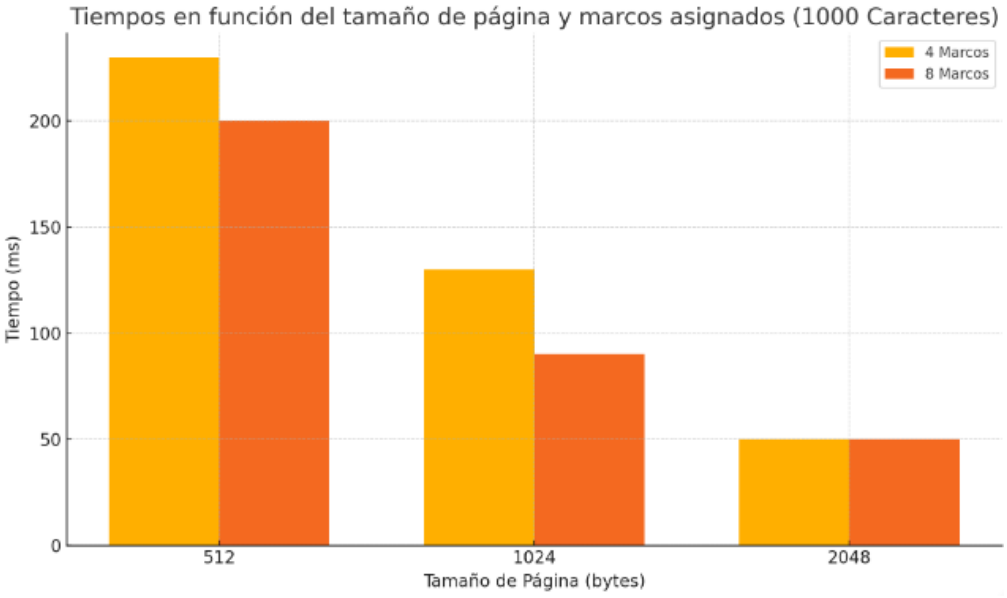


- Archivo parrots_1000.bmp: 1000 caracteres

Páginas de 512, parrots_1000.bmp: 1000 Caracteres					
Marcos Asignados	Total Referencias	Hits	Fallas	%Hits	Tiempo(ms)
4	17016	16993	23	99,86	230
8	17016	16996	20	99,82	200
Páginas de 1024, parrots_1000.bmp: 1000 Caracteres					
Marcos Asignados	Total Referencias	Hits	Fallas	%Hits	Tiempo(ms)
4	17016	17003	13	99,92	130
8	17016	17007	9	99,95	90
Páginas de 2048, parrots_1000.bmp: 1000 Caracteres					
Marcos Asignados	Total Referencias	Hits	Fallas	%Hits	Tiempo(ms)
4	17016	17011	5	99,97	50
8	17016	17011	5	99,97	50



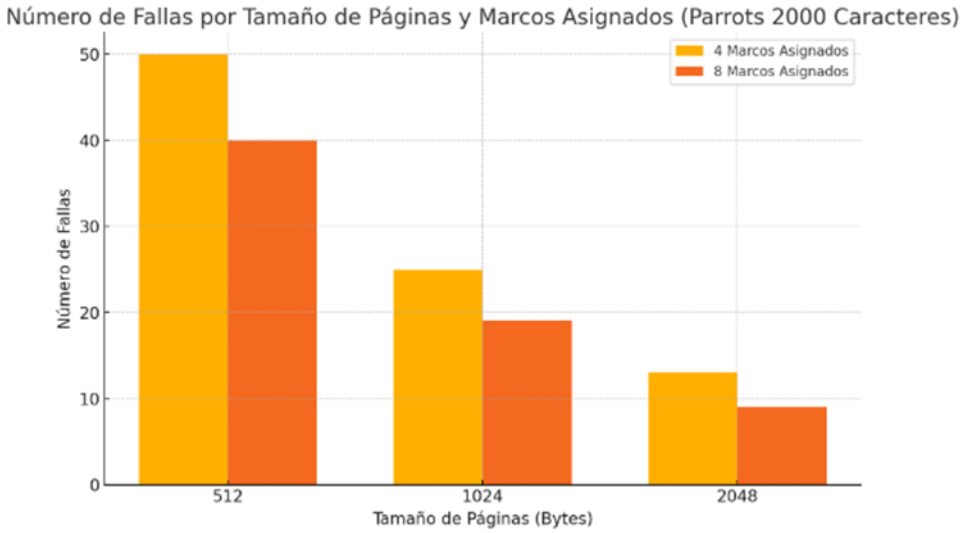
Tiempo:



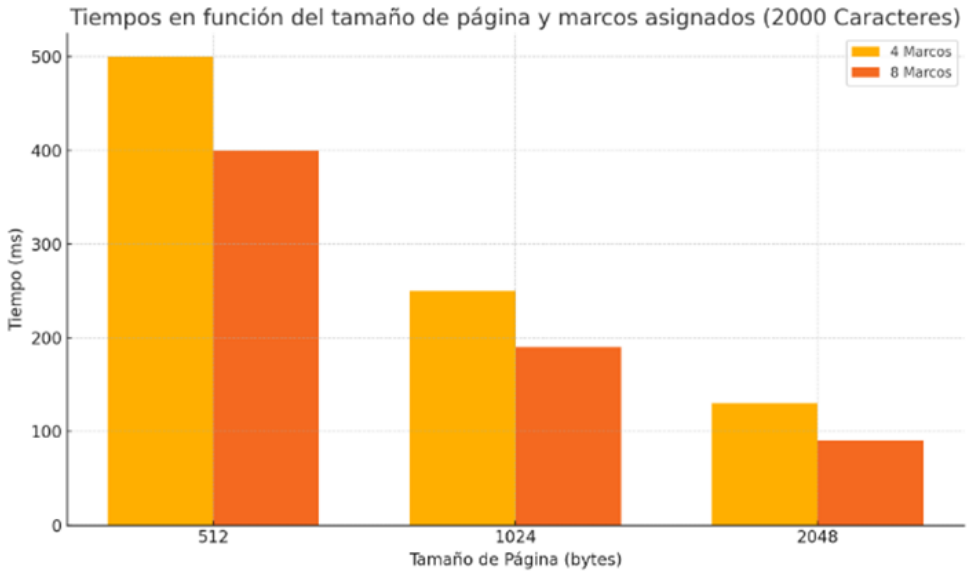
- Archivo parrots_2000.bmp: 2000 caracteres

Páginas de 512, parrots_2000.bmp: 2000 Caracteres					
Marcos Asignados	Total Referencias	Hits	Fallas	%Hits	Tiempo(ms)
4	34016	33966	50	99,85	500
8	34016	33976	40	99,88	400
Páginas de 1024, parrots_2000.bmp: 2000 Caracteres					
Marcos Asignados	Total Referencias	Hits	Fallas	%Hits	Tiempo(ms)
4	34016	33991	25	99,92	250
8	34016	33997	19	99,94	190
Páginas de 2048, parrots_2000.bmp: 2000 Caracteres					

Marcos Asignados	Total Referencias	Hits	Fallas	%Hits	Tiempo(ms)
4	34016	34003	13	99,96	130
8	34016	34007	9	90,97	90



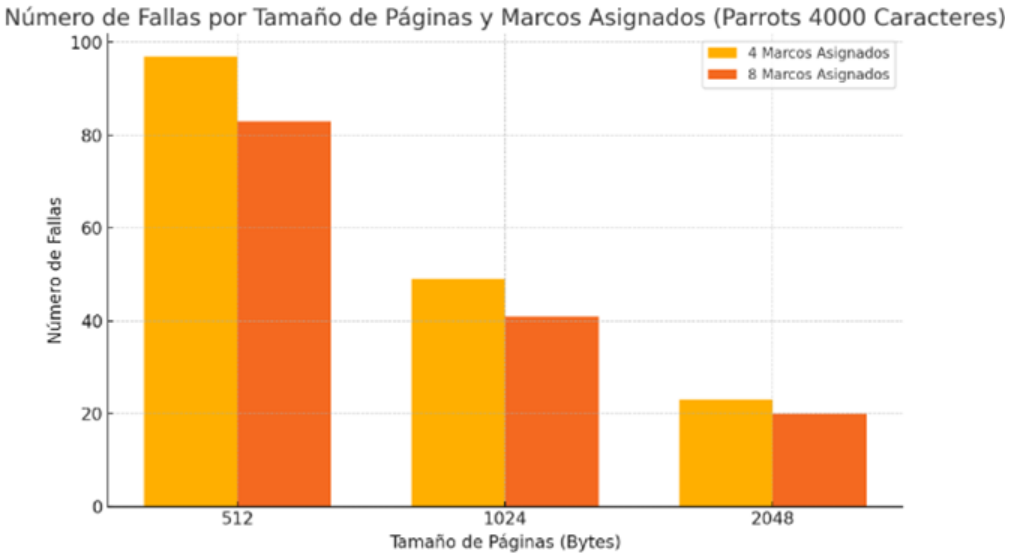
Tiempo:



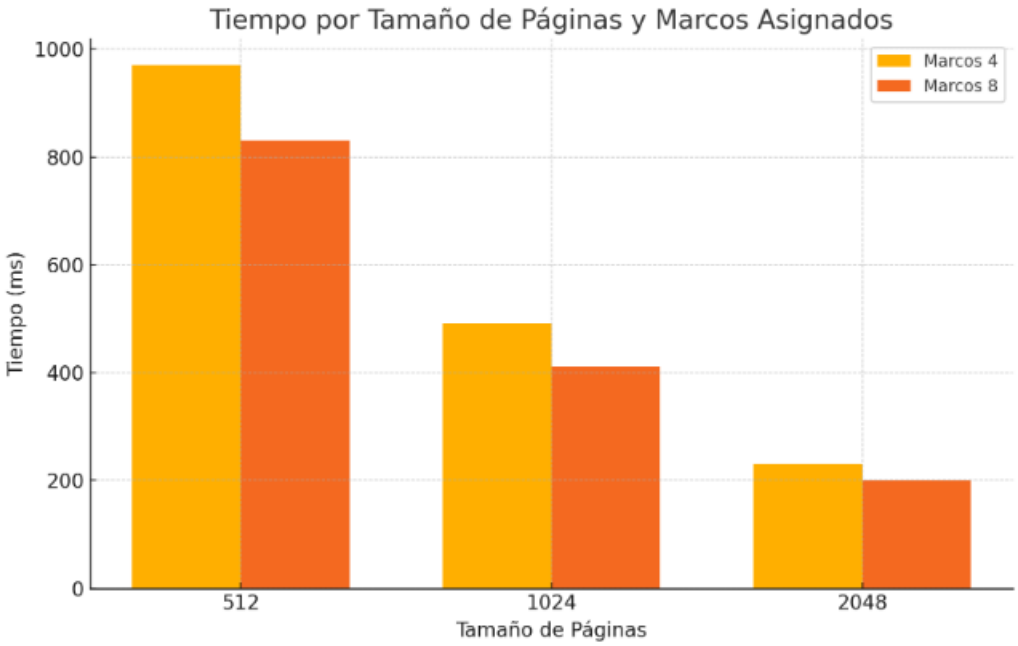
- Archivo parrots_4000.bmp: 4000 caracteres

Páginas de 512, parrots_4000.bmp: 4000 Caracteres					
Marcos Asignados	Total Referencias	Hits	Fallas	%Hits	Tiempo(ms)
4	68016	67919	97	99,86	971
8	68015	67931	84	99,88	830
Páginas de 1024, parrots_4000.bmp: 4000 Caracteres					

Marcos Asignados	Total Referencias	Hits	Fallas	%Hits	Tiempo(ms)
4	68016	67967	49	99,93	491
8	68016	67975	41	99,94	411
Páginas de 2048, parrots_4000.bmp: 4000 Caracteres					
Marcos Asignados	Total Referencias	Hits	Fallas	%Hits	Tiempo(ms)
4	68016	67993	23	99,97	231
8	68016	67996	20	99,97	201

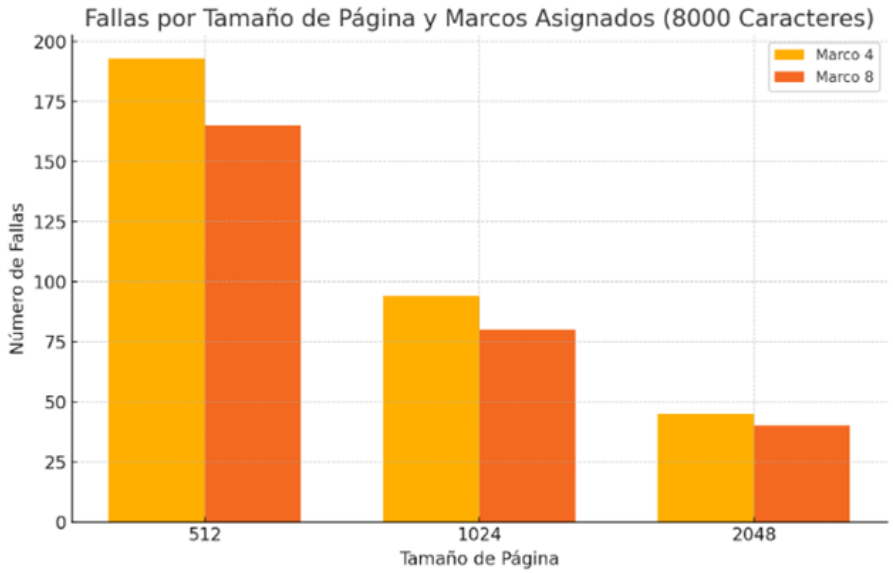


Tiempo:

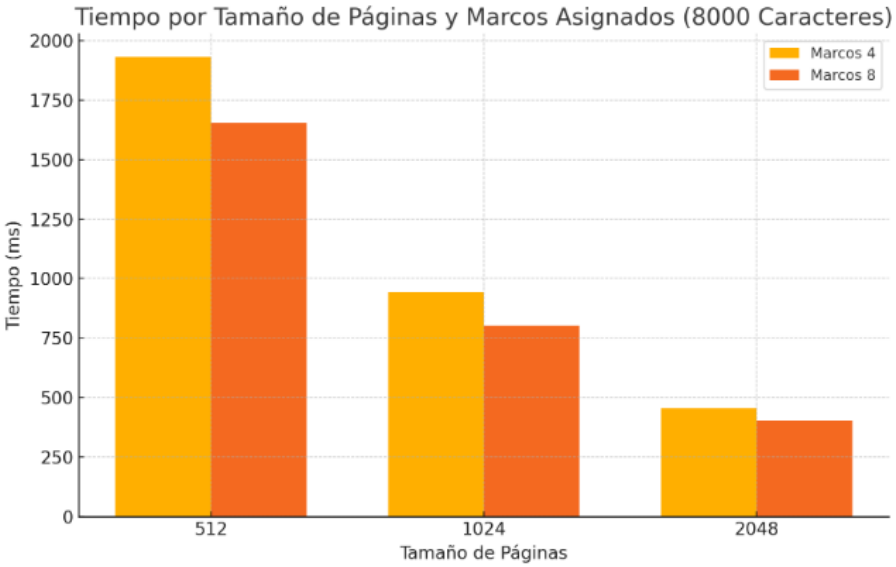


- Archivo parrots_8000.bmp: 8000 caracteres

Páginas de 512, parrots_8000.bmp: 8000 Caracteres					
Marcos Asignados	Total Referencias	Hits	Fallas	%Hits	Tiempo(ms)
4	136016	135823	193	99,86	1.933
8	136016	135851	165	99,88	1.653
Páginas de 1024, parrots_8000.bmp: 8000 Caracteres					
Marcos Asignados	Total Referencias	Hits	Fallas	%Hits	Tiempo(ms)
4	136016	135922	94	99,93	943
8	136016	135936	80	99,94	803
Páginas de 2048, parrots_8000.bmp: 8000 Caracteres					
Marcos Asignados	Total Referencias	Hits	Fallas	%Hits	Tiempo(ms)
4	136016	135971	45	99,97	453
8	136016	135976	40	99,97	403



Tiempo:



Interpretación de los resultados

Por la localidad del problema podemos observar que el cambio de marcos (4-8) no genera un cambio significativo en cuanto a número de fallas, sin embargo, se puede ver que el uso de un 8 marco asignado genera una menor cantidad de fallas que 4 marcos, esto se debe a que cuando se aumentan los marcos disponibles, el sistema puede mantener más páginas en la memoria física al mismo tiempo, significa que hay menos necesidad de reemplazar una página existente cuando se accede a una nueva, reduciendo así la probabilidad de una falla de página. Por otra parte, observamos que un mayor tamaño de páginas disminuye significativamente el número de fallas, ya que esto permite cargar más datos a la vez, disminuir la cantidad de páginas a gestionar y aprovechar mejor el tiempo de transferencia desde el disco. Finalmente, en cuanto al tiempo de respuesta es evidente que aumenta por el tamaño del texto. Todos los resultados obtenidos concuerdan con la teoría esperada y nos dejan entender mejor el funcionamiento de la paginación en memoria virtual.

¿Si la localidad del problema manejado fuera diferente cómo variarían los resultados? Explique su respuesta. (considere una localidad mayor y una localidad menor)

Para simular un escenario diferente se asume que, para recuperar el mensaje oculto dentro de la imagen, este no se encuentra almacenado en los siguientes bits después de donde se encuentra la longitud del mensaje. Bajo este escenario el algoritmo tomaría más tiempo en ejecutarse, pues tendría que acceder a todos los bits de la imagen hasta encontrar el mensaje oculto y después realizar su lectura, provocando un aumento en el tiempo de ejecución pues se debe añadir al cálculo el tiempo que toma acceder a todos los bits de la imagen.

Otras configuraciones que le permitan entender cómo afecta la memoria virtual el desempeño del programa.

En caso de que para la solución se implementaran mecanismos para compartir memoria virtual en lugar del algoritmo LRU el desarrollo de la solución cambia ya que los mecanismos permiten a diferentes procesos compartir memoria y espacios virtuales. Si bien, ofrece flexibilidad, eficiencia y seguridad, tiene costos de desempeño, ya que, al cargar varios procesos a la vez, estos no se cargan por completo, lo cual afecta el flujo de datos y los resultados de las operaciones. De igual manera, si se cargan varias veces librerías compartidas puede llegar a que se desperdicien recursos, pues el sistema operativo es quien decide los espacios de memoria disponibles para los TP.