Universidad de los Andes

Caso 2

Grupo 10:

Manuel Gomez Salazar-202020415

Juan Diego Niebles-202221193

Manuela Lizcano-202122826

Tabla de contenido

1. Algoritmo de generación de referencias de página
2. Estructuras de datos para sistema de paginación
3. Esquema de sincronización
4. Tabla con los datos recopilados
5. Gráficas
6. Otros configuraciones a considerar
7. Gráfica de tiempo
8. Pregunta
9. Diagrama UML

Algoritmo de generación de referencias de página

Antes de iniciar con la descripción de la lógica que se tuvo en cuenta para la creación de las referencias de página, primero en las primeras 5 primeras líneas de archivo se incluyeron las siguientes variables:

P	Tamaño de las Páginas
NF	Número de Filas
NC	Número de Columnas
NR	Número de Referencias
NP	Numero de Paginas

Estas variables nos permiten especificar las características que fueron consideradas para la creación de las referencias de páginas generadas por un proceso de recuperación de un mensaje escondido en una imagen.

Siguiendo con la lógica que fue utilizada para la implementación de la función crearReferenicia, esta función tiene como parámetros:

tamanioPaginas	Tamaño de las páginas
numPaginas	Numero de paginas totales que van a usar
numReferencias	Número de referencias totales

Estos parámetros fueron calculados de la siguiente manera:

1. **tamanioPaginas:** El tamaño de la página es definido según la configuración que requiera para la creación de las referencias.
2. **numPaginas:** Para el cálculo del número de páginas necesarias para la creación de la referencias, se deben tener en cuenta dos consideraciones:
    a. Las dimensiones de la imagen y la longitud del mensaje que está escondido. Con el objetivo de encontrar las dimensiones de la imagen es necesario hacer uso de la función de Imagen(String input), donde se calculan los valores del alto y el ancho de la imagen. De igual manera se debe tener en cuenta los tres diferentes colores, por esta razón, se multiplica por 3 la dimensión, con el objetivo de considerar (RED, GREEN, BLUE).
    b. La longitud es calculada por medio de la función de leerLongitud().

Teniendo esto en cuenta, la función usada para calcular el número de páginas, es la siguiente:

$$numPaginas = ((Alto * Ancho * 3)/tamanioPaginas) + (longitudMensaje/tamanioPaginas)$$

Sin embargo, para ambos casos, se divide sobre el tamaño de la página con el objetivo de lograr identificar la mínima cantidad de páginas necesarias para guardar las referencias creadas.

3. **numReferencias:** Con el objetivo de calcular el número de referencias totales, se debe tener en cuenta el número de accesos en memoria que se realizan. Esto se deduce por medio de la función dada en el enunciado recuperar(char[] cadena, int longitud). Esta función fue implementada con doble for loop, el cual realiza los accesos a memoria en dos ocasiones. En primer lugar, se inicializa el valor con un acceso a memoria, cuando define cadena[posCaracter]=0, finalmente, se accede un total de 16 veces con el objetivo de lograr recuperar el mensaje que está escondido en la imagen. Sin embargo, hay que tener en cuenta que las primeras 16 instancias incluyen la longitud del mensaje, las cuales también deben ser consideradas para calcular el número de referencias.
   Con esto en mente, la función usada para calcular el número de referencias es:

   $$numReferencias = (longitudMensaje * 17 + 16)$$

Para crear las referencias, optamos por dividir el funcionamiento de estas en dos grupos: las referencias de la longitud del mensaje (primeros 16 bytes) y las referencias del contenido del mensaje (resto).

La creación de las referencias, inicia en el índice 6, considerando que las primeras 5 pertenecen a las variables mencionadas anteriormente, las cuales incluyen (P, NF, NC, NR, NP).

Para lograr asegurar cierto nivel de consistencia para la creación de la referencias, creamos las variables de página, desplazamiento, referencia y mensaje, el funcionamiento de cada una de ellas es el siguiente:

página	Esta variable se inicializa en 0 y se encarga de determinar la pagina virtual en la que los registros se encuentran en un momento dado de la ejecución
desplazamiento	Esta variable se inicializa en 0 y determina la posición dentro de cada página y esta varía según el momento de ejecución
referencia	Esta variable se inicializa con un valor de 0 y determina el número total de referencias que se han creado
mensaje	Esta variable fue inicializada como un string vacío, sin embargo esta variable recuperara

	todas las referencias que son generadas según el proceso de ejecución

Ahora, inicialmente, para crear las referencias relacionadas con la longitud del mensaje, se deben tener en cuenta que en los primero 16 bytes se guarda la longitud del mensaje que está escondido, es por esto que estos accesos en memoria siempre ocurren en la fila con índice de 0 y la columna varía según los 16 bytes que guardan estos registros. En el momento en que se realizan las referencias se tiene en cuenta el desplazamiento, el cual permite determinar el número de páginas virtuales que son usadas, en caso de que el desplazamiento alcance el límite del tamaño de la página se opta por hacer uso de una nueva página. Primero se realiza un recorrido por los 16 bytes que incluyen la longitud, con el objetivo de lograr determinar para cada uno de estos, la fila del pixel, la columna del píxel, el valor del componente con la función de aux RGB(k), la página actual, el desplazamiento y al final se incluye la acción (puede ser de $lectura = R$ o $escritura = W$). Para estos primeros 16 bytes, siempre es de lectura, considerando que estamos en el contexto que incluye únicamente la longitud del mensaje.

Teniendo en cuenta la explicación anterior, definimos cada una de las referencias considerando la siguiente estructura:

$Imagen[0][j].auxRGB(k),\ pagina,\ desplazamiento,\ R(lectura)$

Para realizar la implementación de las referencias creadas a partir del contenido del mensaje, tuvimos en cuenta las variables de filaActual, columnaActual, k, paginaMensaje, desplazamientoMensaje y contadorBits. La explicación de cada una de las variables, se encuentra a continuación:

filaActual	Fila actual en la imagen
columnaActual	Columna actual de la imagen
k	Usado para identificar el índice de los colores
contadorMensaje	Contabiliza la cantidad de caracteres que han sido procesados
paginaMensaje	Identifica en que pagina estara un mensaje en específico, según la dimensión de la imagen y el tamaño de la página que se está usando
indiceMensaje	Representa el índice dentro del mensaje que

	se está escribiendo
desplazamientoMensaje	Tiene en cuenta los desplazamientos que han sido realizados en el mensaje a escribir
contadorBits	Lleva el conteo de los bits que han sido procesados, considerando que cada 8 bits, se completa la lectura de un carácter, y justo en ese momento se genera una referencia de escritura
longitud	Lleva la longitud del mensaje que está escondido en la imagen

Seguido de esto, se debe tener en cuenta las referencias creadas a partir del contenido del mensaje, con estas referencias lo que se quiere llegar a lograr es realizar el proceso de lectura y escritura sobre el contenido del archivo. Para este caso, el valor de la fila y columna varía según los límites establecidos con el número de filas y número de columnas. Para el caso de la columna, esta cambia en un factor de 3, considerando que se tiene que tener en cuenta los tres colores(RED, GREEN, BLUE). Cuando el variable k llega 3 esto significa que ya se realizó la lectura, por esto la variable de columnaActual incrementa en una unidad. Sin embargo, en caso de que se llegue al límite de la imagen, es decir que la columnaActual es igual al ancho de la imagen, la variable de filaActual aumenta en una unidad. Para los registros de la lectura, cada 8 bits, el desplazamiento aumenta, es por esto que cada 8 bits, se realiza el proceso de acceder a memoria y esto se ilustra por medio de una escritura. En caso de que todavía no se alcance los 8 bits, de igual manera se realizará el proceso de lectura. Todo este proceso de lectura y escritura del mensaje se realiza hasta que se alcance la longitud final del mensaje.

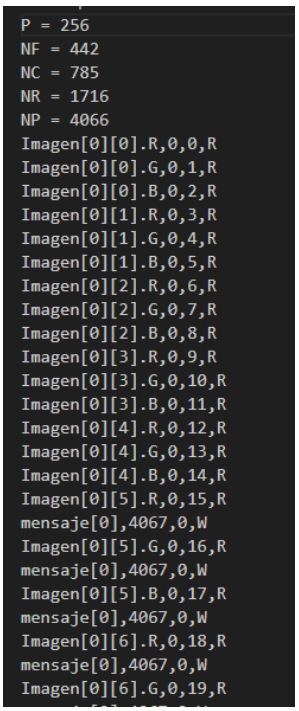
Para el caso de las referencias cuya acción sea lectura, la estructura es la siguiente:
$Imagen[filaActual][columnaActual].auxRGB(k), pagina,\ desplazamiento,\ R$

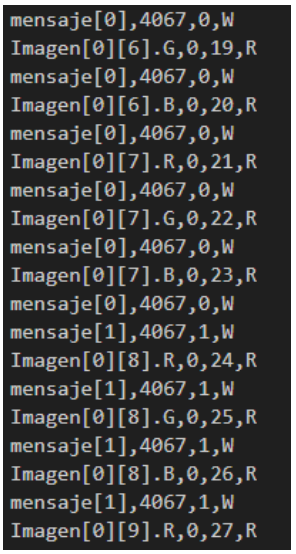
Para el caso de las referencias cuya acción sea escritura, la estructura es la siguiente:
$mensaje[indiceMensaje], pagina,\ desplazamiento,\ W$

La función $auxRGB(k)$ se encarga de lograr convertir los índices numéricos en representación de los colores, (RED, GREEN, BLUE).

k	**Color**
0	Red
1	Green
2	Blue

Así se visualiza el archivo donde se generan las referencias de la imagen analizada.





Estructura de datos para sistemas de paginación

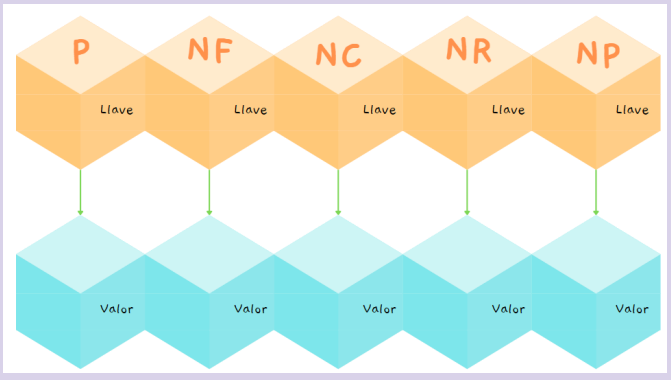
Con el objetivo de lograr ejecutar correctamente el sistema de paginación, se tuvo en cuenta lo siguiente para el cálculo de las fallas y hits.

En primer lugar, en la variable de cabeceras, se guarde información relevante para el cálculo de las fallas y éxitos, para este caso se creó un HashMap el cual contiene información como:

P	Tamaño de las Páginas
NF	Número de Filas

NC	Número de Columnas
NR	Número de Referencias
NP	Numero de Paginas

Estructura explícita:

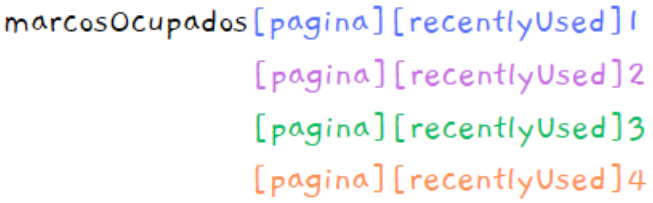


De igual forma, utilizamos un ArrayList, que nos permite guardar las referencias según la configuración que se está analizando. Por esta razón, esta lista está construida usando números enteros que permiten identificar una página que está referenciada. Con esta estructura se guarda el valor de la cantidad de referencias creadas.

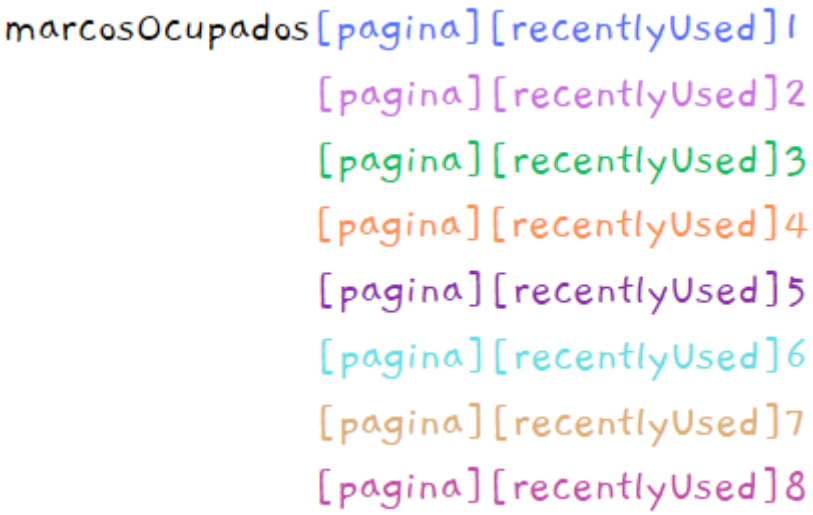
Sin embargo, la estructura de datos, más importante para realizar el sistema de paginación del algoritmo, fue una matriz. La matriz $marcosOcupados$, tiene dimensión de $2 * N$, donde N representa la cantidad de marcos que han sido ocupados en la memoria. Para esta estructura, en la primera posición de cada fila se guarda el número de la página y en el segundo índice se guarda un valor según la acción realizada, por ejemplo, en este contexto usaremos la siguiente idea, en caso de que sea una acción de lectura, esta variable debe incrementar 10 unidades, si es escritura esta variable incrementa en una unidad, esta lógica fue usada considerando que la lectura tienen una mayor prioridad que la acción de escritura. Finalmente, se usa la política de reemplazo según el algoritmo Least Recently Used, con el objetivo de lograr identificar los elementos que no han sido usados recientemente para que estos sean reemplazados. La variable de $marcosOcupados$, se actualiza cada vez que se invoca el método $modificarRecentlyUsed$, este metodo tiene como función, lograr llevar un registro de las páginas y el uso del marco de página. Sin embargo, la variable de $marcosOcupados$, también se actualiza por el método de $buscarIndicePagina$, el cual tiene como objetivo identificar si una página en específico se encuentra en el marco. En caso de que no se encuentre, se invoca al reemplazo de página, haciendo que la variable $marcosOcupados$, se tendrá que actualizar.

Estructura explícita:

Considerando el uso de un marco de tamaño: 4



Considerando el uso de un marco de tamaño: 8



Esquema de sincronización

El uso de threads para esta implementación era necesario para implementar satisfactoriamente el reemplazo de páginas en la memoria, considerando el algoritmo Least Recently Used. Este algoritmo busca reemplazar la página que ha sido usada menos recientemente con el objetivo de poder definir correctamente las referencias que se encuentran en el marco.

Con el objetivo de garantizar que se cumpla la sincronización entre eventos, se realizó la sincronización de los siguientes métodos: $buscarIndicePagina$ y $buscarIndiceMenorRecentlyUsed$.

La primera función, $buscarIndicePagina$, logra identificar si una página en específico ya se encuentra en el marco. En el caso de que se encuentre la página dentro de la estructura de $marcosOcupados$, esto cuenta como hit. Para el caso contrario, donde la página no se encuentra entonces se incrementará en una unidad el contador de fallas. Considerando que esta función, accede constantemente a la matriz de $marcosOcupados$, es indispensable que este método esté sincronizado para evitar que múltiples threads accedan a esta función de manera paralela, haciendo que se puedan generar inconsistencias.

La segunda función, *buscarIndiceMenorRecentlyUsed*, se encarga de encontrar el marco de página que ha sido usado con menor frecuencia el cual depende de la variable RecentlyUsed. Este método asegura mantener las páginas adecuadas dentro del marco. Para este caso, tenemos que tener en cuenta un tiempo de parada de 1 milisegundo, el cual se toma de volver a comenzar con el ciclo de búsqueda de aquellas páginas.
Estos métodos deben estar sincronizados para la implementación del sistema de manejo de memoria para el reemplazo de páginas.

Tabla con los datos recopilados

Con el objetivo de verificar el funcionamiento del algoritmo, utilizamos un serie de casos distribuidos en la tablas a continuación:

Imagen de dimensión: 384 píxeles x 256 píxeles

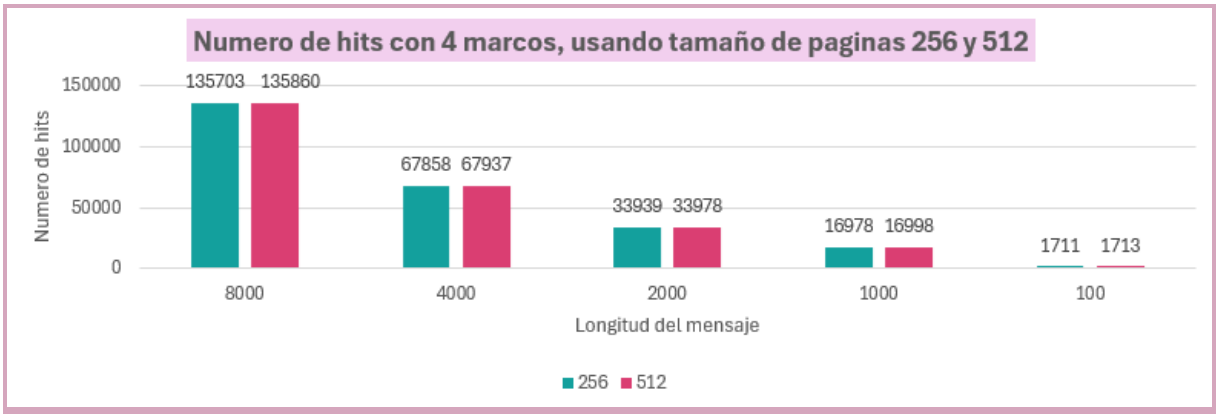
Marco	Tamaño pagina	Dimension imagen	Tamaño mensaje	# Referencias	Hits	% Hits	Fallas	% Fallas	RAM ns	SWAP ms	Total ms
4	256	384 x 256	8000	136016	135703	99.77%	313	0.23%	3392575	3130	33130
8				136016	135707	99.77%	309	0.23%	3392675	3090	33090
4	512			136016	135860	99.89%	156	0.11%	3396500	1560	31560
8				136016	135862	99.89%	154	0.11%	3396550	1540	31540
4	256		4000	68016	67858	99.77%	158	0.23%	1696450	1580	11580
8				68016	67864	99.78%	152	0.22%	1696600	1520	11520
4	512			68016	67937	99.88%	79	0.12%	1698425	790	1790
8				68016	67943	99.89%	73	0.11%	1698575	730	1730
4	256		2000	34016	33939	99.77%	77	0.23%	848475	770	770
8				34016	33943	99.79%	73	0.21%	848575	730	730
4	512			34016	33978	99.89%	38	0.11%	849450	380	380
8				34016	33980	99.89%	36	0.11%	849500	360	360
4	256		1000	17016	16978	99.78%	38	0.22%	424450	380	380
8				17016	16980	99.79%	36	0.21%	424500	360	360
4	512			17016	16998	99.89%	18	0.11%	424950	180	180
8				17016	16998	99.89%	18	0.11%	424950	180	180
4	256		100	1716	1711	99.71%	5	0.29%	42775	50	50
8				1716	1711	99.71%	5	0.29%	42775	50	50
4	512			1716	1713	99.83%	3	0.17%	42825	30	30
8				1716	1713	99.83%	3	0.17%	42825	30	30

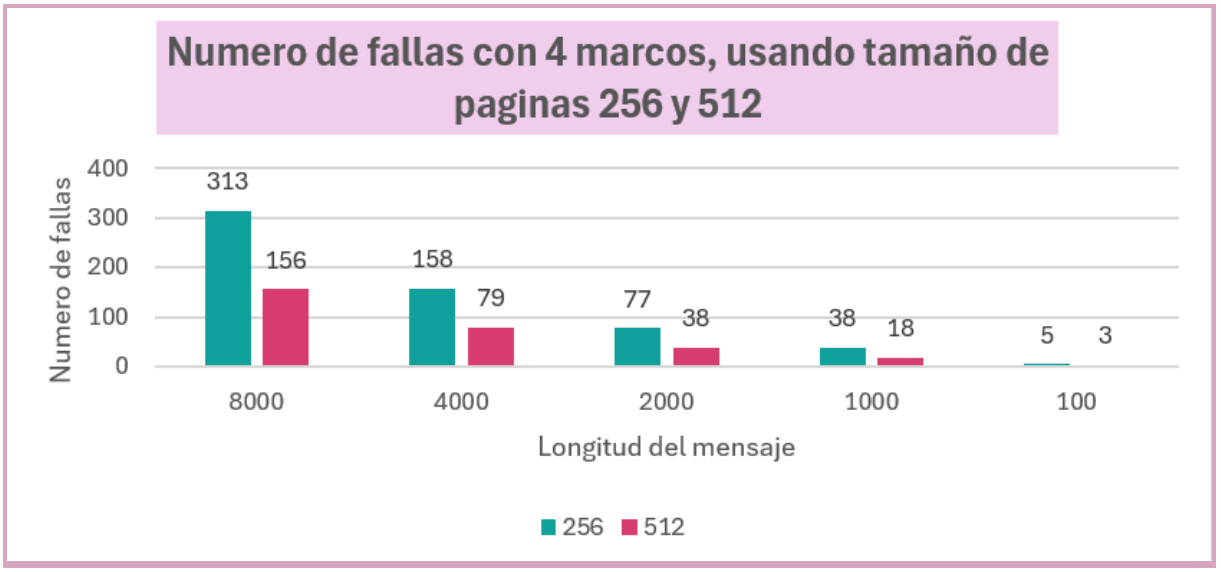
Imagen de dimensión: 785 píxeles x 442 píxeles

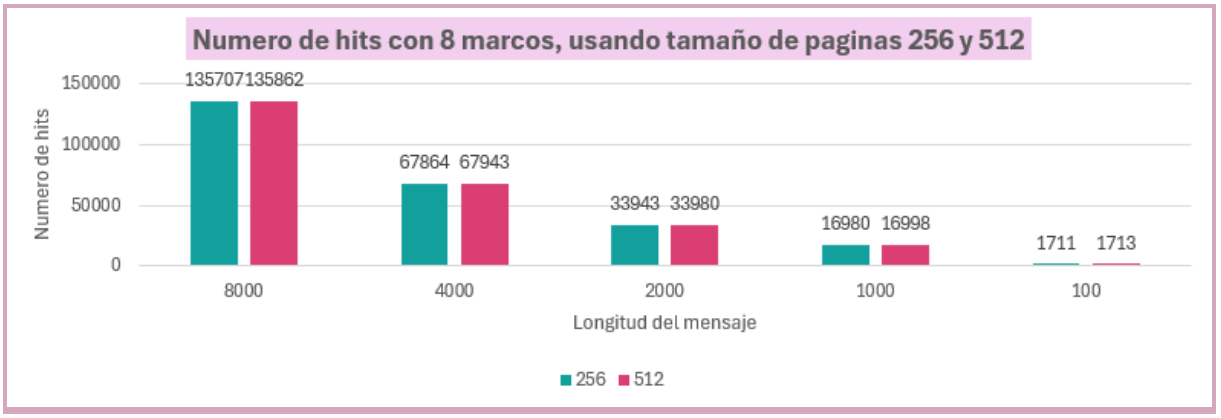
Marco	Tamaño pagina	Dimension imagen	Tamaño mensaje	# Referencias	Hits	% Hits	Fallas	% Fallas	RAM ns	SWAP ms	Total ms
4	256	785x442	8000	136016	135701	99.77%	315	0.23%	3392525	3150	33150
8				136016	135707	99.77%	309	0.23%	3392675	3090	33090
4	512			136016	135856	99.88%	160	0.12%	3396400	1600	31600
8				136016	135864	99.89%	152	0.11%	3396600	1520	31520
4	256		4000	68016	67858	99.77%	158	0.23%	1696450	1580	11580
8				68016	67862	99.77%	154	0.23%	1696550	1540	11540
4	512			68016	67937	99.88%	79	0.12%	1698425	790	1790
8				68016	67943	99.89%	73	0.11%	1698575	730	1730
4	256		2000	34016	33939	99.77%	77	0.23%	848475	770	770
8				34016	33943	99.79%	73	0.21%	848575	730	730
4	512			34016	33978	99.89%	38	0.11%	849450	380	380
8				34016	33980	99.89%	36	0.11%	849500	360	360
4	256		1000	17016	16978	99.78%	38	0.22%	424450	380	380
8				17016	16980	99.79%	36	0.21%	424500	360	360
4	512			17016	16998	99.89%	18	0.11%	424950	180	180
8				17016	16998	99.89%	18	0.11%	424950	180	180
4	256		100	1716	1711	99.71%	5	0.29%	42775	50	50
8				1716	1711	99.71%	5	0.29%	42775	50	50
4	512			1716	1713	99.83%	3	0.17%	42825	30	30
8				1716	1713	99.83%	3	0.17%	42825	30	30

Gráficas

Imagen de dimensión: 384 píxeles x 256 píxeles







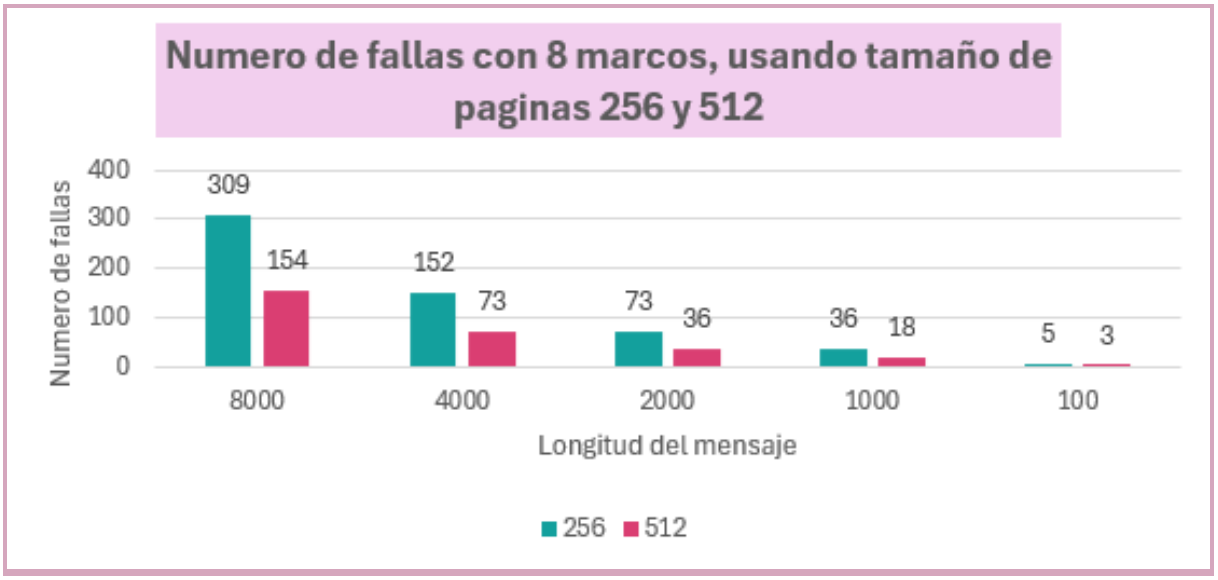
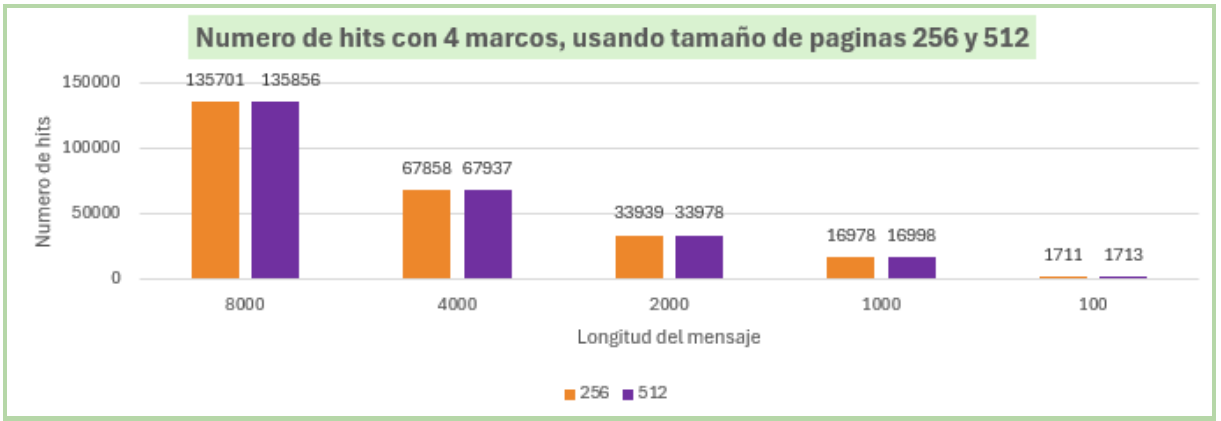
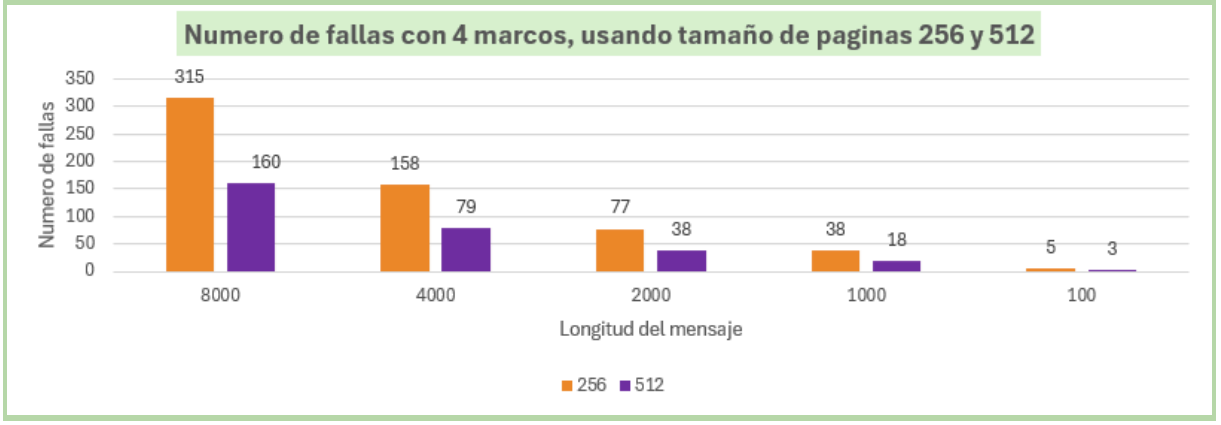
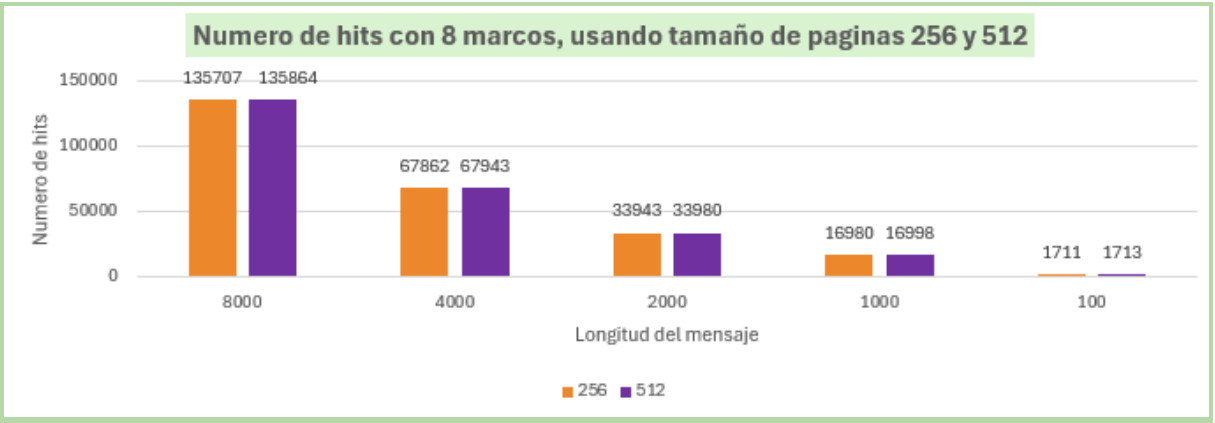
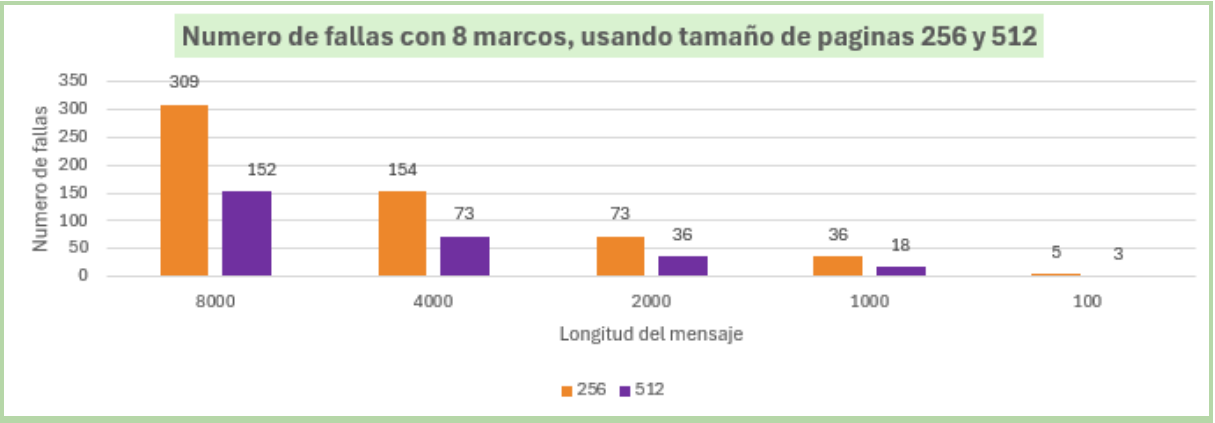


Imagen de dimensión: 785 píxeles x 442 píxeles









Interpretación de resultados

Se evaluaron diferentes escenarios en el sistema de paginación, en estas gráficas se muestra una comparación entre el número de hits y fallas. El mayor número de hits en comparación con las fallas se debe principalmente a la localidad de referencia. Este comportamiento permite que las páginas cargadas en la RAM sean reutilizadas para las siguientes operaciones, reduciendo así la cantidad de fallas. El tamaño de página de 512 resultó en un mejor desempeño, proporcionando un equilibrio adecuado entre el número de páginas almacenadas en la memoria principal y la reducción de la carga en la memoria swap. Esto disminuye el número de fallas de página, lo que permite acceder a los datos de forma más rápida y eficiente.

Otros configuraciones a considerar

Durante el desarrollo del caso, la generación de referencias podían ser creadas cambiando 4 parámetros, las cuales son:

1. Dimensiones de imagen
2. Tamaño de páginas
3. Longitud de mensaje
4. Marcos de página

Al realizar múltiples iteraciones, pudimos ver el comportamiento del algoritmo de paginación bajo diferentes escenarios y como este se comportaba al variar los parámetros mencionados anteriormente.

Para nuestro caso, algunas variaciones de estos parámetros fueron:

Dimensiones de imagen

Imagen de dimensión: 384 píxeles x 256 píxeles
Imagen de dimensión: 785 píxeles x 442 píxeles

Tamaño de Páginas

256
512

Longitud de mensaje

8000 caracteres
4000 caracteres
2000 caracteres
1000 caracteres
100 caracteres

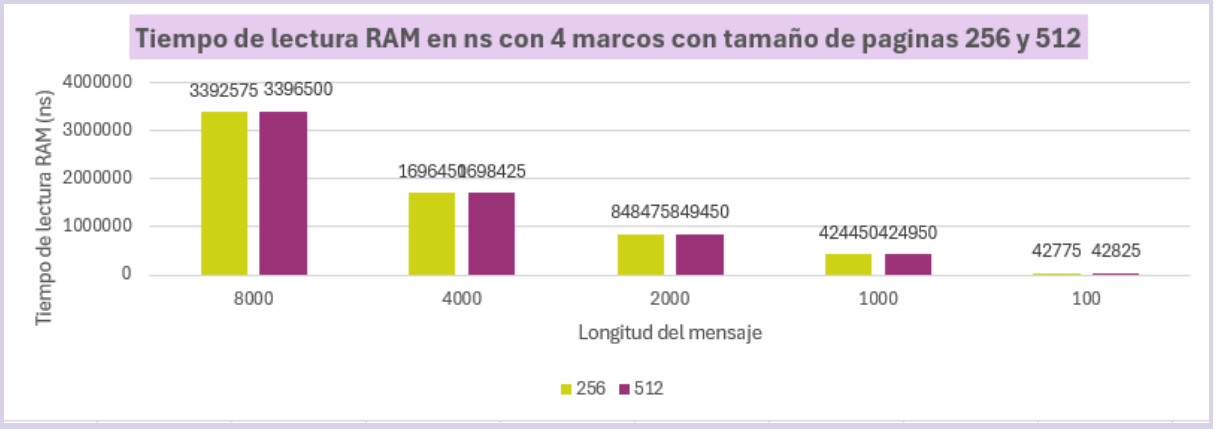
Marcos de paginación

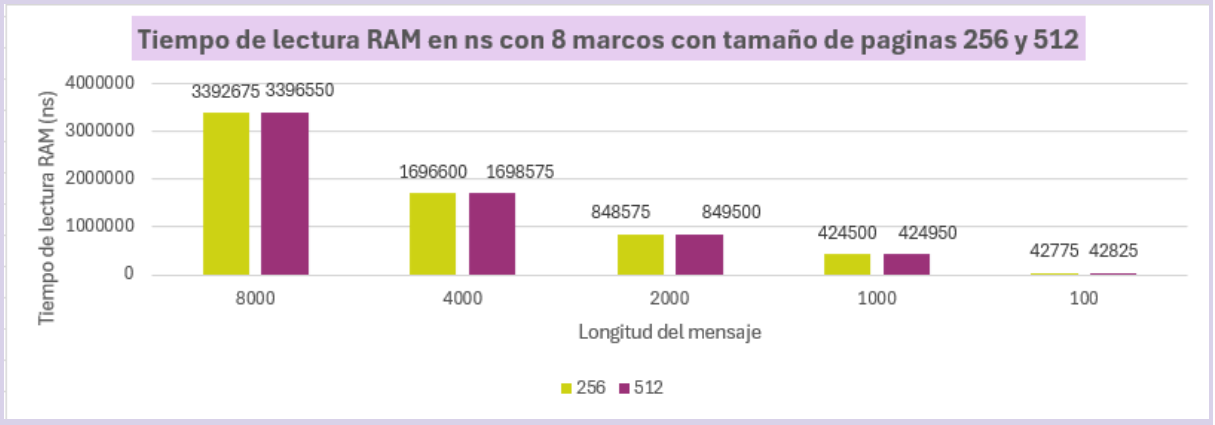
4
8

Gráfica de Tiempo

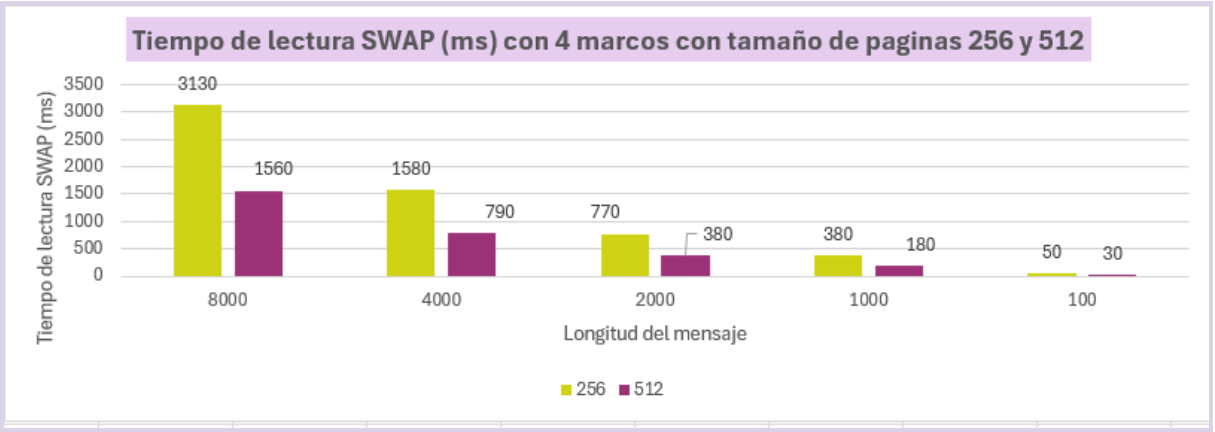
Lectura en RAM

Imagen de dimensión: 384 píxeles x 256 píxeles





Lectura en SWAP



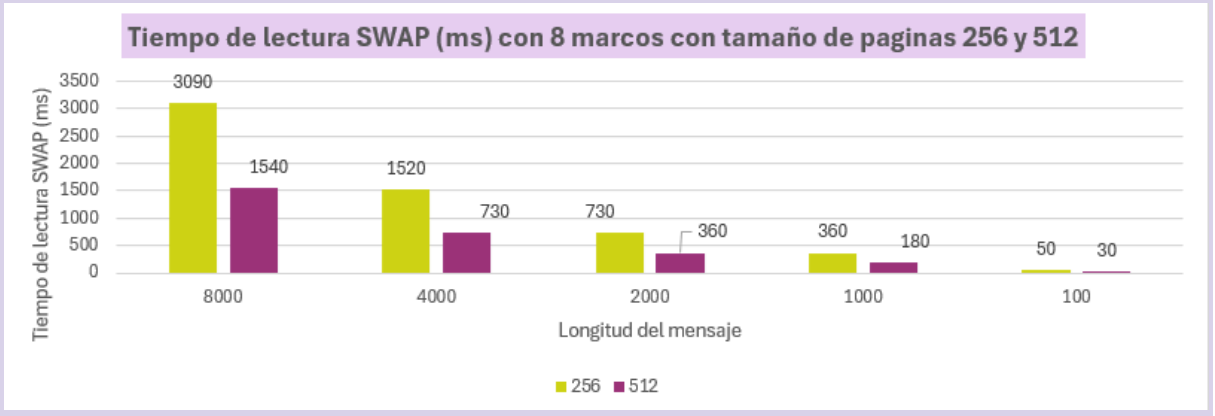
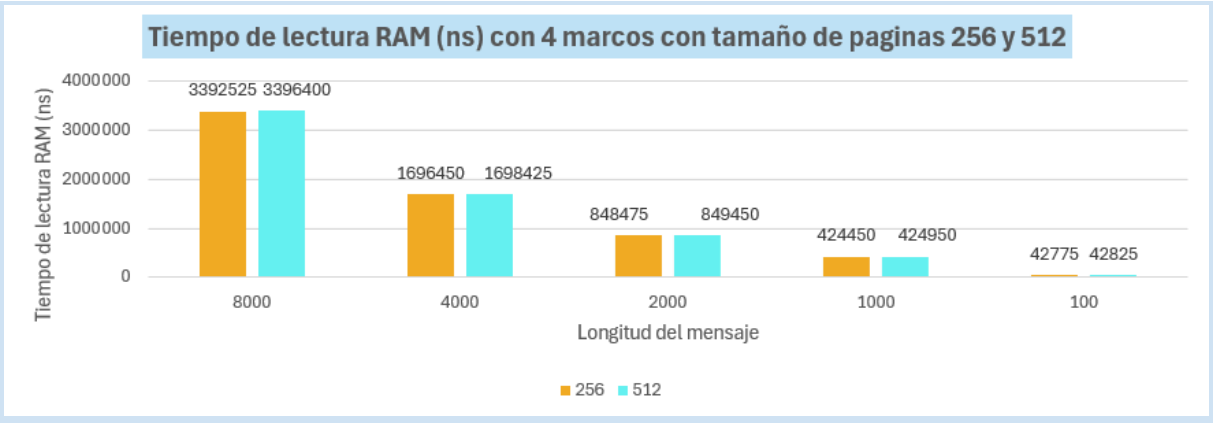
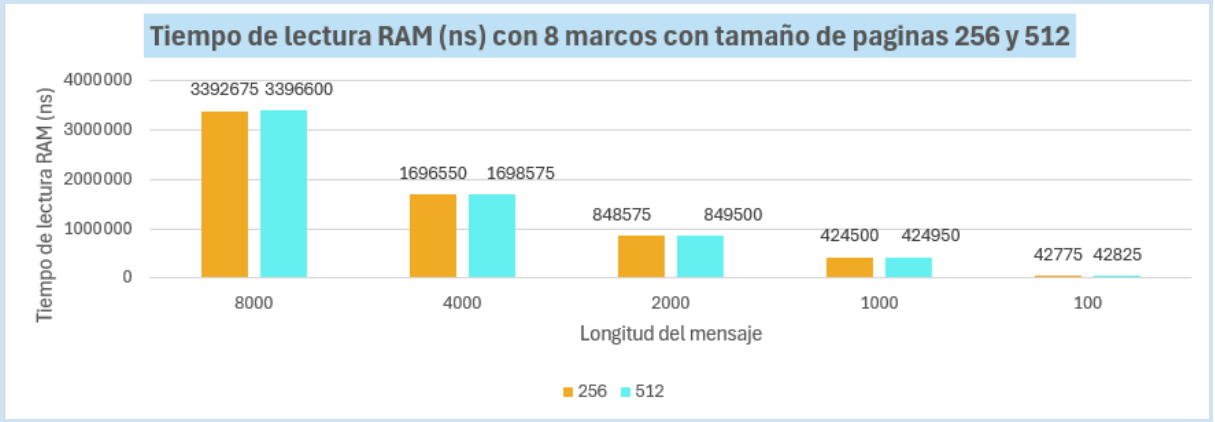


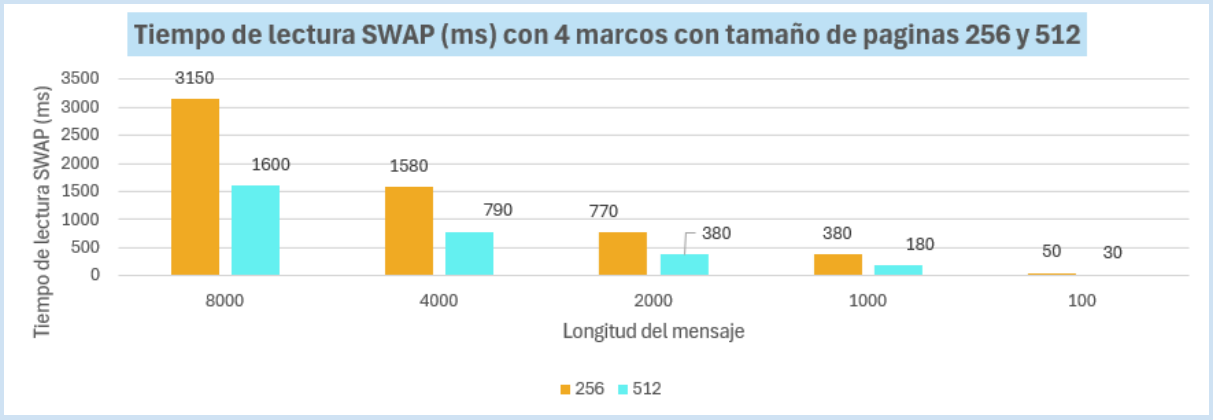
Imagen de dimensión: 785 píxeles x 442 píxeles

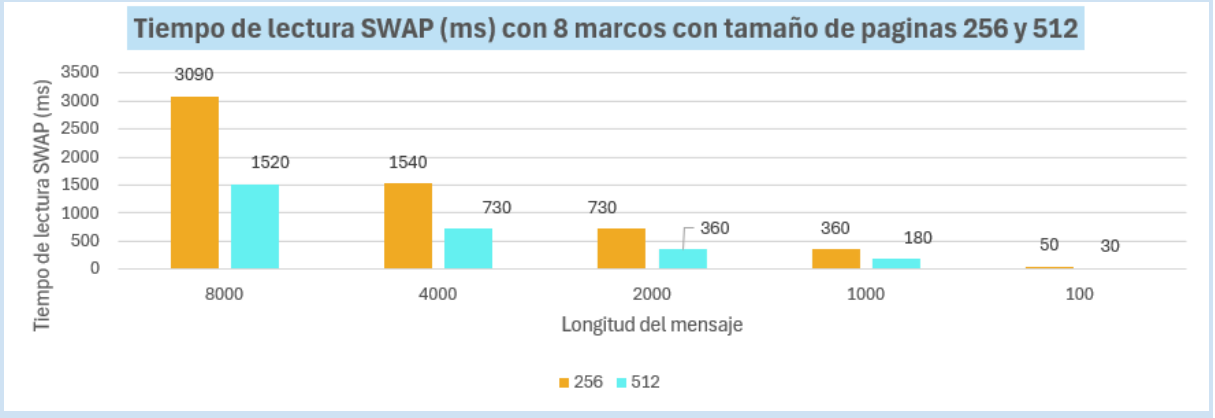
Lectura en RAM



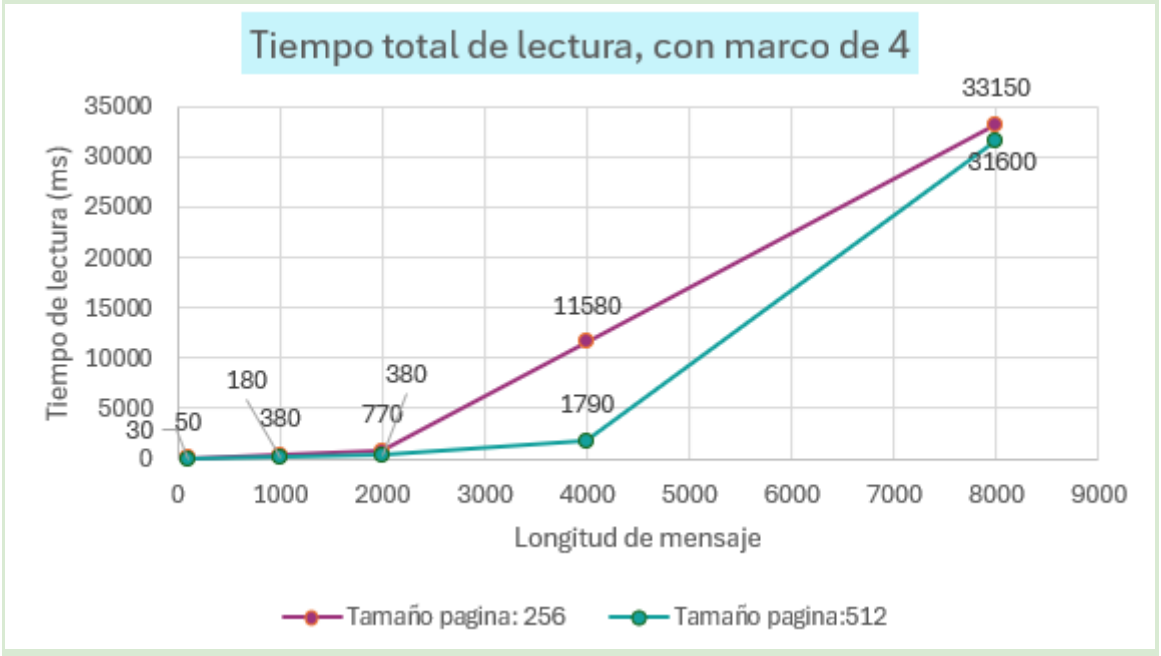


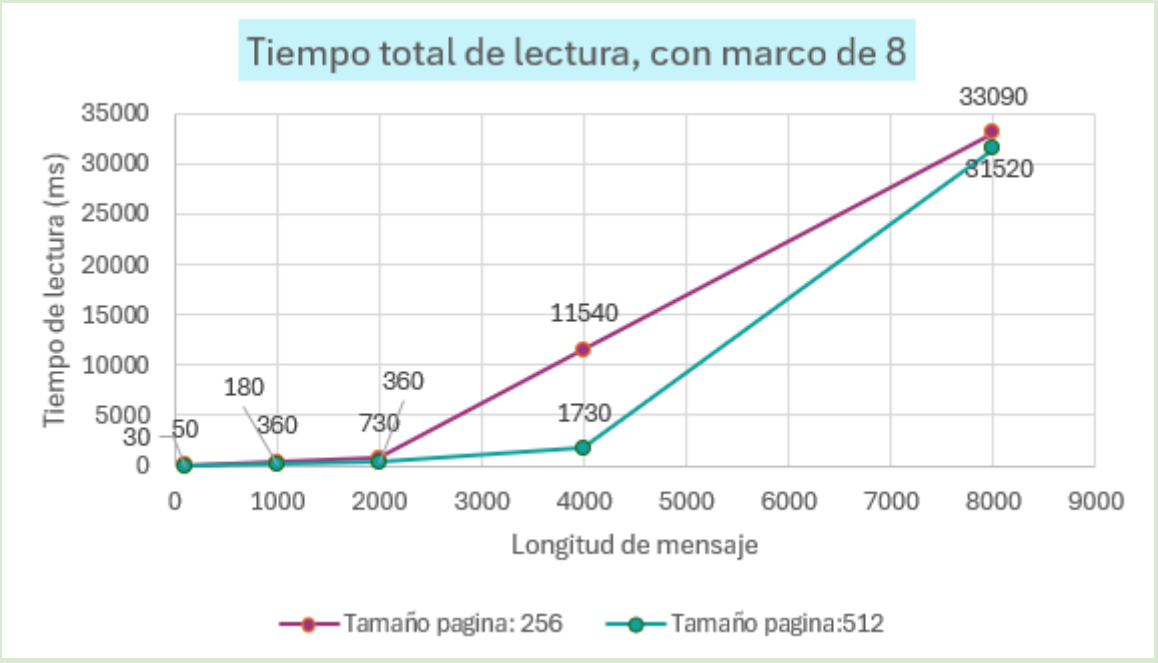
Lectura en SWAP





Tiempo total de lectura





Interpretación de resultados

Se realizaron un total de 40 pruebas, variando 5 longitudes de mensaje, 2 tamaños de marco, 2 tamaños de página y 2 imágenes de diferente tamaño. Los resultados mostrados en las gráficas indican que el tamaño de página de 512 ofrece el mejor rendimiento, reflejado en un mayor número de hits y un tiempo de lectura reducido en la memoria swap. El menor número de fallas sugiere una menor dependencia de la memoria swap, lo que se traduce en una reducción de la latencia. El uso de un tamaño de página de 512 optimiza el uso de la RAM, disminuyendo las operaciones necesarias en la memoria secundaria (swap) y mejorando la eficiencia global del sistema.

Pregunta

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

Mayor localidad: Un aumento en la localidad mejora el rendimiento del programa en el manejo de memoria, ya que se accede con mayor frecuencia a las mismas páginas, lo que reduce la latencia. Una mayor localidad temporal permite reutilizar los datos cargados en memoria, incrementando la cantidad de hits y disminuyendo las fallas. Por otro lado, una mayor localidad espacial asegura que los datos cercanos estén almacenados en las mismas páginas, lo que minimiza la necesidad de acceder repetidamente a la memoria.

Menor localidad: Si la localidad disminuye, el rendimiento del programa se verá afectado negativamente, ya que aumentarán las fallas debido a que los datos estarán más dispersos. Una baja localidad espacial significa que los datos estarán distribuidos

en distintas páginas, lo que incrementa la frecuencia de accesos a la memoria. Una baja localidad temporal implica que los datos no se utilizan con frecuencia, lo que eleva el número de accesos a la memoria y, en consecuencia, aumenta la latencia del programa.

Diagrama UML