Universidad de los Andes

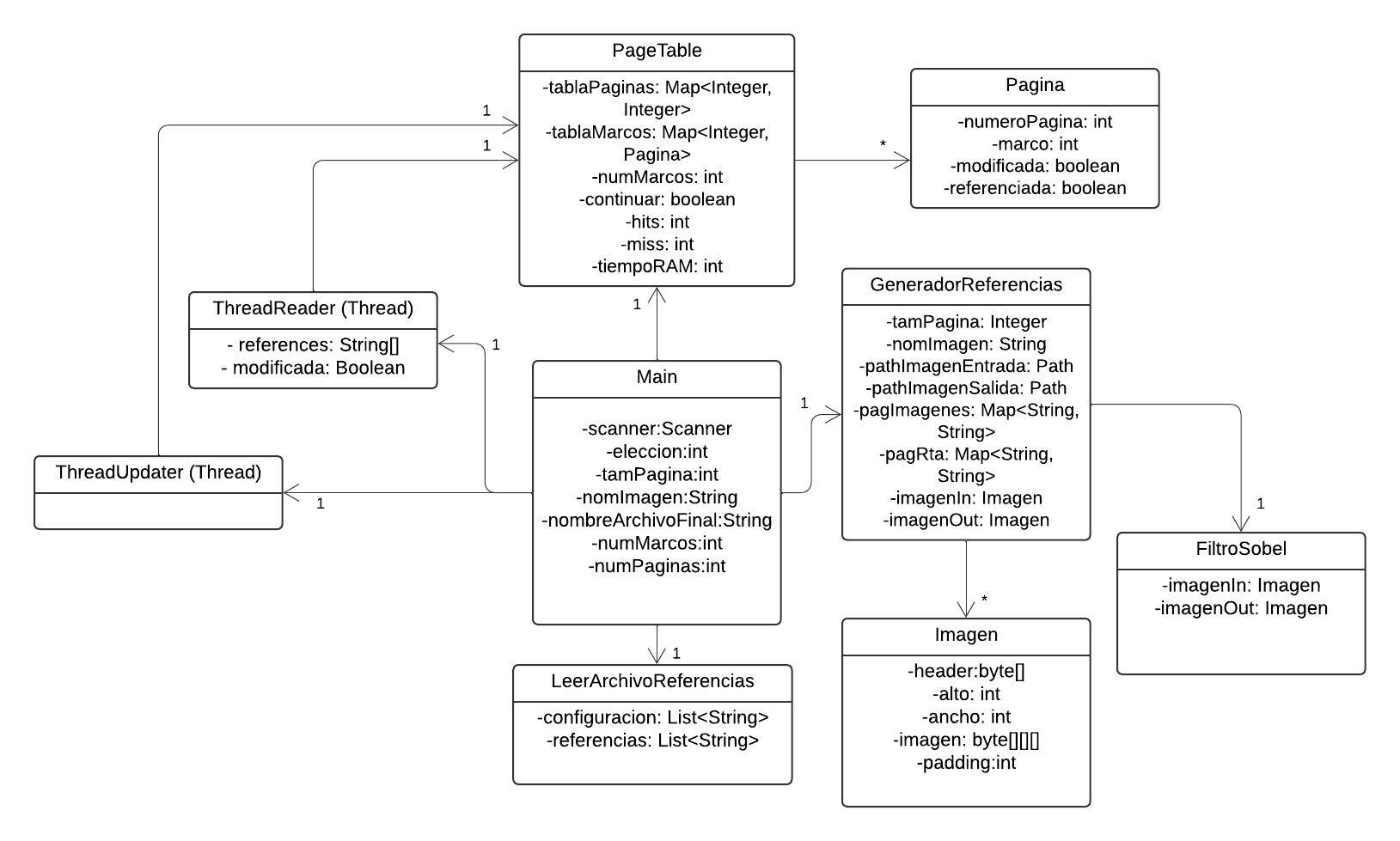
Caso de Estudio 2 – Memoria Virtual – Infraestructura Computacional

Thomas Bonnett – 202225550

Gabriela Zambrano – 202211712

Alejandra Bravo – 201613222

# Diagrama UML



# Descripción del Algoritmo para generar referencias

Para generar las referencias no se utilizó explícitamente un algoritmo especifico. Sin embargo, lo que se realizo fue un análisis del código de applySobel, buscando identificar un patrón en la forma en la que se estaban realizando las referencias. Una vez se identificó esto, se generaron ecuaciones para encontrar el total de las páginas y de las referencias, así como para identificar en que página y desplazamiento iniciaba la referencia de cada estructura. La idea principal era replicar el código que se realiza en applySobel, pero en vez de realizar como las instrucciones, escribir en el archivo cada una de las referencias a las que se accede, implementando contadores para el desplazamiento y la pagina en la que se va. Esto se realizó para las estructuras de imagen, filtroX y filtroY.

Sin embargo, se observó que para la matriz de respuesta esto no se podía realizar así ya que no se referenciaba toda la matriz de respuesta, debido a que no se tenían en cuenta los bordes. Por lo que se generó un diccionario antes de aplicar el algoritmo, teniendo como llave Rta[f][c], donde f y c correspondían a la fila y columna que se estaba accediendo, mientras que el valor era la línea de texto que se iba a escribir en el archivo. Por lo tanto, en el algoritmo general cuando se llegaba al punto de referenciar la respuesta, se buscaba en este diccionario lo que se debía escribir. En resumen, implementando contadores para las páginas y el desplazamiento (el cual se reiniciaba cada que se pasara de página) de cada estructura, en conjunto con una replicación del funcionamiento del algoritmo applySobel se generó el archivo de referencias.

# Descripción de Estructuras de Datos

Para simular el comportamiento del sistema de paginación, se utilizaron dos tablas de hash, en una se representó la tabla de páginas, mientras que en la otra los marcos de página de la RAM. En primer lugar, la tabla de páginas está compuesta por el número de página (llave) y el marco de página (valor). Por otro lado, la tabla de hash para los marcos de página está compuesta por el número del marco (llave) y un objeto llamado Pagina(valor). El objeto Pagina, está compuesto por el numero de la página, el número del marco donde se encuentra esta página y dos atributos booleanos, uno para representar si fue modificada y el otro para representar si se referencio hace poco. Estos dos últimos, se utilizan para manejar el algoritmo de NRU para hacer el remplazo de páginas.

En primer lugar, se hace la verificación si una página solicitada ya está en memoria, para esto se implementa la tabla de páginas, si es así se aumenta el contador de hits y se marca la página como referenciada en la tabla de marcos. Si no es así, se pasa al manejo de fallo de página, en este se verifica si aún hay espacio disponible en la RAM o si toca hacer un remplazo de página. Si aún hay espacio, se asigna un marco a la página, se crea el Objeto Pagina, teniendo referenciado como true y en caso de que se haya escrito sobre la página modificado también se asigna como true, además se agrega la página a la tabla de marcos y de páginas.

No obstante, si la RAM está llena se debe pasar al remplazo de página, cuando esto ocurre se implementa el NRU para ver cuál es la paina candidata a remplazar. Para esto se evalúan todas las páginas que están en los marcos de referencia, con el siguiente orden de prioridad: (no referenciada, no modificada), (no referenciada, modificada), (referenciada, no modificada) y (referenciada, modificada). Una vez identifique alguna página que cumpla con eso, se elimina de la tabla de páginas y en la tabla de marcos se remplaza con la nueva página que está ingresando. Por último, hay una función en la cual se actualiza el estado de todas las páginas en la RAM, para esto se recorren todas las páginas y se cambia el estado de referenciada de cada una a false.

# Esquema de Sincronización

Sincronización en la clase PageTable:

* Los métodos accessPage(), agregarPagina(), manejoFalloPagina(), reemplazarPagLRU(), actualizarEstado() están sincronizados para garantizar que no haya interferencias entre los hilos al acceder o modificar la tabla de páginas o la tabla de marcos. Estos métodos utilizan bloques sincronizados para proteger el acceso a los recursos compartidos y evitar condiciones de carrera.

Hilos (ThreadReader y ThreadUpdater):

* ThreadReader: Lee las referencias de página y accede a la tabla de páginas. Si hay un fallo de página, invoca el método manejoFalloPagina() de forma sincronizada. Además, espera 1 milisegundo cada 10,000 referencias procesadas.
* ThreadUpdater: Ejecuta el método actualizarEstado() en la tabla de páginas cada milisegundo, restableciendo el estado de referencia de las páginas. Este método también se ejecuta de manera sincronizada para evitar modificaciones concurrentes al estado de las páginas.

Interacción entre los hilos:

* Sincronización en ThreadUpdater: Este hilo utiliza synchronized para invocar actualizarEstado() y garantizar que el proceso de restablecer las referencias de las páginas no interfiera con el procesamiento de las referencias de página en ThreadReader.
* Comunicación entre los hilos: El hilo ThreadReader debe detener la ejecución de ThreadUpdater cuando haya terminado de procesar todas las referencias, lo cual está controlado por el estado continuar en la clase PageTable

# Datos recopilados

Se obtuvieron los datos a partir de las simulaciones con las siguientes configuraciones:

* Imágenes imag1.bmp (Ancho 119px Alto 79px), imag2.bmp (Alto 60px Alto 40px)
* Tamaños de página de 512, 1024, 2048 y 4096 bytes
* Marcos de página de 4, 6, 8, 12

Según esto en los resultados se mostraron el número de hits y misses, total de referencias, tiempo de lectura en RAM y SWAP (todo esto respuesta en consola) y un porcentaje de aciertos añadido en el csv creado según las pruebas realizadas.

# Graficas de comportamiento del sistema

Para obtener los gráficos correspondientes se tuvieron en cuenta las configuraciones mencionadas anteriormente, las cuales, al ser ejecutadas en las pruebas, nos dieron los siguientes resultados según imagen y tamaño de página:

|  |  |  |  |
| --- | --- | --- | --- |
| Página de 512B, imag1.bmp | | | |
| Marcos asignados | Total referencias | Hits | Fallas |
| 4 | 756756 | 746554 | 10202 |
| 6 | 756756 | 756644 | 112 |
| 8 | 756756 | 756645 | 111 |
| 12 | 756756 | 756645 | 111 |

|  |  |  |  |
| --- | --- | --- | --- |
| Página de 1024B, imag1.bmp | | | |
| Marcos asignados | Total referencias | Hits | Fallas |
| 4 | 756756 | 753753 | 3003 |
| 6 | 756756 | 756701 | 55 |
| 8 | 756756 | 756701 | 55 |
| 12 | 756756 | 756701 | 55 |

|  |  |  |  |
| --- | --- | --- | --- |
| Página de 2048B, imag1.bmp | | | |
| Marcos asignados | Total referencias | Hits | Fallas |
| 4 | 756756 | 755020 | 1736 |
| 6 | 756756 | 756728 | 28 |
| 8 | 756756 | 756728 | 28 |
| 12 | 756756 | 756728 | 28 |

|  |  |  |  |
| --- | --- | --- | --- |
| Página de 4096B, imag1.bmp | | | |
| Marcos asignados | Total referencias | Hits | Fallas |
| 4 | 756756 | 756568 | 188 |
| 6 | 756756 | 756742 | 14 |
| 8 | 756756 | 756742 | 14 |
| 12 | 756756 | 756742 | 14 |

|  |  |  |  |
| --- | --- | --- | --- |
| Página de 512B, imag2.bmp | | | |
| Marcos asignados | Total referencias | Hits | Fallas |
| 4 | 756756 | 183396 | 1740 |
| 6 | 756756 | 185107 | 29 |
| 8 | 756756 | 185108 | 28 |
| 12 | 756756 | 185108 | 28 |

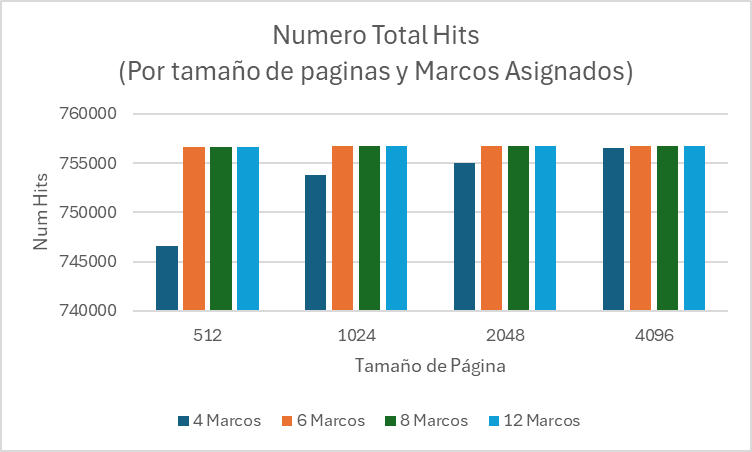
|  |  |  |  |
| --- | --- | --- | --- |
| Página de 1024B, imag2.bmp | | | |
| Marcos asignados | Total referencias | Hits | Fallas |
| 4 | 756756 | 184213 | 923 |
| 6 | 756756 | 185122 | 14 |
| 8 | 756756 | 185122 | 14 |
| 12 | 756756 | 185122 | 14 |

|  |  |  |  |
| --- | --- | --- | --- |
| Página de 2048B, imag2.bmp | | | |
| Marcos asignados | Total referencias | Hits | Fallas |
| 4 | 756756 | 185129 | 7 |
| 6 | 756756 | 185129 | 7 |
| 8 | 756756 | 185129 | 7 |
| 12 | 756756 | 185129 | 7 |

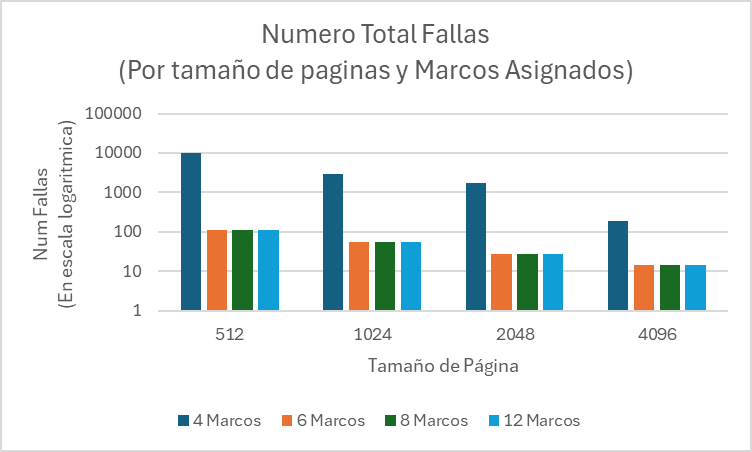
|  |  |  |  |
| --- | --- | --- | --- |
| Página de 4096B, imag2.bmp | | | |
| Marcos asignados | Total referencias | Hits | Fallas |
| 4 | 756756 | 185132 | 4 |
| 6 | 756756 | 185132 | 4 |
| 8 | 756756 | 185132 | 4 |
| 12 | 756756 | 185132 | 4 |

Ya teniendo estos datos recopilados se graficaron el número total de hits y asimismo el número total de cada imagen, teniendo los siguientes resultados:

Imagen 1, Ancho 119px - Alto 79px

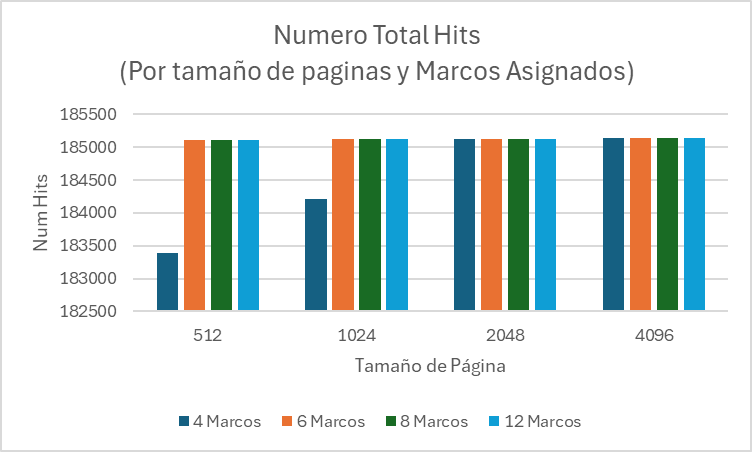


El número total de hits no tiene gran variación según el tamaño de página, ya que como se puede observar se mantiene en una variación entre los 756mil, el único momento donde se puede encontrar una variación del 10% es en un tamaño de página de 512B con 4 marcos asignados, ya que ahí se baja en 10mil el resultado obtenido. Teniendo en cuenta la cercanía entre los distintos resultados obtenidos se puede observar que el número de hits obtenidos al utilizar 4 marcos, pese a ser el que tiene mayor diferencia, tiene un crecimiento a la par del tamaño de página, logrando una cercanía en +/-200 a los demás marcos cuando se utiliza una página de 4096B, mientras que los demás marcos logran una estabilidad entre ellos a partir del tamaño de 1024B.

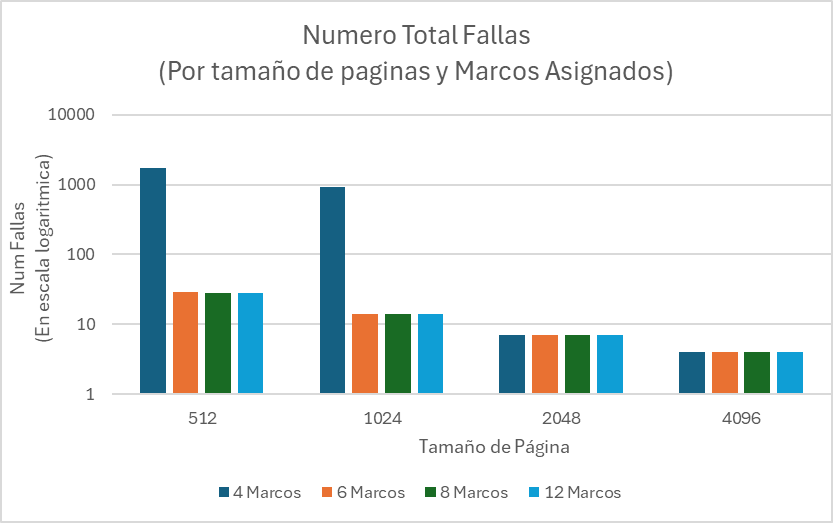


Teniendo en cuenta la gráfica presente se puede observar cómo el número de fallas se mantiene estable entre los marcos de 6 a 12, teniendo una sola variación en el tamaño de página de 512B, donde la diferencia del tamaño 6 a los otros dos es de 1. Al asignar 4 marcos es donde se ve en mayor magnitud el número de fallas, lo cual complementa la gráfica pasada donde se puede ver que a mayor tamaño de página se ve minimizado el error en este marco, aunque a diferencia de los hits, no logra tener una cercanía a los datos obtenidos con los otros marcos como en los resultados dados por los hits, ya que según el tamaño de página se puede observar como las fallas bajan en 50% con el anterior resultado, siendo un decrecimiento estable sin importar el número de marcos.

Imagen 2, Alto 60px Alto 40px



Según la gráfica se puede observar como el número total de hits está considerablemente cercano entre los números de marcos de 6 a 12, ya que el único momento donde hay variación entre ellos es con un tamaño de página de 512B donde la diferencia entre 6 marcos al resto es de 1, logrando todos los marcos una estabilidad en el tamaño de página de 2048B al comenzar a obtener los mismos resultados sin importar el tamaño otorgado o número de marcos.



Considerando la gráfica, se pudo comprobar que el número total de fallas se mantiene estable entre los marcos de 6 a 12. Sin embargo, para el caso de 4 marcos de página, se realizan un total de 1740 fallos de página lo cual sugiere que tener solo 4 marcos implica un alto coste de CPU

En conclusión, se puede ver cómo el tamaño de la imagen afecta los resultados obtenidos, ya que donde mayor diferencia se nota es en la imagen 1 que tiene un tamaño mucho mayor a la 2, requiriendo así un mayor procesamiento y tamaño de página para obtener una cantidad de hits y fallas estables entre marcos.

# Otras configuraciones

Se añadió un archivo de Pruebas que ejecuta automáticamente todos los escenarios requeridos que incluyen 2 imágenes con tamaños distintos, 4 tamaños de página y 4 marcos. Los resultados obtenidos se guardaron en un archivo resultados\_simulacion.csv para después ser utilizados en el análisis de estos datos.

# Interpretación de resultados

Según los resultados obtenidos se puede ver que a medida que sube el valor de marcos se presentan menos fallos de página, aunque se observa que, a partir de cierto punto, el cual oscila entre 6 – 8 marcos, los beneficios dejan de verse. También se puede observar que el tamaño de página se ve relacionado con los fallos debido al número de datos por página y la fragmentación, teniendo así una relación en la cual a menor tamaño de página mayor cantidad de fallos, pero asimismo en cuanto a eficiencia se puede observar que las imágenes pequeñas requieren menor cantidad de marcos. Además, el algoritmo NRU funciona efectivamente cuando se maneja adecuadamente el bit de referencia, evitando así reemplazos costosos, y finalmente se ve que la sincronización con hilos da cierta variación pero que esta misma no altera de manera significativa las conclusiones.

# Problema de localidad: Filtro Sobel

Aplicar el filtro Sobel representa un problema de localidad media, ya que, aunque el algoritmo procesa pequeños bloques de píxeles vecinos (ventana de 3x3), lo que favorece la localidad espacial, el acceso a la memoria no es completamente eficiente en imágenes grandes. A medida que el filtro recorre la imagen, los datos no siempre se encuentran en la memoria caché, lo que puede reducir la localidad temporal. Si la imagen es demasiado grande y no cabe en caché, los accesos a diferentes partes de la imagen serán más dispersos, lo que afecta la eficiencia del sistema. Por lo tanto, la localidad es media porque la estructura de acceso al filtro favorece el acceso cercano, pero se ve afectada por el tamaño de la imagen.