**Jorge David Bustamante Pino – 202210581**

**David Leonardo Almanza Márquez –**

**Juan David Roa Moyano – 202211446**

**Informe caso 2**

A continuación, se encuentra el análisis detallado para la solución para el problema del caso 2:

**Descripción del algoritmo usado para generar las referencias de página (modo uno):**

El algoritmo de generación de referencias de página en la opción 1 está diseñado para registrar el acceso a la imagen y al mensaje escondido en ella, simulando un esquema de paginación. En este modo:

1. **Inicialización del archivo de referencias:** El algoritmo empieza leyendo la longitud del mensaje escondido en la imagen, usando el método *leerLongitud()*. Después, genera un archivo llamado referencias.txt, donde se guardan metadatos clave como el tamaño de página (TP), el número de filas (NF), el número de columnas (NC), el número de páginas requeridas (NP), y el número de referencias generadas (NR). Esto simula la división del espacio de memoria en páginas de un tamaño específico.
2. **Referencias al leer la longitud del mensaje:** Al leer la longitud del mensaje escondido en los primeros 16 bits de la imagen, el método *leerLongitud2()* genera referencias a la imagen, registrando el acceso a cada byte donde se encuentra cada bit del mensaje. Para cada acceso, el programa calcula la página virtual y el desplazamiento dentro de esa página, y lo escribe en el archivo de referencias.
3. **Referencias al recuperar el mensaje escondido:** Luego de obtener la longitud, el algoritmo procede a recuperar el mensaje escondido, bit a bit, y genera referencias para cada uno de los caracteres del mensaje. En el método *recuperar2()*, se realiza un registro detallado de los accesos a la imagen y al mensaje en el archivo de referencias. Cada acceso está descrito por la página virtual, el desplazamiento dentro de la página, el canal de color (RGB) en la imagen, y si el acceso fue de lectura o escritura.

El objetivo del algoritmo es simular cómo se manejarían las referencias de memoria en un sistema con paginación, usando la imagen y el mensaje como datos almacenados.

**Descripción de las estructuras de datos usadas para simular el comportamiento del sistema de paginación:**

1. **marco (CopyOnWriteArrayList<Integer>):** Es una lista de enteros que simula los marcos de página en la memoria física. Cada posición de esta lista representa un marco de página, y su valor es el número de la página virtual que está cargada en ese marco. Si una posición contiene -1, significa que el marco está vacío. El tamaño de esta lista depende del número de marcos de página (num\_marcospag), que se le solicita al usuario en la opción dos.
   * **Cuando se actualiza:** La lista marco se actualiza cuando ocurre una falla de página. Si una página virtual no está presente en la lista de marcos, el sistema debe cargarla, reemplazando alguna página existente.
   * **Actualización:** Cuando se detecta una falta de página, el algoritmo selecciona un marco de página para reemplazar el contenido (la política de reemplazo no está completamente especificada en este fragmento de código, pero podría ser FIFO, LRU, etc.).
2. **bitR (ConcurrentHashMap<Integer, Integer>):** Este es un diccionario que se utiliza para llevar un registro de los bits de referencia asociados con las páginas virtuales. Cada entrada del ConcurrentHashMap tiene como clave el número de página virtual y como valor su bit de referencia (0 o 1). El bit de referencia indica si una página ha sido utilizada recientemente, lo que es útil en algoritmos de reemplazo como el de segunda oportunidad (Second Chance).
   * **Cuando se actualiza:** Se actualiza cada vez que se accede a una página virtual. Si una página es accedida (leída o escrita), se actualiza su bit de referencia a 1.
   * **Actualización:** Si una página es referenciada, su bit en bitR se marca como 1, indicando que ha sido accedida recientemente. Esto ayuda al algoritmo a decidir qué páginas reemplazar en el caso de una falta de página.

Estas estructuras permiten simular el comportamiento de la memoria paginada: marco representa los marcos en la memoria física, y bitR controla el uso reciente de las páginas para optimizar el reemplazo.

**Esquema de sincronización usado y su justificación:**

En el código proporcionado, la sincronización se lleva a cabo utilizando bloques de sincronización para controlar el acceso a los recursos compartidos entre múltiples hilos. Los dos recursos principales que se protegen con este mecanismo son la lista de marcos de página (marco) y el mapa que almacena los bits de referencia (bitR). El objetivo de la sincronización es prevenir condiciones de carrera, donde múltiples hilos podrían acceder y modificar simultáneamente los mismos recursos, lo que podría resultar en comportamientos inesperados o inconsistencias en los datos.

El esquema de sincronización se utiliza en los siguientes puntos clave del código:

1. **En el método *gestionarReferencias()***: Aquí se utiliza un bloque de sincronización para asegurar que el acceso y la modificación de los recursos compartidos marco y bitR sean seguros en un entorno multihilo. En este método, el hilo que gestiona las referencias de las páginas puede encontrar un hit (cuando la página ya está en memoria) o una falla de página (cuando la página no está presente). En ambos casos, se realiza una modificación de los recursos compartidos: en el caso de un hit, se actualiza el bit R de la página; y en el caso de una falla, se realiza un reemplazo de una página en uno de los marcos. Dado que otro hilo podría estar accediendo a estos mismos recursos, por ejemplo, envejeciendo los bits de referencia o gestionando otras páginas al mismo tiempo, es esencial asegurar que estos accesos estén sincronizados para evitar que dos hilos modifiquen los mismos datos al mismo tiempo, lo que podría causar inconsistencias en el estado de los marcos de página o en los bits de referencia.
2. **En el método *envejecimientoBitR()***: Este método también utiliza la sincronización para proteger el acceso al mapa bitR mientras envejece los bits de referencia. El proceso de envejecimiento implica que el bit R de cada página se pone a 0 después de un cierto periodo de tiempo, simulando que las páginas no han sido referenciadas recientemente. Dado que este hilo puede estar ejecutándose al mismo tiempo que otro hilo está gestionando las referencias de las páginas y actualizando los bits R, es necesario sincronizar el acceso al mapa bitR para evitar que ambos hilos intenten modificar el mismo bit R al mismo tiempo. Si esto no se sincronizara, se podrían producir inconsistencias en los valores de los bits, como por ejemplo que una página sea marcada como no referenciada justo después de haber sido actualizada por otro hilo.
3. **En el método *actualizarR()***: Este método es invocado cuando una página es referenciada, y su función es actualizar el bit R de dicha página a 1. El acceso al mapa bitR está sincronizado para asegurar que esta operación se realice de manera segura en un entorno concurrente. Si varios hilos intentaran actualizar simultáneamente el bit R de diferentes páginas sin ningún tipo de sincronización, se podrían perder actualizaciones o corromper los datos del mapa. La palabra clave synchronized garantiza que solo un hilo pueda acceder al mapa bitR en un momento dado, preservando así la coherencia de los datos.
4. **En los métodos *getHits()* y *getFallas()*:** En estos métodos se utiliza la sincronización para que los datos a obtener no sean inconsistentes (hits y fallas).

**Tablas con los datos recopilados**

**Imagen con tamaño 450 x 299**

|  |  |  |  |
| --- | --- | --- | --- |
| Página de 512, gatoConMensaje\_100.bmp, mensaje: 109 caracteres | | | |
| **Marcos Asignados** | **Total referencias** | **Hits** | **Fallas** |
| 4 | 1869 | 1866 | 3 |
| 8 | 1869 | 1866 | 3 |

|  |  |  |  |
| --- | --- | --- | --- |
| Página de 1024, gatoConMensaje\_100.bmp, mensaje: 109 caracteres | | | |
| **Marcos Asignados** | **Total referencias** | **Hits** | **Fallas** |
| 4 | 1869 | 1867 | 2 |
| 8 | 1869 | 1867 | 2 |

|  |  |  |  |
| --- | --- | --- | --- |
| Página de 2048, gatoConMensaje\_100.bmp, mensaje: 109 caracteres | | | |
| **Marcos Asignados** | **Total referencias** | **Hits** | **Fallas** |
| 4 | 1869 | 1867 | 2 |
| 8 | 1869 | 1867 | 2 |

|  |  |  |  |
| --- | --- | --- | --- |
| Página de 512, gatoConMensaje\_1000.bmp, mensaje: 1001 caracteres | | | |
| **Marcos Asignados** | **Total referencias** | **Hits** | **Fallas** |
| 4 | 17033 | 17014 | 19 |
| 8 | 17033 | 17014 | 19 |

|  |  |  |  |
| --- | --- | --- | --- |
| Página de 1024, gatoConMensaje\_1000.bmp, mensaje: 1001 caracteres | | | |
| **Marcos Asignados** | **Total referencias** | **Hits** | **Fallas** |
| 4 | 17033 | 17023 | 10 |
| 8 | 17033 | 17023 | 10 |

|  |  |  |  |
| --- | --- | --- | --- |
| Página de 2048, gatoConMensaje\_1000.bmp, mensaje: 1001 caracteres | | | |
| **Marcos Asignados** | **Total referencias** | **Hits** | **Fallas** |
| 4 | 17033 | 17028 | 5 |
| 8 | 17033 | 17028 | 5 |

|  |  |  |  |
| --- | --- | --- | --- |
| Página de 512, gatoConMensaje\_2000.bmp, mensaje: 2021 caracteres | | | |
| **Marcos Asignados** | **Total referencias** | **Hits** | **Fallas** |
| 4 | 34373 | 34336 | 37 |
| 8 | 34373 | 34335 | 38 |

|  |  |  |  |
| --- | --- | --- | --- |
| Página de 1024, gatoConMensaje\_2000.bmp, mensaje: 2021 caracteres | | | |
| **Marcos Asignados** | **Total referencias** | **Hits** | **Fallas** |
| 4 | 34373 | 34354 | 19 |
| 8 | 34373 | 34354 | 19 |

|  |  |  |  |
| --- | --- | --- | --- |
| Página de 2048, gatoConMensaje\_2000.bmp, mensaje: 2021 caracteres | | | |
| **Marcos Asignados** | **Total referencias** | **Hits** | **Fallas** |
| 4 | 34373 | 34363 | 10 |
| 8 | 34373 | 34363 | 10 |

|  |  |  |  |
| --- | --- | --- | --- |
| Página de 512, gatoConMensaje\_4000.bmp, mensaje: 4097 caracteres | | | |
| **Marcos Asignados** | **Total referencias** | **Hits** | **Fallas** |
| 4 | 69665 | 69590 | 75 |
| 8 | 69665 | 69591 | 74 |

|  |  |  |  |
| --- | --- | --- | --- |
| Página de 1024, gatoConMensaje\_4000.bmp, mensaje: 4097 caracteres | | | |
| Marcos Asignados | **Total referencias** | **Hits** | **Fallas** |
| 4 | 69665 | 69627 | 38 |
| 8 | 69665 | 69627 | 38 |

|  |  |  |  |
| --- | --- | --- | --- |
| Página de 2048, gatoConMensaje\_4000.bmp, mensaje: 4097 caracteres | | | |
| **Marcos Asignados** | **Total referencias** | **Hits** | **Fallas** |
| 4 | 69665 | 69645 | 20 |
| 8 | 69665 | 69645 | 20 |

|  |  |  |  |
| --- | --- | --- | --- |
| Página de 512, gatoConMensaje\_8000.bmp, mensaje: 8167 caracteres | | | |
| **Marcos Asignados** | **Total referencias** | **Hits** | **Fallas** |
| 4 | 138855 | 138710 | 145 |
| 8 | 138855 | 138708 | 147 |

|  |  |  |  |
| --- | --- | --- | --- |
| Página de 1024, gatoConMensaje\_8000.bmp, mensaje: 8167 caracteres | | | |
| **Marcos Asignados** | **Total referencias** | **Hits** | **Fallas** |
| 4 | 138855 | 138781 | 74 |
| 8 | 138855 | 138782 | 73 |

|  |  |  |  |
| --- | --- | --- | --- |
| Página de 2048, gatoConMensaje\_8000.bmp, mensaje: 8167 caracteres | | | |
| **Marcos Asignados** | **Total referencias** | **Hits** | **Fallas** |
| 4 | 138855 | 138818 | 37 |
| 8 | 138855 | 138817 | 38 |

**Imagen con tamaño 270 x 180**

|  |  |  |  |
| --- | --- | --- | --- |
| Página de 512, pikachuConMensaje\_100.bmp, mensaje: 109 caracteres | | | |
| **Marcos Asignados** | **Total referencias** | **Hits** | **Fallas** |
| 4 | 1869 | 1866 | 3 |
| 8 | 1869 | 1866 | 3 |

|  |  |  |  |
| --- | --- | --- | --- |
| Página de 1024, pikachuConMensaje\_100.bmp, mensaje: 109 caracteres | | | |
| **Marcos Asignados** | **Total referencias** | **Hits** | **Fallas** |
| 4 | 1869 | 1867 | 2 |
| 8 | 1869 | 1867 | 2 |

|  |  |  |  |
| --- | --- | --- | --- |
| Página de 2048, pikachuConMensaje\_100.bmp, mensaje: 109 caracteres | | | |
| **Marcos Asignados** | **Total referencias** | **Hits** | **Fallas** |
| 4 | 1869 | 1867 | 2 |
| 8 | 1869 | 1867 | 2 |

|  |  |  |  |
| --- | --- | --- | --- |
| Página de 512, pikachuConMensaje\_1000.bmp, mensaje: 1001 caracteres | | | |
| **Marcos Asignados** | **Total referencias** | **Hits** | **Fallas** |
| 4 | 17033 | 17014 | 19 |
| 8 | 17033 | 17014 | 19 |

|  |  |  |  |
| --- | --- | --- | --- |
| Página de 1024, pikachuConMensaje\_1000.bmp, mensaje: 1001 caracteres | | | |
| **Marcos Asignados** | **Total referencias** | **Hits** | **Fallas** |
| 4 | 17033 | 17023 | 10 |
| 8 | 17033 | 17023 | 10 |

|  |  |  |  |
| --- | --- | --- | --- |
| Página de 2048, pikachuConMensaje\_1000.bmp, mensaje: 1001 caracteres | | | |
| **Marcos Asignados** | **Total referencias** | **Hits** | **Fallas** |
| 4 | 17033 | 17028 | 5 |
| 8 | 17033 | 17028 | 5 |

|  |  |  |  |
| --- | --- | --- | --- |
| Página de 512, pikachuConMensaje\_2000.bmp, mensaje: 2021 caracteres | | | |
| **Marcos Asignados** | **Total referencias** | **Hits** | **Fallas** |
| 4 | 34373 | 34334 | 39 |
| 8 | 34373 | 34335 | 38 |

|  |  |  |  |
| --- | --- | --- | --- |
| Página de 1024, pikachuConMensaje\_2000.bmp, mensaje: 2021 caracteres | | | |
| **Marcos Asignados** | **Total referencias** | **Hits** | **Fallas** |
| 4 | 34373 | 34354 | 19 |
| 8 | 34373 | 34354 | 19 |

|  |  |  |  |
| --- | --- | --- | --- |
| Página de 2048, pikachuConMensaje\_2000.bmp, mensaje: 2021 caracteres | | | |
| **Marcos Asignados** | **Total referencias** | **Hits** | **Fallas** |
| 4 | 34373 | 34363 | 10 |
| 8 | 34373 | 34363 | 10 |

|  |  |  |  |
| --- | --- | --- | --- |
| Página de 512, pikachuConMensaje\_4000.bmp, mensaje: 4097 caracteres | | | |
| **Marcos Asignados** | **Total referencias** | **Hits** | **Fallas** |
| 4 | 69665 | 69588 | 77 |
| 8 | 69665 | 69590 | 75 |

|  |  |  |  |
| --- | --- | --- | --- |
| Página de 1024, pikachuConMensaje\_4000.bmp, mensaje: 4097 caracteres | | | |
| **Marcos Asignados** | **Total referencias** | **Hits** | **Fallas** |
| 4 | 69665 | 69627 | 38 |
| 8 | 69665 | 69627 | 38 |

|  |  |  |  |
| --- | --- | --- | --- |
| Página de 2048, pikachuConMensaje\_4000.bmp, mensaje: 4097 caracteres | | | |
| **Marcos Asignados** | **Total referencias** | **Hits** | **Fallas** |
| 4 | 69665 | 69645 | 20 |
| 8 | 69665 | 69645 | 20 |

|  |  |  |  |
| --- | --- | --- | --- |
| Página de 512, pikachuConMensaje\_8000.bmp, mensaje: 8167 caracteres | | | |
| **Marcos Asignados** | **Total referencias** | **Hits** | **Fallas** |
| 4 | 138855 | 138708 | 147 |
| 8 | 138855 | 138705 | 150 |

|  |  |  |  |
| --- | --- | --- | --- |
| Página de 1024, pikachuConMensaje\_8000.bmp, mensaje: 8167 caracteres | | | |
| **Marcos Asignados** | **Total referencias** | **Hits** | **Fallas** |
| 4 | 138855 | 138782 | 73 |
| 8 | 138855 | 138781 | 74 |

|  |  |  |  |
| --- | --- | --- | --- |
| Página de 2048, pikachuConMensaje\_8000.bmp, mensaje: 8167 caracteres | | | |
| **Marcos Asignados** | **Total referencias** | **Hits** | **Fallas** |
| 4 | 138855 | 138818 | 37 |
| 8 | 138855 | 138818 | 37 |

**Graficas del comportamiento del sistema**

**Graficas de imagen con tamaño 450 x 299 pixeles**

**Graficas de imagen con tamaño 270 x 180 pixeles**

**Graficas de tiempos**

**Graficas de imagen con tamaño 450 x 299 pixeles**

**Graficas de imagen con tamaño 270 x 180 pixeles**

**Análisis de resultados**

Los resultados obtenidos muestran una tendencia clara en el comportamiento del sistema de paginación:

**Efecto del Tamaño de Página:**

A medida que aumenta el tamaño de la página, el número de fallas de página tiende a disminuir. Esto se debe a que un mayor tamaño de página permite almacenar más datos en cada página, reduciendo la frecuencia con la que se necesita acceder al almacenamiento secundario.

Sin embargo, el tiempo de ejecución no muestra una variación significativa con el cambio en el tamaño de la página. Esto sugiere que el tiempo de acceso a la memoria es relativamente constante, independientemente del tamaño de la página.

**Efecto del Número de Marcos Asignados:**

El número de marcos asignados tiene un impacto directo en el número de fallas de página. Un mayor número de marcos permite almacenar más páginas en la memoria física, reduciendo la necesidad de reemplazar páginas y, por lo tanto, disminuyendo el número de fallas de página.

Sin embargo, en algunos casos, el aumento del número de marcos no resulta en una reducción significativa de las fallas de página. Esto puede deberse a que el tamaño del mensaje y el tamaño de la página ya están optimizados para el número de marcos asignados, y cualquier aumento adicional en los marcos no proporciona un beneficio adicional.

**Comparación entre Diferentes Mensajes:**

Los mensajes más largos tienden a generar más fallas de página y tiempos de ejecución más largos. Esto es esperado, ya que un mensaje más largo requiere más páginas para ser almacenado y accedido, aumentando la probabilidad de fallas de página y el tiempo necesario para acceder a todos los datos.

En conclusión, los resultados obtenidos corresponden a las expectativas basadas en el comportamiento teórico de un sistema de paginación. El aumento del tamaño de la página y el número de marcos asignados tienden a reducir el número de fallas de página, aunque el impacto en el tiempo de ejecución es menos pronunciado. Los mensajes más largos generan más fallas de página y tiempos de ejecución más largos, lo cual es consistente con la teoría.

Estos resultados destacan la importancia de optimizar tanto el tamaño de la página como el número de marcos asignados para mejorar el rendimiento de un sistema de paginación. Además, sugieren que, en algunos casos, aumentar el número de marcos más allá de un cierto punto puede no proporcionar beneficios adicionales significativos, y que el tamaño del mensaje y el tamaño de la página deben ser considerados conjuntamente para lograr una optimización efectiva.

**Localidad del Problema**

Para abordar cómo variarían los resultados del problema manejado en el informe si la localidad fuera diferente, es importante entender primero qué es la localidad en el contexto de los sistemas de paginación y cómo afecta el rendimiento del sistema. La localidad se refiere a la tendencia de un programa a acceder a un conjunto limitado de páginas de memoria en un periodo corto de tiempo. Existen dos tipos principales de localidad: localidad temporal y localidad espacial.

**Localidad Temporal y Espacial**

**Localidad Temporal:** Si una página de memoria ha sido accedida recientemente, es probable que sea accedida de nuevo en un futuro cercano.

**Localidad Espacial:** Si una página de memoria ha sido accedida, es probable que las páginas cercanas en la memoria también sean accedidas pronto.

**Impacto de la Localidad en el Sistema de Paginación**

El rendimiento de un sistema de paginación está fuertemente influenciado por la localidad. Un alto grado de localidad puede reducir significativamente el número de fallos de página, ya que las páginas necesarias ya están en la memoria física. Por otro lado, una baja localidad puede aumentar los fallos de página, ya que el sistema necesita cargar y descargar páginas con mayor frecuencia.

**Localidad Mayor**

Si la localidad del problema manejado fuera mayor, es decir, si el acceso a las páginas de memoria fuera más concentrado en un conjunto reducido de páginas, los resultados variarían de la siguiente manera:

**Reducción de Fallos de Página:** Con una mayor localidad, el número de fallos de página disminuiría. Esto se debe a que las páginas que se necesitan con frecuencia ya estarían en la memoria física, reduciendo la necesidad de cargar nuevas páginas desde el disco.

**Mejora en el Rendimiento:** Menos fallos de página significan menos interrupciones en la ejecución del programa, lo que lleva a un mejor rendimiento general. El tiempo de ejecución del programa se reduciría, ya que el sistema pasaría menos tiempo gestionando fallos de página y más tiempo ejecutando instrucciones útiles.

**Eficiencia en el Uso de la Memoria:** La memoria física sería utilizada de manera más eficiente, ya que las páginas cargadas en la memoria serían las que realmente se necesitan. Esto reduciría el desperdicio de memoria y mejoraría la eficiencia del sistema.

**Menor Necesidad de Políticas de Reemplazo Complejas:** Con una mayor localidad, las políticas de reemplazo de páginas (como FIFO, LRU, etc.) serían menos críticas, ya que las páginas necesarias estarían en la memoria con mayor frecuencia. Esto simplificaría la gestión de la memoria y reduciría la sobrecarga asociada con la implementación de políticas de reemplazo complejas.

**Localidad Menor**

Si la localidad del problema manejado fuera menor, es decir, si el acceso a las páginas de memoria fuera más disperso y menos predecible, los resultados variarían de la siguiente manera:

**Aumento de Fallos de Página:** Con una menor localidad, el número de fallos de página aumentaría. Esto se debe a que las páginas necesarias no estarían en la memoria física con la misma frecuencia, lo que obligaría al sistema a cargar nuevas páginas desde el disco con mayor frecuencia.

**Degradación del Rendimiento:** Más fallos de página significan más interrupciones en la ejecución del programa, lo que lleva a una degradación del rendimiento general. El tiempo de ejecución del programa aumentaría, ya que el sistema pasaría más tiempo gestionando fallos de página y menos tiempo ejecutando instrucciones útiles.

**Ineficiencia en el Uso de la Memoria:** La memoria física sería utilizada de manera menos eficiente, ya que las páginas cargadas en la memoria no serían las que realmente se necesitan con la misma frecuencia. Esto aumentaría el desperdicio de memoria y reduciría la eficiencia del sistema.

**Mayor Necesidad de Políticas de Reemplazo Complejas:** Con una menor localidad, las políticas de reemplazo de páginas serían más críticas, ya que el sistema necesitaría decidir con mayor frecuencia qué páginas reemplazar. Esto aumentaría la sobrecarga asociada con la implementación de políticas de reemplazo complejas y podría requerir algoritmos más sofisticados para gestionar la memoria de manera eficiente.

En resumen, la localidad tiene un impacto significativo en el rendimiento de un sistema de paginación. Una mayor localidad mejora el rendimiento al reducir los fallos de página y aumentar la eficiencia de la memoria, mientras que una menor localidad degrada el rendimiento al aumentar los fallos de página y reducir la eficiencia de la memoria. Por lo tanto, es crucial diseñar algoritmos y estructuras de datos que maximicen la localidad para mejorar el rendimiento del sistema.