CASO 2 - VIRTUALIZACIÓN

Universidad de los Andes Departamento de Ingeniería de Sistemas y Computación
ISIS 2203 Infraestructura Computacional
202220

Andrés Felipe Guerrero Sarmiento 202015143
Juan José Osorio Calad 202021720
Julián Camilo Mora Valbuena 202012747

Contenido

1. Contexto del problema

La memoria RAM es un recurso limitado que debe administrarse con cuidado para garantizar el avance en la ejecución de los procesos creados. En este contexto surge el concepto de memoria virtual, el cual ofrece varias ventajas: independencia de direcciones físicas, posibilidad de compartir memoria y de correr programas más grandes que la memoria física que se les asigna. Queremos entender mejor cómo varía el tiempo de ejecución de un proceso de acuerdo con las características de un sistema de memoria virtual, en particular, TLB y espacio asignado en RAM.

Para poder comenzar con el proceso de diseño, primero fue necesario conocer cómo se lleva a cabo las relaciones que existen entre la RAM, la TLB (Translation Lookaside Buffer), y la TP (tabla de páginas). y, por lo tanto, entender cómo varía el tiempo de ejecución de un proceso de acuerdo con las características de un sistema de memoria virtual.

2. Diagrama de clases

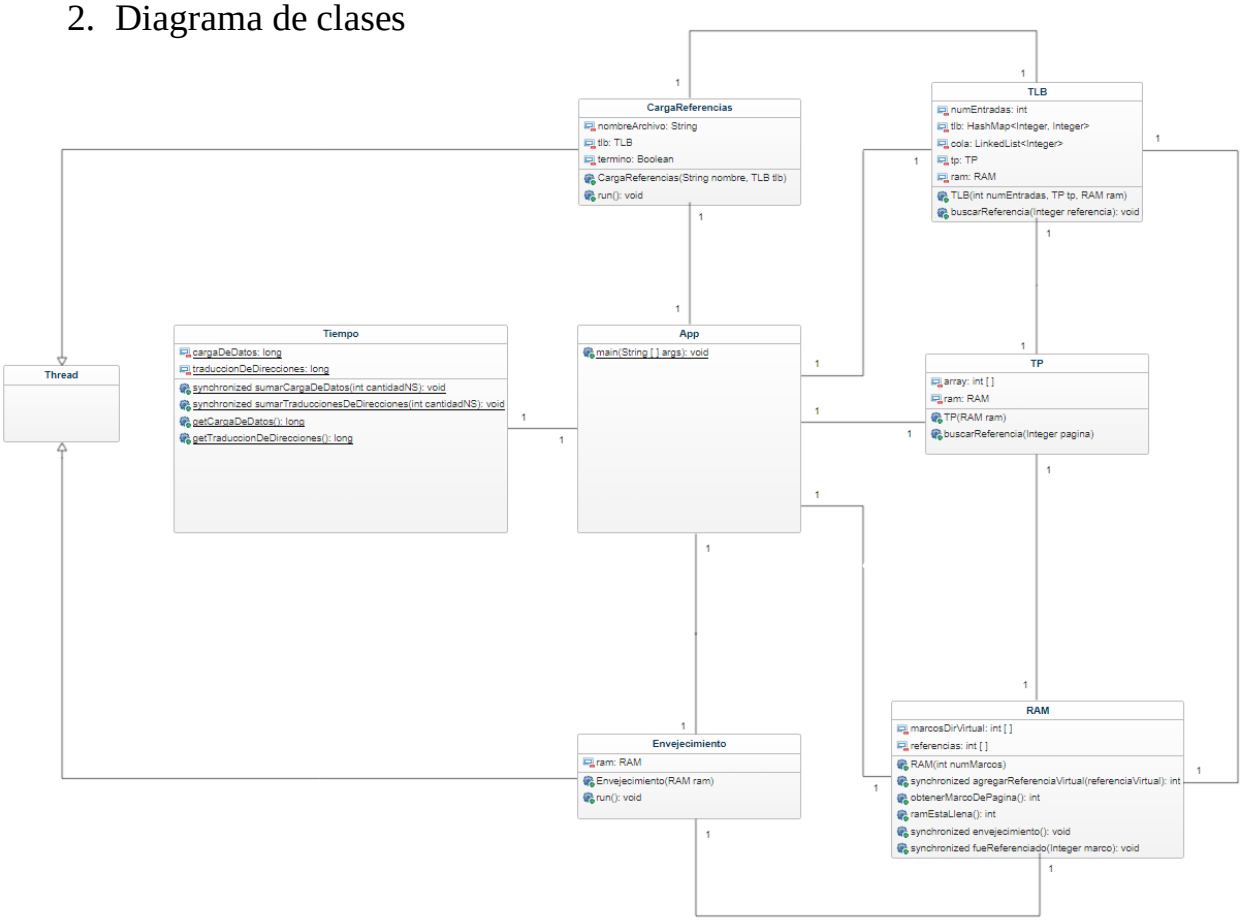


Figura 1: Diagrama de Clases

2.1. Descripción del diagrama

El diagrama de clases refleja el diseño y la distribución utilizada para trabajar el caso. Como se puede observar en la figura 1, el diagrama está compuesto por 7 clases principales.

En primer lugar, la clase App contiene el método main, en el cual se inician los threads, los componentes del sistema de memoria virtual y el espacio asignado en RAM. Siguiendo con el hilo, cada uno de los dos threads tiene una función diferente, uno se encarga de cargar las referencias y actualizar el estado de la TLB, por ello, se vincula con la TLB. El otro, del algoritmo de envejecimiento, indica el estado de los marcos de página, para reemplazar la página que tenga el número más pequeño en su contador, lo cual indica que es página menos usada frecuentemente. Cabe aclarar que, como ambos son threads deben heredar de la clase Thread de Java. Finalmente, la clase Tiempo tiene la responsabilidad de mantener dos variables que van agregando el valor del tiempo en nanosegundos a la carga de datos o direcciones, según corresponda. Esta clase tiene todos sus atributos y métodos *static*, por lo que no se instancia en ninguna otra clase, solo se conecta con el main para su carga e imprimir los resultados.

Adicionalmente, decidimos modelar el programa dedicando una clase a cada componente del sistema de memoria virtual. Primero, la TLB, la cual es el primer lugar en el que se revisa si una dirección está o no. Así, si una dirección se encuentra en la TLB, se realiza su correspondiente carga de datos de la memoria RAM, de lo contrario, se busca en la Tabla de páginas la referencia, por tal motivo, la TLB está conectada con la RAM y la TP.

Segundo, la Tabla de páginas guarda las referencias a buscar e indica si tienen un marco de página asignado. Si lo tienen, se carga el dato desde la RAM, de lo contrario se da un fallo de página y posteriormente se actualiza su valor. Debido a lo anterior, esta clase se debe conectar con la TLB y la RAM. Tercero, en la RAM tenemos los marcos de página con la dirección virtual y física. Si hay un fallo de página, este se resuelve acá, con la respectiva ayuda del algoritmo de envejecimiento, por lo cual, esta clase también va conectada con dicho thread.

3. Descripción de la solución

Cuando el programa comienza a ejecutarse lo primero que ocurre es que se le pide al usuario el número de entradas de la TLB, el número de marcos de la RAM y el nombre del archivo que se encuentra en la carpeta Caso-2-Infracomp-ISIS2203/src/ej_paginas/. Al recibir esta información, se crean las instancias de RAM, TP, TLB, el thread que carga las referencias y el thread de envejecimiento y se ejecuta el método start() sobre los últimos mencionados.

El thread de la clase CargaReferencias lee las referencias del archivo de texto cada dos milisegundos. Al leerla ejecuta el método buscarReferencia() sobre la instancia de la TLB. Si la referencia se encuentra en la TLB se suman 2 nanosegundos al tiempo de traducción de direcciones y 30 nanosegundos al

tiempo de la carga de datos. Además, se le notifica a la RAM que aquel marco fue referenciado (utilizando el método fueReferenciado() sobre la instancia de RAM) donde se le agrega un 1 a los bits.

Si la referencia no se encuentra en la TLB se basa a buscar en la TP a través del método buscarReferencia(). Si la página se encuentra en la TP se retorna el marco de página que tiene asignado y se agrega en la TLB mediante la estructura FIFO. Si la referencia de la página no se encuentra en la TP, ocurre un fallo de página y se llama al método agregarReferenciaVirtual() de la instancia de RAM. En este método se revisa si hay marcos de página disponibles. Si los hay, se le asigna el primer marco de página disponible a la referencia de la página. Por otro lado, si no hay marcos de página disponibles se reemplaza el que haya sido referenciado menos veces teniendo en cuenta los valores que quedan gracias al algoritmo de envejecimiento.

De manera concurrente, el thread de envejecimiento ejecuta el método envejecimiento () de la instancia de RAM cada milisegundo. En este método el thread hace un shift hacia la derecha de un bit para cada marco de página.

3.1. Estructuras usadas

TLB:

La TLB está compuesta por los siguientes atributos:

- numEntradas: Número de entradas de la TLB.
- tlb: HashMap de enteros donde la llave es la referencia (página) y el valor es su marco de página respectivo.
- cola: LinkedList que representa la estructura FIFO para saber qué referencia debe ser sacada de la TLB.
- tp: Tabla de páginas del programa.
- ram: Instancia de RAM del programa.

TP:

La TP está compuesta por los siguientes atributos:

- tp: Arreglo de 64 enteros que representa las páginas y su marco de página asignado; es decir, el índice del arreglo representa la referencia y el valor dentro del índice es su marco de página asignado. Si no tiene un marco de página entonces es –1.

RAM:

La RAM está compuesta por los siguientes atributos:

- marcosDirVirtual: Arreglo donde el índice representa el marco de página y el valor es la página (referencia) que tiene a ese marco asignado.
- referencias: Arreglo de enteros donde el índice representa el marco de página y el valor es un entero de 32 bits que es utilizado para el algoritmo de envejecimiento y para decidir que marco será liberado.

CargaReferencias:

El thread de CargaReferencias tiene los siguientes atributos:

- nombreArchivo: Nombre del archivo de texto que contiene las referencias.
- tlb: Instancia de TLB del programa.

Envejecimiento:

El thread de Envejecimiento tiene los siguientes atributos:

- ram: Instancia de RAM del programa.

Tiempo

La clase Tiempo tiene los siguientes atributos:

- cargaDeDatos: Tiempo de la carga de datos.
- TraducciónDeDirecciones: Tiempo de la traducción de direcciones.

3.2. Esquema de sincronización

El esquema de sincronización del programa puede ser visto en la clase RAM. En esta clase, los métodos agregarReferenciaVirtual(), envejecimiento() y fueReferenciado() son sincronizados puesto que estos comparten datos que pueden alterar la ejecución del programa por la ejecución de los dos Threads. Esta sincronización busca evitar que se ejecute el algoritmo de envejecimiento cuando están ocurriendo cambios en la RAM.

4. Resultados de simulaciones

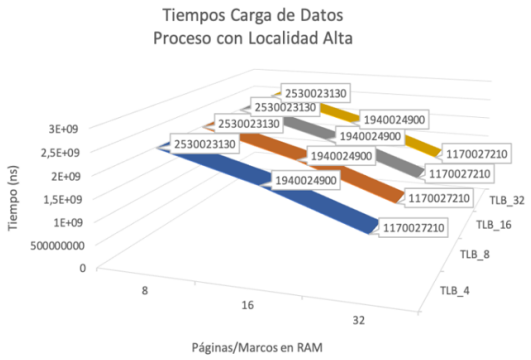
Se realizaron todas las pruebas correspondientes a las diferentes combinaciones entre el tamaño de la TLB y el tamaño de RAM con los archivos de Referencias Alta y Referencias Baja.

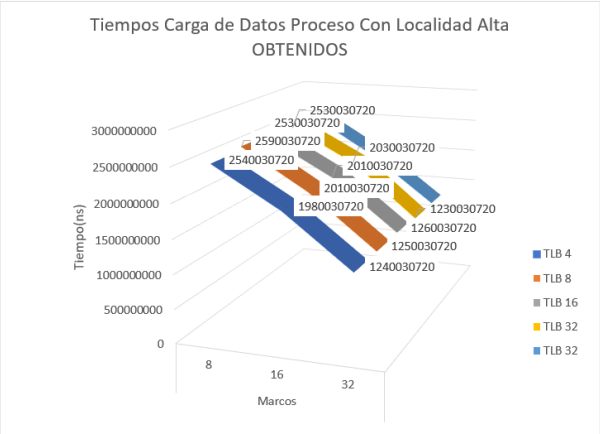
4.1. Referencias Alta

Se realizo la simulación el archivo *ej_Alta_64paginas.txt* y se obtuvieron los siguientes tiempos de Carga de datos (ns):

		TLB				
		4	**8**	**16**	**32**	**64**
Marcos	**8**	2540030720	2590030720	2530030720	2530030720	2530030720
	16	1980030720	2010030720	2010030720	2030030720	1960030720
	32	1240030720	1250030720	1260030720	1230030720	1240030720

Ahora se muestran 2 graficas, la primera es de los tiempos de carga de datos propuestos y la segunda es el tiempo de carga de datos que se obtuvieron al ejecutar la simulación.

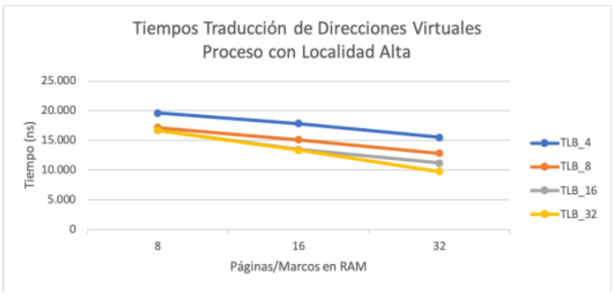


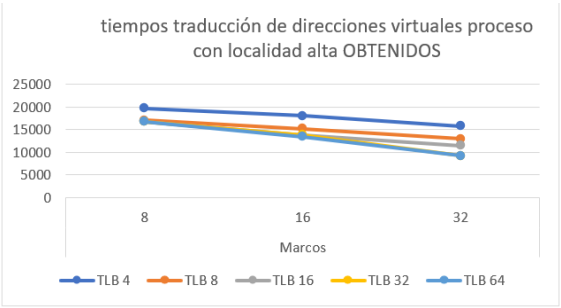


Se realizo la simulación el archivo *ej_Alta_64paginas.txt* y se obtuvo los siguientes tiempos traducción de direcciones virtuales(ns).

		TLB				
		4	**8**	**16**	**32**	**64**
Marcos	**8**	19664	17070	16722	16722	16722
	16	17984	15190	13706	13822	13416
	32	15764	12910	11484	9182	9240

Ahora se muestran 2 graficas, la primera es de los tiempos de traducción de direcciones virtuales propuestos y la segunda es el tiempo de carga de traducción de direcciones virtuales que se obtuvieron al ejecutar la simulación.



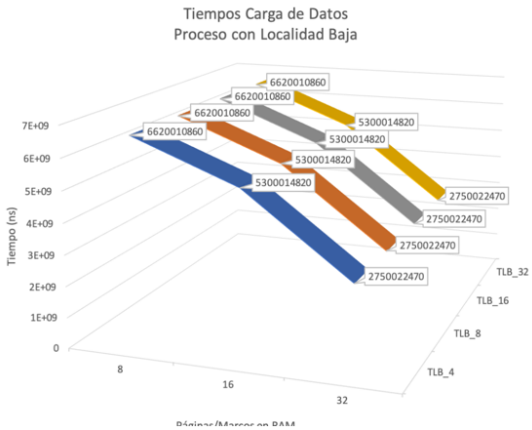


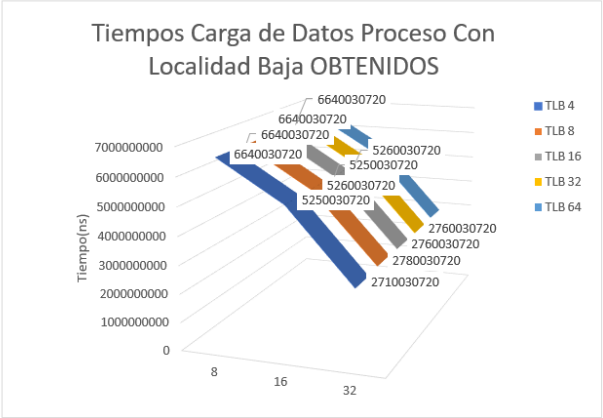
4.2. Referencias Baja

Se realizo la simulación el archivo *ej_Baja_64paginas.txt* y se obtuvo los siguientes tiempos de Carga de datos (ns):

		TLB				
		4	**8**	**16**	**32**	64
Marcos	**8**	6640030720	6640030720	6640030720	6640030720	6640030720
	16	5250030720	5260030720	5250030720	5260030720	5290030720
	32	2710030720	2780030720	2760030720	2760030720	2800030720

Ahora se muestran 2 graficas, la primera es de los tiempos de carga de datos propuestos y la segunda es el tiempo de carga de datos que se obtuvieron al ejecutar la simulación.

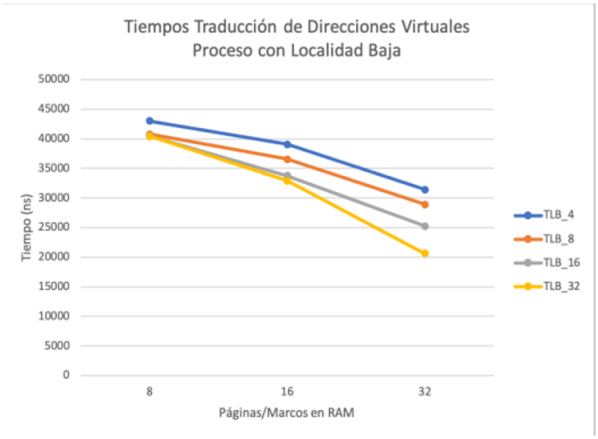


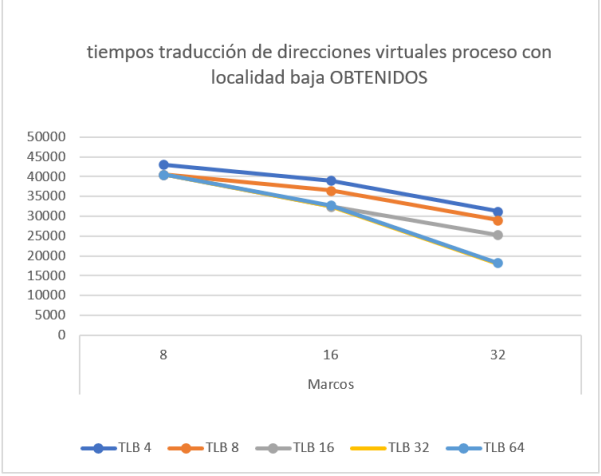


Se realizo la simulación el archivo *ej_Baja_64paginas.txt* y se obtuvo los siguientes tiempos traducción de direcciones virtuales(ns).

		TLB				
		4	**8**	**16**	**32**	**64**
Marcos	**8**	43080	40560	40560	40560	40560
	16	39000	36476	32498	32556	32730
	32	31290	29036	25280	18056	18288

Ahora se muestran 2 graficas, la primera es de los tiempos de traducción de direcciones virtuales propuestos y la segunda es el tiempo de carga de traducción de direcciones virtuales que se obtuvieron al ejecutar la simulación.





5. Interpretación de los resultados y conclusión.

Lo primero que podemos analizar es el por que se usaron 2 archivos .txt , esto es por que en uno quería representar la localidad alta y en el otro a la localidad baja. El principio de localidad representa que tanto o que tan poco se llama a una referencia y a que tan juntos o separados se encuentran los llamados. El archivo de localidad alta se refiere a que los llamados de la misma página están muy juntos en su mayoría. por el contrario, en el archivo de localidad baja se quiere representar unos llamados menos ocurrentes entre sí. Esto hace que los tiempos entre el archivo con localidad alta sea mucho menor a el archivo de localidad Baja. Al permitir que se referencien las páginas cuando ya están en la ram o en la TLB, se reducen los tiempos de carga y los tiempos de traducción. Estos tiempos se reducen por que al estar cargados en la ram van a generar menos fallos de páginas.

Los tiempos de carga no tiene nada que ver con el tamaño de la TLB, los tiempos de carga solo tienen en cuenta los llamados a la ram, los tiempos de traducción si tienen en cuenta los tiempos de la TLB ya que la función de la TLB es almacenar las traducciones, si bien los datos obtenidos son muy cercanos a los datos propuestos, no son exactamente los mismos. Al hacer la simulación con 64 paginas de TLB se tuvo los mismos tiempos a cuando la TLB estaba en 35, esto se debe a que una traducción no puede estar en la TLB sin estar asociada en la RAM, al no estar en la memoria real, no existe una traducción de memoria virtual a memoria real que se pueda almacenar en la TLB, cuando se llega al limite de la ram, no existe la posibilidad de guardar más traducciones que datos en la RAM.

Así mismo, se observó que al simular la misma combinación de tamaños entre TLB y ram en repetidas ocasiones, los tiempos variaban dentro del margen de error, los tiempos no eran los mismos en cada ocasión. Esto se debe a la concurrencia, Se tiene que el corrimiento del envejecimiento se lleva a cabo cada 1 milisegundo y la carga de referencias se lleva a cabo cada 2 milisegundos, puede que, alguna página al no ser llamada durante bastante tiempo, puede que no se alcance a apreciar en el contador y puede que en algún momento varios datos de la RAM aparenten tener la misma vejes y por como esta construida la simulación, termina quitando la primera pagina que se encuentra, cuando realmente la mejor opción hubiese sido eliminar alguna de las otras páginas.