Integrantes

Santiago Martínez Novoa - s.martinezn - 202112020

Juan Camilo Bonet De Vivero - j.bonet - 202022466

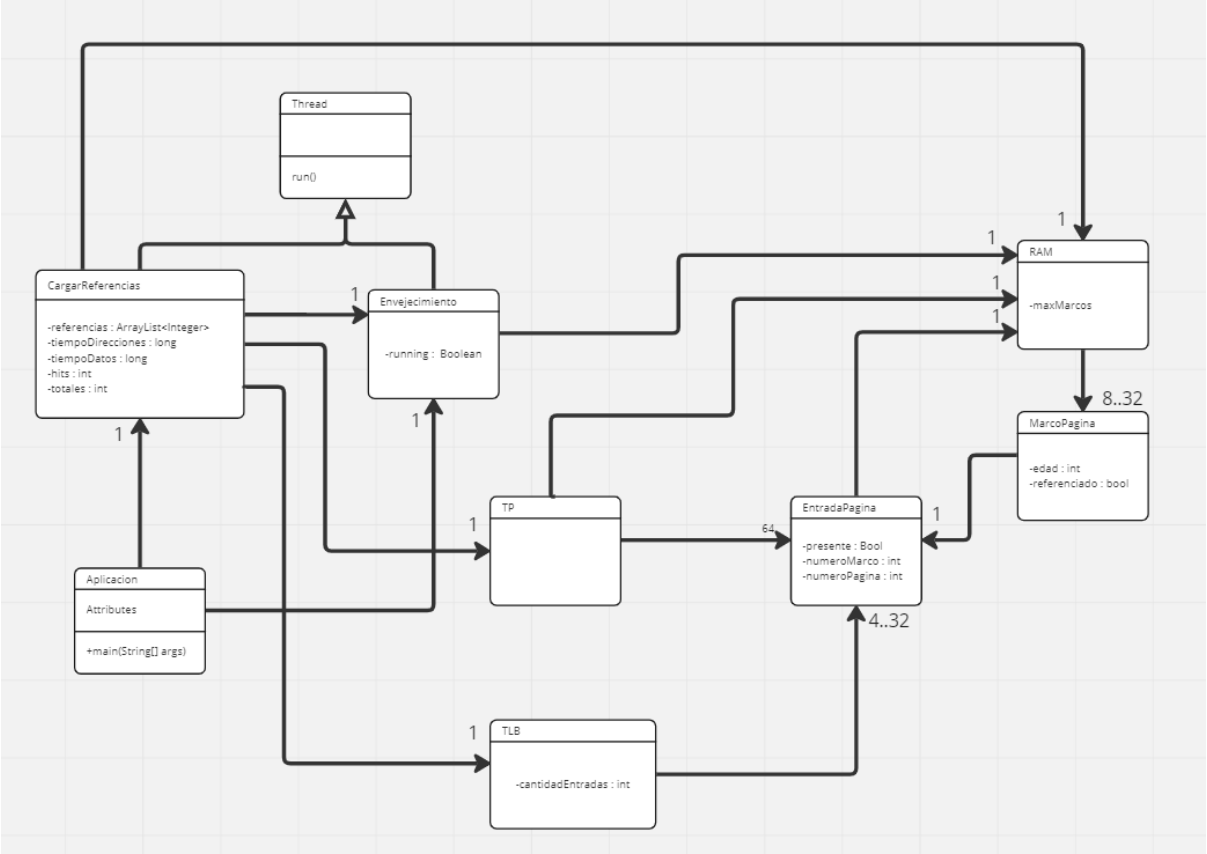
Sebastián Alberto Umaña Peinado - s.umanap - 202013778

Informe de resultados – Caso 2

Infraestructura Computacional

Descripción de estructuras de datos:

Para aproximarnos al problema decidimos generar un diagrama UML para organizar todo de una mejor manera:



Como se puede observar los dos Threads son, CargarReferencias y Envejecimiento, el primero tiene como atributos un elemento de la clase RAM, uno de clase TLB, uno de clase TP y una ArrayList de referencias que se obtienen gracias a la lectura que se hace en el método main de Aplicación. De esta manera, cuando se corre el Thread CargarReferencia

esta iterará sobre todas las referencias recibidas y luego terminará su proceso junto con el del Thread de envejecimiento. Para simular el comportamiento del sistema de paginación se utilizaron las siguientes estructuras de datos:

- **TLB**: Es una clase con dos atributos, el primero siendo la cantidad de entradas máximas que va a tener y una LinkedList donde se van a almacenar las entradas de páginas (Esto es una clase creada que posee las identificaciones en RAM, si está referenciado, su número de marco y su número de página). Esta estructura facilita las cosas, pues tienen los métodos addFirst() y removeLast(), que permiten fácilmente implementar el algoritmo FIFO: Si la TLB está llena se eliminará el primero que haya ingresado y luego se añadirá en primera posición a la nueva referencia. La otra razón por la que cambia la TLB y existe un método llamado quitarEntrada es cuando se genera un fallo de página. En estos casos, como esta entrada de página ya no existe en la RAM, se debe también retirar de la TLB porque la referencia ya no está en la memoria real y ahora se encuentra en el área de swap, de manera que el método simplemente elimina dicha referencia sin importar si es el primero que se ha ingresado para luego ejecutar con normalidad el algoritmo FIFO.
- **TP**: Es una clase que también tiene dos atributos: una RAM y un ArrayList con todas las entradas de la tabla de páginas. En el constructor se crean las 64 páginas que tiene la TP y se asocian con la RAM que se está utilizando. Cada entrada de páginas puede ser modificada por la TP en caso de que esté en la RAM (Verificado con el método getPresente()) y marcarlo como referenciado en la memoria real. Sin embargo, también la TP puede modificar la entrada de página en caso de que ya no esté referenciada en la RAM debido a un fallo de página, en este caso su atributo de presente pasa a ser False.
- **RAM:** Es una clase que tiene dos atributos: la cantidad de marcos que tiene y una lista con los marcos de página que existen en memoria real. Los marcos de página se hicieron también como una clase, la cual contiene su posición dentro de la RAM, si ha sido referenciado en algún momento durante un ciclo, un entero que guarda la edad del marco de página, y la entrada de página que se encuentra actualmente en ese marco de página. Los marcos de página son modificados constantemente por el envejecimiento, el cual revisa si su atributo referenciado es True o False. Primero se elimina el bit menos significativo y en caso de que sea True se sumará un 1 al bit más significativo que representa uso. Cuando finaliza el ciclo, el atributo referenciado de todos los marcos de página es puesto de nuevo en False listo para el siguiente ciclo de envejecimiento.

Esquema de sincronización utilizado:

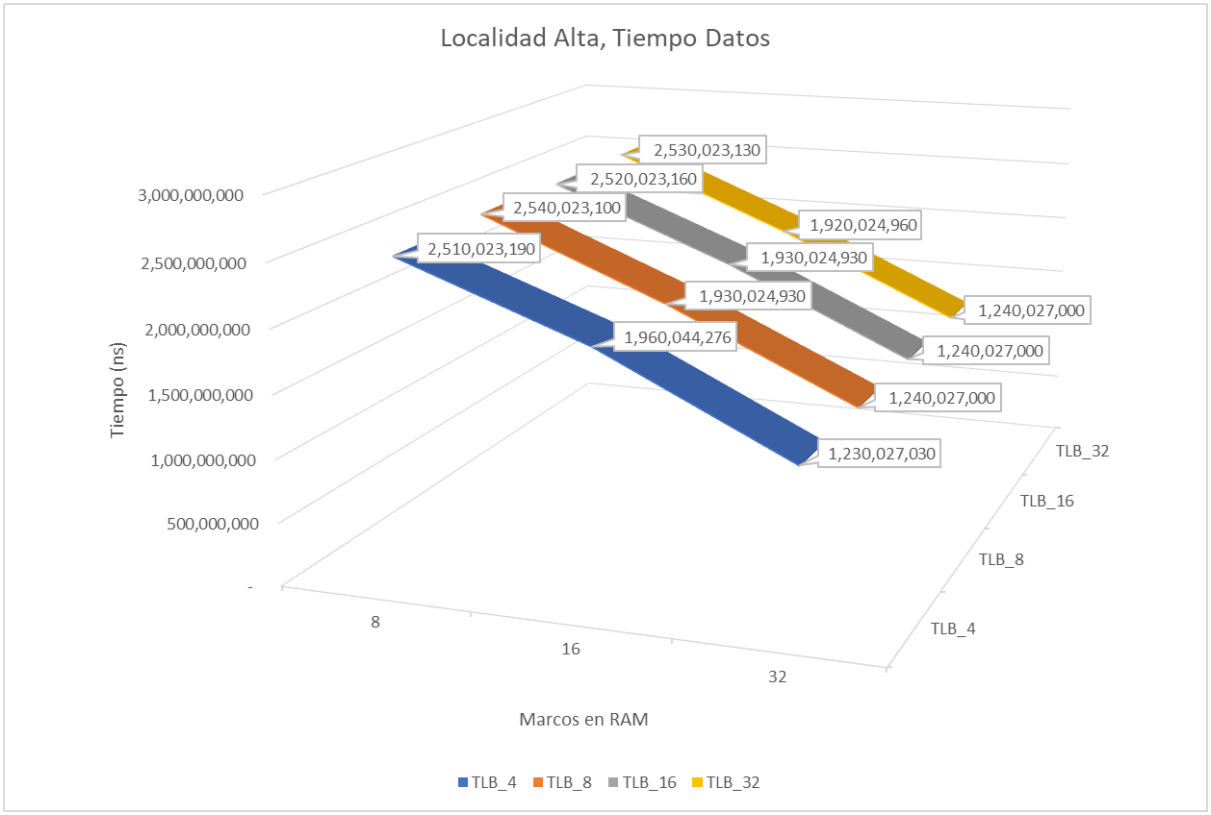
La sincronización debe hacerse cuando haya dos o más Threads que modifiquen o trabajen sobre los mismos datos durante su ejecución. En este caso tenemos solo dos Threads, y estos comparten un solo objeto en común, el envejecimiento de los marcos de página, pues uno de ellos constantemente lo cambia, y el otro depende de dichos cambios para realizar el reemplazo en los marcos de página.

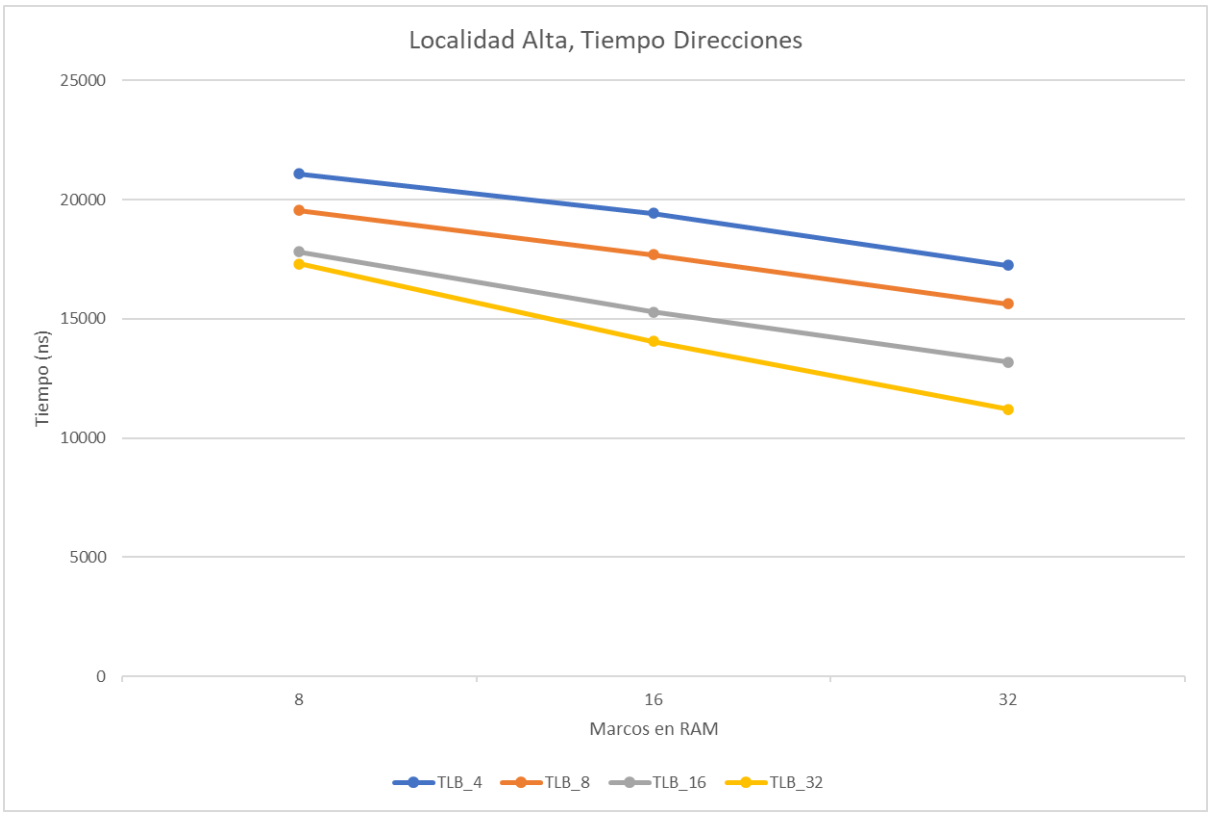
Para solucionar este problema tuvo que realizarse sincronización en ciertos métodos de la RAM, porque estos métodos hacen referencia a la edad de los marcos de página, y como se necesita que mientras se realiza el envejecimiento o se consulta el mismo no haya modificaciones de ningún tipo es necesario que estos métodos sean sincronizados.

Datos

Localidad Alta

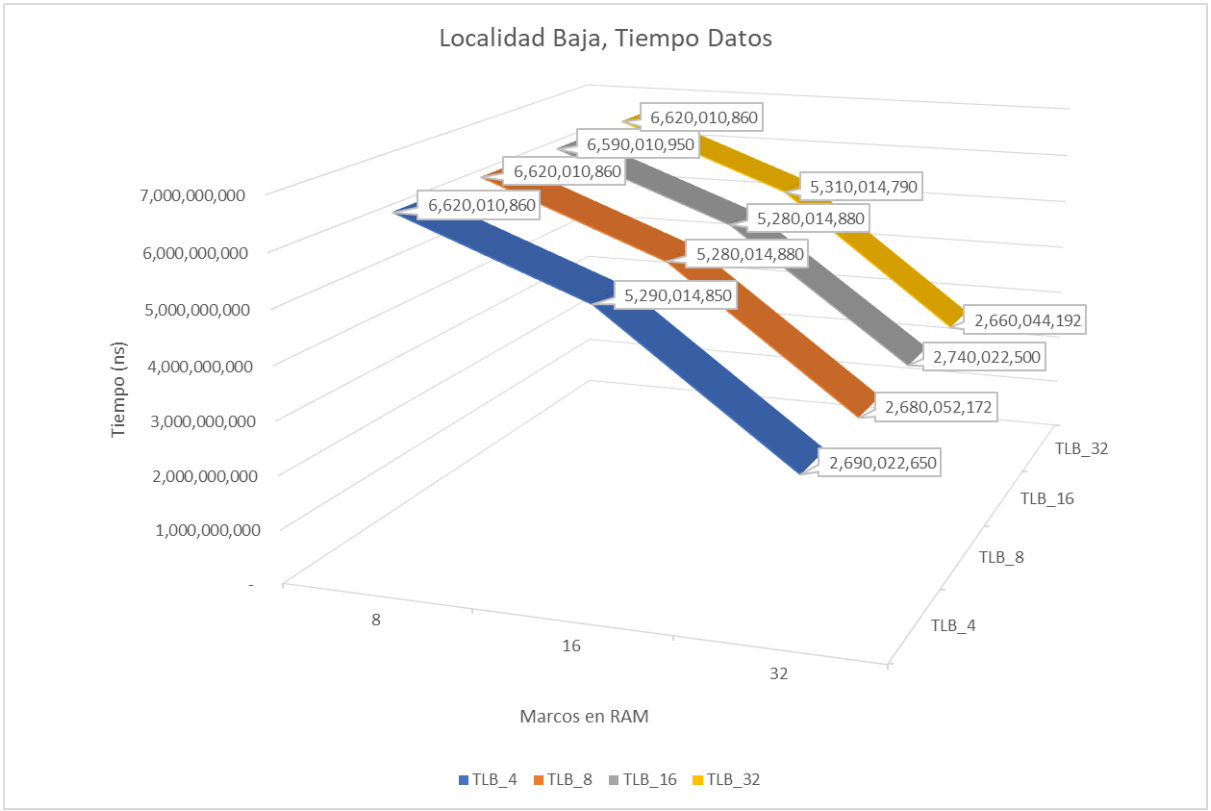
# Entradas TLB	# Marcos en RAM	Tiempo datos (ns)	Tiempo direcciones (ns)	Hit ratio
4	8	2,510,023,190	21086	0.598
4	16	1,960,044,276	19436	0.598
4	32	1,230,027,030	17246	0.598
8	8	2,540,023,100	19552	0.655
8	16	1,930,024,930	17694	0.656
8	32	1,240,027,000	15624	0.656
16	8	2,520,023,160	17812	0.714
16	16	1,930,024,930	15286	0.740
16	32	1,240,027,000	13188	0.741
32	8	2,530,023,130	17310	0.732
32	16	1,920,024,960	14052	0.782
32	32	1,240,027,000	11200	0.810

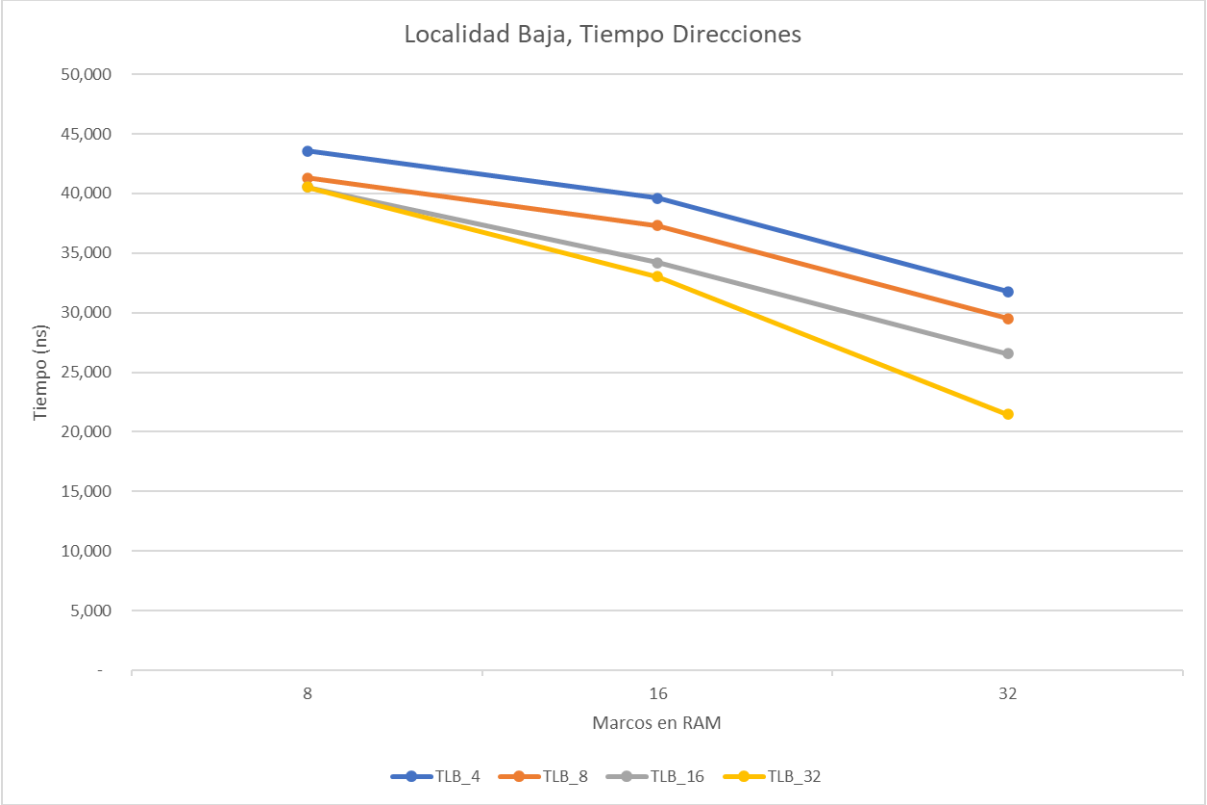




Localidad Baja

# Entradas TLB	# Marcos en RAM	Tiempo datos (ns)	Tiempo direcciones (ns)	Hit ratio
4	8	6,620,010,860	43,580	0.244
4	16	5,290,014,850	39,618	0.243
4	32	2,690,022,650	31,790	0.244
8	8	6,620,010,860	41,312	0.323
8	16	5,280,014,880	37,292	0.323
8	32	2,680,052,172	29,492	0.323
16	8	6,590,010,950	40,522	0.348
16	16	5,280,014,880	34,212	0.431
16	32	2,740,022,500	26,564	0.431
32	8	6,620,010,860	40,556	0.350
32	16	5,310,014,790	33,014	0.476
32	32	2,660,044,192	21,452	0.602





Interpretación de los datos

Hay varias observaciones que se pueden hacer sobre los datos. Se puede notar que al subir el número de marcos de página en el RAM, bajan los tiempos para cargar datos y para resolver direcciones. Esto tiene sentido, ya que si hay más espacio en el RAM, van a ocurrir menos fallas de páginas, la operación más costosa en términos de tiempo de carga y de direcciones. También podemos hacer la observación de que la cantidad de entradas en la tabla TLB básicamente no afecta los tiempos de carga de datos. Al principio puede parecer algo un poco contradictorio, pero tiene una explicación. La TLB contiene una subconjunto de entradas de la tabla de páginas, o sea que si una operación causa una falla de páginas en la TP, igualmente va a causar una falla de páginas en el TLB. Con la TLB nos ahorramos tiempo de lectura del TP, lo cual, además de no afectar el tiempo de carga de datos, es un número tan insignificante comparado al costo de resolver una falla de páginas que su magnitud casi que no afecta el tiempo total. Si afecta el tiempo de resolver direcciones, como se evidencia en las gráficas de tiempos de direccion.

En términos de localidad baja contra alta, los resultados también tienen sentido. Los programas con localidad alta tendrán menos fallas de páginas, y esto se evidencia en los tiempos de carga de datos. Todos los tiempos de localidad alta fueron más bajos que sus correspondientes tiempos en un programa con localidad baja. También se puede distinguir que en los programas con localidad baja hay una sensibilidad más alta al subir el número de marcos en RAM. Como estos tienden a usar más referencias en un tiempo corto, al subir el número de marcos estamos bajando la posibilidad de una falla de página por un factor más grande que los programas con localidad alta.

Otro fenómeno interesante pasa al subir el número de entradas del TLB en los tiempos de traducción de direcciones. Podemos ver que cuando hay pocos marcos en RAM, hay menos diferencia comparado a cuando hay más marcos. Esta diferencia se nota bastante cuando hay localidad baja. La razón de este fenómeno es que, al tener menos marcos en RAM, van a ocurrir más fallas de páginas. No importa si una entrada esta en la TLB, si esta genera una falla de página las ganancias en tiempo serán anuladas por el tiempo que toma resolver la falla.

Además de los datos pedidos, decidimos registrar el hit rate para cada caso. Este hit rate está definido en el libro de Tanenbaum como el número de veces que encontramos una entrada en el TLB sobre el número de referencias totales. Podemos ver que este es un variable que depende mucho del número de entradas del TLB y no tanto de la cantidad de marcos en RAM. Hubiera sido interesante usar diferentes algoritmos para manejar el TLB y ver cómo varía el hit rate.

Referencias:

• *Modern Operating Systems. Andrew S. Tanenbaum. Editorial Pearson. Edición 3, año 2009.*