Bases de Datos

Trabajo Práctico Nro. 2

Autores:
Federico Huel
Luciano Leveroni
Matias Dinota

Fecha: 16 de Noviembre de 2013

Tabla de contenido

Introducción al algoritmo “Touch Count”

Junto con el lanzamiento de la versión 8i de su base de datos, Oracle introduce un nuevo algoritmo de *buffer management* llamado *Touch Count*, con el objetivo de mejorar la performance de su motor. La motivación principal para hacer un cambio tiene que ver con que los algoritmos originales fueron pensados en una época donde el tamaño de caché disponibles era mucho más acotado. El nuevo algoritmo está pensando para tener un comportamiento superior al usar cachés de gran tamaño.

En líneas generales, el algoritmo de *Touch Count* logra que un frame tenga que “ganarse” el lugar para permanecer en el buffer pool. Para esto asigna un contador a cada frame que refleja de algún modo su popularidad. Al momento de que un frame es accedido, el contador se incrementa. De todos modos, en la práctica, para evitar una ráfaga de accesos que hagan que un contador aumente demasiado rápido, se limita que el contador pueda aumentar una única unidad en cierto lapso de tiempo (el *paper* menciona 3 segundos).

En concreto, el algoritmo dispone de una cola con cada frame del buffer pool (en realidad, es sólo un puntero) separando la misma en 2 partes:

- Cold region (parte izquierda de la cola): Aquí se ubican los frames menos usados que se encuentran próximos a ser quitados del buffer.
- Hot region (parte derecha): Aquellos frames más populares. Los frames más populares de todos se ubican en la cabeza de la cola.

Por defecto, la hot region representa el 50% de la lista, aunque este valor es configurable.

Al momento de insertar un nuevo frame, este se inserta en un punto medio entre la cold region y la hot region.

Al momento de buscar un frame a liberar, el algoritmo se encarga primero de mover de la cold region a la hot a aquellos frames cuyo touch counter haya superado cierto umbral (por defecto 2). A dichos frames (nuevamente, en la práctica son punteros a los frames) se los desplaza hacia el final de la hot region (es decir a la cabeza de la cola) y su contador se resetea a 0. Como consecuencia de dicho movimiento, dado que la cantidad de frames en hot region es constante, el frame más de la izquierda de la hot region pasará a estar en la cold region. Cuando eso sucede, se restablece el contador de dicho frame a 1 (nuevamente, esto es configurable).

Implementación de estrategias de reemplazo de página

Se implementaron las estrategias de reemplazo de páginas LRU, MRU y Touch Count para luego poder comparar su efectividad en cuanto al porcentaje de páginas encontradas en memoria. Aprovechando la estructura del código proporcionado por la cátedra, se decidió utilizar la técnica TDD para el desarrollo de las nuevas funcionalidades.

LRU y MRU

Las primeras estrategias implementadas fueron la LRU y la MRU. Se creó la clase RUBufferFrame que es utilizada por ambas estrategias. El RUBufferFrame cuenta con la variable lastTimeUsed que es la que es chequeada para verificar cual de los frames presentes en memoria fue el último utilizado. Esta variable se actualiza en el momento en que se realiza un pin del frame, ya que esto indica que se utilizará la página.

Touch Count

Luego se realizó la implementación de la estrategia Touch Count. En este caso, tuvieron que tomarse varias decisiones, dado que había muchos aspectos que no estaban definidos en la descripción del algoritmo y no hacían a la esencia del mismo.

Lo primero que se tuvo que decidir fue cómo implementar la estructura que utiliza la estrategia. Para esto, se creó una clase a la que se llamó LRUChain que cumple con el funcionamiento de la cadena requerida por el algoritmo. Esto, además de ser una decisión conceptual en la cual se separan las responsabilidades, también permitió poder testear la estructura por separado del algoritmo utilizado por la estrategia.

La LRUChain cuenta con dos colas de dos finales (ArrayDeque) que representan la regiones HotRegion y ColdRegion. Esto es altamente aprovechado en las funcionalidades necesarias. De esta forma, la inserción en el punto medio es simplemente encolar al final de la coldRegion el buffer a agregar. Además, facilita la tarea de pasar un buffer de una región a la otra y poder respetar el orden correcto dentro de cada región, ya que simplemente se van agregando los buffers al final de cada cola.

Cada Buffer tiene su propio Touch Counter que se incrementa cuando se hace un pin. La estrategia actualiza la estructuca de la LRUChain, es decir, cada una de las regiones, al momento en que tiene que buscar qué candidato decidirá reemplazar. Al tener esto esparado, se garantiza que el incremento del TC y el movimiento de los bloques sea independiente.

Para buscar el buffer a reemplazar, se recorre primero los frames de la coldRegion y luego de la hotRegion en orden hasta encontrar uno que no esté siendo utilizado. Cuando el frame es encontrado se elimina de memoria y se inserta la nueva página al final de la coldRegion.

La forma en la que se va llenando la LRUChain cuando no está llena es un aspecto que no está especificado en el algoritmo. Es por esto que tomamos, arbitrariamente, la decisión de agregar todos los elementos en la hotRegion primero y cuando esta está llena, empezar a argegarlos en la coldRegion. Para verificar cuándo las regiones están llenas ,se utilizan las variables sizeColdRegion y sizeHotRegion que representan la cantidad màxima de frames que pueden estar presentes en cada cola.

Evaluación de desempeño de las estrategias implementadas

Con el objetivo de analizar el comportamiento de cada una de las estrategias implementadas se crearon trazas de pedidos de página utilizando las herramientas de generación de trazas provistas por la cátedra. En particular, se crearon trazas básicas de File Scan, Index Scan Y Block Nested Loops Join (BNLJ) y se combinaron entre sí para obtener casos de análisis interesantes. Es decir, se intentó entrelazar trazas de manera tal que generaran *Hits* en los pedidos de página y así poder evaluar el comportamiento de las estrategias de reemplazo.

Además, por cada uno de estos casos, que llamaremos escenarios, se generaron variantes aumentando la cantidad de páginas de los archivos referenciados en las trazas. Finalmente, cabe mencionar que la efectividad de las estrategias de reemplazo va a estar medida por el *Hit rate* y el *Hit rate with memory*.

A continuación se presentan los resultados obtenidos con las distintas estrategias para cada uno de los escenarios, variando el tamaño del Buffer Pool. Además, se realizará un análisis de cada uno de los escenarios con el objeto de determinar que estrategia tuvo el mejor comportamiento.

Referencias:

- P_A: cantidad de páginas del archivo A.
- X_I: altura del árbol del índice I.
- Los valores de las celdas son de la forma (*Hit rate* / *Hit rate with memory*)
- Las celdas en verde muestran el máximo *Hit Rate* obtenido para todas las estrategias, con un mismo tamaño de buffer pool.
- Las celdas en amarillo muestran empates en el *Hit Rate* máximo entre dos o más estrategias.

Escenario A:

Traza de BNLJ de Producto x Producto.
$P_{producto}$ = 100

Estrategia \ tamaño pool	50	100	150	200	250	300	350	400	450	500
FIFO	0.12 / 0.14	0.90 / 1	0.90 / 1	0.90 / 1	0.90 / 1	0.90 / 1	0.90 / 1	0.90 / 1	0.90 / 1	0.90 / 1
LRU	0.13 / 0.14	0.90 / 1	0.90 / 1	0.90 / 1	0.90 / 1	0.90 / 1	0.90 / 1	0.90 / 1	0.90 / 1	0.90 / 1
MRU	0.45 / 0.50	0.90 / 1	0.90 / 1	0.90 / 1	0.90 / 1	0.90 / 1	0.90 / 1	0.90 / 1	0.90 / 1	0.90 / 1
TC	0.31 / 0.34	0.90 / 1	0.90 / 1	0.90 / 1	0.90 / 1	0.90 / 1	0.90 / 1	0.90 / 1	0.90 / 1	0.90 / 1

Escenario A1:
Traza de BNLJ de Producto x Producto.
$P_{producto}$ = 500

Estrategia \ tamaño pool	50	100	150	200	250	300	350	400	450	500
FIFO	0.02 / 0.02	0.03 / 0.03	0.03 / 0.03	0.04 / 0.04	0.08 / 0.09	0.22 / 0.22	0.41 / 0.42	0.57 / 0.58	0.78 / 0.79	0.98 / 1
LRU	0.03 / 0.03	0.03 / 0.03	0.03/ 0.04	0.03 / 0.03	0.04 / 0.04	0.09 / 0.10	0.17 / 0.17	0.29 / 0.30	0.62 / 0.63	0.98 / 1
MRU	0.09 / 0.09	0.19 / 0.19	0.29/ 0.29	0.38 / 0.39	0.48 / 0.49	0.58 / 0.59	0.68 / 0.69	0.78 / 0.79	0.87 / 0.89	0.98 / 1
TC	0.06 / 0.06	0.11 / 0.12	0.16 / 0.17	0.21 / 0.22	0.26 / 0.27	0.31 / 0.32	0.36 / 0.37	0.41 / 0.42	0.46 / 0.47	0.98 / 1

Escenario A2:
Traza de BNLJ de Producto x Producto.
$P_{producto}$ = 1000

Estrategia \ tamaño pool	50	100	150	200	250	300	350	400	450	500
FIFO	0.01 / 0.01	0.01 / 0.01	0.01 / 0.01	0.01 / 0.01	0.01 / 0.01	0.01 / 0.01	0.01 / 0.01	0.01 / 0.01	0.01 / 0.01	0.01 / 0.01
LRU	0.01 / 0.01	0.01 / 0.01	0.01 / 0.01	0.01 / 0.01	0.01 / 0.01	0.01 / 0.01	0.01 / 0.01	0.01 / 0.01	0.01 / 0.01	0.01 / 0.01
MRU	0.04 / 0.04	0.09 / 0.09	0.14 / 0.14	0.19 / 0.19	0.24 / 0.24	0.29 / 0.29	0.34 / 0.35	0.39 / 0.39	0.44 / 0.44	0.49 / 0.49
TC	0.03 / 0.03	0.05 / 0.06	0.08 / 0.08	0.10 / 0.11	0.13 / 0.13	0.15 / 0.16	0.18 / 0.18	0.20 / 0.21	0.23 / 0.23	0.25 / 0.26

Como se puede notar a simple vista a partir de los escenarios A1 y A2 y del escenario original A, en menor medida, la estrategia con mejor comportamiento es MRU. Esto, como se verá más adelante, es un resultado que no se vuelve a repetir, por lo que es probable que se deba a que el reemplazo MRU es eficiente cuando se tiene una traza de BNLJ sobre un mismo archivo.

Escenario B:
Traza de File Scan de Producto entrelazada con traza BNLJ de Producto x Vendedor
$P_{producto}$ = 100
$P_{vendedor}$ = 100

Estrategia \ tamaño pool	50	100	150	200	250	300	350	400	450	500
FIFO	0.13 / 0.19	0.31 / 0.43	0.51 / 0.72	0.71 / 1	0.71 / 1	0.71 / 1	0.71 / 1	0.71 / 1	0.71 / 1	0.71 / 1
LRU	0.13 / 0.18	0.34 / 0.47	0.53 / 0.75	0.71 / 1	0.71 / 1	0.71 / 1	0.71 / 1	0.71 / 1	0.71 / 1	0.71 / 1
MRU	0.13 / 0.19	0.31 / 0.44	0.50 / 0.70	0.71 / 1	0.71 / 1	0.71 / 1	0.71 / 1	0.71 / 1	0.71 / 1	0.71 / 1
TC	0.05 / 0.08	0.17 / 0.23	0.34 / 0.48	0.71 / 1	0.71 / 1	0.71 / 1	0.71 / 1	0.71 / 1	0.71 / 1	0.71 / 1

Escenario B1:
Traza de File Scan de Producto entrelazada con traza BNLJ de Producto x Vendedor

$P_{producto}$ = 500
$P_{vendedor}$ = 250

Estrategia \ tamaño pool	50	100	150	200	250	300	350	400	450	500
FIFO	0.24 / 0.49	0.36 / 0.73	0.44 / 0.89	0.48 / 0.96	0.5 / 1	0.5 / 1	0.5 / 1	0.5 / 1	0.5 / 1	0.5 / 1
LRU	0.28 / 0.56	0.39 / 0.79	0.42 / 0.85	0.48 / 0.96	0.5 / 1	0.5 / 1	0.5 / 1	0.5 / 1	0.5 / 1	0.5 / 1
MRU	0.11 / 0.23	0.14 / 0.28	0.22 / 0.44	0.31 / 0.63	0.29 / 0.59	0.35 / 0.70	0.43 / 0.87	0.45 / 0.90	0.46 / 0.93	0.5 / 1
TC	0.5 / 1	0.5 / 1	0.5 / 1	0.5 / 1	0.5 / 1	0.5 / 1	0.5 / 1	0.5 / 1	0.5 / 1	0.5 / 1

Escenario B2:
Traza de File Scan de Producto entrelazada con traza BNLJ de Producto x Vendedor
$P_{producto}$ = 1000
$P_{vendedor}$ = 500

Estrategia \ tamaño pool	50	100	150	200	250	300	350	400	450	500
FIFO	0.24 / 0.48	0.36 / 0.72	0.43 / 0.87	0.48 / 0.96	0.5 / 1	0.5 / 1	0.5 / 1	0.5 / 1	0.5 / 1	0.5 / 1
LRU	0.23 / 0.46	0.35 / 0.71	0.43 / 0.87	0.49 / 0.99	0.5 / 1	0.5 / 1	0.5 / 1	0.5 / 1	0.5 / 1	0.5 / 1
MRU	0.08 / 0.16	0.10 / 0.21	0.14 / 0.28	0.13 / 0.27	0.14 / 0.29	0.20 / 0.41	0.19 / 0.38	0.23 / 0.46	0.26 / 0.52	0.26 / 0.52
TC	0.44 / 0.89	0.5 / 1	0.5 / 1	0.5 / 1	0.5 / 1	0.5 / 1	0.5 / 1	0.5 / 1	0.5 / 1	0.5 / 1

En este escenario, la estrategia que mejores resultados entregó fue *Touch Count.* Esto se puede apreciar sobre todo en los casos B1 y B2 en los que la cantidad de páginas es mayor. Contrario a lo que se esperaría y a lo mencionado en la introducción del algoritmo *Touch Count,*

en este escenario la estrategia fue eficiente aún en casos cuando el tamaño del buffer pool es pequeño.

Escenario C:
Traza de File Scan de Producto entrelazada con traza de Index Scan de Producto con un índice clustered.
$P_{producto} = 500$
$X_i = 4$

Estrategia \ tamaño pool	**50**	**100**	**150**	**200**	**250**	**300**	**350**	**400**	**450**	**500**
FIFO	0.24 / 0.48	0.38 / 0.77	0.47 / 0.94	0.49 / 0.98	0.49 / 0.99	0.5 / 1	0.5 / 1	0.5 / 1	0.5 / 1	0.5 / 1
LRU	0.24 / 0.48	0.36 / 0.73	0.45 / 0.91	0.22 / 1	0.47/ 0.94	0.49/ 0.99	0.5 / 1	0.5 / 1	0.5 / 1	0.5 / 1
MRU	0.11 / 0.22	0.14 / 0.28	0.23 / 0.46	0.29 / 0.59	0.27 / 0.55	0.36 / 0.72	0.42 / 0.84	0.46 / 0.92	0.47 / 0.95	0.5 / 1
TC	0.5 / 1	0.5 / 1	0.5 / 1	0.5 / 1	0.5 / 1	0.5 / 1	0.5 / 1	0.5 / 1	0.5 / 1	0.5 / 1

Escenario C1:
Traza de File Scan de Producto entrelazada con traza de Index Scan de Producto con un índice clustered.
$P_{producto} = 1000$
$X_i = 4$

Estrategia \ tamaño pool	**50**	**100**	**150**	**200**	**250**	**300**	**350**	**400**	**450**	**500**
FIFO	0.04 / 0.15	0.09 / 0.32	0.17 / 0.62	0.22 / 0.79	0.27/ 0.95	0.25 / 0.91	0.26 / 0.91	0.27 / 0.97	0.27 / 0.96	0.28 / 1
LRU	0.04 / 0.15	0.09 / 0.31	0.14 / 0.51	0.20 / 0.71	0.24 / 0.85	0.26 / 0.93	0.28 / 0.99	0.27 / 0.95	0.28 / 1	0.28 / 1
MRU	0.03 / 0.11	0.06 / 0.23	0.10 / 0.38	0.14 / 0.50	0.15 / 0.54	0.18 / 0.66	0.22 / 0.80	0.24 / 0.87	0.26 / 0.93	0.28 / 0.99

TC	0.01 / 0.01	0.01 / 0.05	0.02 / 0.08	0.05 / 0.18	0.28 / 1	0.28 / 1	0.28 / 1	0.28 / 1	0.28 / 1	0.28 / 1

Escenario C2:
Traza de File Scan de Producto entrelazada con traza de Index Scan de Producto con un índice clustered.
$P_{producto} = 2000$
$X_i = 4$

Estrategia \ tamaño pool	**50**	**100**	**150**	**200**	**250**	**300**	**350**	**400**	**450**	**500**
FIFO	0.13 / 0.26	0.26 / 0.52	0.36 / 0.73	0.44 / 0.88	0.48 / 0.97	0.49 / 0.99	0.5 / 1	0.5 / 1	0.5 / 1	0.5 / 1
LRU	0.13 / 0.26	0.24 / 0.49	0.34 / 0.69	0.42 / 0.85	0.46 / 0.93	0.49 / 0.99	0.49 / 0.99	0.5 / 1	0.5 / 1	0.5 / 1
MRU	0.01 / 0.02	0.04 / 0.09	0.05 / 0.11	0.06 / 0.13	0.07 / 0.14	0.09 / 0.19	0.10 / 0.21	0.11 / 0.23	0.12 / 0.25	0.15 / 0.3
TC	0.08 / 0.17	0.20 / 0.41	0.31 / 0.63	0.5 / 1	0.5 / 1	0.5 / 1	0.5 / 1	0.5 / 1	0.5 / 1	0.5 / 1

En este escenario se puede notar como la estrategia de *Touch Count* obtuvo mejores resultados que otras (en este caso FIFO) a medida que el tamaño del buffer pool aumenta. Este es uno de los resultados esperados según de lo que se presentó en la introducción y se corresponde con lo que se menciona en la documentación sobre la implementación del algoritmo en *Oracle.*

Escenario D:
Traza de BNLJ de Producto x Venta entrelazada con traza BNLJ de Venta x Producto
$P_{producto} = 100$
$P_{venta} = 80$

Estrategia \ tamaño pool	**50**	**100**	**150**	**200**	**250**	**300**	**350**	**400**	**450**	**500**

FIFO	0.33 / 0.41	0.57 / 0.70	0.75 / 0.93	0.81 / 1	0.81 / 1	0.81 / 1	0.81 / 1	0.81 / 1	0.81 / 1	0.81 / 1
LRU	0.31 / 0.39	0.61 / 0.76	0.77 / 0.95	0.81 / 1	0.81 / 1	0.81 / 1	0.81 / 1	0.81 / 1	0.81 / 1	0.81 / 1
MRU	0.28 / 0.34	0.49 / 0.61	0.76 / 0.93	0.81 / 1	0.81 / 1	0.81 / 1	0.81 / 1	0.81 / 1	0.81 / 1	0.81 / 1
TC	0.28 / 0.34	0.59 / 0.73	0.78 / 0.96	0.81 / 1	0.81 / 1	0.81 / 1	0.81 / 1	0.81 / 1	0.81 / 1	0.81 / 1

Escenario D1:
Traza de BNLJ de Producto x Venta entrelazada con traza BNLJ de Venta x Producto
$P_{producto}$ = 500
P_{venta} = 300

Estrategia \ tamaño pool	**50**	**100**	**150**	**200**	**250**	**300**	**350**	**400**	**450**	**500**
FIFO	-	-	0.12 / 0.14	0.13 / 0.15	0.14 / 0.16	0.15 / 0.17	0.15 / 0.18	0.16 / 0.18	0.19 / 0.21	0.23/ 0.26
LRU	-	-	0.12 / 0.14	0.13 / 0.15	0.14 / 0.16	0.14 / 0.16	0.16 / 0.18	0.16 / 0.18	0.17 / 0.19	0.20 / 0.23
MRU	-	-	0.18 / 0.21	0.26 / 0.29	0.31 / 0.35	0.37 / 0.42	0.42 / 0.48	0.47 / 0.53	0.52 / 0.59	0.58 / 0.65
TC	-	-	0.13 / 0.14	0.18 / 0.21	0.26 / 0.29	0.29 / 0.33	0.32 / 0.36	0.35 / 0.40	0.38 / 0.43	0.42 / 0.48

Escenario D2:
Traza de BNLJ de Producto x Venta entrelazada con traza BNLJ de Venta x Producto
$P_{producto}$ = 1000
P_{venta} = 750

Estrategia \ tamaño pool	**50**	**100**	**150**	**200**	**250**	**300**	**350**	**400**	**450**	**500**

FIFO	-	-	-	0.31 / 0.33	0.39 / 0.42	0.46 / 0.50	0.49 / 0.53	0.49 / 0.54	0.49 / 0.54	0.50/ 0.54
LRU	-	-	-	0.29 / 0.31	0.35 / 0.38	0.41 / 0.44	0.45 / 0.49	0.48 / 0.52	0.49 / 0.53	0.49 / 0.54
MRU	-	-	-	0.07 / 0.08	0.09 / 0.10	0.11 / 0.11	0.12 / 0.13	0.13 / 0.14	0.14 / 0.16	0.16 / 0.17
TC	-	-	-	0.21 / 0.23	0.30 / 0.33	0.33 / 0.36	0.34 / 0.37	0.41 / 0.45	0.50 / 0.45	0.57 / 0.62

Nuevamente, en este escenario se puede notar la misma tendencia que en el anterior. Para buffers de tamaño grande la estrategia de *Touch Count* obtiene mejores resultados que las otras tres implementadas. Esto se ve en el escenario D2 especialmente ya que la cantidad de páginas permite apreciarlo.

Conclusiones

A partir de las experiencias realizadas y del análisis de los resultados se puede concluir que ninguna de las estrategias tiene un comportamiento efectivo para todos los casos. Presentando cuatro escenarios distintos y sus variantes se pudo comprobar que hay estrategias que tienen una tendencia a funcionar mejor en ciertas condiciones pero no se puede generalizar y determinar qué estrategia de reemplazo de página es mejor. En particular, la estrategia de reemplazo de página que utiliza el algoritmo *Touch Count* se comporta bien en los casos en que el tamaño del *buffer pool* es grande. Sin embargo, como se mencionó, esta tendencia no aplica a todos los escenarios presentados.

Probablemente, para hacer un estudio más profundo de estas estrategias se hubiera tenido que contar con volúmenes de datos significativamente más grandes. Como se mencionó en la introducción del trabajo, el algoritmo de *Touch Count* fue pensado para tener resultados eficientes en ambientes con buffers de varios *gigabytes* y pedidos de páginas proporcionalmente grandes y dichos volúmenes de datos exceden los límites del trabajo.

Finalmente, cabe mencionar que durante el desarrollo del trabajo elaborado, se incorporaron conocimientos de las distintas partes del motor de una base de datos que participan del manejo de la memoria y el acceso a disco y cómo interactúan, a grandes rasgos, para cumplir con los requerimientos.