UNIVERSIDAD DE CHILE
FACULTAD DE CIENCIAS FÍSICAS Y MATEMÁTICAS
DEPARTAMENTO DE CIENCIAS DE LA COMPUTACIÓN

ESTRUCTURAS COMPACTAS DINÁMICAS MÁS EFICIENTES PARA BASES DE DATOS DE GRAFOS CON ATRIBUTOS

RESULTADOS DE LOS EXPERIMENTOS SOBRE TRES $K^2 - TREES$ PARTE 2

DAVID DE LA PUENTE VÉLIZ

PROFESOR GUÍA:
GONZALO NAVARRO
PROFESOR GUÍA 2:
DIEGO ARROYUELO

SANTIAGO DE CHILE
2021

Preliminar

En este documento pusimos a prueba las dos estructuras que quedaron como candidatas a ser incrustadas en la base de datos. La basada en bloques de arreglos de bits dinámicos [1], a la que llamamos "blocks". Y la basada en la unión de k^2-trees estáticos [2], a la que llamamos "Ustatic". Para estas estructuras se midió su rendimiento con un nuevo tipo de experimento que intenta poner a prueba las estructuras para oleadas de operaciones de un mismo tipo, Es decir, primero una oleada de inserciones, luego una oleada de borrados, y así sucesivamente. En lo que sigue del informe se presentan dos experimentos.

datasets

Se usa el mismo dataset (indochina-2014) usado en en los experimentos básicos.

exp1: Oleadas 20-10

Este experimento consiste en insertar 20 millones de aristas, luego borrar 10 millones, luego partir desde la primera arista borrada e insertar nuevamente 20 millones. repetimos este proceso hasta que la estructura alcanza un tamaño de 190 millones de aristas. Es importante notar que las aristas borradas luego son reinsertadas, junto a 10 millones de aristas nuevas.

tiempo de inserción por arista

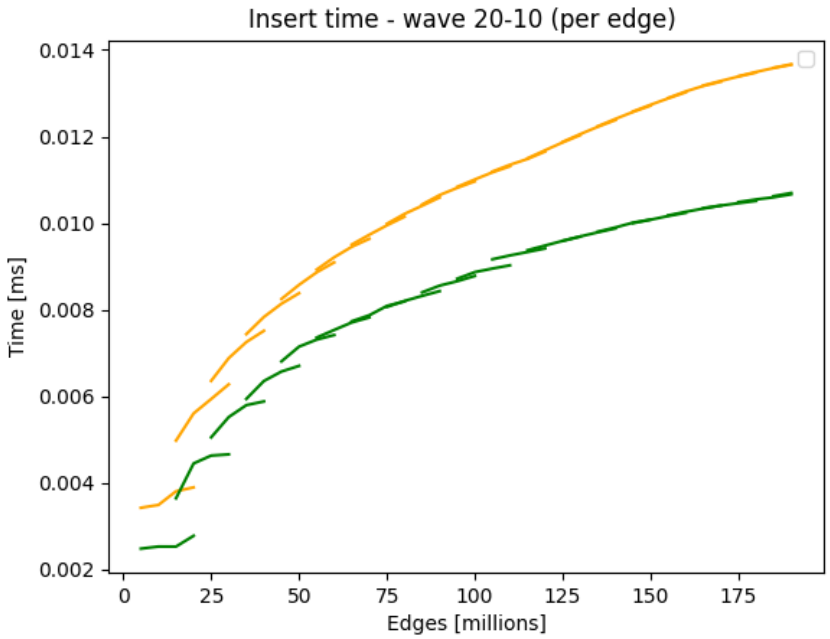

Figura 1: Tiempo de inserción por arista

En la figura 1, el color anaranjado representa a blocks, y el color verde a Ustatic. Lo que se muestra en la figura son varios gráficos que representan las oleadas de inserciones en cada estructura (por eso se ven como cortados). Es decir cada uno de estos mini gráficos representa la inserción de 20 millones de aristas (luego de que se le borraran 10 millones). Observamos que en ambos casos, se entorpece la inserción de aristas (los tiempos son mayores que en los experimentos básicos), pero se sigue manteniendo el comportamiento logarítmico. Ustatic gana.

tiempo de borrado por arista

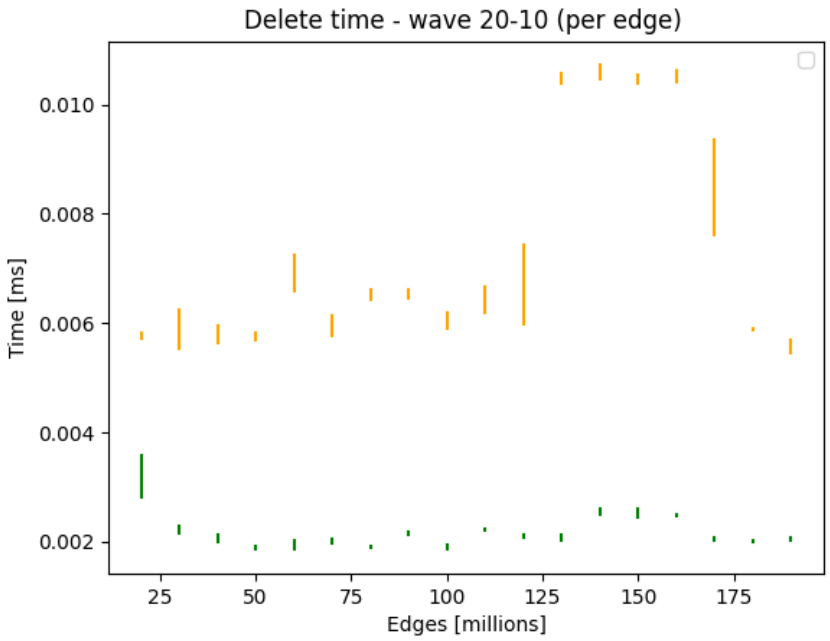

Figura 2: Tiempo de borrado por arista

En la figura 2, el color anaranjado representa a blocks, y el color verde a Ustatic. Como a las estructuras siempre se le borran 10 millones de aristas. La forma de mostrar los resultados es distintas. En el eje "x" tenemos el tamaño (en numero de aristas) de la estructura cuando se le borraron los 10 millones, y en el eje "y" tenemos el tiempo promedio al borrar 5 y 10 millones de aristas (por eso tenemos una linea, por que son dos valores de "y" por cada "x"). Por ejemplo si "x"=150 millones, para blocks obtendremos una linea naranja representada por dos puntos, un punto representa el tiempo promedio de borrar 5 millones de aristas para una estructura con 150 millones de aristas. Y el otro punto representa el tiempo promedio de borrar 10 millones de aristas para una estructura con 150 millones de aristas. Podemos observar que el tiempo de borrado por arista se mantiene casi igual que en los experimentos del informe anterior (excepto entre los 125 y 175 millones donde en ambas estructuras aumenta el tiempo, algo está pasando ahí). Gana Ustatic.

Memoria total

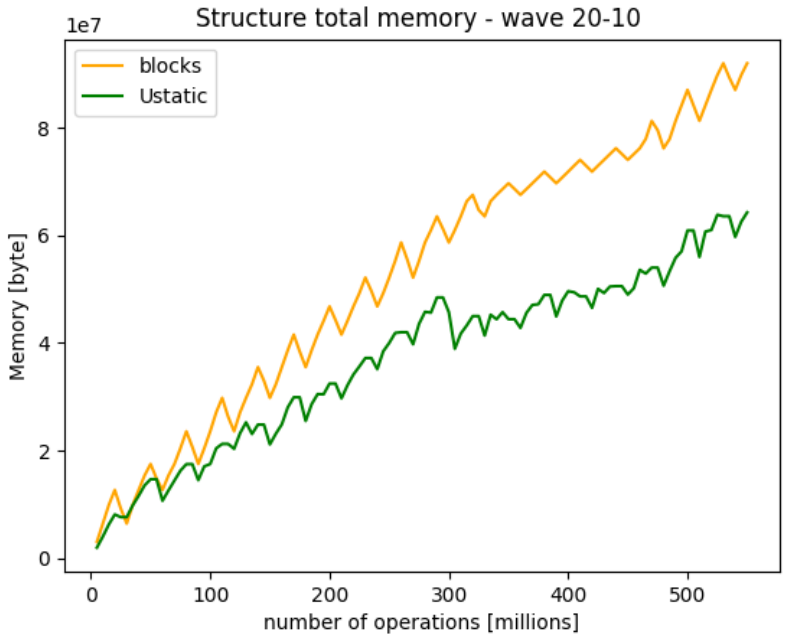

Figura 3: Memoria total ocupada por la estructura

En la figura 3 podemos ver el comportamiento de la memoria de las estructuras a medida que va transcurriendo el tiempo. En el eje "x" tenemos el numero de operaciones que han ocurrido hasta el momento (no hay distinción entre si es la operación es de inserción o borrado. En ambos casos podemos ver que la memoria tiene forma de "olas" (por eso el nombre del experimento) de tal forma que la memoria aumenta cuando se le insertan aristas, y baja cuando se le borran. gana Ustatic. (nuevamente en ambos experimentos ocurre algo en la zona donde se insertan las aristas 125M y 175M. En este gráfico vendría siendo como entre las operaciones entre 300M y 500M).

exp2: Oleadas 20-15

Este experimento, es muy similar al anterior, consiste en insertar 20 millones de aristas, luego borrar 15 millones. Pero esta vez NO reinsertamos las aristas borradas, sino que insertamos 20 millones de aristas nuevas. Repetimos este proceso hasta que se hayan insertado un total de 190 millones de aristas.

tiempo de inserción por arista

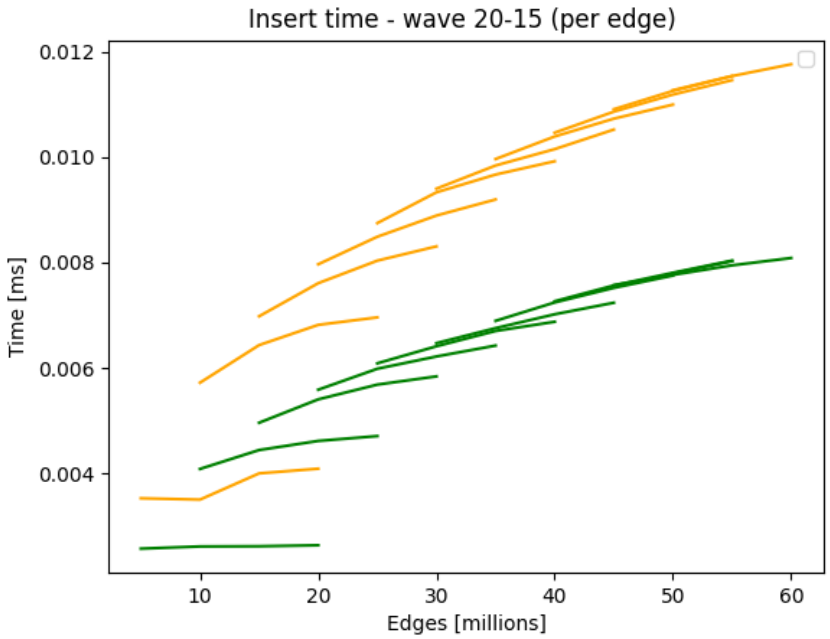

Figura 4: Tiempo de inserción por arista

En la figura 4, el color anaranjado representa a blocks, y el color verde a Ustatic. No hay mucha diferencia con respecto al experimento anterior. Ustatic gana.

tiempo de borrado por arista

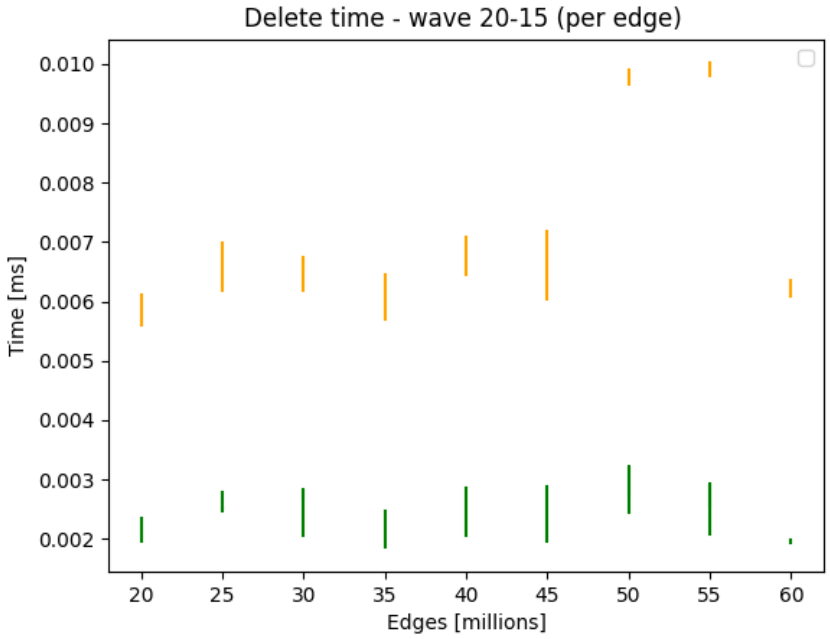

Figura 5: Tiempo de borrado por arista

En la figura 5, el color anaranjado representa a blocks, y el color verde a Ustatic. No hay mucha diferencia con respecto al experimento anterior. Gana Ustatic.

Memoria total

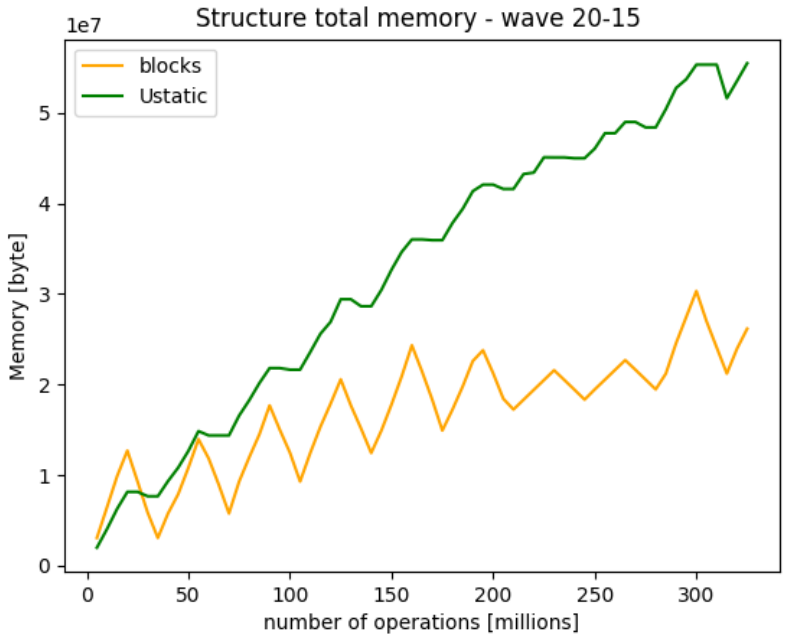

Figura 6: Memoria total ocupada por la estructura

En la figura 6 podemos ver el comportamiento de la memoria de las estructuras a medida que va transcurriendo el tiempo. Esta vez podemos notar algo interesante, la memoria ocupada por Ustatic empieza a crecer bastante, casi como si no se le borraran aristas. Esto se debe a que Ustatic no suele borrar inmediatamente las aristas borradas, si no que solo apaga sus bits. La diferencia con el experimento anterior es que esta vez no se reinsertan las mismas aristas borradas (solo tendría que prender sus bits), sino que son nuevas y tiene que pedir mas memoria.

Conclusiones

1) Ustatic es mas rápido que blocks.

2) El manejo de memoria de blocks es mejor que el de Ustatic.

3) En lo personal me quedaría con blocks por ser mas estable en sus operaciones.

4) (esto no es una conclusión pero si una hipótesis) Creo que existe una familia de experimentos que pueden arruinar la performance de Ustatic. Esos experimentos estarían enfocados en construir un stream de inserciones y borrados de tal forma que se aprovechen de que Ustatic no borra inmediatamente sus aristas. Por ejemplo un experimento donde se insertan "x" aristas y luego se borra el 99 % de "x", luego se insertan otros "x" que sean nuevos.

Referencias

[1] Arroyuelo, Diego, Guillermo de Bernardo, Travis Gagie y Gonzalo Navarro: *Faster Dynamic Compressed d-ary Relations*. Lecture Notes in Computer Science, página 419–433, 2019, ISSN 1611-3349. http://dx.doi.org/10.1007/978-3-030-32686-9_30.

[2] Coimbra, Miguel E., Alexandre P. Francisco, Luís M. S. Russo, Guillermo de Bernardo, Susana Ladra y Gonzalo Navarro: *On dynamic succinct graph representations*, 2019.