**OBSERVACIONES DEL LA PRACTICA**

Sebastián Contreras Salazar Cod 202020903

Valentina Goyeneche Cod 201922380

|  |  |  |
| --- | --- | --- |
|  | Máquina 1 | Máquina 2 |
| Procesadores | 1.6 GHz Intel Core i5 | Intel Core i5-8265U CPU |
| Memoria RAM (GB) | 8 GB 1600 MHz DDR3 | 1.6 GHz |
| Sistema Operativo | macOS High Sierra versión 10.13.6 | Windows |

Tabla 1. Especificaciones de las máquinas para ejecutar las pruebas de rendimiento.

# **Maquina 1**

## **Resultados**

|  |  |  |
| --- | --- | --- |
| ***Carga de Catálogo PROBING*** | | |
| **Factor de Carga (PROBING)** | **Consumo de Datos [kB]** | **Tiempo de Ejecución [ms]** |
| 0.30 | 1403135.95 | 48297.46 |
| 0.50 | 1403135.95 | 55018.77 |
| 0.80 | 1403135.95 | 73320.79 |

Tabla 2. Comparación de consumo de datos y tiempo de ejecución para carga de catálogo con el índice por categorías utilizando PROBING en la Maquina 1.

|  |  |  |
| --- | --- | --- |
| ***Carga de Catálogo CHAINING*** | | |
| **Factor de Carga (CHAINING)** | **Consumo de Datos [kB]** | **Tiempo de Ejecución [ms]** |
| 2.00 | 1403135.80 | 50904.72 |
| 4.00 | 1403135.80 | 58882.06 |
| 6.00 | 1403135.80 | 82042.91 |

Tabla 3. Comparación de consumo de datos y tiempo de ejecución para carga de catálogo con el índice por categorías utilizando CHAINING en la Maquina 1.

## **Graficas**

La gráfica generada por los resultados de las pruebas de rendimiento en la **Maquina 1.**

* Comparación de memoria y tiempo de ejecución para PROBING y CHAINING

# **Maquina 2**

**Resultados**

|  |  |  |
| --- | --- | --- |
| ***Carga de Catálogo PROBING*** | | |
| **Factor de Carga** (PROBING) | **Consumo de Datos [kB]** | **Tiempo de Ejecución [ms]** |
| 0,30 | 3858,02 | 111,68 |
| 0,50 | 3858,02 | 153,88 |
| 0,80 | 3858,02 | 158,27 |
| ***Carga de Catálogo CHAINING*** | | |
| **Factor de Carga (CHAINING)** | **Consumo de Datos [kB]** | **Tiempo de Ejecución [ms]** |
| 2,00 | 3857,85 | 119,75 |
| 4,00 | 3857,85 | 126,95 |
| 6,00 | 3857,85 | 118,50 |

## **Graficas**

La gráfica generada por los resultados de las pruebas de rendimiento en la **Maquina 2.**

* Comparación de memoria y tiempo de ejecución para PROBING y CHAINING

# **Preguntas de análisis**

1. ¿Por qué en la función **getTime()** se utiliza **time.perf\_counter()** en ves de la previamente conocida **time.process\_time()**?

En la función getTime() se utiliza time.perf\_counter() en vez de la previamente conocida time.process\_time(), puesto que el perf\_counter devuelve el valor en float del contador de rendimiento y sirve para medir una duración corta. Por otro lado, el time.process\_time() devuelve el valor en fracciones de segundos de la suma del tiempo del CPU del sistema y del usuario del proceso actual, con el Kernel o el nucleo Kernel (comunicación entre Python y procesador del computador).

1. ¿Por qué son importantes las funciones **start()** y **stop()** de la librería **tracemalloc**?

Son importantes las funciones start() y stop() dentro de la librería de tracemalloc, puesto que con start() se inicia el proceso de toma de datos antes de ejecutar la función del model.py e igualmente, por la función stop(), se finaliza el proceso.

1. ¿Qué cambios percibe en el **tiempo de ejecución** al modificar el factor de carga máximo para cargar el catálogo de videos?

Los cambios que se pueden percibir en el tiempo de ejecución al modificar el factor de carga máximo para cargar el catalogo de videos es que si se usa PROBING y se va aumentando el factor de carga de, por ejemplo, 0.3 a 0.5 o de 0.5 a 0.8, se va aumentar el tiempo de ejecución. Igualmente, si se usa CHAINING ocurre lo mismo que con PROBING, puesto que a medida que aumenta el Factor de Carga, aumenta tambien el tiempo de ejecución.

1. ¿Qué cambios percibe en el **consumo de memoria** al modificar el factor de carga máximo para cargar el catálogo de videos?

Los cambios que se perciben en el consumo de memoria al modificar el factor de carga maximo para cargar el catalogo de videos, en nuestro caso, se mantendra constante dependiendo del factor de carga que se use, puesto que en caso de que se use CHAINING, se obtendra un consumo de datos constante, pero es diferente, por poco, al del PROBING. Igualmente, esto se debe a que no hay un cambio en el subset de los datos y en el numero de elementos que se usan para crear los maps. Ahora bien, en PROBING, el consumo de memoria es tambien el mismo para los diferentes factores de carga, puesto que ocurre lo mismo que con el CHAINING, que es que no hay cambio en el subset de los datos o en el numero de elementos.

1. ¿Qué cambios percibe en el **tiempo de ejecución** al modificar el esquema de colisiones?, si los percibe, describa las diferencias y argumente su respuesta.

Los cambios que percibimos en el tiempo de ejecución al modificar el esquema de colisiones es que el tiempo de ejecución en va ser menor si se usa PROBING, puesto que por ejemplo, si se usa un factor de carga 0.3 en PROBING se tiene un tiempo de ejecución menor que los demás. Ahora bien, se puede decir que en tanto en PROBING y en CHAINING los tiempo de ejecución van a aumentar a medida que aumenta el factor de carga, dependiendo de si se usa CHAINING y PROBING. Del mismo modo, es posible argumentar lo dicho previamente con los resultados obtenidos en las dos maquinas, puesto que como se puede ver, el que menor tiempo de ejecución tuvo de todos los resultados fue el primero con un factor de carga de 0.3 en PROBING. Finalmente, es posible decir que hay mayores cambios en el tiempo de ejecución en el PROBING que en el CHAINING, debido a que hay una mayor diferencia entre los resultados y una mayor dispersión.

1. ¿Qué cambios percibe en el **consumo de memoria** al modificar el esquema de colisiones?, si los percibe, describa las diferencias y argumente su respuesta.

Se observa que el cambio en el consumo de memoria entre probing y chaning es mínimo, esto se debe a que por parámetro a ambos se les entrega el mismo número de elementos para generar la tabla inicial, este valor es bastante grande con el fin de suprimir la necesidad de generar un rehash en algún punto, es posible que consuma un poco más de memoria de la necesaria en ambos casos ya que se busca reservar los espacios para futuros elementos.