**OBSERVACIONES DEL LA PRACTICA**

Estudiante 1: Juan Sebastián Ortega Romero Cod 202021703

Estudiante 2: Yesid Camilo Almanza Cod 201921773

|  |  |  |
| --- | --- | --- |
|  | Máquina 1 | Máquina 2 |
| Procesadores | Procesador Intel(R) Core(TM) i7-7700HQ CPU @ 2.80GHz | Intel(R) Core(TM) i5-9300H CPU @ 2.40GHz |
| Memoria RAM (GB) | 16GB | 16GB |
| Sistema Operativo | Windows 10 Home 64-bits | Windows 10 Home 64-bits |

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 | 1469630.227 | 41541.130 |
| 0.50 | 1469630.227 | 42653.207 |
| 0.80 | 1469630.263 | 42143.161 |

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 | 1469639.981 | 44116.465 |
| 4.00 | 1469640.083 | 42705.495 |
| 6.00 | 1469640.047 | 42761.001 |

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 | 1469663.351 | 30758.606 |
| 0.50 | 1469647.230 | 31886.107 |
| 0.80 | 1469647.230 | 31796.501 |

Tabla 4. Comparación de consumo de datos y tiempo de ejecución para carga de catálogo con el índice por categorías utilizando PROBING en la Maquina 2.

|  |  |  |
| --- | --- | --- |
| ***Carga de Catálogo CHAINING*** | | |
| **Factor de Carga (CHAINING)** | **Consumo de Datos [kB]** | **Tiempo de Ejecución [ms]** |
| 2.00 | 1469639.937 | 33960.697 |
| 4.00 | 1469664.982 | 31800.484 |
| 6.00 | 1469639.937 | 34211.919 |

Tabla 5. Comparación de consumo de datos y tiempo de ejecución para carga de catálogo con el índice por categorías utilizando CHAINING en la Maquina 2.

## **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()**?

La función ***time.process\_time()*** retorna la sumatoria del tiempo que tarda la CPU en resolver un proceso determinado (en este caso sería la carga de datos). Sin embargo, esta función está limitada a un único proceso por definición, por lo tanto el esfuerzo que dicho proceso pueda tener en el resto del sistema es ignorado lo cual se distancia de una ejecución real.

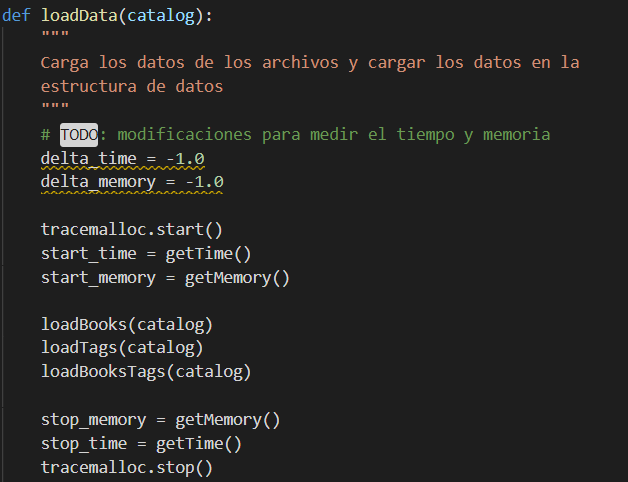
De forma contraria, la función ***time.perf\_counter()*** se basa en un reloj con la máxima resolución posible para medir un corto periodo de tiempo. De esta forma, esta función toma en cuenta la velocidad de la totalidad del sistema, haciendo que sus resultados sean mucho más precisos y apegados a un panorama real.

En adición, ya que se está evaluando el espacio utilizado en memoria es posible concluir que el programa está afectando al hardware del sistema en cuestión lo cual puede incidir directamente en la eficiencia que este tiene para resolver procesos. Por esto es preferible evaluar a la totalidad del sistema y no a un solo proceso.

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

Tanto la función **start()** como la función **stop()** cumplen con el propósito de rastrear la memoria utilizada por Python instalando y desinstalando ganchos que se alojan en la memoria del programa en cuestión. Es debido a esto que ambas funciones permiten estimar el tamaño que ocupa un proceso determinado.

El uso de ambas funciones genera un proceso cuantificable con relación al espacio que este va ocupando en el sistema. Son importantes porque sirven como puntos de referencia para la medición aproximada del tamaño que puede llegar a tomar dicho proceso bien delimitado. En este caso, la función ***start()*** debe realizarse justo antes de realizar la carga de los datos para que el proceso recién iniciado solo tenga en cuenta dicha carga. A continuación, es necesario realizar la carga de datos para que el proceso contador lo tenga en cuenta, y finalmente se utiliza la función **stop()** para finalizar el proceso contador.



Finalización de proceso de captura

Carga de datos

Inicio de proceso de captura

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

Tanto en la **Máquina 1** con en la **Máquina 2** se observo una leve tendencia en la que el tiempo de ejecución aumentaba a medida que también lo hacia el factor de carga. Esto puede ser observado tanto en *CHAINING* como en *PROBING,* no obstante,también se observaron excepciones a esta tendencia (especialmente en la máquina 2) lo que no nos permite realizar una afirmación definitiva respecto a esta regla.

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

En el caso de la **Máquina 1** es posible observar que virtualmente no hubo ningún cambio en el consumo de memoria, dado que tanto en *PROBING* como en *CHAINING* el espacio requerido por el proceso de carga de datos fue bastante similar y no presentó una diferencia considerable. Por otro lado, la **Máquina 2** si presenta cierta varianza notable en cuanto al consumo de memoria. Sin embargo, este sigue siendo mínima.

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.

En ambas máquinas es posible observar que al establecer un esquema de colisiones de tipo *CHAINING* se tiende a necesitar un mayor tiempo de ejecución a comparación de *PROBING*. Esto puede ser debido a que iterar sobre cada bucket individual resulta menos eficiente que iterar únicamente sobre la lista principal de llaves (como si pasa en PROBING).

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.

En cuanto al consumo de memoria se repite un fenómeno similar al visto en el tiempo de ejecución, por lo que el esquema de colisiones de tipo *CHAINING* tiende a ocupar un mayor espacio a comparación de *PROBING.* Este hecho puede atribuirse al hecho de que la creación de sub-listas o buckets ocupa un mayor espacio en memoria.