**OBSERVACIONES DE LA PRÁCTICA**

Jose Luis Tavera Ruiz - 201821999

Juan Diego Yepes Parra - 202022391

|  |  |  |
| --- | --- | --- |
|  | Máquina 1 | Máquina 2 |
| Procesadores | Apple M1 8-core GPU | Intel Core i5 dos núcleos 1,3 GHz |
| Memoria RAM (GB) | 16 GB | 4 GB |
| Sistema Operativo | macOS Big Sur | macOS Catalina |

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

# **Máquina 1**

## **Resultados Laboratorio**

|  |  |  |
| --- | --- | --- |
| ***Carga de Catálogo PROBING*** | | |
| **Factor de Carga (PROBING)** | **Consumo de Datos [kB]** | **Tiempo de Ejecución [ms]** |
| 0.30 | 293039,15 | 58296,88 |
| 0.50 | 290888,12 | 56931,74 |
| 0.80 | 289092,56 | 82925,50 |

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 | 327239,97 | 60023,10 |
| 4.00 | 326669,51 | 73379,70 |
| 6.00 | 326400,29 | 73745,50 |

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.

## **Gráficas**

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

## **Resultados Reto**

|  |  |  |
| --- | --- | --- |
| ***Carga de Catálogo PROBING*** | | |
| **Factor de Carga (PROBING)** | **Consumo de Datos [kB]** | **Tiempo de Ejecución [ms]** |
| 0.30 | 1488313,72 | 14346,06 |
| 0.50 | 1488313,70 | 14342,51 |
| 0.80 | 1488334,04 | 14540,87 |

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 | 1488321,88 | 14858,76 |
| 4.00 | 1488321,85 | 14663,75 |
| 6.00 | 1488321,85 | 14532,01 |

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.

## **Gráficas**

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

# **Nota:**

Tiene sentido que la memoria utilizada sea casi igual (o igual) teniendo en cuenta que los mapas implementados en el reto son muy pequeños. Ya que corresponden a: i) mapa de países de 20 elementos y ii) un mapa de categorías de 40 elementos. Asimismo, el cambio en el factor de carga no afecta el número de rehash en ninguna de las situaciones por lo que existe un comportamiento similar en los tiempos. Asimismo, esta situación corresponde únicamente al reto ya que los mapas del laboratorio tienen muchos más elementos.

# **Maquina 2**

## **Resultados**

|  |  |  |
| --- | --- | --- |
| ***Carga de Catálogo PROBING*** | | |
| **Factor de Carga** (PROBING) | **Consumo de Datos [kB]** | **Tiempo de Ejecución [ms]** |
| 0,30 | 293057,17 | 115278,10 |
| 0,50 | 290875,28 | 106708,63 |
| 0,80 | 289716,20 | 203400,52 |

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 2.

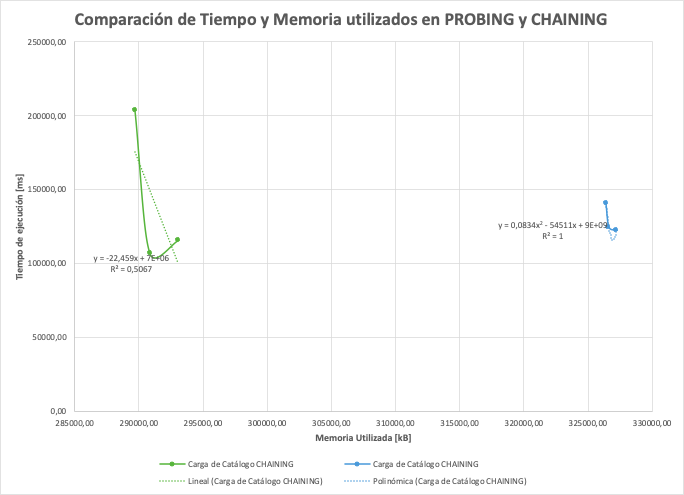
|  |  |  |
| --- | --- | --- |
| ***Carga de Catálogo CHAINING*** | | |
| **Factor de Carga (CHAINING)** | **Consumo de Datos [kB]** | **Tiempo de Ejecución [ms]** |
| 2,00 | 327219,84 | 122405,33 |
| 4,00 | 326623,19 | 124514,69 |
| 6,00 | 326399,38 | 140613,98 |

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 2.

## **Gráficas**

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



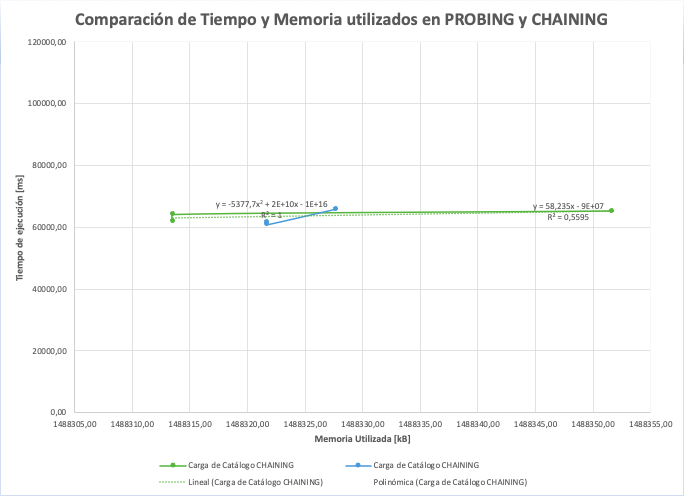
## **Resultados Reto 2**

|  |  |  |
| --- | --- | --- |
| ***Carga de Catálogo PROBING*** | | |
| **Factor de Carga** (PROBING) | **Consumo de Datos [kB]** | **Tiempo de Ejecución [ms]** |
| 0,30 | 1488351,73 | 65134,65 |
| 0,50 | 1488313,58 | 64052,66 |
| 0,80 | 1488313,56 | 61774,38 |

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 | 1488327,73 | 65797,30 |
| 4,00 | 1488321,71 | 60833,79 |
| 6,00 | 1488321,73 | 61594,14 |

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.



# **Nota:**

Tiene sentido que la memoria utilizada sea casi igual (o igual) teniendo en cuenta que los mapas implementados en el reto son muy pequeños. Ya que corresponden a: i) mapa de países de 20 elementos y ii) un mapa de categorías de 40 elementos. Asimismo, el cambio en el factor de carga no afecta el número de rehash en ninguna de las situaciones por lo que existe un comportamiento similar en los tiempos Asimismo, esta situación corresponde únicamente al reto ya que los mapas del laboratorio tienen muchos más elementos.

# **Preguntas de análisis**

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

Según la documentación de la librería time, perf\_counter() actúa como un cronómetro, ya que mide la cantidad real de tiempo que tarda un proceso, mientras que process\_time() retorna el tiempo empleado por la computadora para el proceso actual, y no para cualquier otro proceso que esté ejecutando la computadora. Es decir, se enfoca solo en una acción o una parte del código y no en todo lo que esté ejecutando la máquina.

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

Como explica la documentación del código, estas funciones inician y detienen respectivamente el proceso para medir memoria; es decir, indican cuándo empezar y dejar de medir. Esto es importante ya que la memoria utilizada varía dependiendo la función, el tiempo de ejecución y el tamaño del archivo a procesar.

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

Son directamente proporcionales; al aumentar el factor de carga aumenta el tiempo de ejecución. Ya mayores factores de carga implican más posibilidades de colisión , y por ende, más aplicaciones de rehash

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

Son inversamente proporcionales; al aumentar el factor de carga disminuye el consumo de memoria.

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.

Para el separate chaining toma más tiempo de ejecución en todos los casos en comparación con linear probing. Esto puede explicarse por su estructura interna, ya que linear probing funciona con un array\_list, que tarda O(1) en hacer las comparaciones. Sin embargo el contraste de tiempos no es muy significativo.

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.

El consumo de memoria es mucho menor en el linear probing. Esto se puede explicar en la manera como funcionan ambos métodos de colisiones, ya que para separate chaining, la manera de resolverlo es agregando al mismo bucket, un elemento a esa *mini* lista; mientras que para el linear probing se busca un espacio dentro del mapa grande existente. Es por ello que el linear probing utiliza mejor el espacio.