Informe sobre Cache Misses

1. Problemática

   Cuando se desea recorrer los elementos de una matriz se puede hacerlo de dos maneras, la primera es recorrer fila por fila y la segunda es recorrer columna por columna, a simple parecería que no hay una diferencia importante, pero se debe tener en cuenta el funcionamiento de la memoria cache, conociendo los conceptos de “spatial locality” y “temporal locality”, podemos decir por este primero que cada línea de la memoria cache es capaz de almacenar una fila de una matriz, cuyos elementos son contiguos, entonces el problema radica es el segundo concepto, ya que la temporalidad de estos datos en la cache hará que una implementación tenga mejor rendimiento que la otra.

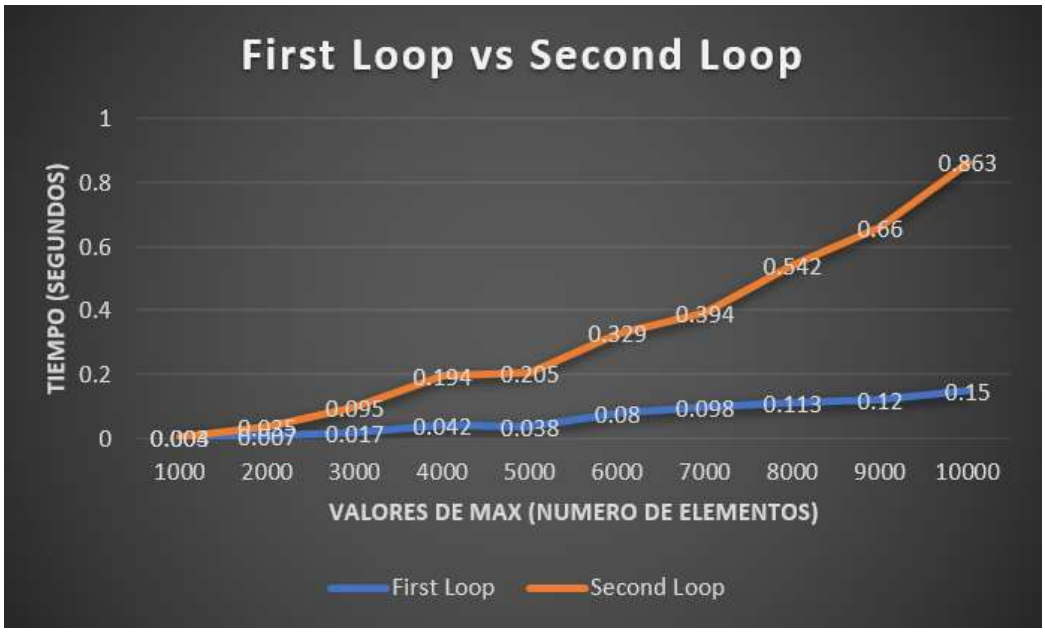
2. Experimento

   Se implemento ambas propuestas en el lenguaje de programación C++ utilizando memoria dinámica (para asegurar que los datos sean contiguos).

```
1 for (int i = 0; i < MAX; i++)
2         for (int j = 0; j < MAX; j++)
3             y[i] += A[i][j] * x[j];
```

```
1 for (int j = 0; j < MAX; j++)
2         for (int i = 0; i < MAX; i++)
3             y[i] += A[i][j] * x[j];
```

   Ambas implementaciones recorrerán exactamente los mismos arreglos y la misma matriz, por lo que están en igualdad de condiciones, entonces se procede a calcular el tiempo de ejecución (también en igualdad de condiciones), y el resultado es que recorrer fila por fila es mas eficiente que recorrer columna por columna, como se aprecia en la siguiente grafica.



   Con estos resultados surge la pregunta de ¿Por qué sucede esto?, la respuesta esta en la forma en la que los datos son tratados en la memoria cache, supongamos que la memoria cache esta vacia al momento de empezar a ejecutar ambas implementaciones, para la primera implementación (desde ahora First Loop) la memoria cache lo que hará por “spatial locality” es recuperar en cada línea cada fila de la matriz (el número de líneas en la cache es limitado), lo que es conveniente para First Loop ya que cada línea

será recorrida y todos los datos de la cache serán usados, por lo que sería una máxima eficiencia, a diferencia de la segunda implementación (desde ahora Second Loop), ya que este manejo de datos es completamente desfavorecedor para Second Loop, volvamos a pensar que la memoria cache esta vacia al momento de ejecutar, primero la memoria cache se llenara con las primeras filas de la matriz (por "spatial locality"), pero ahora solo se usara los primeros números de cada una de estas filas y luego se tendrá que vaciar el cache u volverlo a llenar con las siguientes filas para otra vez solo utilizar los primeros datos de estas, cuando se termine de usar los primeros datos empezaremos todo esto una y otra vez hasta llegar al final de cada fila, por lo que la cantidad de "cache misses" será mucho mayor que en First Loop.

3. Conclusiones

   First Loop es más rápido que Second Loop por la afinidad que tiene con el comportamiento de la memoria cache, lo que le da una eficiencia máxima a esta forma de recorrer una matriz, esto se debe a que la cantidad de cache misses de Second Loop será cuadrática, mientras que en First Loop es lineal, como se aprecia en las siguientes graficas.