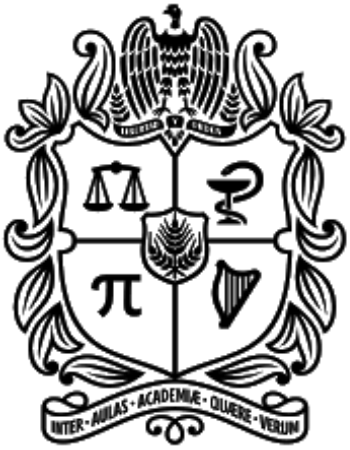


UNIVERSIDAD
NACIONAL
DE COLOMBIA

Proyecto intermedio HC

by

Alejandro Gómez alegomezse@unal.edu.co
Oscar David Peñuela odpenuelar@unal.edu.co

Este informe busca comparar la eficiencia de 3 algoritmos diferentes en el calculo de dos operaciones matriciales
Numericamente y las implementadas por dos librerias diferentes
Eigen y Armadillo

Profesor:
William Oquendo woquendo@gmail.com

May, 2021

Desarrollo

El problema consiste en analizar la rapidez de varios algoritmos que calculan la transpuesta y la multiplicación de matrices. Esto se logra midiendo la cantidad de Flops por unidad de tiempo por cada algoritmo.

- versión linux: Linux 4.16.11-100.fc26.x86 64
- compilador: gcc (GCC) 7.3.1 20180130 (Red Hat 7.3.1-2)
- sistema operativo: Fedora release 26
- librerias: papi-6.0.0.1,gnuplot-5.2.7, Armadillo 9.800.3,Eigen-3.3.8 y las librerias estandar de c y c++.

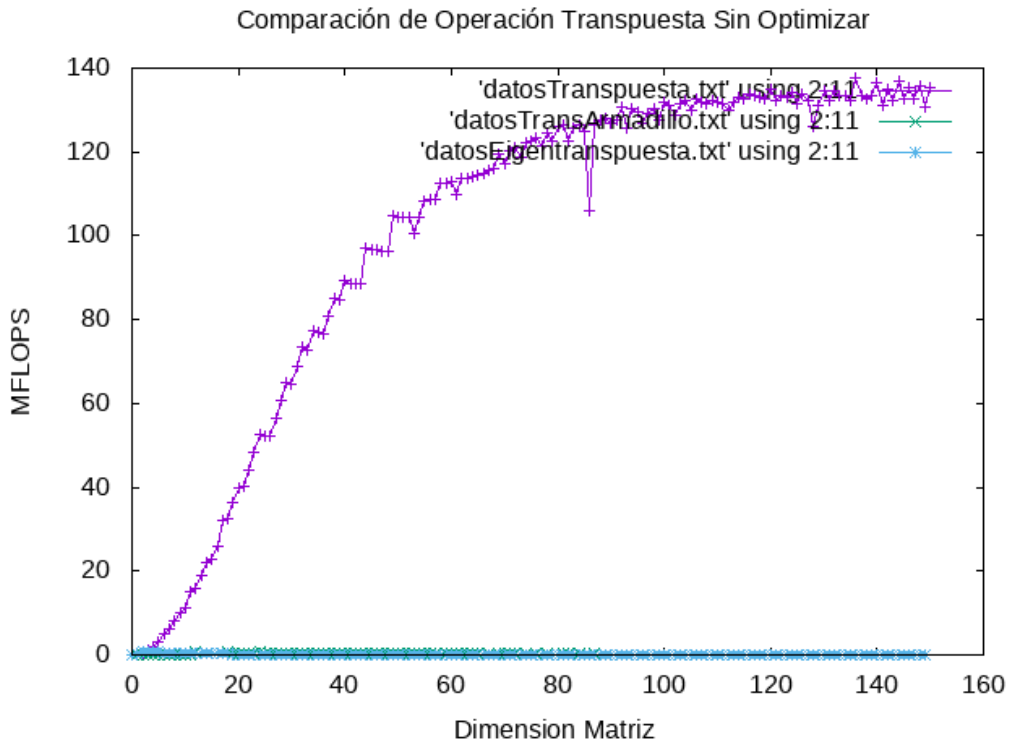
Uno de los problemas mas grandes fue la instalación y uso correcto de la libreria papi, ya que el compilador de C daba muchos problemas y cuando se lograron corregir los errores, resulta que la maquina en que inicialmente se realizaron los algoritmos no permitía operaciones de punto flotante para medir los flops. Este problema se soluciono usando el cluster de biología donde inicialmente hubo una cantidad de problemas en la instalación de armadillo y Eigen además de problemas con la ubicación de spack. Finalmente logramos resolver los problemas de dependencias usando los modulos disponibles y cambiando el /.bashrc . Al graficar los resultados obtuvimos este desempeño por caso y tipo de optimización:

Chapter 1

Resultados

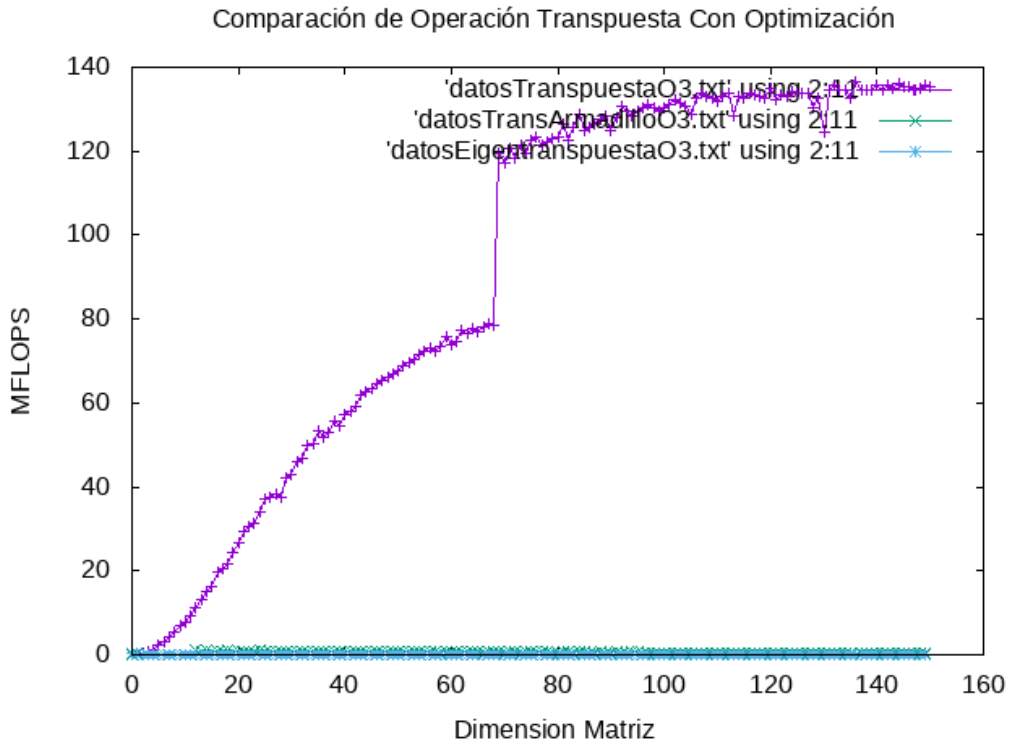
1.1 transpuesta

1.2 optimización o0



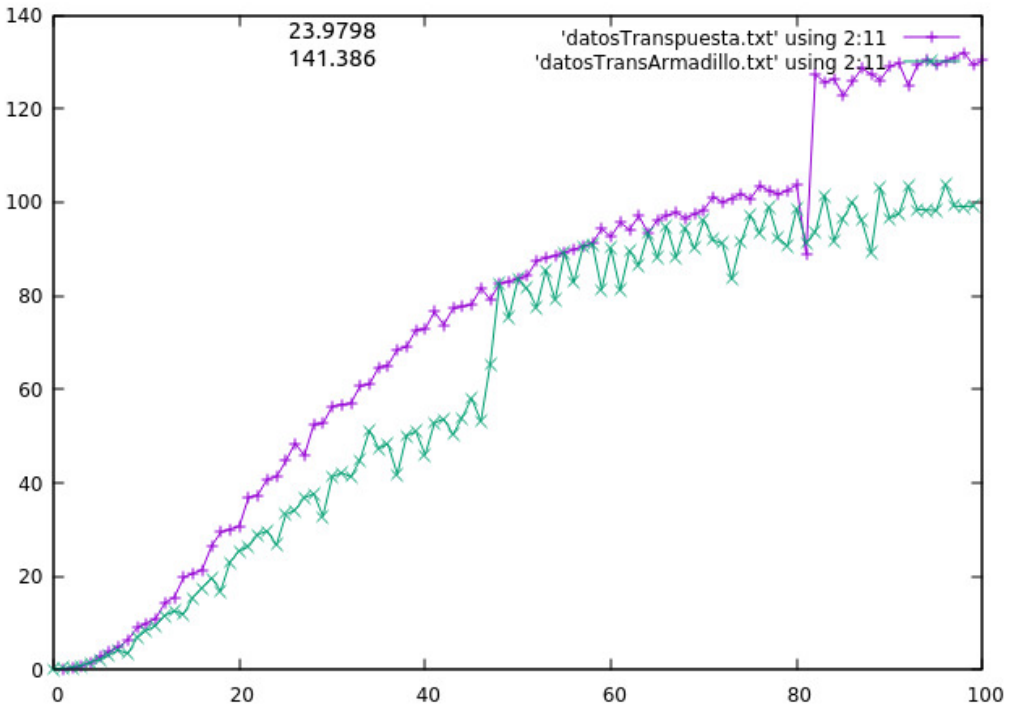
Al realizar el programa sin optimización se observa que los flops de la transpuesta son muy altos en comparación con los de los algoritmos de las librerias, las cuales se mantienen cercanas a cero.

1.3 optimización o3



los resultados son iguales para los algoritmos de las librerias, sin embargo para el calculo de la transpuesta el valor de los flops disminuye drasticamente hasta llegar a 70 donde se mantiene muy similar al caso sin optimización

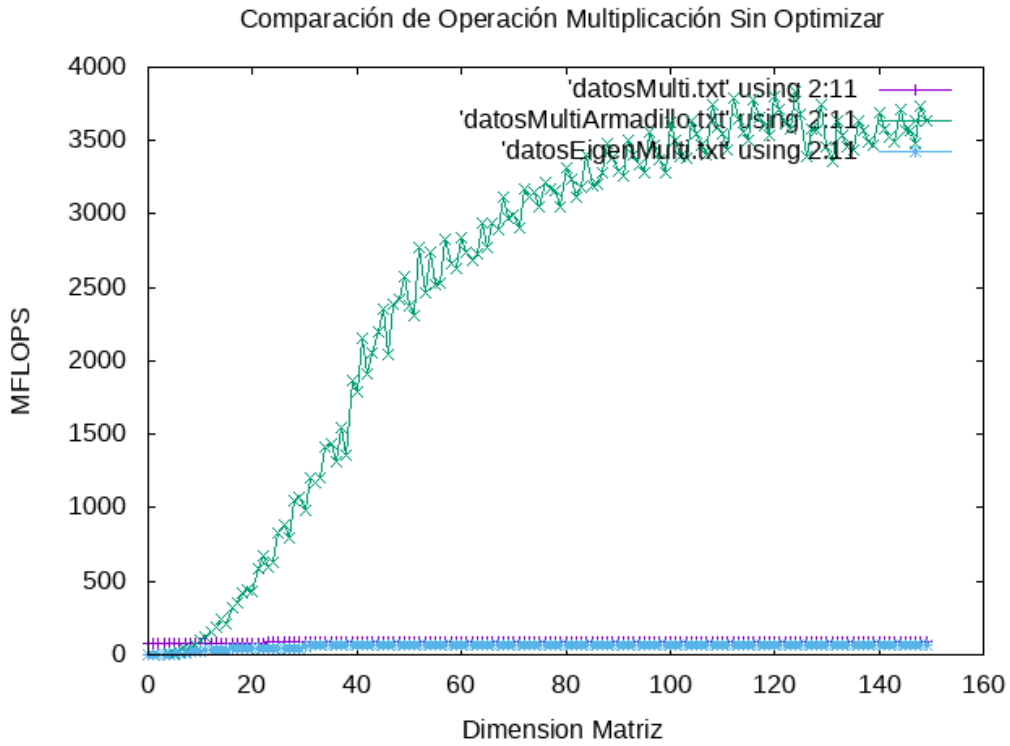
1.4 Multiplicación



optimización o0

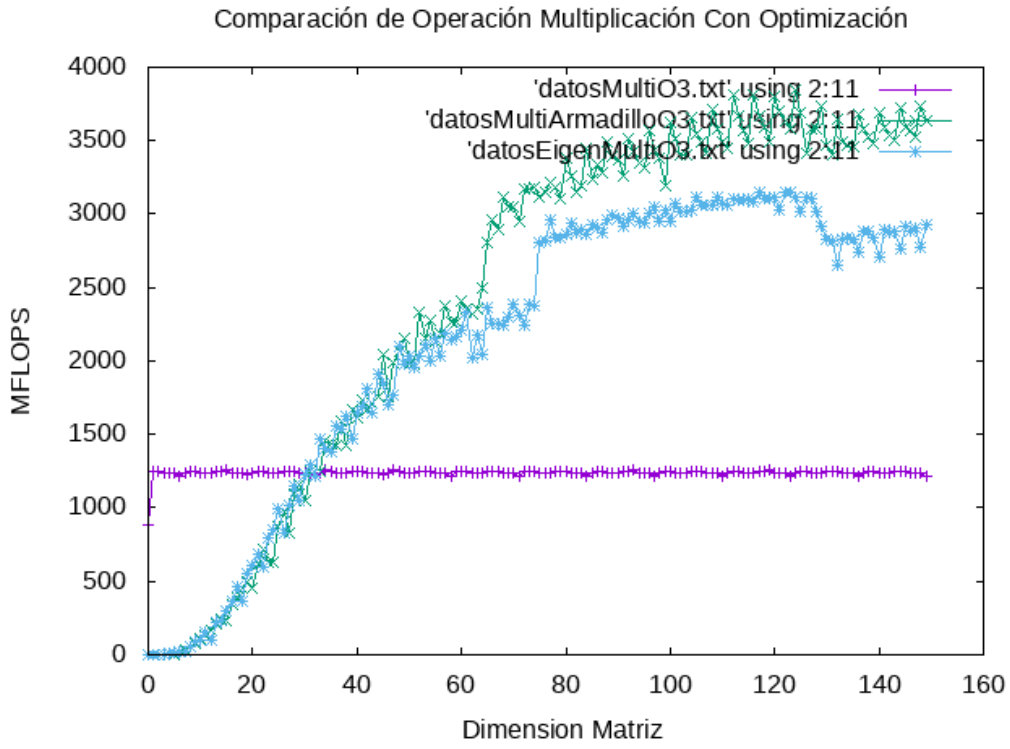
Al realizar el programa sin optimización se observa que a pesar de que las librerias hacen un mejor trabajo la mayoría del tiempo, si se encuentran puntos donde la cantidad de flops son muy similares a el calculo directo o donde la diferencia no es muy grande.

1.5 optimización o1



La optimización lleva a valores muy cercanos a 0 , excepto armadillo, esto parece ser un error en el calculo de los flops, pues este sube exponencialmente mientras que el calculo directo y Armadillo se aproximan a cero.

1.6 optimización o3



El calculo directo se mantiene constante en las optimizaciones, mientras que eigen y armadillo a pesar de ser mejores al principio, aumentan casí de manera exponencial