**Proyecto Análisis de Rendimiento de Dotplot en secuencias de ADN o proteínas**

**(Junio de 2023)**

Felipe Giraldo Diaz Ing. en Sistemas y Computación Universidad de Caldas Manizales, Colombia luis.1701721492@ucaldas.edu.co

Sergio Cardona Ramírez, Ing. en Sistemas y Computación Universidad de Caldas Manizales, Colombia Sergio.1701910394@ucaldas.edu.co

Jose David Gonzalez Ing. en Sistemas y Computación Universidad de Caldas Manizales, Colombia Jose.1701912141@ucaldas.edu.co

**Resumen-**

1. INTRODUCCION

En la Biomédica existe la técnica Dotplot que consiste en identificar coincidencias entre 2 secuencias de ADN o hasta pueden ser proteínas, esto por medio de diferentes patrones, Y ahí es donde se aplica el Dotplot, además de que también se tiene que aplicar una técnica de filtrado que sea capaz de poder correctamente la diagonal que va generando dicho Dotplot obviamente también optimizada, y además los archivos tiene que estar en el formato FASTA(que más adelante será explicado en el marco teórico)

Lo que se buscara en este Proyecto es poder implementar esta valiosa técnica, pero no de una manera normal y corriente que es hasta cierto punto sencillo, lo que se busca es que sea de una forma óptima, y para eso utilizaremos diferentes técnicas y herramientas de paralelización que nos ayudaran a que sea mas sencillo sacar dichos resultados.

1. JUSTIFICACION

Tal vez se piensan por que es que se harán pruebas con paralelización si ya se tiene el mismo resultado con el de forma lineal, pero el problema es el tiempo y los recursos, para poder generar dicho elemento Dotplot es algo que consume no muchos si no demasiados recursos y el objetivo de hacer diferentes pruebas es que se encuentre la manera en la que se pueda llevar a cabo dichos procesos ahorrando recursos y tiempo.

En resumen, lo que se busca es ahorrar al que quiera hacer dicho proceso tantos recursos de su máquina y su tiempo ya que el Dotplot sin paralelización podría costar horas y una maquina bastante potente de HPC

1. OBJETIVO GENERAL

El Objetivo General de este Proyecto es el siguiente:

Poder encontrar una forma en al que se pueda ejecutar de forma rápida y eficiente el Dotplot

1. OBJETIVOS ESPEFICFICOS
2. Generar los scripts del Dotplot: Lo que se debe hacer al principio de todo y a la vez lo más complicado será hacer todos los scripts que nos permitirá hacer el Dotplot en diferentes herramientas normalmente basadas en Python o hasta en C.
3. Ser capaz de visualizar correctamente Los Resultados: El proceso aun siento lo que mas nos importa, lo es también el resultado ya que tenemos que verificar de forma constante que los resultados deben ser los correctos en la cantidad de datos que vamos ingresando, para poder evitar errores en la paralelización del proceso.
4. Analizar el rendimiento de dichos procesos: lo que seguiría seria poder sacar las conclusiones de forma correcta seria para poder saber cómo es que es mejorado el tiempo de los procesos por medio de diferentes librerías y estrategias.
5. Filtrado de línea horizontal: Para poder ya terminar ya lo único que nos falta para poder generar un filtro que nos permita ver de forma más clara la línea que se genera en un Dotplot, entre estos se podrá.
6. MARCO TEORICO
7. Dotplot:

También Conocido como alineamiento grafico es la representación en forma de matriz de puntos, estas pueden generar varios resultados en una misma imagen como estructuras repetidas, inversiones de diferentes secuencias o hasta región conservadas [1].

1. FASTA:

Es un fichero informático. Es usado para representar las secuencias de ácidos nucleicos, de péptido y en el que los pares de bases o los aminoácidos se representan usando códigos de una única letra [2]

1. Bioinformática:

Para analizar grandes cantidades de datos a nivel molecular y genómico. **La bioinformática**obtiene estos datos y crear nuevos algoritmos para interpretarlos y adaptarlos a las necesidades del proyecto en el que están trabajando, parar mejor los sistemas de salud o para aportar al desarrollo de las comunidades a través de investigaciones y análisis.[3]

1. Paralelización:

La computación paralela implica el uso de múltiples procesadores para ejecutar distintas partes de un programa de manera simultánea, mejorando la eficiencia y reduciendo el tiempo de procesamiento en comparación con la computación secuencial. Este enfoque es crucial para manejar tareas complejas en ciencia e ingeniería, como simulaciones numéricas y procesamiento de grandes volúmenes de datos​ [4]

1. Secuencias De ADN:

Cadena que está formada por una cadena principal (esqueleto) compuesta por moléculas de azúcar y fosfato que se unen a una de cuatro bases: la adenina (A), la timina (T), la guanina (G) y la citosina (C). Las bases se emparejan entre ellas (A con T y G con C) para formar enlaces químicos que actúan como peldaños de una escalera [5]

1. Multiprocessing:

Librería de uso en Python que ofrece concurrencia local y remota, por medio de subprocesos en vez de hilos, lo que ayudara a aprovechar lo máximo la maquina con múltiples procesos [6]

1. Mpi4py:

Librería que proporciona una paralelización que se ejecutara por medio de diferentes directivas que ayudan al manejo de datos:

1. Scatter: Permite enviar diferente tipo de información a cada uno de los núcleos derivado de otra información.
2. BroadCast: Consiste en enviarle a todos los procesos la misma información
3. Gather: Consiste en toda la información dividida con el Scatter y la vuelve a unir

Permitiendo un manejo más personalizable. [7]

1. PyCUDA: Librería de Python que se encarga de del poder acceder a la memoria GPU de las tarjetas NVIDIA y con eso poder hacer mejor uso de toda la potencia de la misma por medio de un control por matrices [8]
2. METOLOGIA

Línea de comando para poder ejecutar los diferentes scripts:

mpirun: Comando de ejecución básico que permitía correr el programa.

-n #: Determina el número de tareas que va a usar cada librería.

python3: El comando que permite que lenguaje piensa correr

pruebasMulti.py: Escoger el archivo que piensa utilizar en el que se va a ejecutar con los núcleos seleccionados anteriormente

--file1=data/Salmonella.fna: Selecciona uno de los archivos a los que se va a aplicar el DotPlot

--file2=data/E\_coli.fna: Selecciona el segundo archivo a los que se va a aplicar el DotPlot

--limite=10000: Limite de elementos que se usaran para evitar la sobrecarga en equipos limitados.

--output=img/Multi2cores.png: Se escogerá del nombre del resultado que se dará sin ningún tipo de filtro.

Todos los archivos serán ejecutados en un mismo equipo para que los resultados sean coherentes y en igualdad de condiciones.

Especificaciones:

* RAM:
* Procesador:
* SSD:
* Velocidad:
* Núcleos:
* Procesadores Lógicos:

Excepto el de PyCUDA que se ejecutara en Google Colabs por limitación de hardware de parte de nuestro equipo que posee una GPU de:

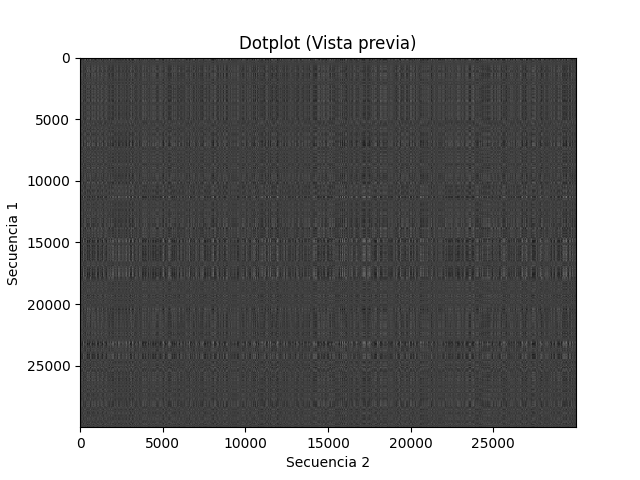
* GPU: Tesla T4
* Memoria:14 GB
* Capacidad de Computación: 7.5

1. DISCUSIÓN

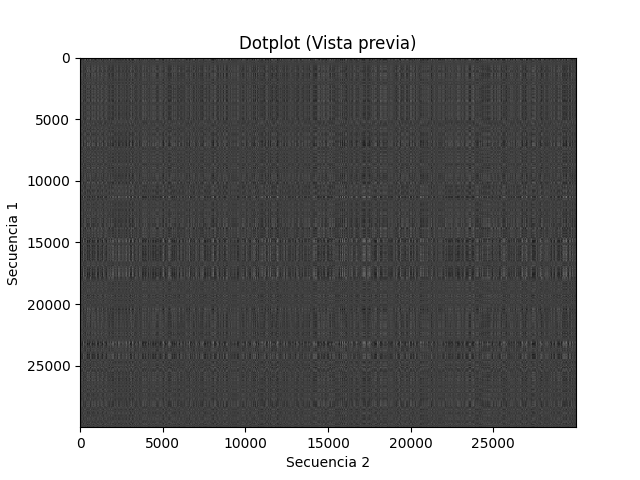
Mientras que los resultados que dieron los procesos no fueron en si sorpresas lo que si genero cierto impacto en nosotros es ver que es en ninguna de nuestras maquina se podría llegar a resolver todos los datos posibles, lo que genero cierta incertidumbre si la paralelización en muchos casos era correcta pero después de ciertos cambios nos dimos cuenta que la maquina en si era una limitante que no nos dejaría poder generar dicha operación de forma completa.

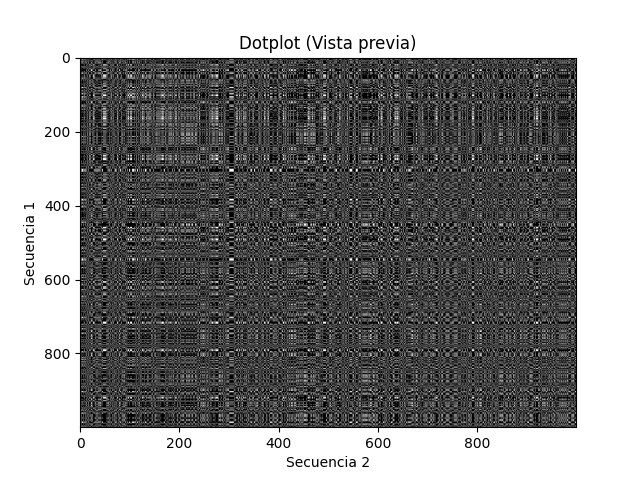
1. RESULTADOS

Secuencial:

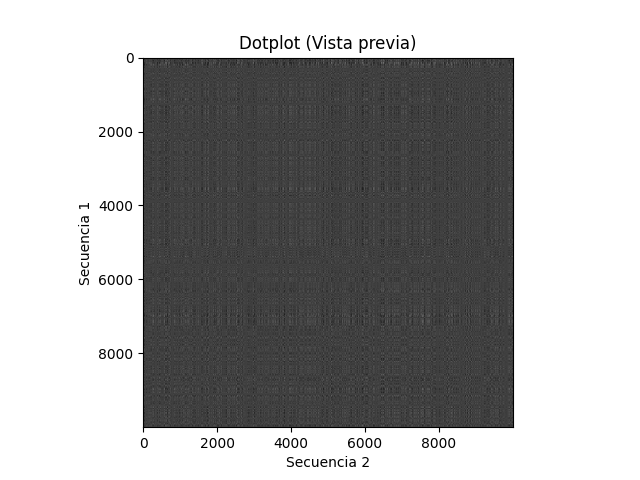


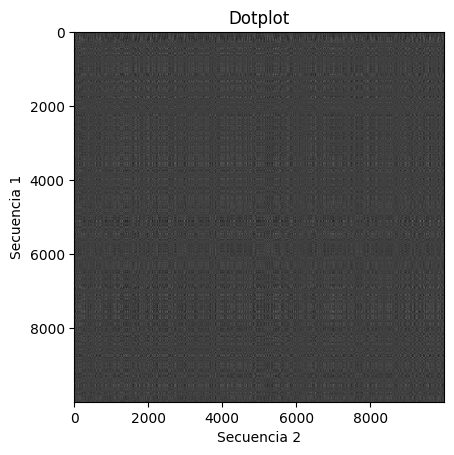
Hilos:



MPI4PY: 

Multiprocessing:



PyCUDA: 

1. CONCLUSIONES

La Conclusión a la que llegamos es que se puede llegar a optimizar el proceso de una gran manera y hasta de forma efectiva, pero lo que no se puede llegar a hacer es una que función que pueda recibir todos los elementos y quede completamente Funcional por limitaciones de maquina a pesar de que los tiempos mejoran en todas las formas de paralelización que se probaron con respecto a la Secuencial.

La especificación del Filtrado en todas partes lo llegamos a considerar hacer con sus mismas herramientas más la que poseíamos ya era optima por lo que pensamos que seria innecesario crear otro filtro que también sea paralelizado.

En el siguiente enlace llevara a todos lo elementos como a este informe si es deseado: <https://github.com/ShiomeDiaz/performance_analysis.git>

1. REFERENCIAS

[1] L. Gómez et al., "Alineamiento gráfico de secuencias a través de programación paralela: un enfoque desde la era postgenómica," ResearchGate, 2020. [En línea]. Disponible: <https://www.researchgate.net/publication/344254403_Alineamiento_grafico_de_secuencias_a_traves_de_programacion_paralela_un_enfoque_desde_la_era_postgenomica>.

[2] "FASTA format," SEQ2FUN, 2024. [En línea]. Disponible: [https://seq2fun.dcmb.med.umich.edu//FASTA/](https://seq2fun.dcmb.med.umich.edu/FASTA/).

[3] Universidad El Bosque, "Bioinformática: qué estudia y cómo se utiliza," Blog Universidad El Bosque, 2024. [En línea]. Disponible: <https://www.unbosque.edu.co/blog-universidad-el-bosque/bioinformatica-que-estudia-y-como-se-utiliza>.

[4] G. E. Karniadakis, R. M. Kirby II, y M. Y. Hussaini, "Introduction to Parallel Computing," Cambridge University Press, 2020. [En línea]. Disponible: <https://assets.cambridge.org/97811071/74399/frontmatter/9781107174399_frontmatter.pdf>.

[5] Instituto Nacional del Cáncer, "ADN," Diccionario de cáncer, 2024. [En línea]. Disponible: <https://www.cancer.gov/espanol/publicaciones/diccionarios/diccionario-cancer/def/adn>.

[6] Python Software Foundation, "multiprocessing — Process-based parallelism," Python 3.8.6 documentation, 2024. [En línea]. Disponible: <https://docs.python.org/3/library/multiprocessing.html>.

[7] L. Dalcin et al., "mpi4py: MPI for Python," PyPI, 2024. [En línea]. Disponible: <https://pypi.org/project/mpi4py/>.

[8] A. Klöckner et al., "PyCUDA," PyPI, 2024. [En línea]. Disponible: <https://pypi.org/project/pycuda/>.