[1]

PROYECTO ANÁLISIS DE RENDIMIENTO DE DOTPLOT SECUENCIAL VS PARALELIZACIÓN (Junio de 2023)

Santiago Salazar P., Jeffrey Loaiza Q., Maria Camila Gil P.

En el siguiente trabajo se expondrá el desarrollo y análisis de rendimiento de diversas formas de realizar un dotplot para la comparación de secuencias de ADN o proteínas; esto planteado como proyecto final para el curso de programación concurrente y distribuida.

I. INTRODUCCIÓN

Una de las tareas más comunes y fundamentales en la investigación genómica y de la bioinformática es la comparación de secuencias de ADN, la cual consiste en tomar dos o más secuencias de nucleótidos o proteínas en busca de las semejanzas o diferencias que estas poseen entre sí. Dentro de los métodos más usados que existen para realizar la comparación de secuencias de ADN, encontramos la comparación grafica de secuencias, la cual normalmente es representada con una matriz de puntos conocida como dot-plots, este método nos permite observar las zonas con mayor similitud, así como reconocer patrones de dos secuencias, las cuales se ubican una en el eje X y otra en el eje Y, marcando con un punto cuando se encuentra una coincidencia entre estas.

A medida que avanza la investigación genómica, se van generando cada vez más grandes volúmenes de datos de moléculas que contienen la información genética de los organismos vivos, este nuevo conocimiento nos permite entender mejor el mundo que nos rodea y nos brinda nuevas posibilidades de dar solución a problemas existentes, pero esto también conlleva desafíos computacionales en términos de la eficiencia, ya que comparar secuencias de ADN de grandes tamaños pueden llegar a tener un elevado costo en el tiempo requerido para llevar acabo estos análisis. Es por esto que la paralización y el uso de bibliotecas especializadas pueden brindarnos herramientas efectivas para la aceleración del proceso de comparación de las secuencias, lo que permite dividir la tarea en partes más pequeñas e independientes, que trabajan de forma simultanea diferentes partes de los datos, y de esta forma aprovechar los recursos de computo de manera más eficiente, reduciendo drásticamente los tiempos de ejecución que estos análisis requieren.

II. JUSTIFICACIÓN

La comparación de secuencias de ADN es una tarea fundamental en la investigación genómica y la bioinformática, ya que nos brinda información de gran relevancia sobre la estructura, la función y la evolución de los genes y organismos; por su lado a medida que aumenta la cantidad y complejidad de los datos disponibles para su investigación, es necesario desarrollar y usar métodos y herramientas que nos proporcionen una mayor eficiencia y precisión a la hora de llevar a cabo el análisis de estos datos, siendo un punto crítico el tiempo que toman realizar estos análisis.

La necesidad de mejorar la eficiencia y rendimiento de la comparación de secuencias de ADN nos obliga a buscar formas de acelerar el tiempo de ejecución de estos análisis, por esto es que la paralización y el uso de bibliotecas especializadas como mpi4py nos brindan una solución rápida y económica frente a este problema que también está presente en otros campos de la investigación.

Este proyecto nos brinda una visión clara de los diferentes enfoques que tenemos disponibles en el uso de los recursos computacionales durante la implementación y análisis de la información durante una investigación dada, lo que nos permitirán tener un mayor criterio a la hora de identificar la estrategia más eficiente y escalable según las condiciones del problema dado.

.

III. OBJETIVO GENERAL

Implementar y analizar el rendimiento de tres formas de realizar un dotplot, una técnica comúnmente utilizada en bioinformática para comparar secuencias de ADN o proteínas.

IV. OBJETIVOS ESPECÍFICOS

- Implementar un enfoque secuencial y paralelo para realizar el dotplot en la comparación de secuencias de ADN, utilizando un algoritmo estándar.
- Evaluar y comparar el rendimiento de los tres enfoques implementados en términos de tiempo de ejecución.
- Determinar la eficiencia y la escalabilidad de cada enfoque en la comparación de secuencias de ADN.

V. MARCO TEÓRICO

a. Dotplot:

[1]

Es una técnica ampliamente utilizada en bioinformática para comparar secuencias de ADN o proteínas. Consiste en la representación gráfica de las similitudes entre las secuencias, donde cada punto en la matriz representa una coincidencia entre dos caracteres de las secuencias. El dotplot es útil para identificar patrones, detectar similitudes y explorar la estructura de las secuencias. La idea básica detrás del dotplot como bien dice Darren R. Flower (1997) es que si dos secuencias comparten regiones similares, entonces habrá puntos en la matriz en las posiciones correspondientes a esas regiones.

b. Paralelización en el procesamiento de datos:

La paralelización es una técnica que permite dividir una tarea en subprocesos independientes que se ejecutan simultáneamente, aprovechando así el poder de procesamiento de múltiples recursos. Según Weizhong Li (2004) La paralelización en bioinformática abarca una amplia gama de aplicaciones, desde alineación de secuencias y búsqueda de similitudes hasta análisis de expresión génica y modelado molecular. En el contexto de este proyecto, se explorarán dos enfoques de paralelización: multiprocessing y mpi4py.

- *Multiprocessing:* es un módulo de Python que permite la ejecución de tareas en paralelo utilizando múltiples procesos en un sistema multicore. Cada proceso tiene su espacio de memoria independiente y puede realizar cálculos simultáneos.

- *MPI4PY: e*s una biblioteca de Python que proporciona una interfaz para la programación paralela utilizando el estándar de paso de mensajes (Message Passing Interface, MPI). MPI permite la comunicación y coordinación entre múltiples procesos distribuidos en un clúster o sistema de cómputo de alto rendimiento.

c. Métricas de rendimiento

Las métricas de rendimiento son medidas utilizadas para evaluar el desempeño y eficiencia de las implementaciones del dotplot. Le permiten a los investigadores y desarrolladores seleccionar la opción más adecuada para sus necesidades de análisis de secuencias genómicas. Xu, Z. Z. (2016)

Las siguientes métricas serán calculadas y analizadas:

- *Tiempos de ejecución:*
  Se medirán los tiempos de ejecución totales y parciales de las implementaciones secuencial, multiprocessing y mpi4py. Esto permitirá comparar el rendimiento de cada enfoque y determinar la eficiencia en términos de tiempo requerido para generar el dotplot.

- *Tiempo de carga de datos y generación de imagen:*
  Se analizará el tiempo necesario para cargar los datos de las secuencias en memoria y generar la imagen del dotplot. Esta métrica permitirá identificar posibles cuellos de botella y optimizar la eficiencia en la manipulación de datos.

- *Tiempo muerto:*
  El tiempo muerto es el tiempo en el que los procesos no están realizando cálculos o están esperando la finalización de otras tareas. Se registrará y analizará el tiempo muerto en cada implementación para identificar oportunidades de optimización y maximizar la utilización de los recursos computacionales.

- Aceleración y eficiencia:
  La aceleración y eficiencia son medidas de cuánto se mejora el rendimiento mediante la paralelización en comparación con la implementación secuencial
  . La aceleración se calcula como la relación entre el tiempo secuencial y el tiempo paralelo (Si este valor es mayor que 1, significa que el enfoque paralelo es más rápido, y si es igual a 1, significa que ambos enfoques tienen un rendimiento similar), mientras que la eficiencia se calcula como la relación entre la aceleración y el número de procesos utilizados (Cuanto más cerca esté la eficiencia de 1, más eficiente es el enfoque paralelo en términos de utilizar los recursos disponibles.)

- Escalabilidad:
  La escalabilidad es la capacidad de una implementación para mantener o mejorar su rendimiento a medida que se aumenta el número de procesos o recursos utilizados. Se analizará la escalabilidad de las implementaciones paralelas mediante la comparación de los tiempos de ejecución y el desempeño a medida que se incrementa el número de procesos o hilos utilizados.

  a. Fuerte: se mantiene el tamaño de la secuencia constante y se incrementa el número de procesadores utilizados. El objetivo es observar cómo el tiempo de ejecución varía a medida que se aumenta la cantidad de procesadores sin cambiar el tamaño del problema
  b. Débil: se mantiene el número de procesadores constante y se incrementa el tamaño de la secuencia. El objetivo es observar cómo el tiempo de ejecución varía a medida que se aumenta el tamaño del problema sin cambiar la cantidad de procesadores utilizados.

d. Optimización del rendimiento

La optimización de rendimiento es un aspecto crítico en los flujos de trabajo bioinformáticos para mejorar la eficiencia y reducir los tiempos de ejecución, es posible lograr un

rendimiento óptimo en los flujos de trabajo bioinformáticos, lo que permite un análisis eficiente de grandes conjuntos de datos biológicos. Liang, T (2018)

Para optimizar el rendimiento de las implementaciones paralelas, se pueden considerar las siguientes estrategias:

1. *Uso eficiente de recursos computacionales:*
   Es fundamental maximizar la utilización de los recursos computacionales disponibles, como los núcleos de procesamiento y la memoria. Esto implica distribuir adecuadamente las tareas entre los procesos, evitar la sobrecarga de recursos y utilizar técnicas de programación eficiente.

2. *Minimización de la comunicación entre procesos:*
   La comunicación entre procesos puede ser costosa en términos de rendimiento. Se deben aplicar técnicas para minimizar la cantidad de comunicación necesaria, como reducir el intercambio de datos entre los procesos y utilizar algoritmos de comunicación eficientes.

3. *Optimización de algoritmos y estructuras de datos:*
   Es importante analizar y optimizar los algoritmos y estructuras de datos utilizados en el proceso de generación del dotplot. Esto puede incluir la implementación de algoritmos más eficientes, la utilización de estructuras de datos optimizadas para el acceso y manipulación de las secuencias, y la aplicación de técnicas de paralelización específicas.

e. Filtrado de imágenes

El filtrado de imágenes juega un papel crucial en la bioinformática, ya que permite mejorar la calidad de las imágenes y extraer características relevantes para el análisis de datos biológicos. Hassan, U., Khattak (2017). Facilita el análisis de datos más precisos y contribuye al avance de la investigación en esta área.

En resumen, en este proyecto se implementará y analizará el rendimiento de tres formas de realizar el dotplot: una versión secuencial, una versión paralela utilizando multiprocessing y una versión paralela utilizando mpi4py. Se medirán y compararán diferentes métricas de rendimiento para evaluar la eficiencia y escalabilidad de las implementaciones. El objetivo es optimizar el rendimiento de las versiones paralelas y obtener conclusiones sobre la viabilidad de la paralelización en el contexto del dotplot.

VI. METODOLOGÍA

La metodología utilizada en este proyecto para implementar y analizar el rendimiento del dotplot se puede describir de la siguiente manera:

1. Entender el problema y concepto de dotplot: Se realiza una revisión del concepto de dotplot y su aplicación en bioinformática para comparar secuencias de ADN o proteínas. Se exploran las características y utilidades del dotplot en el análisis de similitudes y estructuras de secuencias.

2. Implementación secuencial del dotplot: Se implementa la generación del dotplot secuencialmente, leyendo las secuencias desde archivos FASTA y calculando el dotplot. Se genera la imagen resultante del dotplot.

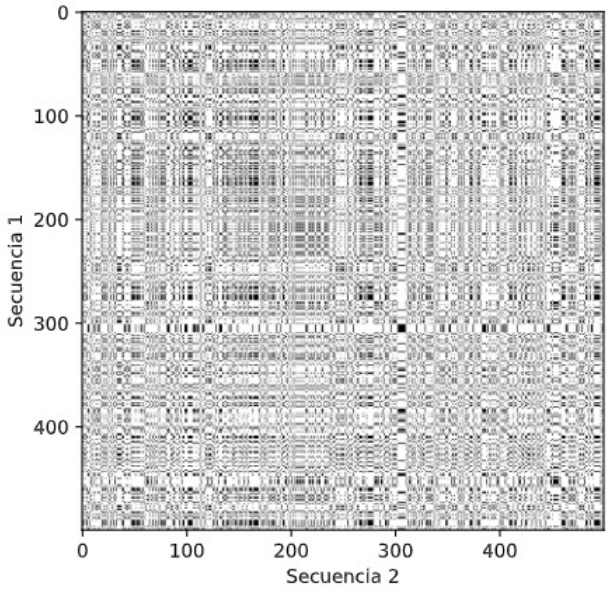


Figura 1: Generación secuencial del dotplot

3. Implementación paralela con multiprocessing: Se implementa una versión paralela del dotplot utilizando la biblioteca `multiprocessing` de Python. Se identifican las partes del código que pueden ejecutarse en paralelo y se utilizan múltiples procesos para acelerar el proceso de generación del dotplot.

4. Implementación paralela con mpi4py: Se implementa otra versión paralela del dotplot utilizando la biblioteca `mpi4py`. En esta implementación, se divide las secuencias en partes y se distribuye el trabajo entre los procesos de MPI. Se utilizan las funciones de comunicación adecuadas, como `comm.send()` y `comm.recv()`, para enviar y recibir datos entre los procesos.

5. Implementación de la función de filtrado de imagen: Se implementa una función paralela para realizar el filtrado de la imagen generada por el dotplot y detectar las líneas diagonales. Esta función puede ser implementada de manera general para todas las versiones (secuencial, paralela con multiprocessing y paralela con mpi4py), adaptando la forma de implementación según la biblioteca o enfoque utilizado.

6. Análisis de rendimiento: Se calculan y analizan las métricas de rendimiento solicitadas para cada implementación. Se registran los tiempos de ejecución totales y parciales, el tiempo de carga de datos y generación de imagen, el tiempo muerto, la aceleración, la eficiencia y la escalabilidad. Estas métricas permiten comparar el rendimiento de las implementaciones y evaluar su eficiencia y escalabilidad en función de los recursos utilizados.

1. Tiempos de ejecución

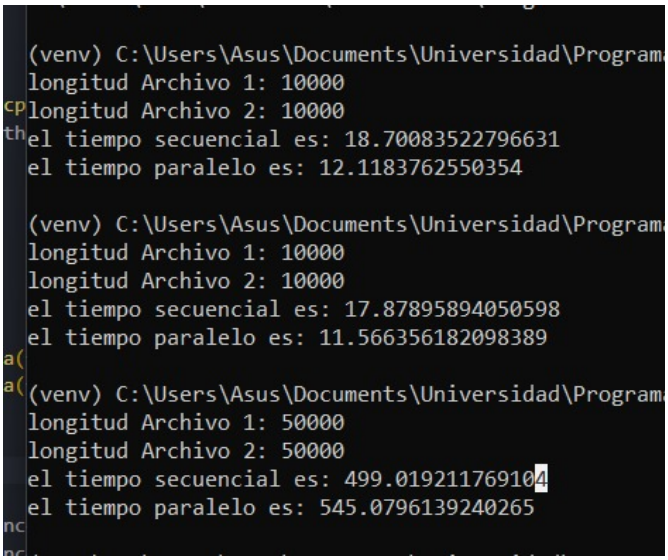


Figura 2: Comparación de tiempos entre las implementaciones secuencial y paralela.

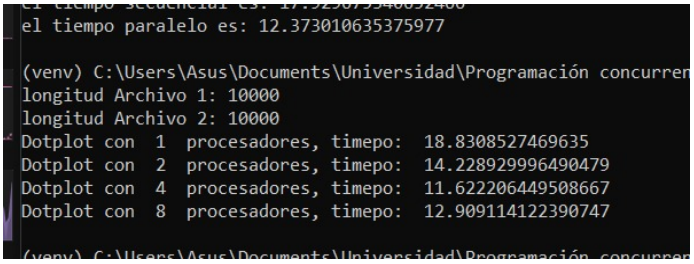


Figura 3: Tiempo de ejecución con diferente número de procesadores con enfoque paralelo

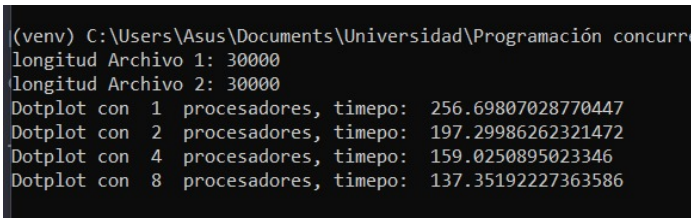


Figura 4: Tiempo de ejecución con aumento en el tamaño de las secuencias y con diferente número y procesadores con enfoque paralelo.

2. Escalabilidad (Fuerte: mayor tamaño de secuencias, Debil: mayor número de procesadores)

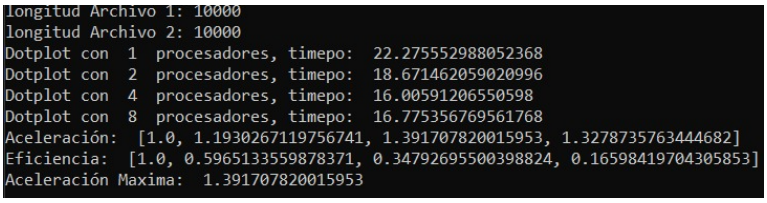


Figura 5: Cálculo de la aceleración (división entre el tiempo secuencial (times[0]) por cada tiempo paralelo en la lista times) y de la eficiencia que indica qué tan eficientemente se están utilizando los recursos paralelos en relación con el número de procesadores utilizados (Mayor tamaño de secuencias)

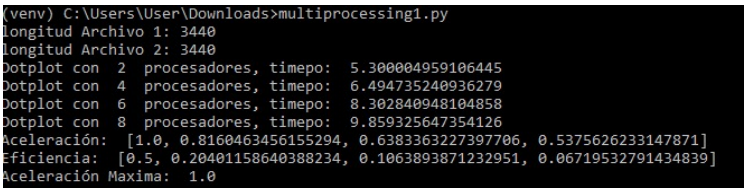


Figura 6: Cálculo de la aceleración (división entre el tiempo secuencial (times[0]) por cada tiempo paralelo en la lista times) y de la eficiencia que indica qué tan eficientemente se están utilizando los recursos paralelos en relación con el número de procesadores utilizados (menor tamaño de las secuencias)

3. Aceleración y eficiencia

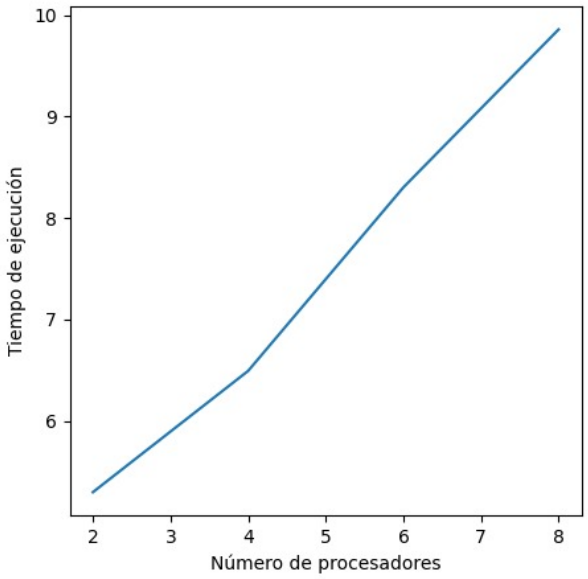


Figura 7: Gráfico que muestra cómo varía el tiempo de ejecución en función del número de procesadores utilizados en el enfoque paralelo.

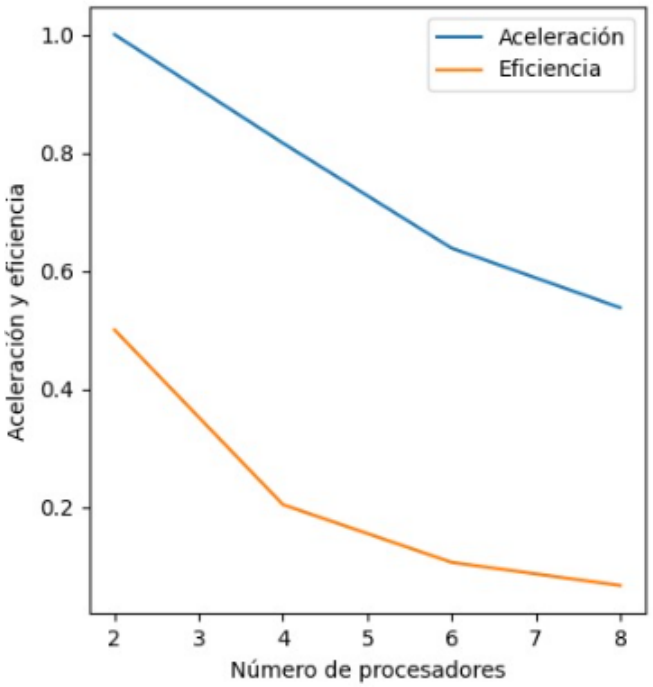


Figura 8: Gráfico que muestra la aceleración y la eficiencia en función del número de procesadores.

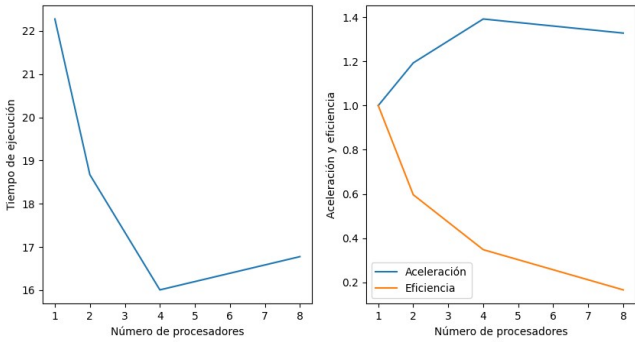
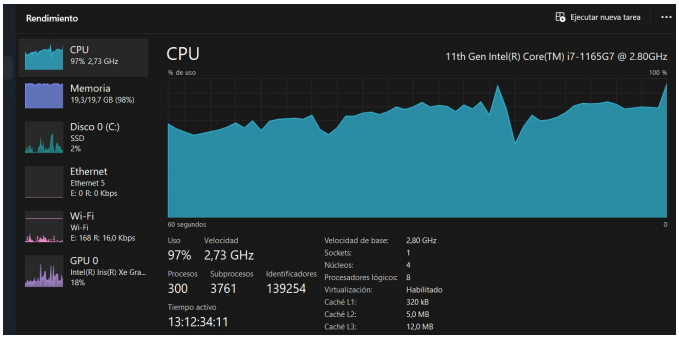


Figura 9: Figura 8: Gráfico que muestra la aceleración y la eficiencia en función del número de procesadores.(Con auto en el tamaño de las secuencias)

7. Optimización del rendimiento: Se exploran estrategias para optimizar el rendimiento de las implementaciones paralelas, como el uso eficiente de recursos computacionales, la minimización de la comunicación entre procesos y la optimización de algoritmos y estructuras de datos utilizados en el proceso del dotplot.



Gráfica 9: Evidencia del rendimiento después de los procesos aplicados.

.

VII. Discusión

Poder dividir la tarea en múltiples subprocesos independientes permite una reducción drástica del tiempo de ejecución, ya que nos brinda una capacidad de trabajo simultaneo más eficiente, lo cual, respaldado con los resultados obtenidos, nos muestra que la paralización y el uso de mpi4py pueden mejorar significativamente el rendimiento del análisis de grandes volúmenes de datos como lo hicimos con el dotplot para la comparación de secuencias de ADN.

Es importante destacar que el rendimiento de los enfoques paralelos y librerías como mpi4py son mucho más notables en volúmenes de datos mayores, aunque estos requieren de recursos adicionales, como hardware especifico, conocimientos y más tiempo para su implementación a diferencia del enfoque secuencial

VIII. Conclusiones

Durante el desarrollo del proyecto se implementaron y compararon los enfoques: secuencial, paralelo y uso de la librería mpi4py, para realizar el dotplot en la comparación de diversas secuencias de ADN, demostrando que tanto el enfoque en paralelo como el mpi4py presentaron mejoras significativas en el rendimiento del procesamiento de los datos en comparación con el enfoque secuencial. El uso de las diferentes métricas utilizadas para este análisis demuestra la importancia de considerar estrategias de optimización y paralelización en el procesamiento de datos genómicos; sin olvidarnos del enfoque secuencial a pesar de las limitaciones que este posee en términos de tiempo de ejecución con respecto a los otros enfoques, este posee una mayor simplicidad y facilidad de implementación, además de la capacidad para manejar conjuntos de datos pequeños, también nos permite trabajar en diferentes entornos de trabajo en donde no se cuentan con la infraestructura de computo requerida.

Es gracias a las diferentes ventajas que pueden ofrecernos los diferentes enfoques frente a los procesos de análisis de datos, como el permitir realizar cálculos y análisis de cada vez

mayores cantidades de datos, la investigación genómica, así como de otros campos de la ciencia pueden aumentar nuestra comprensión del mundo que nos rodea.

IX. Referencias

[1] J. S. Piña, S. Orozco-Arias, N. Tobón-Orozco, L. Camargo-Forero y R. Tabares-Soto, «G-SAIP: Graphical Sequence Alignment Through Parallel Programming in the Post-Genomic Era,» Sage Journals, vol. 19, pp. 1-10, 2022.

[2] J. López Fernández, M. Rubio Camarillo y S. Peris Iniesta, «Universidad Complutense Madrid,» 06 02 2014.

[3] F. Alvarez, «Universidad de la Republica,» 26 06 2020.

[4] Hassan, U., Khattak, N. S., Basit, A., Latif, K., & Jaffar, A. (2017). Image Filtering Techniques for Bioinformatics Applications. IEEE Access, 5, 15191-15211

[5] Flower, D. R., & Attwood, T. K. (1997). Dot Plots in Bioinformatics. Bioinformatics, 13(5), 516–522.

[6[Liang, T., Rivera, R., Li, W., & Mirabello, C. (2018) Performance Optimization Techniques for Bioinformatics Workflows. Future Generation Computer Systems, 88, 703-714.

[7] Li, W., & Zola, J. (2004). Parallel Computing in Bioinformatics. Briefings in Bioinformatics, 5(3), 237–248.

[8] Xu, Z. Z., & Chen, J. (2016). Performance Evaluation of Dotplot Implementations for Genomic Sequence Analysis. BMC Bioinformatics, 17(Suppl 11), 390