Universidad Nacional de San Agustín de Arequipa

Facultad de Ingeniería de Producción y Servicios

Escuela Profesional de Ciencia de la Computación

Tesis Titulada:

Optimización del Algoritmo BLAST en el Alineamiento de Secuencias de ADN Basado en Procesamiento Masivamente Paralelo y Distribuido

Tesis presentada por el bachiller ***Franklin Luis Antonio Cruz Gamero*** *para optar el Título Profesional de* ***Licenciado en Ciencia de la Computación****.*

Asesor: Dr. Juan Carlos Gutierrez Caceres

AREQUIPA - PERÚ
2019

Dedicado a mi esposa Candy por su enorme apoyo en la culminación de esta hermosa profesión y a mis hermosos hijos Joaquin y Maria Jose, por ser todos la luz que ilumina mi sendero

Agradecimientos

A Dios y la Santísima Virgen María, por darme la existencia y orientar en cada instante mis pasos en el sendero de la vida.

A mis padres Luz Maria y Victor Raúl por haberme permitido nacer y gozar de una familia.

A mi asesor Juan Carlos Gutierrez Caceres por su apoyo incondicional y guía permanente en la obtención de los objetivos propuestos en esta hermosa carrera de Ciencia de la Computación.

A la Universidad Nacional de San Agustín de Arequipa, por haberme dado la oportunidad de formarme como profesional y apoyarme con la subvención a este proyecto de tesis con número de contrato TT-0150-2016

A los monitores de UNSA INVESTIGA por su apoyo brindado durante la elaboración de informes de avances del presente proyecto.

A mi esposa Candy y a mis hijos Joaquín Francisco y Maria Jose, por todo el cariño y apoyo que me brindaron, con cuya motivación pude culminar exitosamente esta carrera profesional.

Resumen

En Bioinformática intentan definir modelos matemáticos de sistemas biológicos usando grandes cantidades de *Unidades de Procesamiento Centrales* (CPUs), generando aplicaciones poco prácticas. Esto está siendo optimizado por paralelismo usando *Unidad de Procesamiento Gráfico* (GPU) y sistemas distribuidos. En este trabajo se presenta dos algoritmos de optimización del algoritmo *Basic Local Alignment Search Tool* (BLAST), basado en tablas Hash, para el alineamiento de una secuencia (unisecuencial) y para el alineamiento de múltiples secuencias (multisecuencial) de consulta de *Ácido Desoxirribonucleico* (ADN), usando técnicas masivamente paralelas y distribuidas, mediante el modelo de programación con *Compute Unified Device Architecture* (CUDA) y uso de la GPU. Comparando su rendimiento en implementaciones secuenciales usando CPUs e implementaciones con la GPU. Evaluando su rendimiento en tiempo de procesamiento usando secuencias de ADN de referencia obtenidas de las bases de datos públicas *National Center for Biotechnology Information* (NCBI) y EMSEMBL como el genoma humano, mostrando los mejores rendimientos los algoritmos *Cuda Naive* (CN) para BLAST unisecuencial con un speedup de latencia de 1.24X sobre el algoritmo *Knut Morris Pratt* (KMP) y *Cuda Base 5* (CB5) para BLAST multisecuencial con un speedup de 1.23X sobre el algoritmo *Base 5* (B5), ambos con arquitectura GPU (con paralelismo) mejorando en el tiempo de procesamiento la heurística del BLAST.

Palabras Claves: ADN, optimización, paralelismo, alineamiento, GPU, BLAST

Abstract

In Bioinformatics, they try to define mathematical models of biological systems using large amounts of Central Processing Units (CPUs), generating little practical applications. This is being optimized by parallelism using Graphical Processing Unit (GPU) and distributed systems. This paper presents two algorithms for optimization of the Basic Local Alignment Search Tool (BLAST), based on Hash tables, for the alignment of a sequence (unisequential) and for the alignment of multiple sequences (multisequential) of Deoxyribonucleic acid (DNA), using massively parallel and distributed techniques, through the programming model with Computer Unified Device Architecture (CUDA) and use of the GPU. Comparing its performance in sequential implementations using CPUs and implementations with the GPU. Evaluating its performance at processing time using reference DNA sequences obtained from the National Center for Biotechnology Information (NCBI) and EMSEMBL databases as the human genome, showing the best performance of the Cuda Naive (CN) algorithms for unisequential BLAST with a speedup of 1.24X latency over the Knut Morris Pratt (KMP) and Base Base 5 (CB5) algorithm for multi-sequential BLAST with a speedup of 1.23X over the Base 5 (B5) algorithm, both with GPU architecture (with parallelism) improving heuristic processing time of BLAST.

Keywords: DNA, optimization, parallelism, alignment, GPU, BLAST

Índice general

Índice de figuras

Índice de tablas

Lista de Abreviaturas

AFD *Autómata Finito Determinista*

ADN *Ácido Desoxirribonucleico*

ARN *Ácido Ribonucleico*

B5 *Base 5*

BF *Brute Force o Naive*

BLAST *Basic Local Alignment Search Tool*

BM *Booyer-Moore*

BWT-SW *Software de la Transformada de Burrows Wheeler*

BWT *Transformada de Burrows Wheeler*

CB5 *Cuda Base 5*

CBM *Cuda Booyer-Moore*

CKMP *Cuda Knut Morris Pratt*

CKR *Cuda Karp Rabin*

CN *Cuda Naive*

CPU *Unidad de Procesamiento Central*

CPUs *Unidades de Procesamiento Centrales*

GB *Giga Bytes*

CUDA *Compute Unified Device Architecture*

GPU *Unidad de Procesamiento Gráfico*

IIAP *Instituto de Investigaciones de la Amazonía Peruana*

KMP *Knut Morris Pratt*

KR *Karp-Rabin*

NCBI *National Center for Biotechnology Information*

pb *pares de bases*

UCSC *Universidad de California Santa Cruz*

Capítulo 1

Introducción

1.1. Contexto

El alineamiento de secuencias biológicas es usado en el campo de la biología molecular y bioinformática como una alternativa a un problema de la biología molecular que es la comparación de secuencias y poder analizar si una secuencia es homóloga a otra, hallar si una función es idéntica o similar a otra, o si tiene una estructura común con otras secuencias. Se han desarrollado diferentes técnicas de extracción de datos y algoritmos de alineamiento desde el primer secuenciamiento en 1975, siendo posible reconstruir secuencias genómicas de nuevas especies. [Elloumi and Zomaya, 2013]

El secuenciamiento de un genoma es el proceso de obtener la secuencia completa de ADN de una especie, así mismo se puede obtener la secuencia de un fragmento de ADN de este genoma (cromosoma, gen, etc.). Existen dos métodos de hacerlo: el primero es reconstruir el genoma completamente desde cero o "*de novo*" y el segundo es reconstruir el genoma tomando como referencia otro genoma de la misma especie.

El genoma humano secuenciado posee alrededor de 3 200 millones de pares de bases (pb). Requiriendo para su secuenciamiento por el segundo método, el uso de algoritmos de alineamiento de secuencias de ADN para alinear múltiples secuencias (multisecuencial), donde las secuencias a alinear o de consulta "**query**" son de corta longitud, siendo de 25 a 35 *pares de bases* (pb) en *Illumina* y *SOLiD* [Mardis, 2008]. Así mismo el alineamiento puede ser de tipo unisecuencial al comparar una secuencia corta de ADN como un gen con un genoma para encontrar secuencias homólogas u otros datos de interés.

Se han creado algoritmos que permiten este alineamiento y se pueden identificar cuatro grupos por sus características: basados en árboles de sufijos, basados en Tablas Hash, relacionados a Merge Sorting [Li and Homer, 2010] y relacionados con algoritmos evolutivos [Ticona, 2003]. Siendo los más usados los dos primeros grupos y dentro de los cuales se puede mencionar algunos como el algoritmo BLAST (basado en Tablas Hash)[Altschul et al., 1990], y BWT [Lam et al., 2008] basado en árboles de sufijos.

Con el paso de los años a aumentado vertiginosamente la cantidad de bases de datos de genomas y proteomas obtenidos por secuenciamiento a nivel mundial por lo cual para diversos estudios en ADN, *Ácido Ribonucleico* (ARN) y proteínas es indispensable usar algoritmos eficientes para procesar en menor tiempo alineamientos con la base de datos en aumento.

En Arequipa y muchas partes del mundo para realizar alineamientos es usado un framework de nombre similar BLAST que tiene como proveedor al NCBI el cual presenta modificaciones del algoritmo, según la molécula biológica en estudio (ADN, ARN o proteínas).

El paralelismo o uso de múltiples recursos simultáneamente con varias CPUs así como el uso de GPU, permite ejecutar varios cálculos en un menor tiempo de procesamiento, permitiendo la optimización de algoritmos frente a un problema dado.

1.2. Motivación

El alineamiento de secuencias permite realizar múltiples estudios en ADN, ARN y proteínas por lo cual es importante tener algoritmos eficientes que permitan optimizar su tiempo de procesamiento, y siendo el algoritmo BLAST el más utilizado, es importante su optimización usando procesamiento masivamente paralelo y distribuido, así como saber las ventajas que ofrece en tiempo de procesamiento frente a su implementación secuencial en CPU y otros algoritmos como BWT basados en árboles de sufijos.

El impacto de este trabajo será brindar a los framework existentes una alternativa de optimización del algoritmo BLAST para realizar alineamientos de secuencias con igual exactitud en sus resultados y un menor coste en tiempo de procesamiento por usar paralelismo con la GPU mediante CUDA.

1.3. Planteamiento del Problema

En la actualidad el algoritmo BLAST es el más usado en el alineamiento de secuencias de ADN por su mayor rapidez de procesamiento en comparación con otros algoritmos, aplicando una heurística de *sembrar y extender* [Li and Homer, 2010]; sin embargo el aumento de bases de datos de genomas se da a un ritmo vertiginoso por lo que es necesario reducir el tiempo de procesamiento del algoritmo BLAST para obtener resultados igual de confiables con un menor tiempo de procesamiento y poder procesar eficientemente los datos en aumento disponibles con más rapidez.

¿Cuál es el máximo porcentaje de reducción del tiempo de procesamiento del algoritmo BLAST aplicado al alineamiento de secuencias de ADN al usar programación masivamente paralela y distribuida? y ¿Cuánta diferencia posee con el algoritmo BWT al procesar un mismo alineamiento de secuencias de ADN?

1.4. Hipótesis

El algoritmo de alineamiento BLAST optimizado en el alineamiento de secuencias de ADN presentará una mayor reducción en el tiempo de procesamiento que los algoritmos BLAST (unisecuencial y multisecuencial) sin optimizar y BWT.

1.5. Objetivos

1.5.1. Objetivo General

Optimizar en tiempo de procesamiento el algoritmo BLAST en el alineamiento de secuencias de ADN usando procesamiento masivamente paralelo y distribuido.

1.5.2. Objetivos Específicos

- Identificar y analizar técnicas de alineamiento de secuencias de ADN.
- Implementar y comparar los algoritmos de alineamiento de secuencias de ADN: BLAST unisecuencial y BLAST multisecuencial con arquitectura CPU (sin paralelismo).
- Optimizar e implementar los algoritmos de alineamiento de secuencias de ADN: BLAST unisecuencial y BLAST multisecuencial con arquitectura GPU (con paralelismo).
- Evaluar y comparar el rendimiento en tiempo de procesamiento de los algoritmos optimizados de alineamiento de secuencias de ADN: BLAST unisecuencial, BLAST multisecuencial, BWT.

1.6. Variables e indicadores

1.6.1. Variable independiente

A: Algoritmo utilizado.
L: Longitud de las secuencias.

Indicadores

De la variable A:

- Tipo de algoritmo usado
- Número de núcleos GPU usados
- Número de hebras de paralelismo

- Número de nodos en el sistema distribuido

De la variable L:

- Tamaño de la secuencia de ADN de referencia
- Tamaño de la secuencia de ADN a alinear

Índices

Tamaño de la secuencia de referencia y la secuencia a alinear por el número de pb que la constituyen.

1.6.2. Variable Dependiente

T: Tiempo de procesamiento durante el alineamiento de secuencias de ADN

Indicadores

Tiempo de procesamiento.

Índices

Tiempo de procesamiento en segundos.

1.7. Aspectos metodológicos

Tipo y Nivel de investigación

Tipo de investigación

Investigación experimental, transversal, cuantitativa.

Nivel de investigación

Investigación aplicada en la Optimización de software.

1.8. Estructura de la tesis

El presente documento está dividido en 6 capítulos, siendo el capítulos 1 para la introducción, 2 para Trabajos Relacionados donde se describe el estado del arte de los principales algoritmos para alineamiento de secuencias y las diferentes optimizaciones del algoritmo BLAST, 3 para el Marco Teórico donde se analiza conceptos e información necesaria para un completo entendimiento de este proyecto, 4 para la propuesta donde se detalla la metodología usada, 5 para resultados, y 6 para las conclusiones.

Capítulo 2

Trabajos Relacionados

2.1. Consideraciones Iniciales

Se han desarrollado varios algoritmos como alternativas para el alineamiento de secuencias biológicas (ADN, ARN y proteínas), los cuales son clasificados en cuatro grupos por sus características: basados en árboles de sufijos, basados en Tablas Hash, relacionados a Merge Sorting [Li and Homer, 2010] y relacionados con algoritmos evolutivos [Ticona, 2003].

2.2. El Algoritmo Blast

El algoritmo BLAST es el más representativo de los algoritmos basados en tablas hash, el mismo que puede tener modificaciones en el sembrado con semillas espaciadas o plantillas determinadas para luego realizar su extensión como optimizaciones en el sembrado [Li and Homer, 2010]. Se demostró que la siembra con coincidencias no consecutivas mejora la sensibilidad, así una plantilla '111010010100110111' que requiere 11 coincidencias en las posiciones '1' es 55 % más sensible que la plantilla predeterminada de BLAST '1111111111' para dos secuencias de 70 % de similitud [Ma et al., 2002].

El NCBI en la actualidad ha implementado una herramienta basada en consultas para el alineamiento de secuencias de ácidos nucleicos y aminoácidos que coincidentemente posee el mismo nombre BLAST que el algoritmo de alineamiento de secuencias BLAST [Altschul et al., 1990]. Esta herramienta utiliza análisis de biosecuencias utilizando implementaciones de perfil oculto de Markov [Eddy, 2011] y posee sub-herramientas como BLASTN, BLASTP y BLASTX que procesan alineamientos de nucleótidos, alineamiento de proteínas y la traducción de ADN a proteínas respectivamente.

La idea de la indexación de una tabla hash se puede atribuir al algoritmo BLAST

[Altschul et al., 1990], el cual se basa en el paradigma de sembrar (propagar) y extender. BLAST mantiene la posición de cada subsecuencia k-mer de la consulta en una tabla hash con la secuencia k-mer como clave, y escanea las secuencias de la base de datos para buscar coincidencias exactas k-mer, llamadas semillas. BLAST extiende y une las semillas primero sin espacios (Sembrado) y luego las refina mediante una alineación de Smith-Waterman (Extensión) [Smith and Waterman, 1981]. Produce alineaciones locales estadísticamente significativas como los resultados finales.

2.2.1. El Sembrado

Existen versiones que modifican la forma como se obtiene el "***sembrado***" como: HS-BLASTN [Chen et al., 2015] que acelera la búsqueda en NCBI-BLASTN al generar un nuevo índice de semillas "*seed index*" a través de un *FM-INDEX* generado por el algoritmo *Burrows-Wheeler transform* BWT derivada de la base de datos, MerAligner [Georganas et al., 2015] que es una nueva herramienta basada en BLAST que genera una tabla hash distribuida como *seed index* basada su estructura en una matriz esparsa y una paralelización en una supercomputadora Cray XC30, donde el tamaño del genoma de trigo es mayor que el genoma humano y al ser procesados por la herramienta MerAligner se evidencia que es directamente proporcional el tamaño del genoma con el tiempo de procesamiento como lo muestra la "**Figura 2.1**".

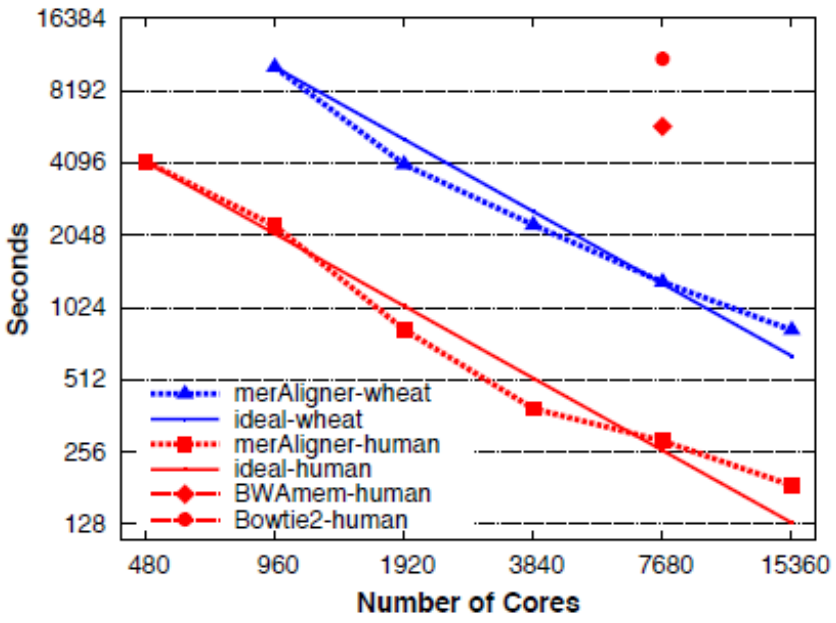


Figura 2.1: Tiempo de procesamiento (segundos) vs. número de núcleos usando tecnología GPU [Georganas et al., 2015]

2.2.2. La Extensión

El algoritmo BLAST posee varias versiones paralelizadas desde su versión inicial [Altschul et al., 1990], cuya característica común es el uso de un entorno distribuido como: GridBLAST [Krishnan, 2005] usando un *grid computing framework*, DC-BLAST [Yim and Cushman, 2017] optimizando el tiempo de las búsquedas en la plataforma NCBI BLAST al distribuir las secuencias de consulta, CloudBLAST [Matsunaga et al., 2008] combina *MapReduce* y Virtualización con el uso de varios nodos, mpiBLAST [Lin et al., 2011, Darling et al., 2003] usando *Message Passing Interface* MPI, HPC-BLAST [Sawyer et al., 2015] implementando BLAST con *High Performance Computing* HPC en clusters Intel Xeon Phi.

Otras versiones paralelizadas incluyen además de un entorno distribuido, el uso de GPU pero enfocado en la segunda fase del algoritmo BLAST que es la búsqueda de secuencias similares o la aplicación del algoritmo "***Smith-Waterman***" [Smith and Waterman, 1981] como: GPU BLAST [Vouzis and Sahinidis, 2010] usando CUDA con OPENMP y MPI exclusivamente en proteínas para acelerar el algoritmo BLAST enfocado en la paralelización con GPU en la segunda fase Smith-Waterman, G-BLASTN [Zhao and Chu, 2014] es una modificación de GPU BLAST para aminoácidos y también enfocado en la segunda fase Smith-Waterman, HCUDABLAST [Khare et al., 2017] que a diferencia de GPU BLAST usa un entorno con *Hadoop*, cuBLASTP [Zhang et al., 2017] paraleliza NCBI-BLASTP enfocado en la fase de búsqueda de secuencias o Smith-Waterman usando GPU, H-BLAST [Ye et al., 2017] paraleliza NCBI-BLASTP y NCBI-BLASTX usando GPU enfocandose en la segunda fase Smith-Waterman.

2.3. Otros Algoritmos de alineamiento

Debemos tomar en cuenta que un algoritmo de alineamiento puede ser usado en secuencias de ADN, ARN o proteínas en su forma primaria.

El algoritmo *Software de la Transformada de Burrows Wheeler* (BWT-SW) que es una de las representaciones de la BWT fue propuesto en el 2008 pero con la clara desventaja ante el BLAST debido a un alto costo en cuanto al rendimiento en tiempo de procesamiento [Lam et al., 2008] y como se muestra en la "**Figura 2.2**".

Query length	5 K	10 K	100 K	1 M	10 M	100 M
BWT-SWaverage time (s)	82	161	1.4 K	8.9 K	34.4 K	218.2 K
BLAST	19.9	29.6	93.4	775	6.7 K	92.2 K

Figura 2.2: Tiempo de procesamiento del algoritmo BWT-SW vs. BLAST sin paralelismo [Lam et al., 2008]

Se ha propuesto un método práctico llamado BCR se demora 12 horas para procesar 100 gigabases en una máquina [Bauer et al., 2013] para obtener la BWT con un tiempo de lectura de cadena similar a BWT-SW y en trabajos posteriores se ha optimizado usando GPU y CUDA creando una herramienta llamada CX1 para la construcción de BWT, que es la primera herramienta que puede aprovechar el paralelismo determinado por GPU, Así una colección lectura corta de hasta 100 gigabases se puede construir en menos de 2 horas [Liu et al., 2014] como se observa en la "**Figura 2.3**".

	100M (100-bp)	50M (200-bp)	25M (400-bp)
BCR	6141	9334	23950
CX1	565	724	1269

Figura 2.3: Tiempo de procesamiento de la herramienta BCR vs. CX1 expresado en segundos[Liu et al., 2014]

2.4. Consideraciones Finales

Tomando en cuenta las revisiones presentadas en este capítulo podemos apreciar en las figuras 2.1, 2.2 y 2.3 diferencias en cuanto a tiempo de procesamiento entre diferentes algoritmos donde destaca la gran reducción de tiempo de procesamiento en los algoritmos de alineamiento optimizados por paralelismo CUDA y uso de GPU.

Todas estas aplicaciones optimizan el tiempo de ejecución del BLAST y hacen uso de diversas bibliotecas y plataformas, sin embargo no han optimizado el algoritmo BLAST en la construcción del índice de sembrado usando GPU.

Capítulo 3

Marco Teórico

3.1. Conceptos de Biología Molecular

3.1.1. La vida y la herencia

El postulado principal de la teoría celular establece que toda célula se genera de una célula similar preexistente. Esta teoría formalizó la vieja observación de que la progenie de una especie biológica, al alcanzar el estado adulto, está compuesta de individuos con características similares a las de sus progenitores. La idea de que todos los organismos están formados por células la propuso por primera vez R. Hooke, alrededor de ¡1660!, al observar al microscopio unos cortes finos de corcho. Sin embargo, la teoría celular, como la conocemos actualmente, se fue construyendo a mediados del siglo XIX, con base en las observaciones de varios investigadores. En 1838, M. Schleiden encontró que los tejidos vegetales están formados por células y, unos cuantos años después, T. Schwann extendió esta observación a los animales. La trascendencia de este conocimiento se comprendió en 1858, cuando R. Virchow demostró que una célula no se genera espontáneamente, sino que proviene de otra célula igual.

Un año después, en 1859, Charles Darwin propuso La teoría de la evolución en su libro El origen de las especies, explicando el origen de la inmensa diversidad de los seres vivos. Esta teoría, como la teoría celular, tiene que ver con la herencia; es decir, con la generación de organismos similares a sus progenitores. Darwin propuso dos ideas principales: 1) los organismos actuales son descendientes modificados de ancestros comunes, y 2) la fuerza principal que dirige los cambios evolutivos de los organismos es la selección natural.

Las reglas de la transmisión de las características de la célula madre a la célula hija y de los padres a los hijos, o herencia, no se entendieron claramente sino hasta principios del siglo XX. El monje G. Mendel estableció las leyes de la herencia en 1865, unos cuantos años después de la publicación del libro El origen de las especies; sin embargo, no se reconoció su importancia sino hasta su "redescubrimiento" en 1905 por De Vries.

3.1.2. Los Ácidos Nucleicos

Las células contienen varias macromoléculas, como son los polisacáridos, las proteínas y los ácidos nucleicos. Estas macromoléculas pueden clasificarse como polímeros; es decir, moléculas que tienen una unidad estructural que se repite muchas veces. Los polisacáridos se forman por la unión de monosacáridos, las proteínas por la de aminoácidos y los ácidos nucleicos por la de nucleótidos. Los ácidos nucleicos son de dos clases: a) ácido desoxirribonucleico o ADN, que tiene como unidad estructural al desoxirribonucleótido o desoxinucleótido, y b) el ácido ribonucleico o ARN, que tiene al ribonucleótido. [Jiménez García, 2003]

3.1.3. El ADN

La molécula de DNA o ADN es un polímero que se forma por enlace covalente de miles de desoxinucleótidos. El desoxinucleótido, o unidad estructural del ADN, contiene un ácido fosfórico, un azúcar de 5 átomos de carbono o pentosa y una base nitrogenada. El ADN contiene 4 bases nitrogenadas: adenina (A) y guanina (G), que derivan de la purina, y citosina (C) y timina (T), que derivan de la pirimidina ("**Figura 3.1**"). El enlace de la pentosa (que en el caso del ADN es una desoxirribosa) y una base nitrogenada forma una molécula denominada deoxinucleósido. La unión de un ácido fosfórico a esta molécula forma el desoxinucleótido. "**Figura 3.1**". [Jiménez García, 2003]

Los desoxinucleótidos se unen para formar el polímero lineal, o polidesoxinucleótido, en el cual el grupo fosfato en posición 5' de la desoxirribosa se une por un enlace éster con el hidroxilo 3' de la desoxirribosa del desoxinucleótido vecino, y así sucesivamente. El polímero forma una hebra donde se alternan una desoxirribosa y un fosfato, mientras que las bases se unen perpendicularmente a ésta. "**Figura 3.2**".

La proporción relativa de cada una de las bases del ADN, así como el orden o secuencia en que se encuentran, varía de organismo a organismo, ya que la secuencia de bases, específica para cada organismo, contiene la información genética que define a ese organismo. Actualmente, existe la tecnología que permite conocer la secuencia de bases de todo el ADN de un organismo.

La estructura secundaria del ADN propuesta por Watson y Crick es una hélice de giro a la derecha formada por dos hebras de polidesoxinucleótidos orientadas en sentido antiparalelo; es decir, el extremo 5' de una hebra queda frente al extremo 3' de la otra. Esto significa que, en uno de los extremos de la molécula de ADN, una hebra tiene el ácido fosfórico que se une al carbono 5' de la desoxirribosa libre y en el otro extremo contiene la desoxirribosa con el –OH 3' libre; la otra hebra tiene, frente al ácido fosfórico 5', la desoxirribosa con el –OH 3' libre y frente a la desoxirribosa con el –OH libre, un ácido fosfórico 5'.

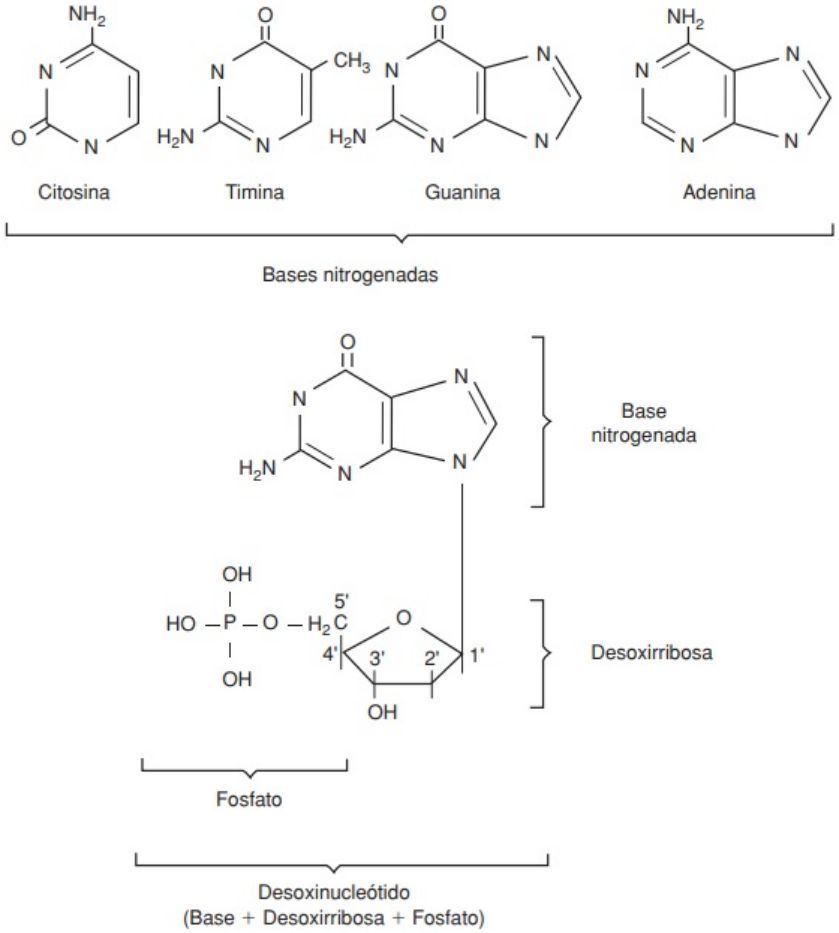


Figura 3.1: Estructura química del ADN. La molécula base en el ADN es el desoxinucleótido. Un desoxinucleótido contiene una base nitrogenada que puede ser citosina, timina, guanina o adenina unida a un azúcar (desoxirribosa) y un ácido fosfórico unido a la desoxirribosa. En la molécula de desoxirribosa se numeran los átomos de los 5 carbonos que la constituyen

3.1.4. Duplicación, Transcripción y Traducción

La información genética de las células se encuentra en el ADN, se transcribe a moléculas de ARN y finalmente se traduce a proteínas. Esta regla, o dogma central de la biología molecular, se completó al descubrirse la transcriptasa reversa, una enzima que sintetiza ADN a partir de una molécula de ARN. "**Figura 3.3**".

Duplicación

El proceso de Duplicación o Replicación del ADN es el mecanismo que permite al ADN duplicarse (es decir, sintetizar una copia idéntica) en la cual se copia el ADN progenitor en moléculas hijas idénticas al ADN progenitor. De esta manera de una molécula de ADN única, se obtienen dos o más réplicas". Esta duplicación del material genético se produce de acuerdo con un mecanismo semiconservador, lo que indica que los dos polímeros complementarios del ADN original, al separarse, sirven de molde cada una para

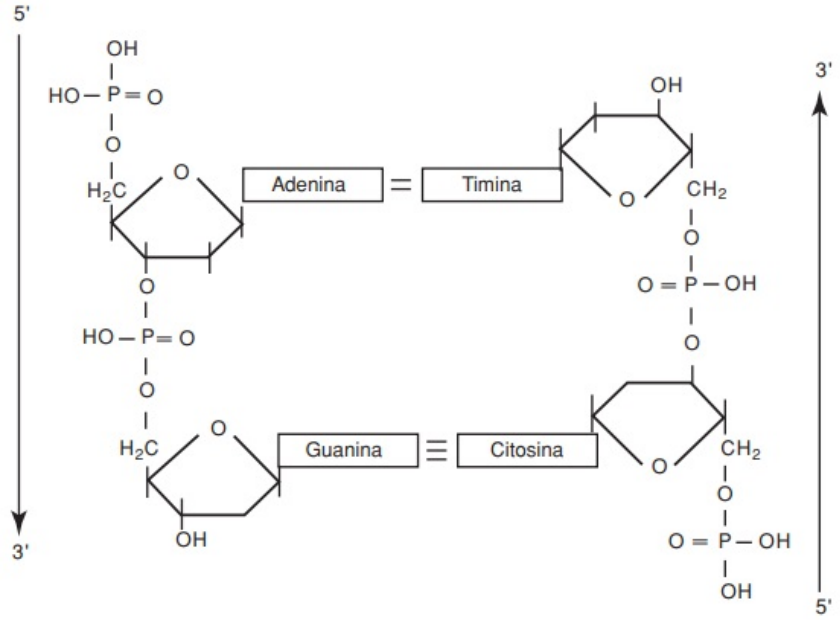


Figura 3.2: Posición antiparalela de las dos hebras de polidesoxinucleótidos en el DNA. En el ADN, las dos hebras de polidesoxinucleótidos están orientadas de manera que una hebra va en dirección 5' a 3' y la otra de 3' a 5'.

la síntesis de una nueva cadena complementaria de la cadena molde, de forma que cada nueva doble hélice contiene una de las cadenas del ADN original.

Transcripción

Es el proceso mediante el cual se transcribe la información genética del ADN al ARN. Durante la transcripción genética, las secuencias de ADN son copiadas a ARN mediante una enzima llamada ARN polimerasa la cual sintetiza un ARN mensajero que mantiene la información de la secuencia del ADN. De esta manera, la transcripción del ADN también podría llamarse síntesis del ARN mensajero.

Traducción

Es el proceso mediante el cual el mensaje cifrado en el idioma de los tripletes de bases (código genético) es descifrado por los ARNt, sintetizándose una proteína.

3.1.5. El Genoma

El genoma es el conjunto de genes contenidos en los cromosomas, lo que puede interpretarse como la totalidad del material genético que posee un organismo o una especie en particular. Se sabe que el genoma humano, por ejemplo, está constituido por 3,300 millones de pares de bases, comprendidos en alrededor de 2 metros lineales de DNA.

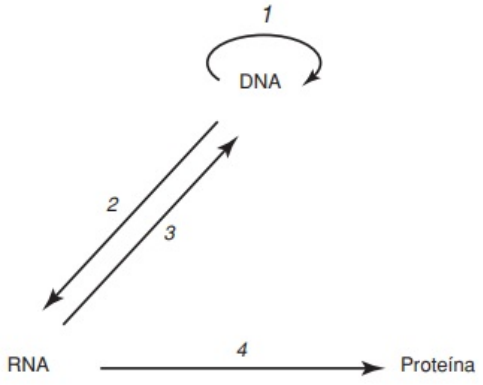


Figura 3.3: Dogma central de la biología molecular. 1. Replicación del ADN 2. Transcripción del ADN 3. Transcripción reversa del ARN 4. Traducción del ARN mensajero

3.1.6. El Gen y el Código Genético

Primera posición (5')	Segunda posición				Tercera posición (3')
	U	C	A	G	
U	UUU Phe UUC Phe UUA Leu UUG Leu	UCU Ser UCC Ser UCA Ser UCG Ser	UAU Tyr UAC Tyr UAA Término UAG Término	UGU Cys UGC Cys UGA Término UGG Trp	U C A G
C	CUU Leu CUC Leu CUA Leu CUG Leu	CCU Pro CCC Pro CCA Pro CCG Pro	CAU His CAC His CAA Gln CAG Gln	CGU Arg CGC Arg CGA Arg CGG Arg	U C A G
A	AUU Ile AUC Ile AUA Ile AUG Met	ACU Thr ACC Thr ACA Thr ACG Thr	AAU Asn AAC Asn AAA Lys AAG Lys	AGU Ser AGC Ser AGA Arg AGG Arg	U C A G
G	GUU Val GUC Val GUA Val GUG Val	GCU Ala GCC Ala GCA Ala GCG Ala	GAU Asp GAC Asp GAA Glu GAG Glu	GGU Gly GGC Gly GGA Gly GGG Gly	U C A G

Figura 3.4: Código genético

En el DNA de las células de un organismo se encuentra toda la información genética que lo define. Un gen, es decir, una unidad de información genética, contiene la información para la síntesis de una molécula de RNA que es complementaria a una de las dos hebras del ADN. Los principales ARN celulares son el ARN ribosomal (ARNr), el ARN de transferencia (ARNt) y el ARN mensajero (ARNm). Cada molécula de ARNm contiene la información para la secuencia de aminoácidos de una proteína, mientras que las moléculas de rRNA y de tRNA forman parte de la maquinaria celular que traduce la información de los mRNA a proteínas.

La información para la secuencia de aminoácidos de una proteína está codificada en el mRNA, en unidades independientes de tres bases llamadas codones. La combinación de cuatro letras, A, T, C y G en el DNA, o A, U, C y G en el mRNA, en unidades de

tres, puede generar 64 codones (4^3). Los 64 codones y el significado de cada uno constituyen el código genético ("**Figura 3.4**"). Una proteína está formada por la combinación de alrededor de 20 aminoácidos diferentes, por lo que varios codones pueden codificar para un mismo aminoácido. Por ejemplo, el aminoácido leucina (Leu) está codificado por los codones UUA, UUG, CUU, CUC, CUA, y CUG ("**Figura 3.4**"). Esta propiedad se denomina "degeneración" del código genético. Tres de los 64 codones no codifican para un aminoácido, sino que funcionan como señal de terminación en la síntesis de proteínas.

3.2. El Secuenciamiento de ADN

La secuenciación del ADN es un conjunto de métodos y técnicas bioquímicas cuya finalidad es la determinación del orden de los nucleótidos (A, C, G y T) en un cadena de ADN. La secuencia de ADN constituye la información genética heredable que forman la base de los programas de desarrollo de los seres vivos (de procariotas, de eucariotas en el núcleo celular, y en los plásmidos, en la mitocondria y en cloroplastos de las plantas). Así pues, determinar la secuencia de ADN es útil en el estudio de la investigación básica de los procesos biológicos fundamentales, así como en campos aplicados, como la investigación forense. Además, se puede utilizar la secuenciación del ADN para conocer las mutaciones somáticas, como las sustituciones de bases, generadas entre distintos organismos.

El desarrollo de la secuenciación del ADN ha acelerado significativamente la investigación y los descubrimientos en biología. Las técnicas actuales permiten realizar esta secuenciación a gran velocidad, lo cual ha sido de gran importancia para proyectos de secuenciación a gran escala como el Proyecto Genoma Humano. Otros proyectos relacionados, en ocasiones fruto de la colaboración investigadora a escala mundial, han establecido la secuencia completa de ADN de muchos genomas de animales, plantas y microorganismos.

3.3. Bases de Datos de ADN

Existen repositorios de acceso público, disponibles para su descarga a través de internety pudiendo albergar desde secuencias completas de genoma como el genoma humano (*Homo sapiens*) con una longitud de más de 3 mil 200 millones de pb, genomas de cromosomas o de algún gen en particular: Entre estas bases de datos destacan: NCBI, Ensembl y la *Universidad de California Santa Cruz* (UCSC).

3.3.1. NCBI

Forma parte de la Biblioteca Nacional de Medicina de Estados Unidos (National Library of Medicine), una rama de los Institutos Nacionales de Salud (National Institutes of Health o NIH). Fue fundado el 4 de noviembre de 1988 con la misión de ser una importante fuente de información de biología molecular. Almacena y constantemente actualiza la

información referente a secuencias genómicas en GenBank, un índice de artículos científicos referentes a biomedicina, biotecnología, bioquímica, genética y genómica en PubMed, una recopilación de enfermedades genéticas humanas en OMIM, además de otros datos biotecnológicos de relevancia en diversas bases de datos.

Todas las bases de datos del NCBI están disponibles en línea de manera gratuita. Además ofrece algunas herramientas bioinformáticas para el análisis de secuencias de ADN, ARN y proteínas.

3.3.2. Ensembl

Ensembl es un proyecto de investigación bioinformática que trata de "desarrollar un sistema de software que produzca y mantenga anotaciones automáticas en los genomas eucariotas seleccionados". Funciona como una colaboración entre el Wellcome Trust Sanger Institute y el Instituto Europeo de Bioinformática, una división del Laboratorio Europeo de Biología Molecular. Toda la información y software generados en el proyecto es de libre uso y acceso. Proporciona tres tipos de archivos por genoma de cromosoma:

- dna_sm - Posee datos enmascarados, donde convierte los nucleótidos repetidos a letras minúsculas.
- dna_rm - Posee datos enmascarados, donde convierte las repeticiones de nucleótidos a N.
- dna . Sin ninguna máscara.

Proporciona dos tipos de archivos por genoma completo por especie:

- toplevel - Incluye información de haplotipos.
- primary_assembly - Base de referencia única por posición

3.3.3. UCSC

La UCSC brinda un servicio de repositorio de base de datos de acceso público de genomas que actualiza cada semana. El 22 de junio de 2000, UCSC y los otros miembros del consorcio del Proyecto Genoma Humano Internacional completaron el primer borrador del ensamble del genoma humano, asegurando para siempre el acceso público gratuito al genoma y la información que contiene. Unas semanas más tarde, el 7 de julio de 2000, el genoma recién ensamblado se lanzó dentro del dominio de la UCSC, junto con el prototipo inicial de una herramienta de visualización gráfica, el navegador Genome UCSC. En los años siguientes, el sitio web ha crecido para incluir una amplia colección de ensambles y anotaciones de organismos vertebrados y modelo, junto con un gran conjunto de herramientas para ver, analizar y descargar datos.

3.4. El alineamiento de secuencias de ADN

El alineamiento de secuencias es una forma de representar y comparar dos o más secuencias biológicas o molécula continua de ADN, ARN, o estructuras primarias proteicas [Mulia et al., 2012] para resaltar sus zonas de similitud como se observa en la "**Figura 3.5**", que podrían indicar relaciones funcionales o evolutivas entre los genes o secuencias de aminoácidos. Los algoritmos de alineamiento de secuencias son usados durante el secuenciamiento de ADN, el cual previamente es sometido a un proceso de duplicación y segmentación para su secuenciamiento mediante alineamiento múltiple que consiste en ordenar dos o más fragmentos de ADN que pueden estar superpuestos hasta encontrar la secuencia general que los fragmentos describen.

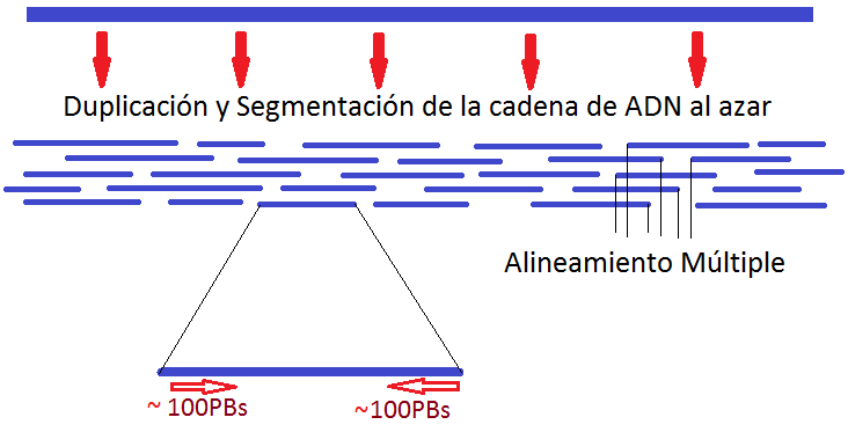


Figura 3.5: Esquema del proceso de alineamiento de segmentos de ADN

3.5. Algoritmos de alineamiento de secuencias

El objetivo de los algoritmos de alineamiento de secuencias es acomodar dos a más secuencias de tal manera que se alcance el máximo de coincidencias entre los elementos de las mismas. Es una forma de representar y comparar dos o más secuencias o cadenas de ADN, ARN, o estructuras primarias proteicas para resaltar sus zonas de similitud, que podrían indicar relaciones funcionales o evolutivas entre los genes o proteínas consultados.

Los alineamientos se pueden clasificar en: globales (cuando se pretende alinear las secuencias enteras, empleando tantos caracteres o símbolos de los extremos de las secuencias como sea posible) y locales (cuando se buscan porciones de las secuencias que poseen mayor concordancia). [Salinas and Lisbona, 2016]

Los algoritmos desarrollados son clasificados en cuatro grupos por sus características: basados en árboles de sufijos, basados en Tablas Hash, relacionados a Merge Sorting [Li and Homer, 2010] y relacionados con algoritmos evolutivos [Ticona, 2003].

3.6. Algoritmos de alineamiento de secuencias basados en tablas Hash

3.6.1. Smith-Waterman

El algoritmo de Smith-Waterman permite un alineamiento local de secuencias biológicas (ADN, ARN o proteínas); es decir que determina regiones similares entre un par de secuencias. Fue propuesto por Temple Smith y Michael Waterman en 1981. Está basado en el uso de algoritmos de programación dinámica, de tal forma que tiene la deseable propiedad de garantizar que el alineamiento local encontrado es óptimo con respecto a un determinado sistema de puntajes que se use (tales como matrices de substitución). [Smith and Waterman, 1981]

Las alternativas básicas para realizar el alineamiento de un par de secuencias son: el alineamiento local y el alineamiento global. Los alineamientos globales pretenden alinear cada símbolo (o residuo) en cada secuencia. Esta estrategia es especialmente útil cuando las secuencias a alinear son altamente similares y aproximadamente del mismo tamaño. En contraste, los alineamientos locales son más útiles cuando las secuencias a alinear poseen grandes diferencias, pero se sospecha que existen regiones de similitud.

3.6.2. BLAST

El *National Center for Biotechnology Information* (NCBI) ha implementado una herramienta con el mismo nombre BLAST, la cual posee sub-herramientas como BLASTN, BLASTP y BLASTX que procesan alineamientos de nucleótidos, alineamiento de proteínas y la traducción de ADN a proteínas respectivamente de manera remota.

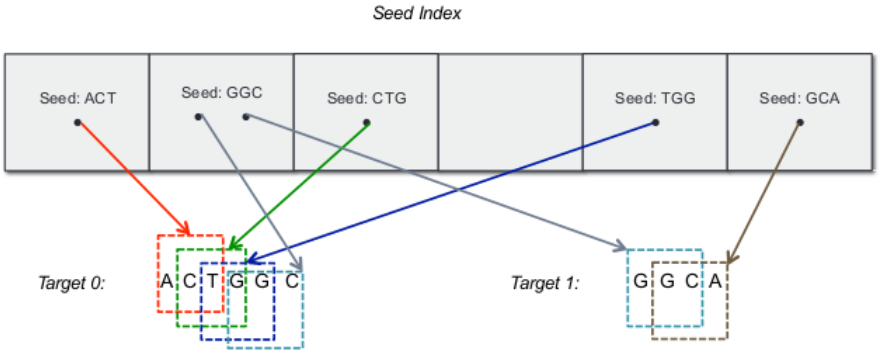


Figura 3.6: Fase de sembrado e indexamiento del algoritmo BLAST

El algoritmo de BLAST [Altschul et al., 1990] posee dos fases en base al paradigma "***sembrar y extender***", siendo la primera fase, la obtención de un índice de semillas que son almacenadas en una Tabla Hash como se observa en la "Figura 3.6" y la segunda fase es la extensión de las coincidencias con la secuencia de consulta usando el algoritmo de

Smith-Waterman [Smith and Waterman, 1981].

3.7. Algoritmos de alineamiento de secuencias basados en árboles de sufijos

Los esquemas de indexación no comprimida (árboles de sufijos) requieren en el peor de los casos $\Omega(nlog(n))$ bits adicionales de espacio. Dichos índices admiten una búsqueda rápida, ya sea en tiempo $O(m+log(n))$ o tiempo O(m), más un costo adicional bajo para listar la cantidad de ocurrencias de cada palabra encontrada.
Existen otras estructuras, llamadas de índices comprimidos que tienen por objetivo indexar usando menos memoria. Indexan el texto basándose en una representación comprimida del arreglo de sufijos. La principal forma de construir y almacenar un índice comprimido se basa en una transformación específica del texto llamada transformada de Burrows-Wheeler (BWT).[Deymonnaz, 2012]

Program	Time (s)	Conf (%)	Err (%)	Time (s)	Conf (%)	Err (%)
Bowtie-32	1271	79.0	0.76	1391	85.7	0.57
BWA-32	823	80.6	0.30	1224	89.6	0.32
MAQ-32	19797	81.0	0.14	21589	87.2	0.07
SOAP2-32	256	78.6	1.16	1909	86.8	0.78
Bowtie-70	1726	86.3	0.20	1580	90.7	0.43
BWA-70	1599	90.7	0.12	1619	96.2	0.11
MAQ-70	17928	91.0	0.13	19046	94.6	0.05
SOAP2-70	317	90.3	0.39	708	94.5	0.34
bowtie-125	1966	88.0	0.07	1701	91.0	0.37
BWA-125	3021	93.0	0.05	3059	97.6	0.04
MAQ-125	17506	92.7	0.08	19388	96.3	0.02
SOAP2-125	555	91.5	0.17	1187	90.8	0.14

Figura 3.7: Evaluación de un millón de pares de lecturas de 32, 70 y 125 pb en el genoma humano [Li and Durbin, 2009]

3.7.1. BWT

BWT-SW fue un software creado en base a la Transformada de Burrow Whealer para el secuenciamiento de ADN específicamente en el año 2008 y de código abierto en la plataforma GITHUB y utilizó la base de datos de codigo abierto usado en el mismo software [Lam et al., 2008].

T^{bwt} se forma recorriendo secuencialmente el arreglo de sufijos A, y concatenando el carácter que precede a cada sufijo. Otra forma de ver la transformada de Burrows-Whealer es tomando todas las secuencias de la forma $T_{i,n}T_{1,i-1}$ ordenadas lexicográficamente. Si se considera una matriz M con todas estas secuencias, la última columna corresponde a T^{bwt}.

3.8. Programación paralela y distribuida

3.8.1. GPU

Una unidad de procesamiento gráfico o GPU (graphics processing unit) es un co-procesador dedicado al procesamiento de gráficos u operaciones de coma flotante, para aligerar la carga de trabajo del procesador central en aplicaciones como los videojuegos o aplicaciones 3D interactivas. De esta forma, mientras gran parte de lo relacionado con los gráficos se procesa en la GPU, la unidad central de procesamiento (CPU) puede dedicarse a otro tipo de cálculos (como la inteligencia artificial o los cálculos mecánicos en el caso de los videojuegos).

La GPU implementa ciertas operaciones gráficas llamadas primitivas optimizadas para el procesamiento gráfico. Una de las primitivas más comunes para el procesamiento gráfico en 3D es el antialiasing, que suaviza los bordes de las figuras para darles un aspecto más realista. Adicionalmente existen primitivas para dibujar rectángulos, triángulos, círculos y arcos. Las GPU actualmente disponen de gran cantidad de primitivas, buscando mayor realismo en los efectos.

3.8.2. CUDA

CUDA son las siglas de Compute Unified Device Architecture (Arquitectura Unificada de Dispositivos de Cómputo) que hace referencia a una plataforma de computación en paralelo incluyendo un compilador y un conjunto de herramientas de desarrollo creadas por nVidia que permiten a los programadores usar una variación del lenguaje de programación C para codificar algoritmos en GPU de nVidia.

Por medio de wrappers se puede usar Python, Fortran y Java en lugar de C/C++. Funciona en todas las GPU nVidia de la serie G8X en adelante, incluyendo GeForce, Quadro, ION y la línea Tesla.

CUDA intenta explotar las ventajas de las GPU frente a las CPU de propósito gene-

ral utilizando el paralelismo que ofrecen sus múltiples núcleos, que permiten el lanzamiento de un altísimo número de hilos simultáneos. Por ello, si una aplicación está diseñada utilizando numerosos hilos que realizan tareas independientes (que es lo que hacen las GPU al procesar gráficos, su tarea natural), una GPU podrá ofrecer un gran rendimiento en campos que podrían ir desde la biología computacional a la criptografía, por ejemplo.

3.8.3. Supercomputador Manati

La supercomputadora del *Instituto de Investigaciones de la Amazonía Peruana* (IIAP) denominada MANATI tiene una arquitectura tipo Cluster, compuesta por 10 nodos (servidores o computadoras con grandes capacidades) distribuidos de la siguiente manera, un nodo coordinador y 9 nodos de procesamiento (6 gráficos y 3 numéricos) "Figura 3.8":

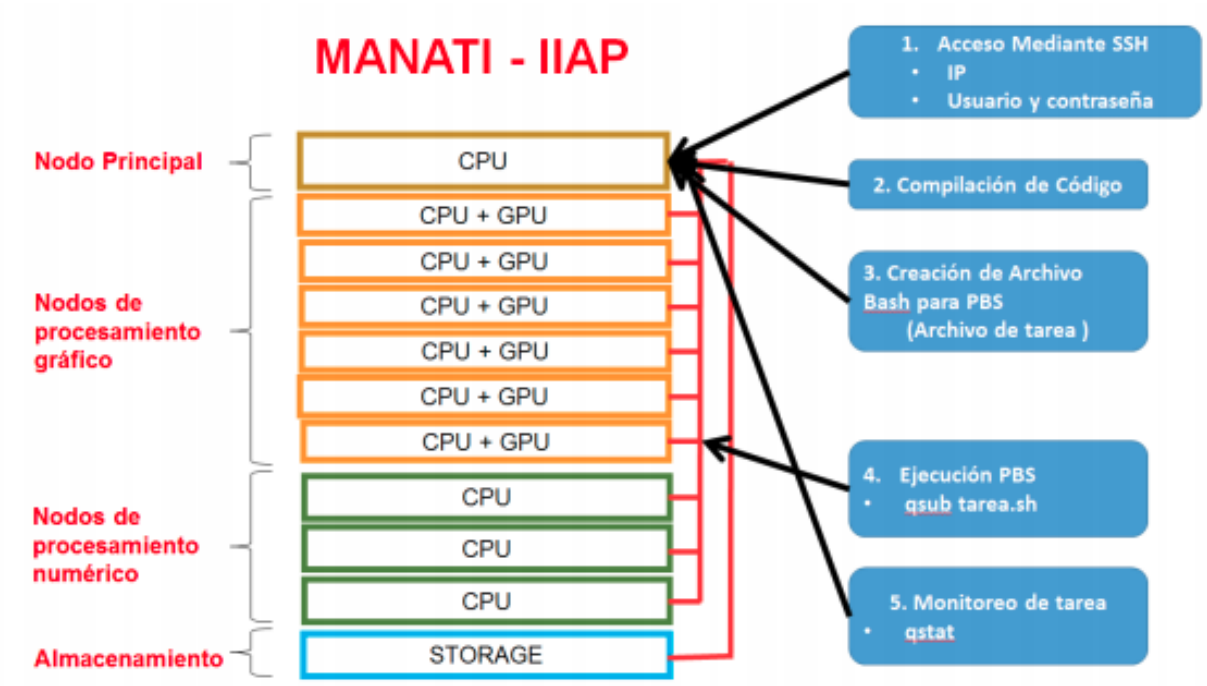


Figura 3.8: Arquitectura del Supercomputador Manati del Instituto de Investigaciones de la Amazonía Peruana [Ocampo Yahuarcani and Cárdenas Vigo, 2017]

- 1 Nodo Principal: CPU Intel Xeon con 56 núcleos, 64 Gb de RAM
- 6 Nodos Procesamiento Gráfico: CPU Intel Xeon con 28 núcleos, GPU NVIDIA TESLA K80 con 4992 Cuda cores, 64 Gb de RAM
- 3 Nodos Procesamiento Gráfico: CPU Intel Xeon con 28 núcleos, 64 Gb de RAM
- Sistema de almacenamiento de 114 Tb.

El sistema operativo de la supercomputadora es Centos 7 y el gestor de colas o tareas que administra los nodos es el PBS (Portable Batch System)
[Ocampo Yahuarcani and Cárdenas Vigo, 2017].

Capítulo 4

Metodología y Propuesta

4.1. Lugar y fecha de ejecución

El presente trabajo de investigación se realizó en el laboratorio de cómputo de la Escuela de Ciencia de la Computación de la Facultad de Ingeniería de Producción y Servicios de la Universidad Nacional de San Agustín, Los experimentos computacionales fueron desarrollados usando una Laptop core i7, una pC core i5 con tarjeta de video GTX 1050 y haciendo uso del SuperComputador "MANATI" del Centro de Alto Rendimiento Computacional de la Amazonía Peruana del IIAP, usando un nodo y un solo dispositivo GPU Tesla K80, durante los meses de enero 2018 a julio del 2018.

4.2. Metodología

4.2.1. Diseño Metodológico

La metodología seguida para desarrollar el presente trabajo se resume de forma secuencial en la "**Figura 4.1**", la cual inicia con la recolección de la base de datos de secuencias de ADN, pasando por el análisis e identificación de técnicas alineamiento de secuencias de ADN, tras lo cual se realizó la implementación y comparación de algoritmos de alineamiento de secuencias de ADN para un BLAST Unisecuencial y multisecuencial sin paralelismo y con paralelismo, para finalizar se realizó pruebas de evaluación y comparación finales.

4.2.2. Recolección y selección de secuencias de ADN

Se recolectaron secuencias de ADN de bases de datos de los portales de acceso público: NCBI [Autores, 2018b] y EMSEMBL [Autores, 2018a], en el formato FASTA en la cantidad de 520 secuencias siendo 20 de ellas genomas completos de especies, 100 para secuencias medianamente largas de cromosomas y 520 para secuencias cortas de genes a

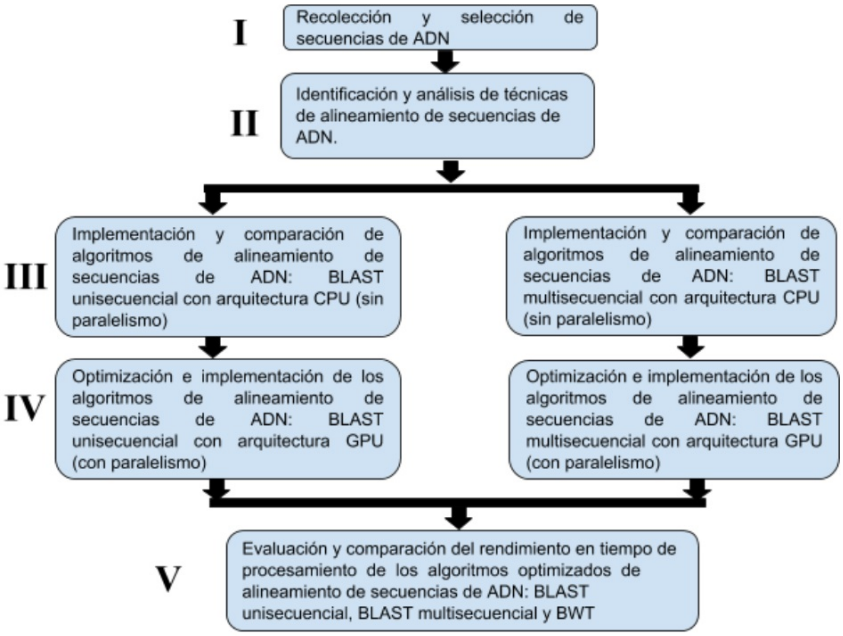


Figura 4.1: Diagrama de flujo de la metodología seguida para optimizar en tiempo de procesamiento el algoritmo BLAST en el alineamiento de secuencias de ADN usando procesamiento masivamente paralelo y distribuido.

Secuencias de ADN	Bases de Datos de Acceso Público	
	NCBI	EMSEMBL
Genomas Completos de Especies	0	20
Secuencias medianamente largas (Cromosomas)	0	80
Secuencias Cortas (Genes)	520	0

Tabla 4.1: Secuencias de ADN Recolectadas de las Bases de Datos de Acceso Público NCBI y EMSEMBL

ser usadas para la validación de los algoritmos optimizados "**Tabla 4.1**".

De manera aleatoria se escogió secuencias de cada grupo para ser usadas como secuencias de referencia en condiciones reales, la secuencia a alinear de 100 pb se extrajo de cada secuencia de referencia aleatoriamente.

4.2.3. Identificación y análisis de técnicas de alineamiento de secuencias de ADN

De los cuatro grupos de algoritmos para el alineamiento de secuencias de ADN: Basados en Tablas Hash, árboles de sufijos, merge and sorting [Li and Homer, 2010, Elloumi and Zomaya, 2013] y algoritmos evolutivos [Ticona, 2003], para su uso en la optimización del algoritmo BLAST se analizaron:

- Para la optimización del BLAST unisecuencial: los algoritmos BF [Cormen et al., 2009], *Booyer-Moore* (BM) [Boyer and Moore, 1977], KMP [Knuth et al., 1977] y *Karp-Rabin* (KR) [Karp and Rabin, 1987],
- Para la optimización del BLAST multisecuencial: los algoritmos B5, KR [Karp and Rabin, 1987] y BWT [Burrows and Wheeler, 1994, Adjeroh et al., 2008].

4.2.4. Implementación y comparación de algoritmos de alineamiento de secuencias de ADN: BLAST unisecuencial y BLAST multisecuencial con arquitectura CPU (sin paralelismo)

Implementación y comparación de algoritmos de alineamiento de secuencias de ADN: BLAST unisecuencial con arquitectura CPU (sin paralelismo)

Se implementaron los algoritmos: BF, BM, KMP y B5 en el sembrado del algoritmo BLAST sin optimizaciones con sólo CPU. Posteriormente se realizaron comparaciones en el supercomputador Manati usando como secuencia de referencia de ADN el genoma humano.

Implementación y comparación de algoritmos de alineamiento de secuencias de ADN: BLAST multisecuencial con arquitectura CPU (sin paralelismo)

Se implementaron los algoritmos: B5, KR en el sembrado del algoritmo BLAST sin optimizaciones con sólo CPU. Posteriormente se realizaron comparaciones en el supercomputador Manati usando como secuencia de referencia de ADN el genoma humano.

4.2.5. Optimización e implementación de los algoritmos de alineamiento de secuencias de ADN: BLAST unisecuencial y BLAST multisecuencial con arquitectura GPU (con paralelismo)

Para el diseño, optimización e implementación de los algoritmos paralelos se utilizó la metodología de Foster [Foster, 1995] que consta de cuatro pasos:

1. **Particionamiento**: El cálculo que se debe realizar y los datos operados por este se descomponen en pequeñas tareas. Cuestiones prácticas como el número de los procesadores en la computadora de destino se ignoran, y la atención se centra en reconocer oportunidades de ejecución paralela.
2. **Comunicación**: Se determina la comunicación requerida para coordinar la ejecución de la tarea y se definen estructuras y algoritmos de comunicación apropiados.

3. **Aglomeración**: La tarea y las estructuras de comunicación definidas en las dos primeras etapas del diseño se evalúa con respecto a los requisitos de rendimiento y los costos de implementación. Si es necesario, las tareas se combinan en tareas más grandes para mejorar el rendimiento o reducir costos de desarrollo.

4. **Mapeo**: Cada tarea se asigna a un procesador de una manera que intenta satisfacer con la finalidad de maximizar la utilización del procesador y minimizar los costos de comunicación. La asignación puede especificarse estáticamente o determinarse en tiempo de ejecución mediante algoritmos de equilibrio de carga

Optimización e implementación de los algoritmos de alineamiento de secuencias de ADN: BLAST unisecuencial con arquitectura GPU (con paralelismo)

Se implementaron los algoritmos: BF, BM, KMP y B5 en el sembrado del algoritmo BLAST con optimizaciones usando la arquitectura GPU con CUDA. Posteriormente se realizaron comparaciones en el supercomputador Manati usando como secuencia de referencia de ADN el genoma humano.

Optimización e implementación de los algoritmos de alineamiento de secuencias de ADN: BLAST multisecuencial con arquitectura GPU (con paralelismo)

Se implementaron los algoritmos: B5, KR en el sembrado del algoritmo BLAST con optimizaciones usando la arquitectura GPU con CUDA. Posteriormente se realizaron comparaciones en el supercomputador Manati usando como secuencia de referencia de ADN el genoma humano.

4.2.6. Evaluación y comparación del rendimiento en tiempo de procesamiento de los algoritmos optimizados de alineamiento de secuencias de ADN: BLAST unisecuencial, BLAST multisecuencial, BWT

Los algoritmos optimizados al procesar una sola cadena de consulta con una secuencia de referencia (unisecuencial) o procesar multiples secuencias de consulta con una secuencia de referencia (multisecuencial) fueron comparadas para determinar el punto de inflexión o de cambio recomendado de un algoritmo al otro.

4.3. Propuesta

4.3.1. Optimización del algoritmo BLAST para alineamientos unisecuenciales

En este trabajo de investigación se propone el algoritmo al que nombraremos CN que es la versión paralelizada con cuda del algoritmo de Naive o Fuerza Bruta (BF) con pseudocódigo en el ***Algoritmo 1***, el cual se basa en el principio del algoritmo BF donde se compara todas las coincidencias, el cual será usado para reducir el tiempo de procesamiento durante el sembrado del algoritmo BLAST.

Algoritmo 1 Pseudocódigo: CudaNaive como optimización del algoritmo BLAST unisecuencial en el sembrado

Input: $sequence\_DB$,$sequence\_Query$ $SIZE\_KEY$, $BLOCK\_SIZE$

1: $length\_seq \leftarrow$ Length sequence
2: Copy $sequence\_DB$ from host memory to Global device memory
3: Copy $sequence\_Query$ from host memory to Global device memory
4: INTO GPU for each block of strands
5: **while** index is in range **do**
6: $gindex \leftarrow threadIdx.x + blockIdx.x * blockDim.x$
7: $lindex \leftarrow threadIdx.x$
8: Copy $sequence_DB$ from Global device memory to Block device memory BLOCK_SIZE + SIZE_KEY -1
9: Copy $sequence\_Query$ from Global device memory to Block device memory
10: synchronize GPU
11: $j \leftarrow 0$
12: **for** $j = 0$ to $SIZE\_KEY - 1$ **do**
13: **if** $sequence\_DB[lindex + j]! = sequence\_Query[j]$ **then**
14: exit the loop
15: **end if**
16: **end for**
17: **if** $j == SIZE\_KEY$ **then**
18: save match with gindex position to Global device memory
19: **end if**
20: **end while**
21: Copy matchs from Global device memory to host memory
22: **for** $i = 0$ to $length\_seq - SIZE\_KEY + 1$ **do**
23: **if** Match[i] **then**
24: save into vector result with position i
25: **end if**
26: **end for**=0

4.3.2. Optimización del algoritmo BLAST para alineamientos multisecuenciales

En este trabajo de investigación se propone el algoritmo al que nombraremos CB5 que es la versión optimizada usando cuda del algoritmo de conversión en base 5, con pseudocódigo en el ***Algoritmo 2***, el cual se basa en el principio del algoritmo B5 que es la transformación de un alfabeto de base 5 a un alfabeto decimal o de base 10, el cual será usado para reducir el tiempo de procesamiento durante el sembrado del algoritmo BLAST indexando de forma óptima todas las combinaciones de semillas con un valor numérico que representará su ubicación en una tabla Hash bidimensional.

Algoritmo 2 Pseudocódigo: CB5 Optimización del algortmo BLAST multisecuencial en el indexamiento del sembrado

Input: Pre-processed reference *sequence*, $SIZE\_KEY$, $BLOCK\_SIZE$
Output: two-dimensional Hash Table filled with seed index

1: $length\_seq \leftarrow$ Length reference sequence
2: Copy *sequence* from host memory to device memory
3: Create $Matrix$ with Max_Length_Hash
4: **while** index is in range **do**
5: share block memory BLOCK_SIZE + SIZE_KEY -1
6: synchronize GPU
7: $HashKey[index] \leftarrow 0$
8: $fact \leftarrow 1$
9: **for** $i = SIZE\_KEY - 1$ to 0 **do**
10: $HashKey[index] \leftarrow HashKey[index] + fact * sequence[index + i]$
11: $fact \leftarrow fact * 5$
12: **end for**
13: **end while**
14: Copy HashKeys from device memory to host memory
15: **for** $i = 0$ to $length\_seq - SIZE\_KEY + 1$ **do**
16: $Matrix[HashKey[i]] \leftarrow$ Insert i
17: **end for**=0

Capítulo 5

Resultados y Discusión

5.1. Recolección y selección de secuencias de ADN

La secuencia de mayor tamaño seleccionada para realizar pruebas con el supercomputador Manati fue el genoma humano, en formato de archivo FASTA de 3.5 *Giga Bytes* (GB) que posee 3 099 750 718 pb. Las pruebas en laptop fueron desarrolladas con el genoma de la gallina (*Gallus gallus*) de 1.3 GB con 1 230 258 557 pb.

Los archivos en formato FASTA obtenido de las bases de datos de secuencias de ADN [Autores, 2018b, Autores, 2018a] poseen la estructura observada en la "**Figura 5.1**", donde la secuencia de ADN está formada por las bases **A** (adenina), **C** (citocina), **G** (guanina) y **T** (timina), además es usual encontrar el caracter **N** que representa una base hipotética que no ha sido identificada en dicha posición, por lo que para su adecuado procesamiento de datos es necesario realizar una codificación como preprocesamientoddddddddddddddddddddd convirtiendo las bases a su representación numérica siendo: N=0, A=1, C=2, G=3 y T=4.

```
1  >gi|263191547|ref|NM_000249.3| Homo sapiens mutL homolog 1 (MLH1), transcript variant 1, mRNA
2  GAAGAGACCCAGCAACCCACAGAGTTGAGAAATTTGACTGGCATTCAAGCTGTCCAATCAATAGCTGCCGCTGAAGGGTG
3  GGGCTGGATGGCGTAAGCTACAGCTGAAGGAAGAACGTGAGCACGAGGCACTGAGGTGATTGGCTGAAGGCACTTCCGTT
4  GAGCATCTAGACGTTTCCTTGGCTCTTCTGGCGCCAAAATGTCGTTCGTGGCAGGGGTTATTCGGCGGCTGGACGAGACA
5  GTGGTGAACCGCATCGCGGCGGGGGAAGTTATCCAGCGGCCAGCTAATGCTATCAAAGAGATGATTGAGAACTGTTTAGA
6  TGCAAAATCCACAAGTATTCAAGTGATTGTTAAAGAGGGAGGCCTGAAGTTGATTCAGATCCAAGACAATGGCACCGGGA
7  TCAGGAAAGAAGATCTGGATATTGTATGTGAAAGGTTCACTACTAGTAAACTGCAGTCCTTTGAGGATTTAGCCAGTATT
8  TCTACCTATGGCTTTCGAGGTGAGGCTTTGGCCAGCATAAGCCATGTGGCTCATGTTACTATTACAACGAAAACAGCTGA
9  TGGAAAGTGTGCATACAGAGCAAGTTACTCAGATGGAAAACTGAAAGCCCCTCCTAAACCATGTGCTGGCAATCAAGGGA
10 CCCAGATCACGGTGGAGGACCTTTTTTACAACATAGCCACGAGGAGAAAAGCTTTAAAAAATCCAAGTGAAGAATATGGG
11 AAAATTTTGGAAGTTGTTGGCAGGTATTCAGTACACAATGCAGGCATTAGTTTCTCAGTTAAAAAACAAGGAGAGACAGT
12 AGCTGATGTTAGGACACTACCCAATGCCTCAACCGTGGACAATATTCGCTCCATCTTTGGAAATGCTGTTAGTCGAGAAC
13 TGATAGAAATTGGATGTGAGGATAAAACCCTAGCCTTCAAAATGAATGGTTACATATCCAATGCAAACTACTCAGTGAAG
14 AAGTGCATCTTCTTACTCTTCATCAACCATCGTCTGGTAGAATCAACTTCCTTGAGAAAAGCCATAGAAACAGTGTATGC
```

Figura 5.1: Secuencia de ADN en formato FASTA de Homo sapiens mutL homolog 1 (MLH1) usada para las pruebas de comparación

5.2. Identificación y análisis de técnicas de alineamiento de secuencias de ADN

Considerando que el algoritmo BLAST posee dos fases que son el sembrado y la extensión y al ser la primera fase (sembrado) la seleccionada para aplicar algoritmos para su optimización, es que se analizaron la inclusión de diferentes algoritmos para su optimización, los cuales se han utilizado para un alineamiento unisecuencial y un alineamiento multisecuencial.

5.2.1. Para Blast unisecuencial

Se identificaron y analizaron los algoritmos:

- BF [Cormen et al., 2009] o llamado tambíen Naive que realiza la búsqueda de una subcadena dentro de la secuencia de ADN, comparando cada k-mer mientras exista coincidencia hasta encontrar una secuencia idéntica y puede ser lento si las secuencias clave del patrón son repetitivas, ejecutando en un tiempo de procesamiento lineal con NM comparaciones. Siendo N la longitud de la secuencia de referencia y M la longitud del patrón.
- BM
  [Boyer and Moore, 1977] utiliza una heurística de caracteres no coincidentes escaneando cada k-mer de derecha a izquierda y puede omitir comparar caracteres que no se encuentren en el patrón al preprocesar el patrón en una tabla índice de ocurrencia por la derecha, ejecutandose en un tiempo lineal pero con un factor mucho más bajo el cual puede ser menor si la cadena patrón es de mayor longitud. EL numero de comparaciones en el mejor de los casos sería de N/M, en el peor de los casos sería de NM comparaciones.
- KMP [Knuth et al., 1977] realiza la búsqueda de coincidencias del patrón con un *Autómata Finito Determinista* (AFD) donde cada comparación es la transición a un nuevo estado dentro del AFD, se procesa en tiempo lineal con M+N comparaciones.
- KR [Karp and Rabin, 1987] Realiza la busqueda de patrones mediante el modulo como función hash, a diferencia de los anteriores algoritmos este debe preprocesar las secuecnias de ADN a un alfabeto numérico de 0 a 5.

5.2.2. Para Blast multisecuencial

Se identificaron y analizaron los algoritmos:

- B5 Entre los sistemas de numeración de uso común en varias ramas de la ciencia podemos encontrar a los sistemas: binario, octal, decimal y hexadecimal. Podemos llamar al sistema binario como sistema con base 2 porque emplea un grupo básico de

dos símbolos (0s y 1s) [Nieves Hurtado and Domínguez Sánchez, 2002]. Al igual que dicho sistema podemos hablar del sistema en base 5 pues emplea un grupo básico de cinco símbolos (0, 1, 2, 3 y 4) o (N, A, C, G y T).

La función Hash de Base 5 usada es la conversión de Base 5 a base decimal como lo ilustra la "Ecuación 5.1" generando la máxima dispersión de las semillas.

$$Hash\_Key = \sum_{i=0}^{n-1} K_i 5^{(n-1)-i} \tag{5.1}$$

- KR [Karp and Rabin, 1987]. Es un algoritmo probabilístico que adapta técnicas de dispersión a la búsqueda de patrones y puede ser usada como una función Hash para la generación de Tablas Hash que permitan optimizar la búsqueda de texto.

  Inicialmente fue formulada teniendo como base el sistema binario donde su alfabeto eran 0s y 1s, este algoritmo fue modificado a un alfabeto de $\{A, C, G, T, N\}$ que es en base 5, siendo su función Hash la detallada en la "Ecuación 5.2 " donde P es un número primo aleatorio y la Ecuación 5.2 necesita de una conversión detallada en la "Ecuación 5.3".

$$Hash\_Key = H(x) mod P \tag{5.2}$$

$$H(x) = \sum_{i=1}^{n} x_i 5^{n-i} \tag{5.3}$$

- BWT [Burrows and Wheeler, 1994, Adjeroh et al., 2008] es un algoritmo utilizado para preparar datos para usar con técnicas de compresión de datos como bzip2, Se implementa usando una matriz de sufijos. las matrices de sufijos se pueden calcular con una complejidad en tiempo lineal y memoria. El BWT se puede definir con respecto a la matriz de sufijos SA del texto T como (indexamiento basado en 1) lo indica la "Ecuación 5.4" [Simpson and Durbin, 2010].

$$BWT[i] \begin{cases} T[SA[i]-1] & if SA[i] < 1 \\ \$ & other\_case \end{cases} \tag{5.4}$$

5.3. Implementación y comparación de algoritmos de alineamiento de secuencias de ADN: BLAST unisecuencial y BLAST multisecuencial con arquitectura CPU (sin paralelismo)

5.3.1. Para Blast unisecuencial

Se implementaron y compararon los algoritmos: BF [Cormen et al., 2009], BM [Boyer and Moore, 1977], KMP [Knuth et al., 1977] y KR [Karp and Rabin, 1987] en la

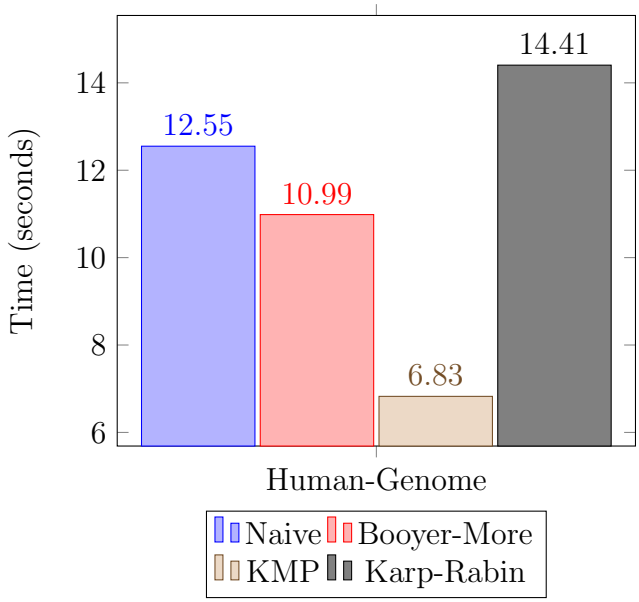


Figura 5.2: Tiempo de procesamiento de los algoritmos BF o Naive, Booyer-More, KMP y Karp-Rabin con arquitectura CPU, como optimización del BLAST unisecuencial en el alineamiento de una secuencia con el genoma humano

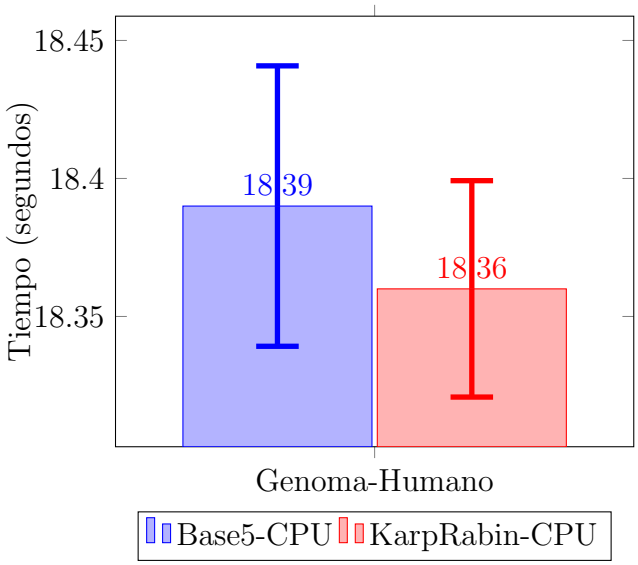
optimización del sembrado del algoritmo BLAST en el tiempo de procesamiento, mostrando un mejor rendimiento el algoritmo KMP al procesar una secuencia clave de 5 pb en el genoma humano usando el supercomputador Manati como se muestra en la "**Figura 5.2**", mostrando un menor tiempo de procesamiento de 6.83s el algoritmo KMP comparado con los algoritmos BM con 10.99s, BF con 12.55s y KR con 14.41s.

5.3.2. Para Blast multisecuencial

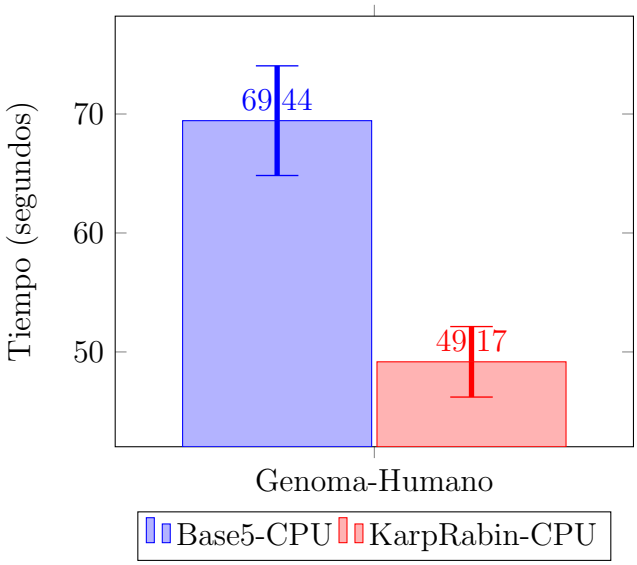
Se implementaron y compararon los algoritmos: B5 y KR [Karp and Rabin, 1987] en el supercomputador Manati. Así mismo se comparó la complejidad en tiempo de procesamiento con el algoritmo BWT [Burrows and Wheeler, 1994, Adjeroh et al., 2008].

El tiempo de generación de las claves Hash e inserción de punteros en la tabla Hash, usando las funciones Hash con Base 5 y Karp-Rabin, en la optimización del sembrado de algoritmo BLAST multisecuencial. se observa en las "**Figuras 5.3a** y **5.3b**" respectivamente y se puede observar la reducción en tiempo de procesamiento del algoritmo KR implementados con arquitectura CPU aparentando una reducción en tiempo de procesamiento del algoritmo BLAST multisecuencial "**Figura 5.5c**".

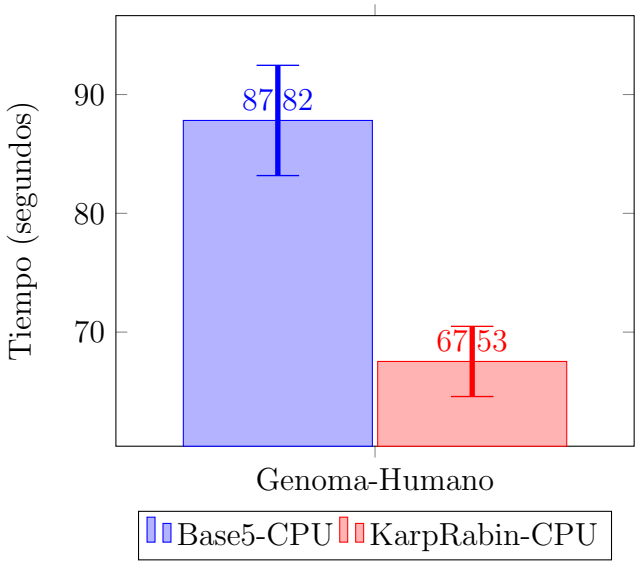
Las tablas hash generadas son de diferente longitud, siendo la de Base 5 de una longitud de 15665 al usar para esta prueba una longitud de clave de 6 K-mer, mientras la longitud para Karp-Rabin fue de 2111, al ser este el número primo empleado. Al obtener la generación de las clave Hash, y el posterior almacenamiento de los punteros en la Tabla Hash, al ser tablas Hash de diferente longitud, es que se observa diferencias en el tiempo



(a) Tiempo de Generación de Claves Hash



(b) Tiempo de Inserción de punteros en la Tabla Hash



(c) Tiempo Total de Indexamiento

Figura 5.3: Tiempo de generación de Claves Hash (a), inserción de los punteros en la tabla Hash (b) y tiempo total de indexamiento (c) en el indexamiento de semillas BLAST multisecuencial con Base5 y KarpRabin con arquitectura CPU en el alineamiento de secuencias de ADN con el genoma humano

Tabla 5.1: Semillas procesadas con Smith-Waterman en 4 consultas de alineamiento

Key	Base 5	Karp-Rabin
CGGCGA	1133695	1133714
ACGTGA	177329	1940721
GACCGA	65850	2250053
TAGGAC	376520	2391178

de procesamiento en el indexamiento "**Figura 5.3**".

Si consideramos el tiempo total de indexamiento de semillas podemos ver una clara ventaja para la implementación con KR "**Figura 5.3**", lo que podría significar un mejor performance de esta función hash con relación a la de Base 5, sin embargo al determinar el número de semillas almacenadas en 4 secuencias diferentes podemos encontrar en dichos casos un número mayor de semillas para ser procesadas por la siguiente fase (extensión) usando el algoritmo de Smith-Waterman [Smith and Waterman, 1981], lo que aumentará significativamente el tiempo de procesamiento en esta segunda fase como se observa en la "Tabla 5.1".

Karp-Rabin podría usarse con esta reducción de la longitud de la tabla Hash, pues es ventajosa si la secuencia de referencia no es de gran tamaño o si la clave posee una longitud de gran tamaño.

Al comparar, el indexamiento FM-INDEX con el algoritmo BWT [Chen et al., 2015], para encontrar las ocurrencias (occ) de un patrón P[1..p] en una secuencia S[1..u] lo hace con un complejidad en tiempo de $O(p + occ * log^{\epsilon} u)$. Sin embargo al utilizar el algoritmo Base5, la complejidad sería de O(1) pues la ubicación de todos los patrones de longitud p son encontrados, ordenados y almacenados por su función hash dentro de la tabla Hash, cuya posición en la tabla está relacionada con el patrón. Asímismo, el algoritmo Karp-Rabin posee una complejidad en tiempo de O(1) para la fase de extensión pero no asegura que los resultados de la búsqueda sean todos iguales al patrón de la consulta.

Meraligner utiliza una matriz esparsa para el indexamiento [Georganas et al., 2015], donde los punteros son derivados de cada 4 primeros caracteres del patrón, por lo que al realizar la búsqueda de un patrón, se realiza la comparación en bloques de 4 caracteres (32 bits) por cada consulta. En nuestra tabla Hash utilizamos un vector bidimensional cuya primera dimensión es la posición o resultado de la función Hash y la segunda dimensión almacena las posiciones dentro de la secuencia de ADN como números (32 bits o 64 bits) o punteros (64 bits), la cual se redimensiona de forma diferente para cada posición, de acuerdo con la cantidad de patrones que alberga.

5.4. Optimización e implementación de los algoritmos de alineamiento de secuencias de ADN: BLAST unisecuencial y BLAST multisecuencial con arquitectura GPU (con paralelismo)

5.4.1. Para Blast unisecuencial

Se implementaron y compararon los algoritmos: CN y las optimizaciones para GPU con CUDA de [Cormen et al., 2009], BM
[Boyer and Moore, 1977], KMP [Knuth et al., 1977] y KR [Karp and Rabin, 1987], mostrando un mejor rendimiento en tiempo de procesamiento el algoritmo CN en la optimización del sembrado del algoritmo BLAST unisecuencial, al encontrar semillas de una secuencia de 5 pb en el genoma humano como se muestra en la "**Figura 5.4**".

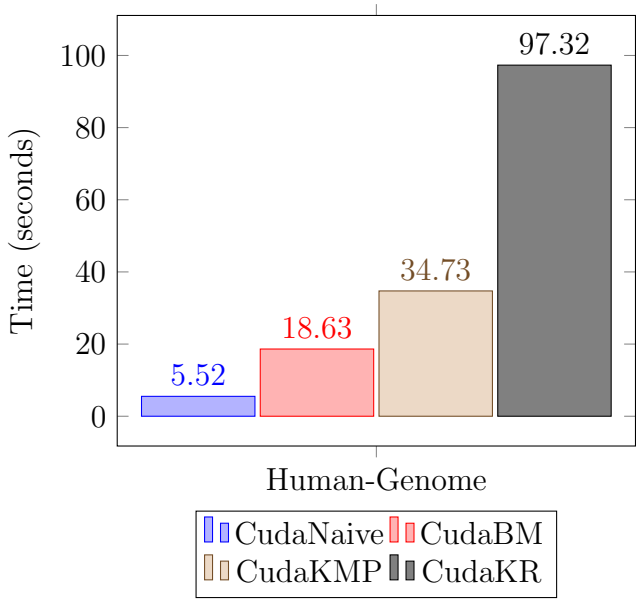


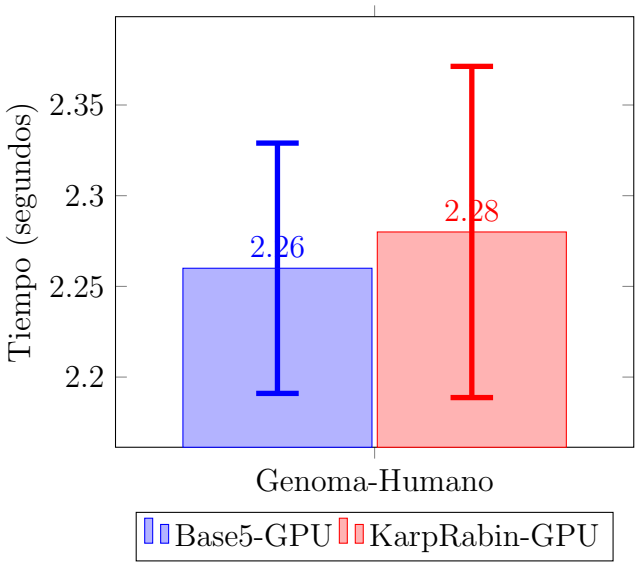
Figura 5.4: Tiempo de procesamiento de los algoritmos CudaNaive, CudaBM, CudaKMP y CudaKR con arquitectura GPU, como optimización del BLAST unisecuencial en el alineamiento de una secuencia con el genoma humano

Todos los algoritmos encontraron un total de 4519420 coincidencias exactas. CudaNaive posee una fase secuencial, que es la generación del vector con todas las coincidencias exactas encontradas, a pesar de lo cual mostró un mejor rendimiento usando GPU con CUDA.

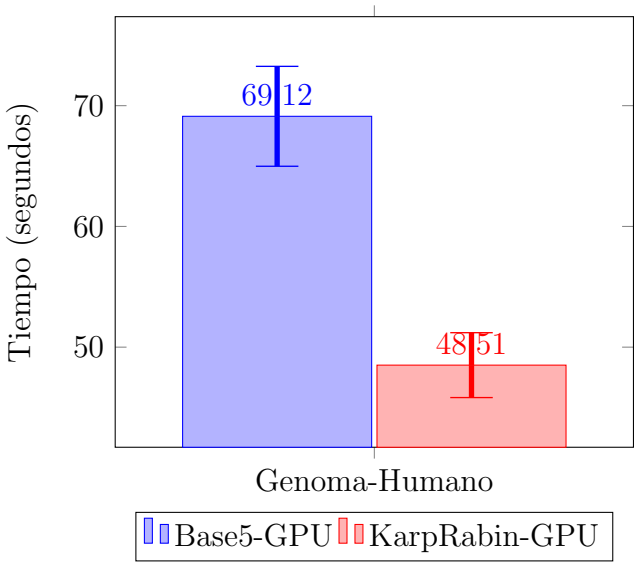
5.4.2. Para Blast multisecuencial

Se implementaron y compararon los algoritmos: CB5 como optimización del B5 y *Cuda Karp Rabin* (CKR) como optimización del KR con arquitectura GPU para opti-

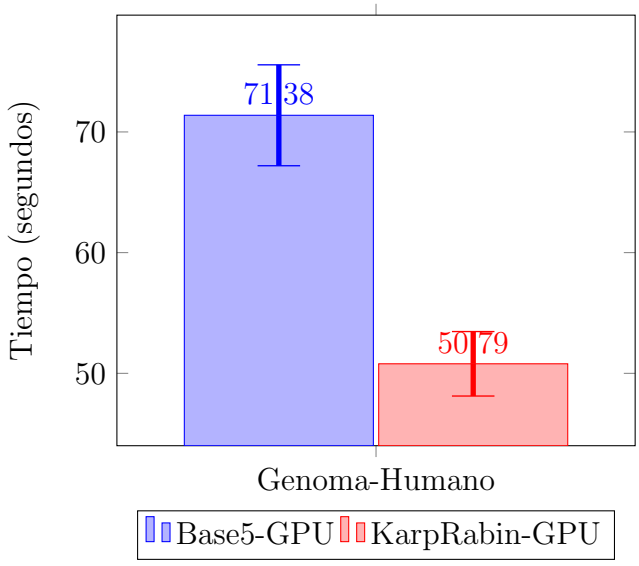
mizar la heurística del BLAST multisecuencial, observando una reducción en tiempo de procesamiento durante el sembrado el algoritmo CKR con arquitectura GPU como se muestra en la "**Figura 5.5**", el cual de forma similar al implementado con arquitectura CPU induciría a un aumento en el tiempo de procesamiento de la fase de extensión del algoritmo BLAST multisecuencial "**Tabla 5.1**", por lo cual el algoritmo sugerido sería el algoritmo CB5.



(a) Tiempo de Generación de Claves Hash



(b) Tiempo de Inserción de punteros en la Tabla Hash



(c) Tiempo Total de Indexamiento

Figura 5.5: Tiempo de generación de Claves Hash (a), inserción de los punteros en la tabla Hash (b) y tiempo total de indexamiento (c) en el indexamiento de semillas BLAST multisecuencial con CudaBase5 y CudaKarpRabin con arquitectura GPU en el alineamiento de secuencias de ADN con el genoma humano

5.5. Evaluación y comparación del rendimiento en tiempo de procesamiento de los algoritmos optimizados de alineamiento de secuencias de ADN: BLAST unisecuencial, BLAST multisecuencial, BWT

5.5.1. Para Blast unisecuencial

Evaluando y comparando el rendimiento de los algoritmos con el mejor tiempo de procesamiento con arquitectura CPU y GPU para la optimización del BLAST unisecuencial, se evidencia que el algoritmo CN que mostró mejor redimiento en tiempo de procesamiento con arquitectura GPU, posee un tiempo de procesamiento menor al algoritmo KMP con arquitectura CPU como se ilustra en la "**Figura 5.6**". Mostranto el algoritmo CudaNaive (CN) optimizado con GPU un speedup de latencia de 1.24X superior al algoritmo KMP sin paralelización con CPU.

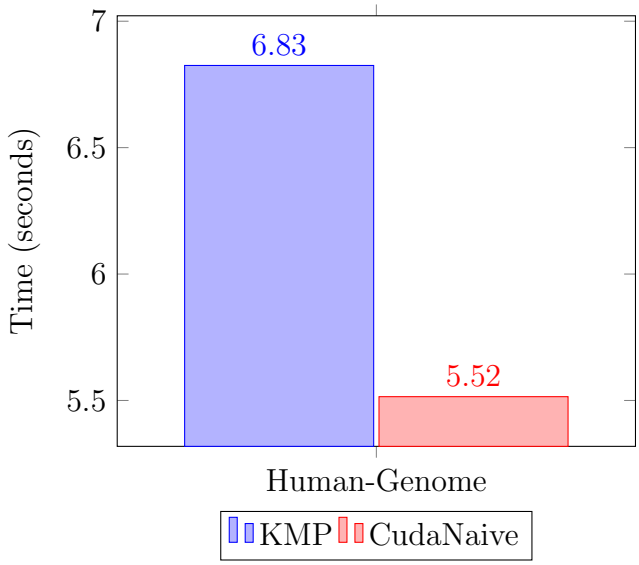


Figura 5.6: Tiempo de procesamiento de los algoritmos KMP con arquitectura CPU y CUdaNaive con arquitectura GPU como optimización del algoritmo BLAST unisecuencial en el alineamiento de una secuencia con el genoma humano

5.5.2. Para Blast multisecuencial

Evaluando y comparando el rendimiento de los algoritmos con el mejor tiempo de procesamiento con arquitectura CPU y GPU para la optimización del BLAST multisecuencial, se evidencia que el algoritmo CB5 mostró mejor redimiento en tiempo de procesamiento con arquitectura GPU, con un tiempo de procesamiento menor al algoritmo B5 con arquitectura CPU con un speedup de latencia de 1.23X, a pesar de mostrar un

aparente mejor rendimiento los algoritmos CKR y KR como se ilustra en la "**Figura 5.7**".

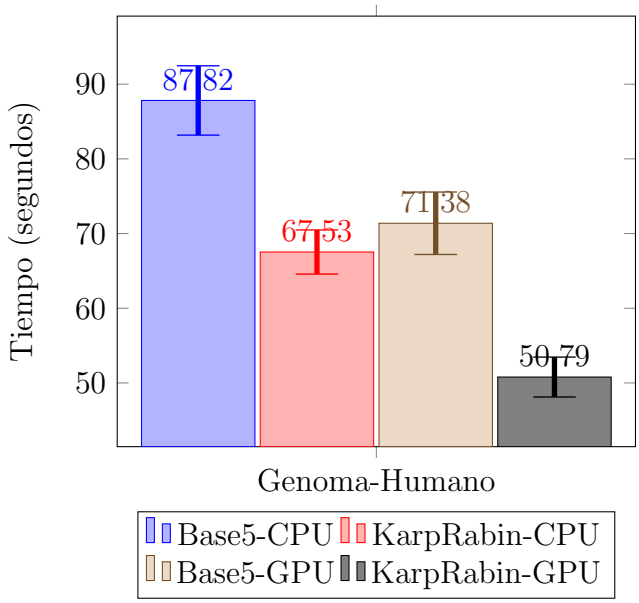


Figura 5.7: Tiempo total de indexamiento en el indexamiento de semillas BLAST multi-secuencial con Base5 y KarpRabin con arquitecturas CPU y GPU en el alineamiento de secuencias de ADN con el genoma humano

Asimismo al realizar la comparación de tiempos de ejecución del CB5 y B5 variando la longitud de la clave hash procesando la secuencia completa del genoma de *Gallus gallus* se muestra un mejor rendimiento en el tiempo de ejecución de la implementación con GPU que la implementación secuencial si su longitud de la clave hash es mayor a 4 como se observa en la "**Figura 5.8**".

Con una longitud de la clave hash de 5 pb se logra una optimización de hasta un 11.24 % del tiempo de ejecución del CB5 en comparación con la implementación secuencial del B5 considerando el tiempo total del indexamiento de semillas (tiempo de generación de semillas + tiempo de inserción de los punteros dentro de la Tabla Hash) del algoritmo BLAST como se observa en la "**Figura. 5.9**".

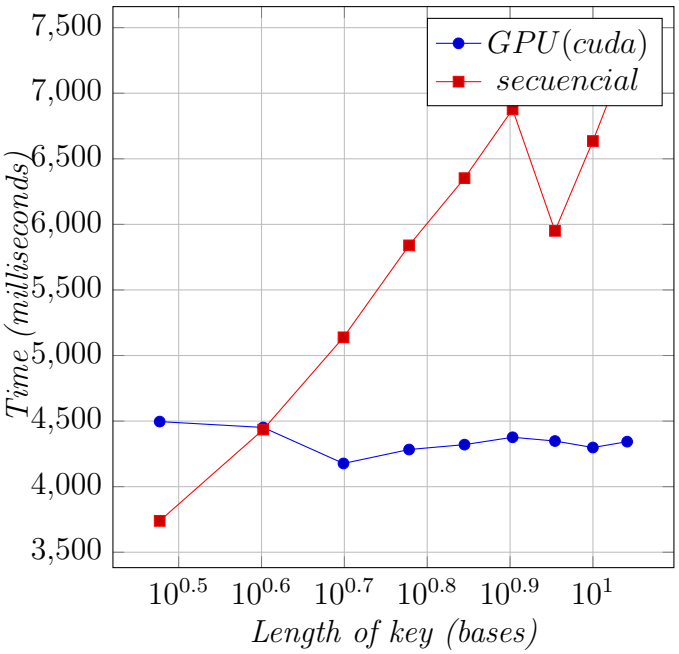


Figura 5.8: Processing time vs. length size_key on DNA sequence of *Gallus gallus* getting the hash keys.

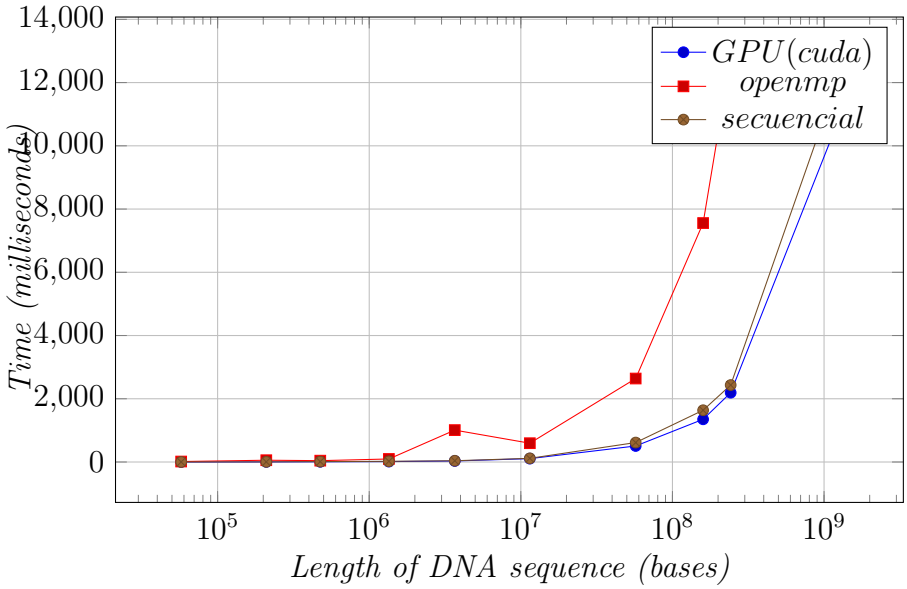


Figura 5.9: Processing time vs. length of DNA reference sequence getting Hash Table of seed index with size_key = 5pb

Capítulo 6

Conclusiones y Trabajos Futuros

Se optimizó en tiempo de procesamiento el algoritmo BLAST en el alineamiento de secuencias de ADN usando procesamiento masivamente paralelo y distribuido, para lo cual se concluye:

- Se identificó y analizó los algoritmos de Fuerza Bruta o Naive (BF), Boyer-Moore (BM), Knut Morris Pratt (KMP), Karp-Rabin (KR), Transformada de Burrows Wheeler (BWT) y Base 5 (B5) para la optimización del algoritmo BLAST (unisecuencial y multisecuencial) durante el sembrado, en el alineamiento de secuencias de ADN.

- Se Implementó y comparó los algoritmos BF, BM, KMP KR y BWT para la optimización del algoritmo BLAST (unisecuencial) y KR y B5 para la optimización del algoritmo BLAST (multisecuencial) en el alineamiento de secuencias de ADN con arquitectura CPU (sin paralelismo) mostrando un menor tiempo de procesamiento de 6.83s el algoritmo KMP al realizar alineamientos sobre el genoma humano de forma unisecuencial y el algoritmo B5 con 87.82s de forma multisecuencial.

- Se Implementó y comparó los algoritmos CN, *Cuda Booyer-Moore* (CBM), *Cuda Knut Morris Pratt* (CKMP) y CKR para la optimización del algoritmo BLAST (unisecuencial) y CKR y CB5 para la optimización del algoritmo BLAST (multisecuencial) en el alineamiento de secuencias de ADN con arquitectura GPU (con paralelismo) usando la metodología de Foster, mostrando un menor tiempo de procesamiento de 5.52s el algoritmo CN al realizar alineamientos sobre el genoma humano de forma unisecuencial y el algoritmo CB5 con 71.38s de forma multisecuencial.

- Se evaluó y comparó el rendimiento en tiempo de procesamiento de los algoritmos optimizados de alineamiento de secuencias de ADN, siendo los algoritmos con mejor rendimiento: CN para BLAST unisecuencial con un speedup de latencia de 1.24X sobre el algoritmo KMP y CB5 para BLAST multisecuencial con un speedup de latencia de 1.23X sobre el algoritmo B5 .

6.1. Problemas encontrados

Los problemas encontrados son:

- Problemas de configuración con CUDA durante la etapa de programación tanto al elegir el sistema operativo como la versión de CUDA.
- Al procesar formato Fasta de secuencias de ADN de gran tamaño como genomas, la dificultad de trabajar directamente con este tipo de archivo se encuentra en separar las secuencias biológicas de las descripciones y comentarios de una forma rápida y con poco gasto de memoria para poder procesar de una mejor forma en entornos distribuidos y el uso de paralelismo con CPU y GPU [Alshammari, 2014].
  .

6.2. Recomendaciones

- Realizar la implementación usando como sistema operativo Ubuntu.
- Usar de preferencia para la comparación de algortimos bases de datos de ADN en formato FASTA y no otro formato.
- Probar los algoritmos usando el mismo computador o supercomputador.

6.3. Trabajos futuros

- Para poder interpretar resultados usando el algoritmo propuesto es necesario la creación de un framework con Interfaz Grafica que muestre los alineamientos, lo cual es una propuesta a posterior.
- Es necesario ampliar la comparación de este algoritmo BLAST optimizado en el sembrado en su optimización para la fase de extensión en entornos distribuidos con GPU.
- Realizar el indexamiento de los algoritmos CB5 o CKR para la optimización del sembrado del algoritmo BLAST multisecuencial en entornos distribuidos, donde la memoria disponible a utilizar sea escalable.
- Comparar en tiempo de procesamiento de los resultados de realizar el alineamiento de secuencias de ADN en la plataforma BLAST dada por el NCBI con el BLAST optimizado propuesto por la presente investigación.

Bibliografía

[Adjeroh et al., 2008] Adjeroh, D., Bell, T., and Mukherjee, A. (2008). *The Burrows-Wheeler Transform:: Data Compression, Suffix Arrays, and Pattern Matching*. Springer Science & Business Media.

[Alshammari, 2014] Alshammari, H. (2014). Dna sequence alignment using hadoop in cloud computing environment. *International Journal of Computer Science and Information Security*, 12(5):19.

[Altschul et al., 1990] Altschul, S. F., Gish, W., Miller, W., Myers, E. W., and Lipman, D. J. (1990). Basic local alignment search tool. *Journal of Molecular Biology*, 215(3):403 – 410.

[Autores, 2018a] Autores, V. (2018a). Emsembl. Accedido el 10-02-2018 a la url https://goo.gl/HJDEqq.

[Autores, 2018b] Autores, V. (2018b). National center for biotechnology information search database (ncbi). Accedido el 05-04-2018 a la url https://goo.gl/HJDEqq.

[Bauer et al., 2013] Bauer, M. J., Cox, A. J., and Rosone, G. (2013). Lightweight algorithms for constructing and inverting the bwt of string collections. *Theoretical Computer Science*, 483:134–148.

[Boyer and Moore, 1977] Boyer, R. S. and Moore, J. S. (1977). A fast string searching algorithm. *Communications of the ACM*, 20(10):762–772.

[Burrows and Wheeler, 1994] Burrows, M. and Wheeler, D. J. (1994). A block-sorting lossless data compression algorithm.

[Chen et al., 2015] Chen, Y., Ye, W., Zhang, Y., and Xu, Y. (2015). High speed blastn: an accelerated megablast search tool. *Nucleic acids research*, 43(16):7762–7768.

[Cormen et al., 2009] Cormen, T. H., Leiserson, C. E., Rivest, R. L., and Stein, C. (2009). *Introduction to algorithms*. MIT press.

[Darling et al., 2003] Darling, A. E., Carey, L., and Feng, W. C. (2003). The design, implementation, and evaluation of mpiblast. Technical report, Los Alamos National Laboratory.

[Deymonnaz, 2012] Deymonnaz, A. (2012). Arreglos de sufijos para alineamiento de secuencias de adn en memoria acotada. Master's thesis, Universidad de Buenos Aires Departamento de Computacion, Buenos Aires Argentina.

[Eddy, 2011] Eddy, S. R. (2011). Accelerated profile hmm searches. *PLoS computational biology*, 7(10):e1002195.

[Elloumi and Zomaya, 2013] Elloumi, M. and Zomaya, A. Y. (2013). *Biological Knowledge Discovery Handbook: Preprocessing, Mining and Postprocessing of Biological Data*, volume 23. John Wiley & Sons.

[Foster, 1995] Foster, I. (1995). *Designing and building parallel programs: concepts and tools for parallel software engineering*. Addison-Wesley Longman Publishing Co., Inc.

[Georganas et al., 2015] Georganas, E., Buluc, A., Chapman, J., Oliker, L., Rokhsar, D., and Yelick, K. (2015). meraligner: A fully parallel sequence aligner. In *Parallel and Distributed Processing Symposium (IPDPS), 2015 IEEE International*, pages 561–570.

[Jiménez García, 2003] Jiménez García, L. F. (2003). *Biología celular y molecular*. Pearson Educación de México.

[Karp and Rabin, 1987] Karp, R. M. and Rabin, M. O. (1987). Efficient randomized pattern-matching algorithms. *IBM journal of research and development*, 31(2):249–260.

[Khare et al., 2017] Khare, N., Khare, A., and Khan, F. (2017). Hcudablast: an implementation of blast on hadoop and cuda. *Journal of Big Data*, 4(1):41.

[Knuth et al., 1977] Knuth, D. E., Morris, Jr, J. H., and Pratt, V. R. (1977). Fast pattern matching in strings. *SIAM journal on computing*, 6(2):323–350.

[Krishnan, 2005] Krishnan, A. (2005). Gridblast: a globus-based high-throughput implementation of blast in a grid computing framework. *Concurrency and Computation: Practice and Experience*, 17(13):1607–1623.

[Lam et al., 2008] Lam, T. W., Sung, W. K., Tam, S. L., Wong, C. K., and Yiu, S. M. (2008). Compressed indexing and local alignment of dna. *Bioinformatics*, 24(6):791–797.

[Li and Durbin, 2009] Li, H. and Durbin, R. (2009). Fast and accurate short read alignment with burrows–wheeler transform. *bioinformatics*, 25(14):1754–1760.

[Li and Homer, 2010] Li, H. and Homer, N. (2010). A survey of sequence alignment algorithms for next-generation sequencing. *Briefings in Bioinformatics*, 11(5):473–483.

[Lin et al., 2011] Lin, H., Ma, X., Feng, W., and Samatova, N. F. (2011). Coordinating computation and i/o in massively parallel sequence search. *IEEE Transactions on Parallel and Distributed Systems*, 22(4):529–543.

[Liu et al., 2014] Liu, C.-M., Luo, R., and Lam, T.-W. (2014). Gpu-accelerated bwt construction for large collection of short reads. *arXiv preprint arXiv:1401.7457*.

[Ma et al., 2002] Ma, B., Tromp, J., and Li, M. (2002). Patternhunter: faster and more sensitive homology search. *Bioinformatics*, 18(3):440–445.

[Mardis, 2008] Mardis, E. R. (2008). Next-generation dna sequencing methods. *Annu. Rev. Genomics Hum. Genet.*, 9:387–402.

[Matsunaga et al., 2008] Matsunaga, A., Tsugawa, M., and Fortes, J. (2008). Cloudblast: Combining mapreduce and virtualization on distributed resources for bioinformatics applications. In *eScience, 2008. eScience'08. IEEE Fourth International Conference on*, pages 222–229. IEEE.

[Mulia et al., 2012] Mulia, S., Mishra, D., and Jena, T. (2012). Profile {HMM} based multiple sequence alignment for {DNA} sequences. *Procedia Engineering*, 38:1783 – 1787.

[Nieves Hurtado and Domínguez Sánchez, 2002] Nieves Hurtado, A. and Domínguez Sánchez, F. C. (2002). *Métodos numéricos: aplicados a la ingeniería*. Grupo Editorial Patria.

[Ocampo Yahuarcani and Cárdenas Vigo, 2017] Ocampo Yahuarcani, I. and Cárdenas Vigo, R. (2017). *Manual de Acceso Remoto a la Supercomputadora MANATI" del IIAP y Ejecución de Programas en la Cola de Trabajo PBS*.

[Salinas and Lisbona, 2016] Salinas, J. C. and Lisbona, F. J. Y. (2016). *Manual de prácticas de Bioinformática*, volume 5. Universidad Almería.

[Sawyer et al., 2015] Sawyer, S. E., Rekepalli, B., Horton, M. D., and Brook, R. G. (2015). Hpc-blast: distributed blast for xeon phi clusters. In *Proceedings of the 6th ACM Conference on Bioinformatics, Computational Biology and Health Informatics*, pages 512–513. ACM.

[Simpson and Durbin, 2010] Simpson, J. T. and Durbin, R. (2010). Efficient construction of an assembly string graph using the fm-index. *Bioinformatics*, 26(12):i367–i373.

[Smith and Waterman, 1981] Smith, T. F. and Waterman, M. S. (1981). Identification of common molecular subsequences. *Journal of molecular biology*, 147(1):195–197.

[Ticona, 2003] Ticona, W. G. C. (2003). *Aplicação de Algoritmos Genéticos Multi-Objetivo para Alinhamento de Seqüências Biológicas*. PhD thesis, Instituto de Ciências Matemáticas e de Computaçao.

[Vouzis and Sahinidis, 2010] Vouzis, P. D. and Sahinidis, N. V. (2010). Gpu-blast: using graphics processors to accelerate protein sequence alignment. *Bioinformatics*, 27(2):182–188.

[Ye et al., 2017] Ye, W., Chen, Y., Zhang, Y., and Xu, Y. (2017). H-blast: a fast protein sequence alignment toolkit on heterogeneous computers with gpus. *Bioinformatics*, 33(8):1130–1138.

[Yim and Cushman, 2017] Yim, W. C. and Cushman, J. C. (2017). Divide and conquer (dc) blast: fast and easy blast execution within hpc environments. *PeerJ*, 5:e3486.

[Zhang et al., 2017] Zhang, J., Wang, H., and Feng, W.-c. (2017). cublastp: Fine-grained parallelization of protein sequence search on cpu+ gpu. *IEEE/ACM Transactions on Computational Biology and Bioinformatics (TCBB)*, 14(4):830–843.

[Zhao and Chu, 2014] Zhao, K. and Chu, X. (2014). G-blastn: accelerating nucleotide alignment by graphics processors. *Bioinformatics*, 30(10):1384–1391.