ESCUELA POLITÉCNICA NACIONAL

FACULTAD DE INGENIERÍA DE SISTEMAS

ANÁLISIS COMPARATIVO DEL DESEMPEÑO COMPUTACIONAL ENTRE DOS GESTORES DE BASES DE DATOS EN CONSULTAS SOBRE LA BASE DE DATOS UNIPROT/SWISS-PROT

TRABAJO DE TITULACIÓN PREVIO A LA OBTENCIÓN DEL TÍTULO DE INGENIERO EN SISTEMAS INFORMÁTICOS Y DE COMPUTACIÓN

KATHERINE GEOMARA VIVANCO VÁSQUEZ

katherine.vivanco@epn.edu.ec

DIRECTOR: IVÁN MARCELO CARRERA IZURIETA

ivan.carrera@epn.edu.ec

Quito, mayo 2019

AVAL

Certifico que el presente trabajo fue desarrollado por Katherine Geomara Vivanco Vásquez, bajo mi supervisión.

Prof. Iván Carrera Izurieta
DIRECTOR DEL TRABAJO DE TITULACIÓN

DECLARACIÓN DE AUTORÍA

Yo, Katherine Geomara Vivanco Vásquez, declaro bajo juramento que el trabajo aquí descrito es de mi autoría; que no ha sido previamente presentada para ningún grado o calificación profesional; y, que he consultado las referencias bibliográficas que se incluyen en este documento.

A través de la presente declaración cedo mis derechos de propiedad intelectual correspondientes a este trabajo, a la Escuela Politécnica Nacional, según lo establecido por la Ley de Propiedad Intelectual, por su Reglamento y por la normatividad institucional vigente.

Katherine Geomara Vivanco Vásquez

DEDICATORIA

Dedico este proyecto a mi familia, padres y hermanos que se han hecho presentes durante todo el desarrollo de mi vida personal como profesional, a ustedes se lo debo todo.

A mis padres Vinicio y Cecilia, mis mejores amigos y ejemplos a seguir. Por haberme apoyado, haber compartido toda mi etapa estudiantil que bajo su protección, desvelo, bendiciones y amor me han impulsado a no decaer, a ser fuerte y perseverante. Hoy puedo sentirme orgullosa de la persona que soy, ya que ustedes han dirigido mi camino. Son mis personas favoritas, mi alegría, mi fuerza y orgullo.

A mi hermano Alexis, futuro colega y mentor. Quien nunca permitió que caiga, me acompañó en cada tropiezo y me prestó su ayuda infinita.

A mi hermana Natalie, quien me dio el empujón para el inicio de mi vida profesional y junto a Juan Sebastián me han brindado fuerza y cariño.

A mi hermana Andrea, mi compañera y amiga. Quien ha hecho mi vida muy interesante acompañado de sube y bajas, gracias a tu positivismo he cumplido mis propósitos.

Se que el trayecto no fue fácil, pero lo logramos.

AGRADECIMIENTO

Agradezco primeramente a Dios, por brindarme salud, infinita bondad y amor. Por haberme brindado la fuerza que se la pedí cuando sentía caer.

Mi inmenso agradecimiento a mi director, Iván Carrera. Quien ha sido una excelente guía en la realización del proyecto y de mi vida estudiantil. Un excelente profesor, quien me dio la oportunidad de empezar y terminar el proyecto de la mejor manera.

Agradezco a Eric Aldas por la paciencia, conocimiento y apoyo en la culminación del proyecto. Supiste orientarme cuando perdía el objetivo. Gracias por ser una parte sustancial de mi vida estudiantil y personal.

A mis compañeros de vida estudiantil y laboral. A Santiago Lema, gracias por compartir conmigo un sinfín de experiencias, viajes y anhelos. Por la confianza depositada en cada proyecto, sin duda formas una parte importante en la culminación de este objetivo. A Cristian Naranjo quien ha compartido conmigo cada alegría en este proyecto.

Y sin hacer de menos a cada uno de mis amigos, compañeros y profesores de la facultad.

ÍNDICE DE CONTENIDO

RESUMEN

La cantidad de datos biológicos a generarse en el futuro inmediato por el uso de nuevas tecnologías de secuenciación requerirá de aplicaciones de alto rendimiento para la manipulación de bases de datos generadas. Aunque los costos del almacenamiento a nivel mundial han ido disminuyendo, la disponibilidad y costo computacional que se requieren para el tratamiento de los datos serán cada vez más elevados; por esto, será necesario una mayor optimización en la manipulación de las bases de datos biológicas.

El presente proyecto propone definir qué gestor de bases de datos NoSQL, sea Elasticsearch o Hadoop, proporciona un mejor desempeño computacional en términos del tiempo de ejecución, para las operaciones CRUD (Create, Read, Update, Delete) sobre los datos en la base de información proteica UniProt/Swiss-Prot. El criterio con el que se definirá cuál de las herramientas es más eficiente en costo computacional en relación con el tiempo será el rango de tiempo que demora la mayoría de las instrucciones en ejecutarse, evaluadas individualmente. Las operaciones serán enviadas desde scripts como benchmarks del presente proyecto mediante servicios web hacia cada gestor sobre la base de datos UniProt/Swiss-Prot. El análisis comparativo se realizará en un entorno de clúster con tres nodos para cada herramienta gestora, estos nodos se los representarán mediante máquinas virtuales de iguales características en cuanto a memoria ram, física y sistema operativo.

PALABRAS CLAVE: bioinformática, NoSQL, Elasticsearch, Hadoop, Swiss-Prot, benchmark

ABSTRACT

The large biological data sets to be generated in the immediate future as a result of the usage of new sequencing technologies requires high performance applications to manipulate databases. Although, the storage costs of the worldwide have decreased, the availability and computational cost required to process the data is still growing. Thus, it is necessary a higher optimization in the database manipulation. Being this way how bioinformatics solves the problem of the organization of information and manipulation of large data sets.

This paper proposes to test which NoSQL database manager, Elasticsearch or Hadoop, provides a better computational performance in terms of execution time for CRUD operations in data of proteics database UniProt/Swiss-Prot. The criteria used to identify which database manager is the more efficient tool in computational cost related to time is the delay of execution of several queries individually evaluated. The CRUD (Create, Read, Update, Delete) operations will be sent from scripts as project benchmarks through web services to each database manager on UniProt/Swiss-Prot database. The comparative analysis will be executed in a cluster environment with three nodes for each manager tool.

The comparative analysis has been realized in a cluster environment with three nodes for each management tool. These nodes will be virtual machines with similar characteristics like same memory ram, storage and operative system.

KEYWORDS: bioinformatics, NoSQL, Elasticsearch, Hadoop, Swiss-Prot, benchmark

1. INTRODUCCIÓN

La bioinformática es una disciplina que nace como vínculo entre la informática y las ciencias biológicas, a partir del desarrollo de la investigación en disciplinas como la genómica o la proteómica, y de las nuevas tecnologías [1]. La bioinformática se define como la aplicación de técnicas computacionales para organizar y reconocer la información asociada a macromoléculas biológicas [1].

La bioinformática se fundamenta en tres objetivos: primero, generar métodos para acceder a los datos y permitir que los investigadores generen nuevas entradas al ritmo como estas se vayan produciendo; segundo, desarrollar herramientas de análisis de datos, el mismo que va de la mano con el tercero, interpretar los resultados [2].

La investigación biológica genera una constante acumulación de información, enriqueciendo así, de manera constante, las bases de datos biológicas. Los investigadores necesitan acceder a la información de estas bases de datos y por lo tanto dependen de su disponibilidad [2].

El principal desafío que presenta el Big Data aplicado en problemas de bioinformática es el volumen de la información a la que tienen acceso los investigadores. De esto han surgido preguntas como: ¿qué sistema de archivos elegir? y ¿cuál es la vía lo suficientemente rápida para acceder a estos datos? [1]. Se ha planteado que es imperativo que la manipulación de los datos sobre las bases de datos biológicas como UniProt/Swiss-Prot sea eficiente, y que la rapidez de respuesta a solicitudes no sea un problema frente a la cantidad de datos que contiene la base de datos. Además, como podemos ver en la Figura 1.1, la curva de crecimiento de UniProt/Swiss-Prot en los últimos años ha sido pronunciada y durante los últimos diez años (de 2009 a 2019) se ha mantenido en un rango de 500,000 entradas de información, datos o registros.

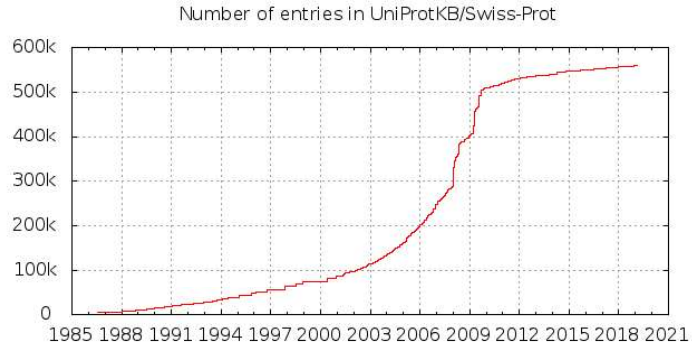

Figura 1.1 Crecimiento de la base de datos UniProt/Swiss-Prot **[3]**

El presente proyecto realizará operaciones CRUD[1] sobre la base de datos UniProt/Swiss-Prot, una base de datos con gran cantidad de información, para indicar cuál de las herramientas propuestas, Elasticsearch y Hadoop, realiza las operaciones de una manera relativamente rápida. Proponiendo de esta manera una herramienta que resuelva el desafío de acceso a los datos de bases de datos biológicas sin importar el volumen. La evaluación de la herramienta gestora de base de datos se realizará siguiendo la metodología de "Evaluación de Desempeño" [4], una metodología que ayudará a elegir la herramienta más eficiente en términos de tiempo, tomando en cuenta todos los factores que pueden influir en los resultados.

La base de datos UniProt/Swiss-Prot, que consta de alrededor de 550,950 entradas de datos de proteínas, en formato XML será la base de datos que se utilizará en el proyecto. Las principales características de la base de datos UniProt/Swiss-Prot son [5]:

1) Anotación: indicando la información de que están conformados cada uno de sus datos.
2) Redundancia mínima: busca minimizar la redundancia de la base de datos.
3) Integración con otras bases de datos: permite la interconectividad entre tres tipos de bases de datos relacionadas con secuencias.
4) Documentación: UniProt/Swiss-Prot proporciona una gran cantidad de documentación especializada.

1.1 Pregunta de investigación

¿Existe una diferencia significativa entre el desempeño computacional en términos de tiempo de ejecución de consultas entre los gestores de bases de datos NoSQL Elasticsearch y Hadoop?

1.2 Objetivo General

Realizar el análisis comparativo del desempeño computacional entre dos gestores de bases de datos en consultas CRUD (creación, lectura, actualización y borrado) sobre la base de datos UniProt/Swiss-Prot.

[1] Acrónimo de Create, Read, Update, Delete que se refiere a las operaciones que se pueden realizar sobre una base de datos [38]

1.3 Objetivos Específicos

a) Estudiar el funcionamiento las herramientas ElasticSearch y Hadoop, para comprender sus servicios y salidas.

b) Conocer la base de datos UniProt/Swiss-Prot mediante el estudio y estructura de sus datos.

c) Levantar el ambiente para la realización de operaciones CRUD, mediante la creación de una API de servicios Web.

d) Comparar los resultados obtenidos al efectuar las operaciones CRUD sobre la base de datos UniProt/Swiss-Prot entre las herramientas gestoras de base de datos ElasticSearch y Hadoop.

1.4 Alcance

El presente trabajo de investigación realizará un análisis de rendimiento computacional en términos de tiempo de ejecución entre dos herramientas gestoras de bases de datos y proveerá, como resultado del proyecto, de una de ellas como la mejor para la manipulación de los datos de la base de datos UniProt/Swiss-Prot y otras bases de datos con características similares.

Para ello se han planteado scripts que contienen operaciones CRUD que se realizarán desde cada gestor de base de datos sobre la base de datos previamente mencionada.

Los datos más relevantes de la base de datos UniProt/Swiss-Prot que se utilizarán para visualizar la información de las pruebas de desempeño son: *accession, name, gene y organism*. Esto ya que son los datos que caracterizan a las proteínas de la base de datos y que son los datos mediante los cuales generalmente se busca información de una proteína en particular.

Para las pruebas de evaluación de desempeño se usarán como gestores de bases de datos las herramientas Elasticsearch y Hadoop. Ambos presentan una estructura de clúster con un modelo de arquitectura Maestro-Esclavo. Adicionalmente, las herramientas fueron levantadas en un entorno conformado con máquinas virtuales de igual hardware y sistema operativo.

1.5 Marco Teórico

La bioinformática moderna es una disciplina científica en constante desarrollo que mediante métodos computacionales analiza una variedad de datos biomédicos [6]. La recopilación de datos biomédicos avanza más rápido que las tecnologías de procesamiento

y análisis de datos, haciendo hincapié en la amplia brecha entre el rápido progreso tecnológico, en la adquisición de datos y la lenta caracterización funcional de la información biomédica [7]. La rapidez de recopilación de información biomédica genera bases de datos biológicas con un volumen de datos relativamente grande como UniProt/Swiss-Prot.

En la actualidad, el concepto de almacenamiento relacional ya no corresponde a grandes volúmenes de datos, una variedad de formatos, una necesidad de compartir datos de distintas partes del mundo y una variedad de consultas de búsqueda [6]. Básicamente, el mecanismo de consultas sobre sistemas gestores de bases de datos relacionales crea tablas temporales que, hablando en términos de grandes cantidades de datos, hace que el trabajo sea ineficiente [6].

Frente a la problemática de tener que lidiar con grandes volúmenes de datos, velocidad y poca escalabilidad de información, grandes empresas como Google y Amazon en el año 2000 generan el concepto de NoSQL para referirse a un sistema de gestión de bases de datos no relacional diseñado para trabajar con grandes volúmenes y variedad de datos [6]. Los sistemas NoSQL no se convertirán en sustitutos de los sistemas relacionales, pero podrían comportarse ventajosamente en ciertos escenarios y bajo varias condiciones [8].

Los gestores de bases de datos Elasticsearch y Hadoop, permiten la manipulación de grandes volúmenes de datos complejos ya que ambas de manera independiente proveen un lenguaje para representarlos y almacenarlos. Considerando que Elasticsearch y Hadoop son dos herramientas que reaccionan de una manera positiva a la rapidez en la manipulación de bases de datos complejas como lo es UniProt/Swiss-Prot, y el tiempo de respuesta de estas herramientas es el recurso a analizar en el proyecto, la metodología de “Evaluación de Desempeño” [4] provee una guía que toma en cuenta todos elementos presentes en un escenario de evaluación comparativa, como parámetros, factores externos e internos y carga de trabajo a los que se encuentran expuestos los sistemas gestores.

A continuación, se detallan los conceptos que forman la base de conocimiento del proyecto:

Gestor de Bases de Datos

Un sistema gestor de bases de datos es un tipo de software que sirve como interfaz entre la base de datos, el usuario y las aplicaciones que acceden y gestionan los datos [9] [10].

Los sistemas gestores de bases de datos están diseñados para manejar grandes bloques de información, que corresponden tanto la información de las estructuras para el almacenamiento como de mecanismos para la gestión de la información [10]. Un Gestor de Bases de datos debe cumplir con las siguientes características [10]:

- Permitir la especificación de la estructura de los datos y la relación entre ellos. Esta información se almacena en el diccionario de datos que, el sistema gestor de base de datos proporcionará mecanismos para la gestión de éste.
- Permitir realizar consultas, inserciones, actualizaciones y borrado, que en todo el proyecto se lo llamará CRUD, mediante un lenguaje de manipulación de datos.
- Controlar el acceso a los datos de la base de datos.
- Mantener la consistencia e integridad de los datos.
- Controlar la concurrencia de usuarios en el acceso a la base de datos.
- Proporcionar un mecanismo para la recuperación de los datos en el caso que el sistema falle.

Bases de datos no estructuradas

El término Bases de datos no estructuradas o NoSQL hace referencia al conjunto de tecnologías, en bases de datos, que buscan alternativas al sistema de bases de datos relacional, en un contexto donde priman la velocidad, el manejo de grandes volúmenes de datos y la posibilidad de tener un sistema distribuido [11]. Las diferentes tecnologías, propuestas hasta ahora, han definido las propiedades de estas bases de datos NoSQL. Se proponen cinco características específicas para poder encasillar este tipo de bases de datos [12]:

- Facilidad de añadir, eliminar o realizar operaciones con elementos (hardware) del sistema, sin afectar el rendimiento.
- Habilidad de replicar y distribuir los datos a través de varios servidores.
- Baja concurrencia comparado con sistemas relacionales.
- Uso eficiente de índices y RAM para almacenamiento de datos.
- Capacidad de agregar nuevos atributos a registro de datos.

Gestor de Base de Datos NoSQL

Por los conceptos de Gestor de base de datos, y de Base de datos no estructurada, se puede indicar que un gestor de base de datos NoSQL es una herramienta mediante la cual se proporciona mecanismos para almacenar y gestionar los datos de bases de datos NoSQL.

Benchmarking

Benchmarking o evaluación comparativa es definida como el proceso de comparar dos o más sistemas mediante medidas obtenidas [13]. Un *benchmark* se define como un programa o conjunto de programas que se ejecuta en varios sistemas para comparar el rendimiento en base a términos definidos. Un *benchmark* se presenta como una condición que determina bajo qué términos dos sistemas pueden compararse y dar resultados válidos [13].

Big Data

Big Data ha sido una de las mayores tendencias tecnológicas en los últimos años, y se involucra en varios campos como: negocios, medicina, ciencia, etc. Pese a todo el conocimiento que se ha podido abarcar acerca de este tema a lo largo de los años, ha sido difícil definir Big Data. Para poder facilitar su entendimiento, Doug Laney en [14] usó las 3Vs para caracterizar de manera más precisa a Big Data: velocidad, variedad y volumen. Velocidad se refiere a la rapidez de entrada y salida de los datos, variedad se refiere al conjunto de datos en distintos formatos y el volumen se refiere al tamaño del conjunto de datos [14].

MapReduce

MapReduce es un modelo de programación para el procesamiento y generación de grandes cantidades de datos [15], y principalmente fue seleccionado para dar soporte a la computación paralela [16].

MapReduce toma un conjunto de entrada de pares clave/valor y genera un conjunto de salida de pares clave valor, este procesamiento se expresa como dos funciones *Map* y *Reduce* [15].

Map: toma un par de clave/valor de entrada y lo procesa para obtener un par clave/valor intermedio de salida [15].

MapReduce toma todos los valores asociados a la clave del intermedio y los pasa a la función *Reduce* [15]. Este paso se lo conoce como *Shuffle* [16].

Reduce: acepta la clave intermedia y un conjunto de valores asociados para esa clave y mezcla estos valores para formar un conjunto de valores más pequeño que el de entrada [15].

Un resumen del algoritmo MapReduce se puede observar en la Figura 1.2.

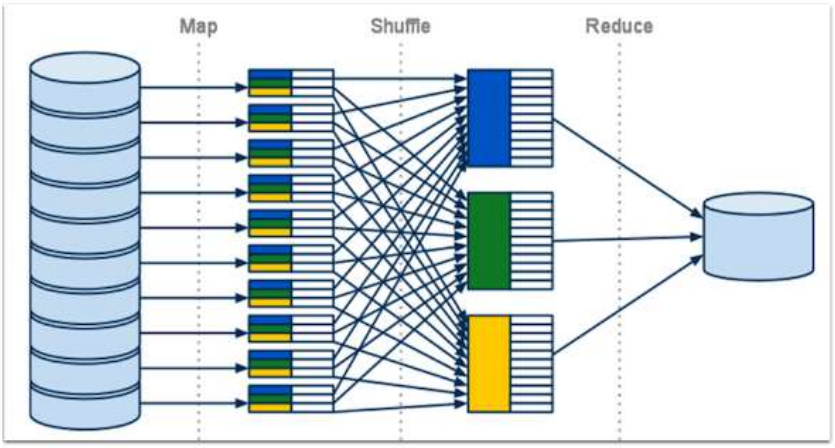

Figura 1.2 Proceso de MapReduce **[17]**

El resto del presente documento se organiza de la siguiente manera:

En el capítulo 2, Metodología, se presenta una descripción más detallada del trabajo realizado, así como los métodos empleados para depurar la base de datos Uniprot/Swissprot además del proceso de análisis de datos y por último las herramientas y el procedimiento de uso de estas para el cumplimiento del objetivo del proyecto.

En el capítulo 3, Resultados y Discusión, principalmente se obtiene la respuesta a la pregunta de investigación presentada en base a la con los resultados de los análisis realizados.

En el capítulo 4, se detallan las conclusiones obtenidos al final del proyecto en base a la experiencia obtenida durante el proyecto.

2. METODOLOGÍA

Para lograr el cumplimiento del objetivo se optó por seguir dos metodologías, la primera, metodología de "Ciclo de Vida del Análisis de Big Data" [18], que preparará los datos para realizar las pruebas y experimentos; y la segunda, metodología de "Evaluación de Desempeño" [4], la cual dirigirá los pasos a realizar para cumplir con el objetivo. La metodología será aplicada en este orden.

1. Metodología de Ciclo de Vida del Análisis de Big Data.
2. Metodología de Evaluación de Desempeño.

2.1 Metodología de Ciclo de Vida del Análisis de Big Data

La metodología para la depuración de los datos es Ciclo de Vida del Análisis de Big Data [18], consta de nueve pasos que se presentan en la Figura 2.1:

Figura 2.1 Representación del Ciclo de Vida del Análisis de Big Data [18]

Dado que el objetivo del proyecto no es análisis de Big Data específicamente la metodología se la redujo a seis pasos, los cuales se presentan en la Figura 2.2.

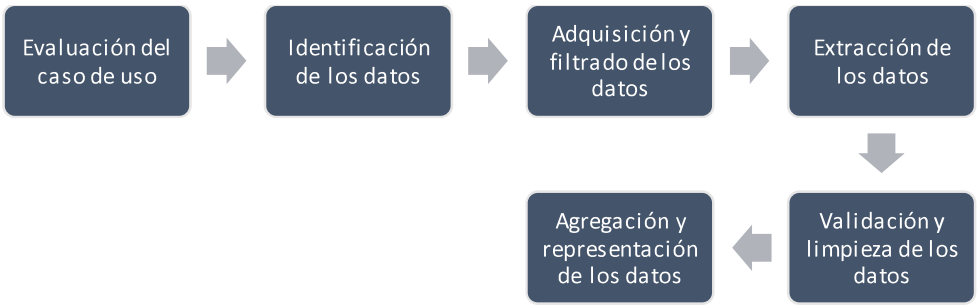

Figura 2.2 Representación del Ciclo de Vida del Análisis de Big Data para el presente proyecto [18]

A continuación, se detallan los pasos a seguir de la metodología del ciclo de vida del análisis de Big Data:

2.1.1 Evaluación del caso de uso

Para este proyecto, se requiere que la estructura de la proteína esté en formato de texto JSON. Se optó por generar archivos con este formato y cuya extensión será ".json", puesto que es un formato que aceptan ambas herramientas, Elasticsearch y Hadoop. Además, se requiere que los datos que van a ser el foco central del proyecto: *gene, accession, organism* y *name,* sean de un mismo tipo de datos y que tanto la clave como el valor no tengan caracteres especiales.

2.1.2 Identificación de los datos

Los datos que se usarán en el caso de uso son pertenecientes a la base de datos UniProt/Swiss-Prot. La fuente de descarga en [19] provee la base de datos en distintos formatos, uno de ellos XML, el cual se tomará como formato de entrada para el tratamiento de los datos. La versión de la base de datos UniProt/Swiss-Prot con la cual se trabajará será la versión disponible a la fecha 16 de marzo del 2019, la cual consta de 559,050 entradas de proteínas [18].

Una entrada de proteína consta campos que se pueden visualizar en la Tabla 2.1 Descripción de los campos de la entrada de una proteína de la base de datos UniProt-Swiss-Prot

Tabla 2.1 Descripción de los campos de la entrada de una proteína de la base de datos UniProt-Swiss-Prot [20]

Campo	**Descripción**	**Índice de ocurrencia**
created	Indica cuándo fue creada una entrada de una proteína en UniProt/Swiss-Prot.	Una vez
comment	Comentario o notas de la base de datos.	Opcional
organismHost	Indica si el organismo de la proteína es susceptible a ser infectado por un virus.	Opcional
xmlns:ns0	Indica que los datos de entrada son de tipo XML y su tipo de codificación.	Una vez
dbReference	Se utilizan como punteros a la información externa asociada con una entrada de una proteína. También indica la fuente en donde el autor o curador de la proteína obtuvo y publicó su información.	Opcional
evidence	Información de la entrada de la proteína. Es un recurso de la base de datos mediante la cual la referencia es obtenida.	Obligatorio
keyword	Proporcionan información que se puede usar para generar índices de las entradas de	Opcional

	secuencias basadas en categorías funcionales y estructurales.	
modified	Indica cuando la secuencia de información fue modificada por última vez.	Una vez
feature	Contiene la información mediante la cual se identificaron y anotaron los datos de la secuencia.	Una o muchas veces
dataset	Indica que sección de la base se está usando, si UniProt/TrEMBL o UniProt/Swiss-Prot.	Una vez
reference	Funcionan como punteros a la información relacionada con las entradas de proteínas UniProt/Swiss-Prot. Estas referencias marcan a UniProt/Swiss-Prot como el foco local de interconexión de bases de datos biomoleculares.	Una o muchas veces
proteinExistence	Indica la evidencia que se tiene actualmente para demostrar la existencia de una proteína.	Una vez
version	La versión de entrada de una proteína se incrementa cada vez que se modifica cualquier dato en la representación del archivo plano de entrada.	Una vez
accession	Identificador de una secuencia cuta función es proveer una vía estable para identificar la secuencia versión por versión.	Una o muchas
name	Es el primer elemento de una entrada de una secuencia, es útil para identificar una secuencia, pero no es un identificador estable como el *accession.*	1 vez
gene	Indica el nombre o conjunto de nombres de los genes que codifican la secuencia de proteínas almacenadas.	Opcional
organism	Identifica el organismo que fue la fuente de la secuencia almacenada.	Una o muchas
protein	Contiene la información de la secuencia almacenada.	1 vez

2.1.3 Adquisición y filtrado de los datos

Durante el paso de formato se eliminarán los campos de la proteína que no son relevantes para el proyecto, señalando de esta manera que los datos relevantes serán *sequence, accession, organism, protein, name* y *gene*. Esto ya que son los datos que contienen la información de la proteína más no de la base de datos como tal. Para ello se ha optado por la realización de un script disponible en el ANEXO I, el mismo que se lo puede visualizar como pseudocódigo en la Figura 2.3.

```
10. Crear array de elementos no relevantes para el caso de estudio
20. Cargar las proteínas
21.     Recorrer array
22.     Eliminar elemento
23.Guardar archivo proteína
```

Figura 2.3 Pseudocódigo filtrado de datos

2.1.4 Extracción de los datos

En este paso los datos se pasarán de formato XML a JSON. Se decidió llevar a cabo la transformación debido a que es un formato que permiten ambos gestores, Elasticsearch y Hadoop. Para ello se ha optado por la realización de un script disponible en el ANEXO I, el mismo el pseudocódigo de la Figura 2.4.

```
10. Leer el archivo XML "uniprot.xml"
20. Buscar la etiqueta "entry"
21. Definir k como una variable que será el número de proteína que se está transformando
22. Transformar todas las etiquetas dentro de "entry" a string
23. Guardar proteína en archivo "proteink.json" (k = número de proteína transformada)
24. Incrementar k
```

Figura 2.4 Pseudocódigo extracción de datos

2.1.5 Validación y limpieza de los datos

El objetivo de este paso es depurar los datos de manera que estos sean válidos para trabajar sobre los gestores. En este paso se eliminarán los caracteres especiales de las claves y valores, que se generaron durante la transformación del formato XML a JSON, ya que estos caracteres especiales pueden confundirse con los caracteres especiales propios de cada *key/value* de la base de datos UniProt/Swiss-Prot.

Además, en este paso se valida que todos los nodos contengan las mismas claves y valores, siempre y cuando estas sean relevantes para el estudio. Se determinó que las claves que deben estar siempre presentes en las entradas de las proteínas o datos serán las siguientes: *sequence, accession, name, protein, gene, organism;* debido a que son los datos más relevantes del estudio.

Además de que estos nodos deben estar presentes, se decidió cuáles campos de estos nodos deben visualizarse. Los nodos de una entrada de proteína disponen de campos que a su vez también son nodos en otros niveles de cardinalidad que describen la información de su nodo principal. Por ejemplo, el nodo *protein,* previamente descrito en la Tabla 2.1, contiene campos representados como categorías que enriquecen su información y estos también tienen subcategorías y se los describirá en la Tabla 2.2 de manera más detallada para mejorar su entendimiento:

Tabla 2.2 Descripción de las categorías y subcategorías del nodo *protein* [20]

Categoría/Nodo	Subcategoría	Índice de ocurrencia	Descripción
recommendedName		1	El nombre de una proteína asignado por el consorcio de UniProt.
	fullName	1	Nombre completo
	shortName	0-n	Abreviación del nombre completo
	ecNumber	0-n	Abreviatura de (*Enzyme Commission numbers*) que son un esquema para clasificación numérica de enzimas [21].
alternativeName		0-n	Un sinónimo de *recommendedName*
	fullName	1	Nombre completo
	shortName	0-n	Abreviación del nombre completo
	ecNumber	0-n	Abreviatura de (*Enzyme Commission numbers*) que son un esquema para clasificación numérica de enzimas [21].
domain		0-n	Dominios funcionales de una proteína
component		0-n	Componentes funcionales de una proteína

Si se conoce que una proteína contiene múltiples dominios funcionales los cuales especifican el dominio de la estructura de la proteína, cada uno de los cuales se describe con un nombre diferente, la descripción de la proteína comienza con el nombre general de la proteína, *recommendedName* y *alternativeName,* descrito en la Tabla 2.2, seguida de la sección de los dominios llamada *domain.* La sección *domain* tiene un índice de ocurrencia 0-n que quiere decir que puede o no estar presente como parte de la información de una proteína, pero en el proyecto se decidió que va a estar presente en todas las proteínas.

Se detalla a continuación, en la Tabla 2.3, las categorías de la sección de dominio de una proteína.

Tabla 2.3 Descripción de las categorías del dominio de una proteína [20]

Categoría/Nodo	Índice de ocurrencia	Descripción
recommendedName	1	El nombre de una proteína asignado por el consorcio de UniProt.
alternativeName	0-n	Un sinónimo de *recommendedName*
fullName	1	Nombre completo
shortName	0-n	Abreviación del nombre completo
ecNumber	0-n	Abreviatura de (*Enzyme Commission numbers*) que son un esquema para clasificación numérica de enzimas [21].

De la misma manera, las categorías *recommendedName* y *alternativeName* de la sección de dominios contienen las mismas subcategorías explicadas en la Tabla 2.2.

Si se conoce que una proteína contiene múltiples componentes funcionales, la descripción de la proteína comienza con el nombre general de la proteína, *recommendedName* y *alternativeName,* descrito en la Tabla 2.2, seguida de la sección de los componentes llamada *component.* La sección *component* tiene un índice de ocurrencia 0-n que quiere decir que puede o no estar presente como parte de la información de una proteína, pero en el proyecto va a estar presente en todas las proteínas.

Se detalla a continuación en la Tabla 2.4 las categorías de la sección de componentes de una proteína.

Tabla 2.4 Descripción de las categorías del dominio de una proteína [20]

Categoría/Nodo	Índice de ocurrencia	Descripción
fullName	1	Nombre completo
shortName	0-n	Abreviación del nombre completo
ecNumber	0-n	Abreviatura de (*Enzyme Commission numbers*) que son un esquema para clasificación numérica de enzimas [21].

Los nodos *gene* y *organism* también disponen de subcategorías, estas se describen en la Tabla 2.5.

Tabla 2.5 Descripción de las categorías de *organism* y *gene*

Categoría/Nodo	Subcategoría	Índice de ocurrencia	Descripción
name		1-n	Contiene el nombre o nombres de los genes y organismos que codifican la secuencia de proteínas almacenada.

Una vez presentados y descritos los nodos con sus categorías y subcategorías, se concluye cuáles serán los que deben estar presentes en la estructura JSON de una proteína.

- *Gene:* deberá tener obligadamente el nodo *name.*
- *Protein:* deberá tener obligadamente los nodos o claves *alternativeName, recommendedName, domain, component.*
    - Los nodos *alternativeName, recommendedName, domain, component* presentes para la proteína, deberán tener obligadamente las claves *fullName, shortName, ecNumber*
- *Organism:* deberá tener obligadamente el nodo *name.*

Para cumplir con el objetivo de este paso, se optó por realizar un script en Python el mismo que se lo puede visualizar en el ANEXO I y se lo presenta como pseudocódigo en Figura 2.5.

```
10. Leer proteína
20. Crear array de nodos eliminar
20. Crear array de nodos a no eliminar
31. Definir k como una variable que será el número de proteína que se está transformando
32. Transformar todas las etiquetas dentro de "entry" a string
33. Guardar proteína en archivo "proteink.json" (k = número de proteína transformada)
34. Incrementar k
```

Figura 2.5 Pseudocódigo validación y limpieza de datos

2.1.6 Agregación y representación de los datos

En este paso, se determinó el tipo de dato que van a tener los campos más relevantes para el proyecto, mencionados en el apartado 2.1.4. Se decidió que todos los datos van a ser del mismo tipo, así concluimos que estos van a ser:

- *Accession*: se va a presentar como una lista de strings

- *Gene:* se presentará como una lista de objetos *name*.
    - *Name:* se presentará como una lista de objetos (clave, valor)
- *Organism:* será una un objeto de objetos *name,*
    - *Name:* será una lista
- *Protein:* se presentará como una lista de objetos *alternativeName, recommendedName, domain, component.*

Con estas especificaciones, se obtienen los datos en un mismo formato. Para cumplir con el objetivo de este paso, se optó por realizar un script en Python el mismo que se lo puede visualizar en el ANEXO I y se lo presenta como pseudocódigo en la Figura 2.6

```
10. Leer proteína
11.     Cambiar formato accession, gene, protein, domain, component, recommendedName,
        alternativeName, fullName, shortName, ecNumber a lista
12.     Cambiar gene -> name a diccionario
13.     Cambiar organismo -> name a diccionario
20. Crear array de nodos a no eliminar
31.         Definir k como una variable que será el número de proteína que se está transformando
32.     Transformar todas las etiquetas dentro de "entry" a string
33.         Guardar proteína en archivo "proteink.json" (k = número de proteína transformada)
34.     Incrementar k
```

Figura 2.6 Pseudocódigo agregación y representación de datos

Una vez realizado todo el proceso de agregación y representación de los datos, se presenta a continuación un ejemplo del modelo final de una entrada correspondiente a una proteína de la base de datos UniProt/Swiss-Prot. Su transformación de formato XML, Figura 2.7, a formato JSON, Figura 2.8, y sin los datos no relevantes para el estudio.

```xml
<?xml version='1.0' encoding='UTF-8'?>
<uniprot
    xmlns="http://uniprot.org/uniprot"
    <entry dataset="Swiss-Prot" created="2008-01-15" modified="2019-01-16" version="74">
        <accession>A1AU17</accession>
        <name>NDK_PELPD</name>
        <protein>
            <recommendedName>
                <fullName evidence="1">Nucleoside diphosphate kinase</fullName>
                <shortName evidence="1">NDK</shortName>
                <shortName evidence="1">NDP kinase</shortName>
                <ecNumber evidence="1">2.7.4.6</ecNumber>
            </recommendedName>
            <alternativeName>
                <fullName evidence="1">Nucleoside-2-P kinase</fullName>
            </alternativeName>
        </protein>
        <gene>
            <name type="primary" evidence="1">ndk</name>
            <name type="ordered locus">Ppro_3244</name>
        </gene>
        <organism>
            <name type="scientific">Pelobacter propionicus (strain DSM 2379 / NBRC 103807 / OttBd1)</name>
            <dbReference type="NCBI Taxonomy" id="338966"/>
            <lineage>
                <taxon>Bacteria</taxon>
                <taxon>Proteobacteria</taxon>
                <taxon>Deltaproteobacteria</taxon>
                <taxon>Desulfuromonadales</taxon>
                <taxon>Desulfuromonadaceae</taxon>
                <taxon>Pelobacter</taxon>
            </lineage>
        </organism>
```

Figura 2.7 Ejemplo XML de entrada

El ejemplo completo de la primera proteína en formato XML se lo puede observar en el ANEXO II.

En el siguiente ejemplo se muestra un extracto de la primera proteína en formato JSON.


```
{"entry": {"sequence": {"length": "137", "text":
  "MERTFAIIKPDAVERRLAGTVIDRIEANGFTIVGMKKIKLSKEQAGGFYCVHRERPFFGE\nLC
  DFMSRSPVIVLCLEKENAIADWRKLMGATNPANAEPGTIRRDFALSLSENSAHGSDAP\nETAAF
  EIAYFFNALELV", "checksum": "80B8C78FC879BD67", "modified": "2007
  -01-23", "version": "1", "mass": "15229"}, "name": "NDK_PELPD",
  "accession": ["A1AU17"], "organism": {"lineage": {"taxon":
  ["Bacteria", "Proteobacteria", "Deltaproteobacteria",
  "Desulfuromonadales", "Desulfuromonadaceae", "Pelobacter"]},
  "name": [{"type": "scientific", "text": "Pelobacter propionicus
  (strain DSM 2379 / NBRC 103807 / OttBd1)"}], "dbReference":
  {"type": "NCBI Taxonomy", "id": "338966"}}, "protein":
  [{"component": [{"fullName": [{}], "ecNumber": [{}], "shortName":
  [{}]}], "domain": [{"fullName": [{}], "ecNumber": [{}],
  "shortName": [{}]}], "alternativeName": [{"fullName":
  [{"evidence": "1", "text": "Nucleoside-2-P kinase"}],
  "recommendedName": [{"fullName": [{}], "ecNumber": [{}],
  "shortName": [{}]}], "alternativeName": [{"fullName": [{}],
  "ecNumber": [{}], "shortName": [{}]}], "shortName": [{}],
  "ecNumber": [{}]}], "recommendedName": [{"alternativeName":
  [{"fullName": [{}], "ecNumber": [{}], "shortName": [{}]}],
  "fullName": [{"evidence": "1", "text": "Nucleoside diphosphate
  kinase"}], "recommendedName": [{"fullName": [{}], "ecNumber":
  [{}], "shortName": [{}]}], "ecNumber": [{"evidence": "1", "text":
  "2.7.4.6"}], "shortName": [{"evidence": "1", "text": "NDK"},
  {"evidence": "1", "text": "NDP kinase"}]}]}], "gene": [{"name":
  [{"evidence": "1", "text": "ndk", "type": "primary"}, {"type":
  "ordered locus", "text": "Ppro_3244"}]}]}}
```


Figura 2.8 Ejemplo de una proteína en formato JSON

La etapa de depuración de los datos es un proceso fundamental para la realización del proyecto de una manera efectiva. De esta manera, se puede presentar el pseudocódigo general. La, que contiene todas las etapas de la metodología que se han detallado anteriormente y el mismo que se encuentra como código en el ANEXO I.

```
10. Crear array de elementos no relevantes para el caso de estudio
20. Cargar las proteínas
21. Recorrer array
22. Eliminar elemento
23. Guardar archivo proteína
30. Leer el archivo XML "uniprot.xml"
40. Buscar la etiqueta "entry"
41. Definir k como una variable que será el número de proteína que se está transformando
42. Transformar todas las etiquetas dentro de "entry" a string
43. Guardar proteína en archivo "proteink.json" (k = número de proteína transformada)
44. Incrementar k
50. Leer proteína
60. Crear array de nodos eliminar
60. Crear array de nodos a no el1minar
71. Definir k como una variable que será el número de proteína que se está transformando
72. Transformar todas las etiquetas dentro de "entry" a string
73. Guardar proteína en archivo "proteink.json" (k = número de proteína transformada)
74. Incrementar k
80. Leer proteína
81. Cambiar formato accession, gene, protein, domain, component, recommendedName,
    alternativeName, fullName, shortName, ecNumber a lista
82. Cambiar gene -> name a diccionario
83. Cambiar organismo -> name a diccionario
90. Crear array de nodos a no eliminar
91. Definir k como una variable que será el número de proteína que se está transformando
92. Transformar todas las etiquetas dentro de "entry" a string
93. Guardar proteína en archivo "proteink.json" (k = número de proteína transformada)  .
94. Incrementar k
```

Figura 2.9 Pseudocódigo de la depuración de la base de datos UniProt/Swiss-Prot

2.2 Metodología de Evaluación de Desempeño

La metodología de Evaluación de Desempeño se especifica en [4], y consta de diez pasos los cuales se los puede visualizar en la Figura 2.10.

Figura 2.10 Representación de la metodología de Evaluación de Desempeño

A continuación, se detallan los pasos de la metodología Evaluación de Desempeño.

2.2.1 Establecer metas y definir el sistema:

Las herramientas que conforman el sistema son:

Hadoop: Hadoop es un *framework* que permite el procesamiento distribuido de grandes conjuntos de datos a través de clústeres de computadoras usando simples modelos de programación. Es escalable y ofrece un gran almacenamiento y control a fallos debido a su alta disponibilidad [16]. Para el procesamiento de los grandes conjuntos de datos, Hadoop utiliza el modelo de programación MapReduce [16]. Un *job*, o trabajo de MapReduce, es una ejecución del modelo MapReduce en Hadoop y básicamente divide el conjunto de datos de entrada en fragmentos independientes que son procesados por las tareas del *Map* de una manera completamente paralela; y a continuación, se ordenan los resultados de los *Maps*, y se ingresan en las tareas de *Reduce* [16].

Comúnmente, tanto la entrada como la salida del *Job* se almacenan en un sistema de archivos denominado HDFS. El entorno MapReduce y el sistema de archivos se ejecutan en los mismos nodos del clúster. El *framework* se encarga de programar las tareas, monitorearlas y volver a ejecutar las tareas fallidas [16].

A continuación, se detallan los procesos que corren sobre un clúster Hadoop [22].

- ***NameNode*:** es un servidor maestro que administra el espacio de nombres del sistema de archivos y regula el acceso de los clientes a esos archivos.
- ***DataNode*:** generalmente se configura uno por nodo y administra el almacenamiento adjunto a los nodos en los que se ejecutan.

Los *DataNode* almancenan los datos por medio de bloques y realizan las escrituras a los archivos del HDFS mediante instrucciones del *NameNode* [22].

- ***SecondaryNameNode*:** es un nodo dedicado al HDFS cuya función es tomar puntos de restauración del sistema de archivos que reside en el *NameNode* [23].

Como una capa del ecosistema Hadoop se presenta YARN[2] el cual es esencialmente un sistema para administrar aplicaciones distribuidas. Consiste en un *ResourceManager* que es el nodo que administra los recursos disponibles del clúster y por lo tanto administra las aplicaciones que se levantan sobre YARN [24] y de *NodeManager* que toma las instrucciones del *ResourceManager* y administra los recursos del nodo en el que reside [24].

Hive: Hadoop fue la base para otras soluciones de alto nivel como Apache Hive, una infraestructura de almacenamiento de datos distribuidos para proporcionar resumen de datos, consultas y análisis. Apache Hive admite el análisis de grandes conjuntos de datos [1], asimismo Hive permite que Hadoop opere como un *Data Warehouse*, superponiéndose sobre la capa de almacenamiento de datos HDFS y permitiendo consultas sobre los datos almacenados del mismo [25]. Proporciona un lenguaje similar a SQL llamado HiveQL manteniendo el soporte completo para Map/Reduce. Es decir, el lenguaje HiveQL es un lenguaje que transforma las consultas en *Jobs* Map/Reduce. Asimismo, Hive acelera consultas, proporciona índices, incluyendo índices de mapa de bits, y vale la pena explotarlo en varias aplicaciones bioinformáticas [1].

ElasticSearch: Elasticsearch es un motor de búsqueda orientado a JSON capaz de ser accedido mediante una API [26].

Debido a sus características, Elasticsearch puede realizar operaciones sobre los datos con una arquitectura de un nodo como de miles de nodos obteniendo resultados similares [27]. Las características que permiten a Elasticsearch reaccionar de esta manera son [27]:

- Distribuye los documentos a través de distintos shards (containers), en el caso de un clúster multi-nodo los *shards* residen en los distintos nodos.
- Con la replicación duplica cada *shard* y provee un clúster con redundancia de datos y tolerante a fallos.

[2] Apache Hadoop YARN por las siglas en ingles de Yet Another Resource Negotiator (Otro Negociador de Recursos) [24]

- Enruta las solicitudes hacia los nodos o *shards* que contienen los datos específicos que se necesitan.

Implementación del entorno de pruebas

El objetivo del caso de estudio es comparar el rendimiento de los dos gestores de bases de datos Hive/Hadoop o Elasticsearch. Los elementos claves para su cumplimiento son:

a) Un conjunto estructurado de operaciones CRUD son el *benchmark* del presente proyecto, cuyo objetivo es obtener resultados necesarios, el tiempo y cantidad de registros manipulados por operación, y concluir cuál herramienta es significativamente más rápida en cuestión de tiempo.

b) Base de datos NoSQL UniProt/Swiss-Prot en formato JSON.

c) Gestores de bases de datos Hadoop y Elasticsearch desplegados en clústeres pseudo distribuidos.

El clúster para cada herramienta consiste en tres máquinas virtuales con las siguientes características:

- 2 GB de memoria ram cada una.
- Sistema Operativo Centos
- 40 GB de memoria en disco.

Las máquinas virtuales de cada clúster se encuentran en distintos discos de un servidor conectadas a través de una red privada. En el clúster Hadoop las máquinas virtuales se encuentran distribuidas de la siguiente manera: Un nodo máster en el cual se levantan los procesos *NameNode*, *SecondaryNameNode* y *ResourceManager* de la aplicación Hive y dos nodos esclavos en los cuales se levantan en cada uno de ellos los procesos *DataNode* y *NodeManager*; tal y como se puede observar en Figura 2.11.

Además, en la Figura 2.11 se pueden observar los detalles de las redes, virtualizador y Sistema Operativo utilizado en cada máquina virtual del clúster Hadoop.

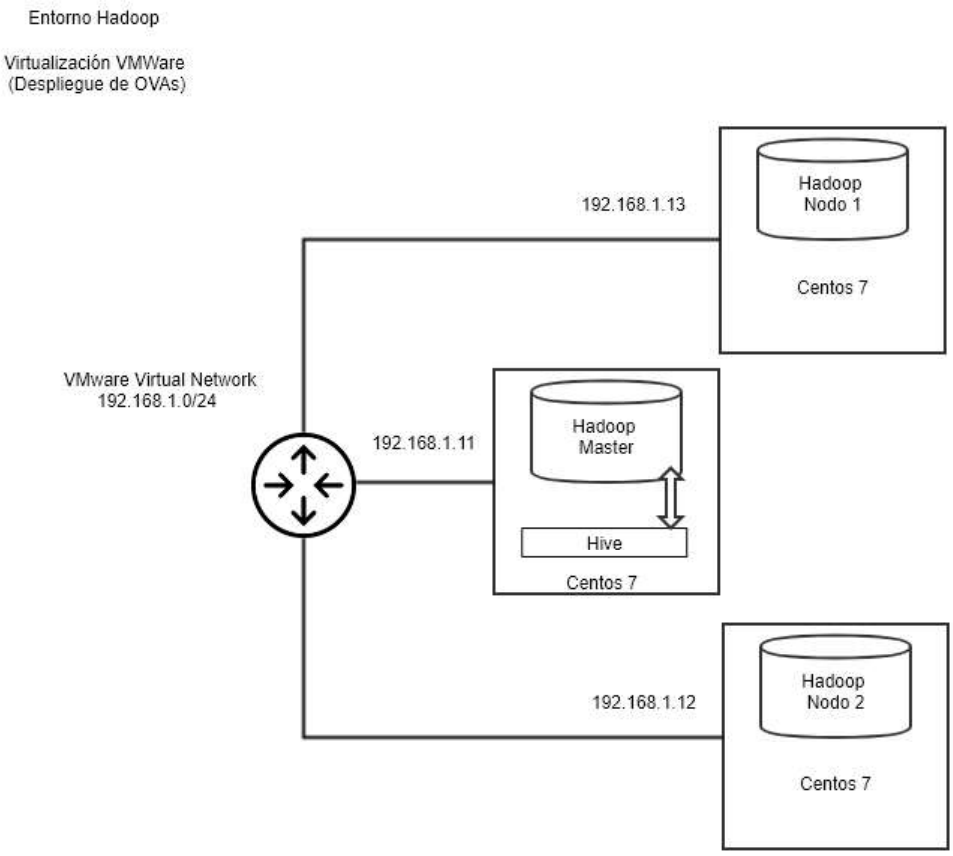

Figura 2.11 Clúster Hadoop configurado para el proyecto

En el clúster Elasticsearch las máquinas virtuales se encuentran distribuidas de la siguiente manera: Un nodo máster, dos nodos esclavos de los cuales uno de ellos desempeña la función de nodo máster en el caso de que este dejara de funcionar; tal y como se puede observar en Figura 2.12.

Además, en la Figura 2.12 se pueden observar los detalles de las redes, virtualizador y Sistema Operativo utilizado en cada máquina virtual del clúster Elasticsearch.

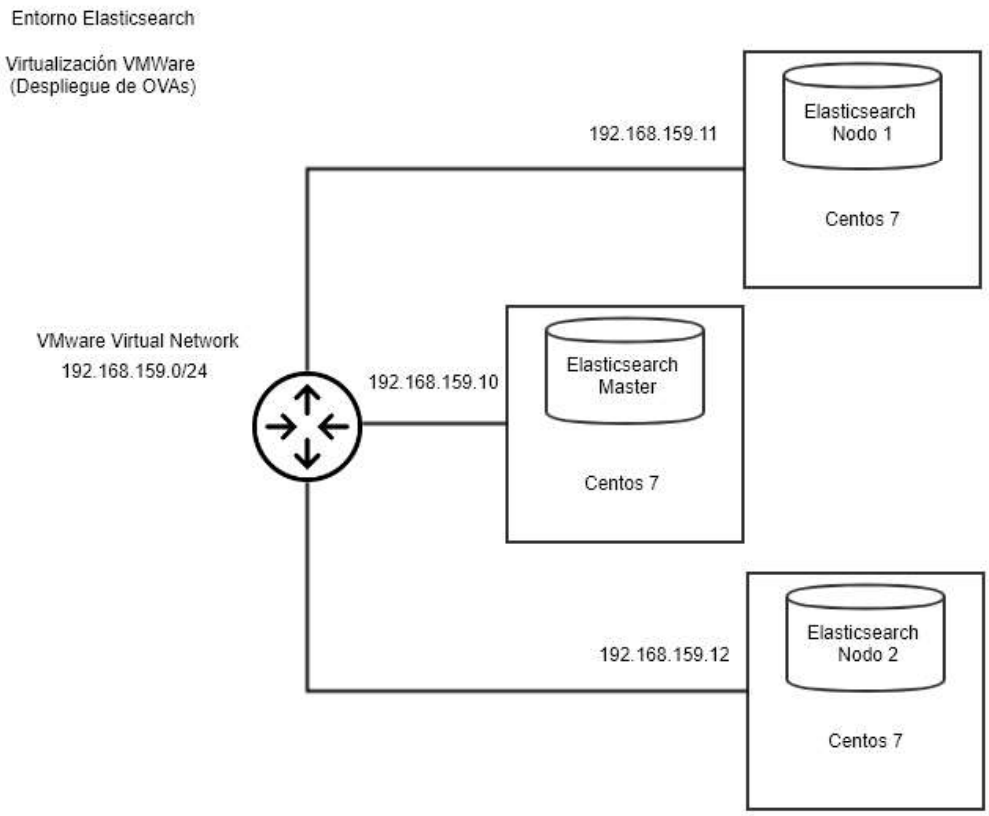

Figura 2.12 Clúster Elasticsearch configurado para el proyecto

La distribución de nodos en el clúster se la puede observar en la Tabla 2.6

Tabla 2.6 Distribución y carga de los nodos de los clústeres Hadoop y Elasticsearch

Gestor	**Nodo 1**	**Nodo 2**	**Nodo 3**
Hadoop	Nodo Máster: Levanta los procesos a) NameNode b) SecondaryNameNode c) ResourceManager	Nodo Esclavo 1: Levanta los procesos DataNode NodeManager	Nodo Esclavo 2: Levanta los procesos DataNode NodeManager
Elasticsearch	Nodo Máster: Recibe la petición	Nodo Data 1: Shard 1 Nodo Máster Backup	Nodo Data 2: Shard 2

Además, se incluye un cliente cuyo objetivo será enviar peticiones a los gestores de bases de datos para el procesamiento de operaciones CRUD y su respectiva evaluación de tiempo de procesamiento de estas y contendrá los logs con las respuestas a las peticiones.

El cliente estará conectado al servidor que contiene las máquinas virtuales, que conforman los clústeres pseudo distribuidos explicados previamente, mediante una red cableada.

Finalmente, se puede observar la arquitectura final del proyecto en la Figura 1.1 y se detallará a continuación de manera resumida su funcionamiento e interacción.

Detallado de izquierda a derecha, la interacción cliente-servidor en el clúster Elasticsearch:

1) El cliente ejecuta un script de creado, actualizado y borrado de datos, presente en el ANEXO III, y un script de consulta, presente en el ANEXO IV, realizados en Python, estos scripts enviarán peticiones al servicio REST proporcionado por Elasticsearch al servidor host que redirecciona las peticiones al nodo Máster y este distribuirá el trabajo entre los nodos del clúster.

2) El nodo Máster devolverá un resultado que lo capturarán los scripts de creado, actualizado y borrado de datos, presente en el ANEXO III, y de consulta, presente en el ANEXO IV y escribirá sobre archivos logs que capturarán el desempeño.

Detallado de izquierda a derecha, la interacción cliente-servidor en el clúster Hadoop:

1) El cliente ejecuta un script de creado, actualizado y borrado de datos, presente en el ANEXO V, y un script de consulta, presente en el ANEXO VI, realizados en Python, estos scripts enviarán peticiones al servicio REST elaborado mediante Python, presente en el ANEXO VII, que redirecciona las peticiones al nodo Máster y este distribuirá el trabajo entre los nodos del clúster.

2) El nodo Máster devolverá un resultado que lo capturarán los scripts de creado, actualizado y borrado de datos, presente en el ANEXO V, y de consulta, presente en el ANEXO VI y escribirá sobre archivos logs que capturarán el desempeño.

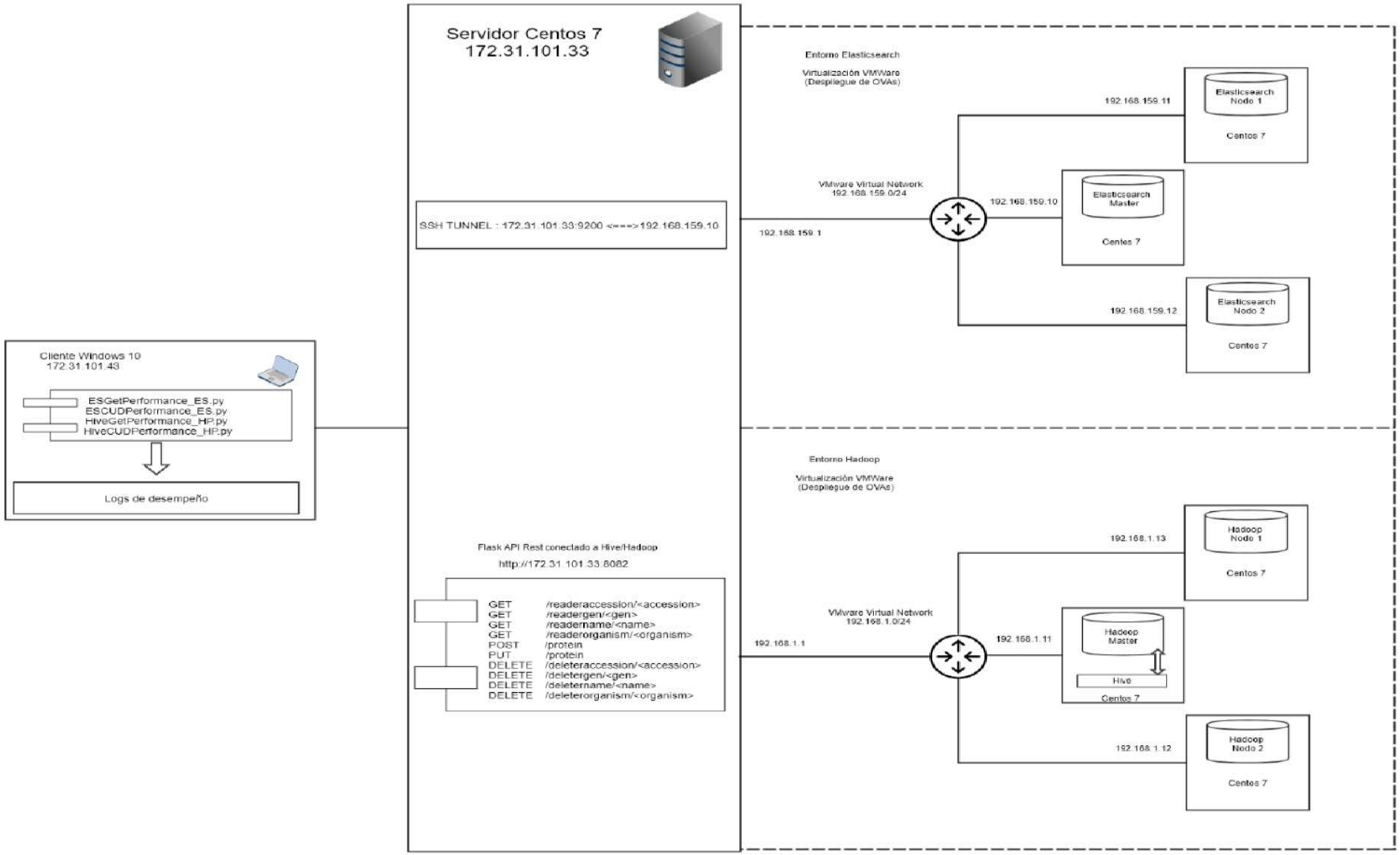

Figura 2.13 Arquitectura final del proyecto

2.2.2 Listar los servicios y salidas:

Los servicios ofrecidos por el sistema son

- Dos APIs REST, una para cada gestor, que permite enviar peticiones de forma remota a los gestores de bases de datos sin importar el medio por el cual se hagan las consultas. En el proyecto se lo realizó mediante un script en Python.
- Los gestores de bases de datos procesarán las operaciones CRUD

Las salidas corresponderán al tiempo empleado en el procesamiento y respuestas a las operaciones CRUD y dependerá de la cantidad de registros o documentos de respuesta, además de la latencia de la red, velocidad de procesamiento de cada gestor y funcionamiento del clúster y serán almacenadas en logs de archivos.

2.2.3 Seleccionar métricas:

Para cada servicio se tomará en cuenta la velocidad de respuesta; se analizará el tiempo que emplea en retornar una salida información en cada operación CRUD enviada, esto se lo llevará a cabo mediante las siguientes métricas:

a) Tiempo medio transcurrido en generar una respuesta al cliente por operación CRUD enviada, esté será determinado desde que se envía la petición al servidor y este presenta la respuesta al cliente.

b) Número de registros o documentos afectados en cada una de las respuestas al cliente.

2.2.4 Listar los parámetros:

Los parámetros de sistema que afectan al rendimiento son los siguientes

a) Velocidad de procesamiento de los gestores de bases de datos.

b) Velocidad de la red.

c) Confiabilidad de la red.

d) Gestor de base de datos.

Los parámetros de carga de trabajo son los siguientes:

a) Tamaño de registro o documento de respuesta.

b) Cantidad de registros o documentos de respuesta.

c) Número de peticiones sucesivas

2.2.5 Seleccionar factores de estudio:

Los factores que se escogieron para este estudio son:

a) Gestor de base de datos.

2.2.6 Seleccionar la técnica de evaluación:

La metodología de evaluación de desempeño seguida propone el uso de tres técnicas de evaluación: modelo analítico, simulación y medición, una metodología basada en prototipos. Un cuadro comparativo de las tres técnicas se lo puede observar en la Tabla 2.7.

Tabla 2.7 Cuadro comparativo de técnicas de evaluación de Evaluación de Desempeño

Criterio	Modelo analítico	Simulación	Medición
1.Escenario	Distintos	Distintos	Post-Prototipos
2.Tiempo Requerido	Pequeño	Mediano	Variado
3.Herramientas	Analista	Lenguajes de Programación	Instrumentación
4.Precisión	Bajo	Moderado	Variado
	Fácil	Moderado	Difícil
6.Costo	Pequeño	Mediano	Alto
7.Convincente	Bajo	Mediano	Alto

[4]

Basado en el ciclo de vida en el que se encuentra el sistema que se ha montado para el presente proyecto, la técnica de evaluación seleccionada es Medición basada en prototipos cuyas características son las siguientes:

a) **Escenario:** el escenario fue definido en base a la arquitectura de cada una de las herramientas del caso de uso. Este escenario presenta una interacción cliente-servidor, en el cual el servidor corresponde al clúster de cada herramienta, Elasticsearch y Hadoop, y como cliente es una PC externa conectada al servidor.

b) **Tiempo requerido:** El proyecto dispone del tiempo necesario para hacer las pruebas sobre las cuales se van a definir los resultados, por lo cual el tiempo no es un limitante.

c) **Herramientas:** En base a la disponibilidad de herramientas, como parte del sistema se requería de un servidor en el cual se puedan alojar los nodos en discos diferentes y puedan trabajar paralelamente y no en cola. Este servidor fue suministrado por la

Escuela Politécnica Nacional. Y por la pequeña complejidad de la arquitectura de los clústeres levantados, fue suficiente para realizar las pruebas mencionadas.

d) **Exactitud:** Se tiene claro que la técnica de evaluación seleccionada dispone de varios tipos de resultados tales como precisos o moderados, debido a varios agentes externos que pueden influir en estos. En base a esto y a que se necesitan resultados lo más precisos para definir que herramienta se ajusta más al objetivo, se han controlado todos los factores y parámetros explicados en los puntos 2.2.4 y 2.2.5, que pueden influir en los resultados, mediante el diseño del experimento.

e) **Costo:** La técnica de medición demanda equipo, instrumentación y tiempos reales, resultando ser la técnica más costosa.

f) **Confiabilidad:** Al ser una técnica cuyo conocimiento y resultados se basan en un entorno real, tiende a ser la más confiable para las personas. Además, de presentar resultados no complejos, entendibles y fácil de visualizar.

2.2.7 Seleccionar la carga de trabajo:

La carga de trabajo, como se lo ha venido mencionando en el documento, serán las operaciones CRUD a los gestores de bases de datos. Estos serán generados y presentados como una serie de scripts en Python, en forma de peticiones a APIs configuradas para cada gestor:

Hadoop-Hive: El lenguaje que maneja Hive es Hive-SQL que es un lenguaje basado en SQL que traduce las sentencias a Jobs Map-Reduce y es procesado por el clúster de Hadoop.

Elasticsearch: El lenguaje que maneja Elasticsearch es Query DSL (Lenguaje de dominio específico), en el proyecto no topamos el lenguaje de una manera directa, sino mediante el manejo de URLs.

Los lenguajes de cada gestor se resumen en APIs REST que manejamos desde la capa de cliente; que a su vez serán gestionados por scripts Python que genera cada una de las operaciones CRUD, hace peticiones a cada uno de los gestores y registra sus salidas en un archivo .log.

2.2.8 Diseñar Experimentos

Basándose en “30-sample rule-of-thumb” de William Sealy Gosset, [28] 30 peticiones de la misma operación CRUD serán los experimentos.

Los scripts Python se realizaron en dos partes para cada gestor. Un script que contiene las peticiones de consulta, Read en CRUD, y un script diferente que realizará las peticiones de inserción de un nuevo registro, actualización de registros y borrado de registros, Create-Update-Delete de CRUD.

Pseudocódigo de operación *create*

Su código en Python se encuentra disponible en el ANEXO III.

```
Crear Proteína:
10. Hacer 30 veces:
11.     Crear proteína con 'name', 'accession', 'organism' y 'gene' aleatorios desde una plantilla
12.             Guardar proteína creada en archivo.
13.             Leer proteína almacenada en archivo
14.             Tomar tiempo Inicial
15.     Crear proteína a través del servicio Web (API de Elasticsearch/Hadoop) (POST)
16.             Tomar Tiempo Final
17.             Guardar registro de tiempo en log correspondiente (create_protein.log)
18.             Imprimir registro de tiempo
```

Figura 2.14 Pseudocódigo operación *create*

Pseudocódigo de operación *read*

Su código en Python se encuentra disponible en el ANEXO IV.

```
Buscar Proteínas
10. Define Arreglo_parametros_Busqueda = ['organism', 'gene', 'accession', 'name']
20. Por cada parámetro en Arreglo_parametros_Busqueda hacer:
30.     Leer listado de datos de un archivo correspondientes al parámetro a buscar
31.             Almacenar listado de datos en Arreglo_datos
32.             Por cada dato en Arreglo_datos hacer:
33.                     Hacer 30 veces:
34.                             Tomar tiempo Inicial
35.                             Realizar consulta hacia el servicio Web (API de Elasticsearch/Hadoop)
                                (GET)
36.                             Tomar Tiempo Final
37.                             Guardar registro de tiempo en log correspondiente (organism.log, gene.log, accession.log, name.log)
38.                             Imprimir registro de tiempo
```

Figura 2.15 Pseudocódigo operación *read*

Pseudocódigo de operación update

Su código en Python se encuentra disponible en el ANEXO III.

```
Actualizar Proteína
10.     Hacer 30 veces:
11.     Leer un 'name' aleatorio desde archivo y almacenar en random_name
12.             Crear un 'name' aleatorio y almacenar en name_updated
13.             Tomar tiempo Inicial
14.     Actualizar proteína con random_name y remplazarlo con name_updated a través del servicio Web (API de Elasticsearch/Hadoop) (POST)
15.             Tomar Tiempo Final
16.             Guardar registro de tiempo en log correspondiente (update_protein_name.log)
17.             Imprimir registro de tiempo
20.     Hacer 30 veces:
21.     Leer un 'accession' aleatorio desde archivo y almacenar en random_accession
22.             Crear un 'name' aleatorio y almacenar en accession_updated
23.             Tomar tiempo Inicial
24.     Actualizar proteína con random_accession y remplazarlo con accession_updated a través del servicio Web (API de Elasticsearch/Hadoop) (POST)
25.             Tomar Tiempo Final
26.             Guardar registro de tiempo en log correspondiente (update_protein_accesion.log)
27.             Imprimir registro de tiempo
30.     Hacer 30 veces:
31.     Leer un 'organism' aleatorio desde archivo y almacenar en random_organism
32.             Crear un 'organism' aleatorio y almacenar en organism_updated
33.             Tomar tiempo Inicial
34.     Actualizar proteína con random_organism y remplazarlo con organism_updated a través del servicio Web (API de Elasticsearch/Hadoop) (POST)
35.             Tomar Tiempo Final
36.             Guardar registro de tiempo en log correspondiente (update_protein_organism.log)
37.             Imprimir registro de tiempo
40.     Hacer 30 veces:
41.     Leer un 'gene' aleatorio desde archivo y almacenar en random_gene
42.             Crear un 'organism' aleatorio y almacenar en gene_updated
43.             Tomar tiempo Inicial
44.     Actualizar proteína con random_gene y remplazarlo con gene_updated a través del servicio Web (API de Elasticsearch/Hadoop) (POST)
45.             Tomar Tiempo Final
46.             Guardar registro de tiempo en log correspondiente (update_protein_gene.log)
47.             Imprimir registro de tiempo
```

Figura 2.16 Pseudocódigo operación *update*

Pseudocódigo de operación delete

Su código en Python se encuentra disponible en el ANEXO III.

```
Eliminar Proteína
10.     Hacer 30 veces:
11.     Leer un 'name' aleatorio desde archivo y almacenar en random_name
12.             Tomar tiempo Inicial
13.     Eliminar proteína con random_name a través del servicio Web (API de Elasticsearch/Hadoop) (POST)
14.             Tomar Tiempo Final
15.             Guardar registro de tiempo en log correspondiente (delete_protein_name.log)
15.             Imprimir registro de tiempo
20.     Hacer 30 veces:
21.     Leer un 'accession' aleatorio desde archivo y almacenar en random_accession
22.             Tomar tiempo Inicial
23.     Eliminar proteína con random_accession a través del servicio Web (API de Elasticsearch/Hadoop) (POST)
24.             Tomar Tiempo Final
25.             Guardar registro de tiempo en log correspondiente (delete_protein_accesion.log)
26.             Imprimir registro de tiempo
30.     Hacer 30 veces:
31.     Leer un 'organism' aleatorio desde archivo y almacenar en random_organism
32.             Tomar tiempo Inicial
33.     Eliminar proteína con random_organism a través del servicio Web (API de Elasticsearch/Hadoop) (POST)
34.             Tomar Tiempo Final
35.             Guardar registro de tiempo en log correspondiente (delete_protein_organism.log)
36.             Imprimir registro de tiempo
40.     Hacer 30 veces:
41.     Leer un 'gene' aleatorio desde archivo y almacenar en random_gene
43.             Tomar tiempo Inicial
44.     Eliminar proteína con random_gene a través del servicio Web (API de Elasticsearch/Hadoop) (POST)
45.             Tomar Tiempo Final
46.             Guardar registro de tiempo en log correspondiente (delete_protein_gene.log)
47.             Imprimir registro de tiempo
```

Figura 2.17 Pseudocódigo operación *delete*

Adicionalmente, tal y como se explicó en el primer paso de la metodología cada clúster o factor del proyecto, Elasticsearch y Hadoop, ejecutarán las consultas y los procesos que conlleven a ello en tres nodos, un máster y dos esclavos.

Los pasos “Presentar resultados” y “Analizar e interpretar datos” que dicta la metodología “Evaluación de Desempeño” [4] se encuentran detallados en los apartados 3.1 Resultados y 3.2 Discusión respectivamente.

3. RESULTADOS Y DISCUSIÓN

3.1. Resultados

Después de realizar las operaciones CRUD sobre los gestores de bases de datos Elasticsearch y Hadoop. Se han obtenido los resultados descritos a continuación, y su interpretación se encuentra en el apartado 3.2 Discusión.

Los resultados se los van a representar mediante histogramas y diagramas de dispersión. Un ejemplo de cómo se obtuvieron las figuras presentadas en el apartado de resultados se lo puede observar en el ANEXO XXIV.

CREATE

Resultados operación C*reate* Gestor de Base de Datos Elasticsearch

En la Figura 3.1 se visualiza un histograma de los valores obtenidos para el tiempo de ejecución de la operación de inserción sobre la base de datos UniProt/Swiss-Prot mediante el gestor Elasticsearch. El rango de tiempo más común se representa en la gráfica en dos dimensiones, eje *x* y eje *y,* siendo el eje *x* los tiempos de respuesta comprendidos entre 0 y 0.030 segundos y en el eje *y* la frecuencia como cantidad de resultados, presentes en ANEXO IX, contenidos en los tiempos de respuesta, eje *x*.

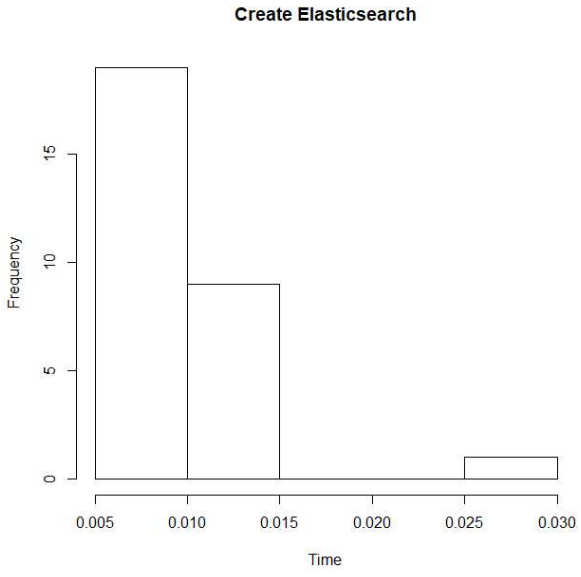

Figura 3.1 Histograma inserción de datos a través de Elasticsearch con tiempo de respuesta de 0 a 0.030 segundos.

Resultados *Create* Gestor de Base de Datos Hadoop

En la Figura 3.2 se visualiza el histograma de la frecuencia de valores de tiempo de ejecución para la operación de inserción sobre la base de datos UniProt/Swiss-Prot mediante el gestor Hadoop.

El rango de tiempo más común se lo representa en la gráfica en dos dimensiones, eje *x* y eje *y,* siendo el eje *x* los tiempos de respuesta y en el eje *y* la frecuencia como cantidad de resultados, presentes en ANEXO XVI, contenidos en los tiempos de respuesta, eje *x*.

Se pueden observar otros resultados, en la Figura 3.3 y la Figura 3.4, pero con la variante del tiempo. En la Figura 3.3 los tiempos de respuesta están comprendidos entre 0 y 50 segundos y en la Figura 3.4 los tiempos de respuesta están comprendidos entre 0 y 100 segundos

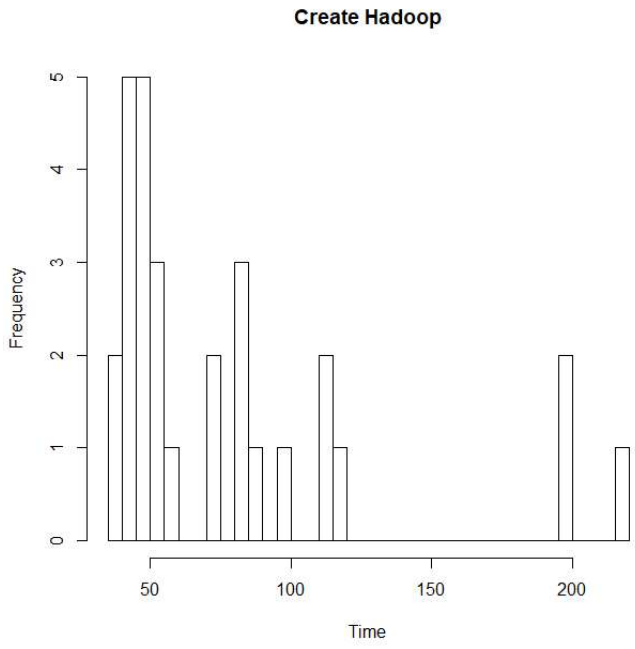

Figura 3.2 Histograma inserción de datos a través de Hadoop

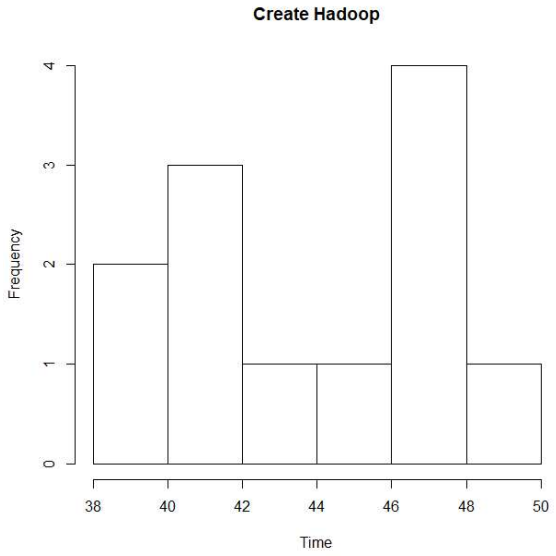

Figura 3.3 Histograma inserción de datos a través de Hadoop con tiempo de respuesta de 0 a 50 segundos.

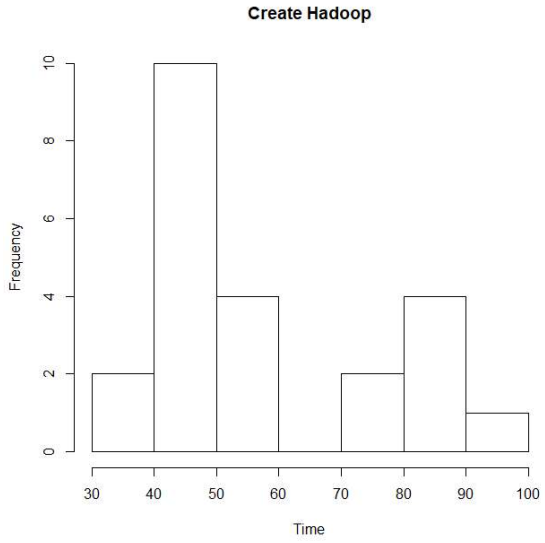

Figura 3.4 Histograma inserción de datos a través de Hadoop con tiempo de respuesta de 0 a 100 segundos.

READ

Se realizó la operación búsqueda, READ en CRUD sobre la base de datos UniProt\Swiss-Prot mediante cuatro parámetros: *name, accession, organism* y *gene.*

Como resultados de la operación se presentan dos tipos de gráficas:

- Gráfica de dispersión: La gráfica de dispersión se representa en dos dimensiones, el Tiempo De Operación, eje *y,* en función la Cantidad De Registros que devuelve dicha operación, eje *x*.
- Histograma: El histograma representa la tendencia de resultados de tiempo que demora la operación en realizar búsquedas de los parámetros adjuntos en los anexos IX, X, XI y XII, sobre la base de datos UniProt/Swiss-Prot a través de los gestores Elasticsearch y Hadoop. El rango de tiempo más común se lo representa en la gráfica en dos dimensiones, eje *x* y eje *y,* siendo el eje *x* los tiempos de respuesta y en el eje *y* la frecuencia como cantidad de resultados contenidos en los tiempos de respuesta, eje *x*.

Resultados *Read* Gestor de Base de Datos Elasticsearch

A continuación, se presentan resultados de las búsquedas hechas mediante el gestor Elasticsearch.

- ***Read Name***

En la Figura 3.5 se obtiene un histograma con tiempos de respuesta comprendidos entre 0 y 0.030 segundos en la operación de búsqueda de un *name*, los resultados se los puede observar en el ANEXO X.

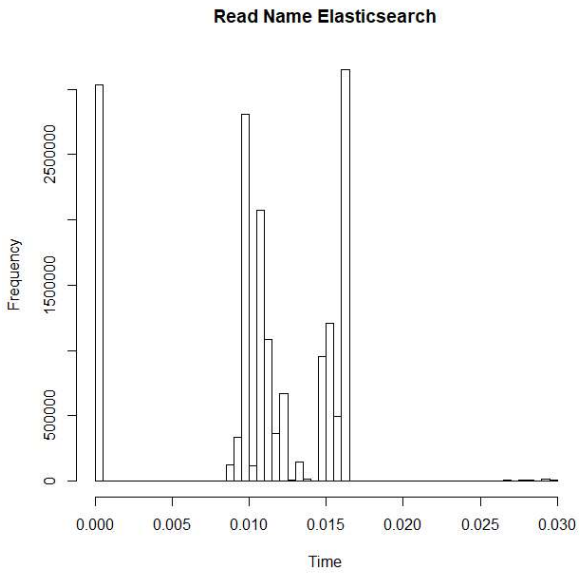

Figura 3.5 Histograma búsqueda de *name* a través de Elasticsearch con tiempo de respuesta de 0 a 0.030 segundos

- ***Read Accession***

En la Figura 3.6 se obtiene un histograma con tiempos de respuesta comprendidos entre 0 y 0.030 segundos en la operación de búsqueda de un *accession,* los resultados se los puede observar en el ANEXO XI.

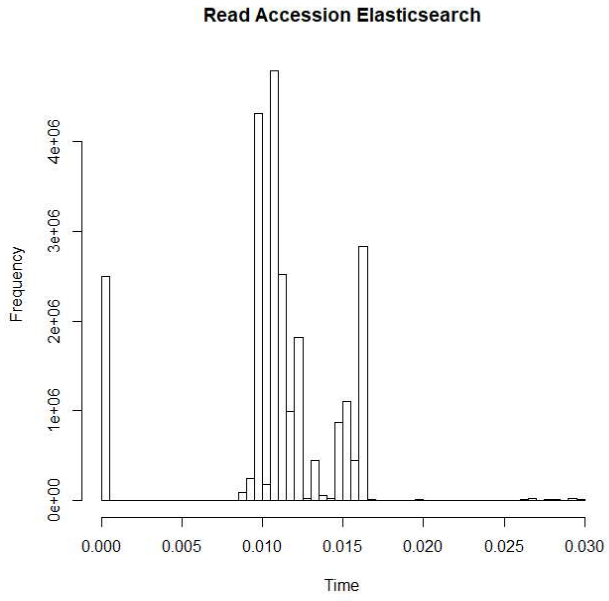

Figura 3.6 Histograma búsqueda de *accession* a través de Elasticsearch con tiempo de respuesta de 0 a 0.030 segundos.

- ***Read Organism***

En la Figura 3.7 se obtiene un diagrama de dispersión con todos los resultados de la operación de búsqueda de un *organism*, los resultados se los puede observar en el ANEXO XII.

En la Figura 3.11 se obtiene un histograma con tiempos de respuesta comprendidos entre 0 y 0.030 segundos en la operación de búsqueda de un *organism*.

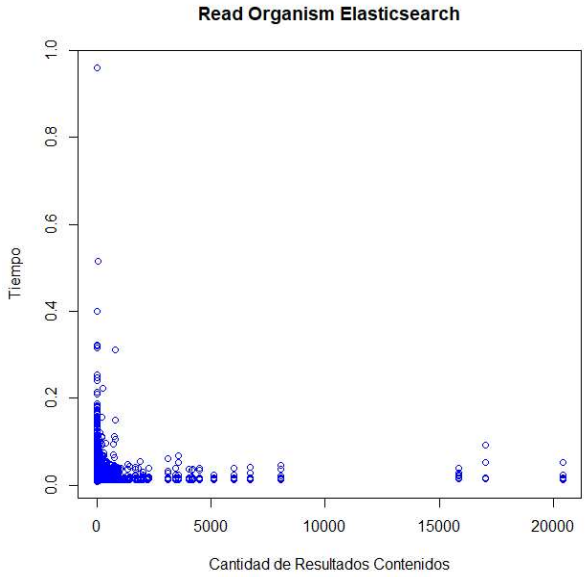

Figura 3.7 Diagrama de dispersión Tiempo De Operación vs Cantidad De Registros con parámetro *organism*

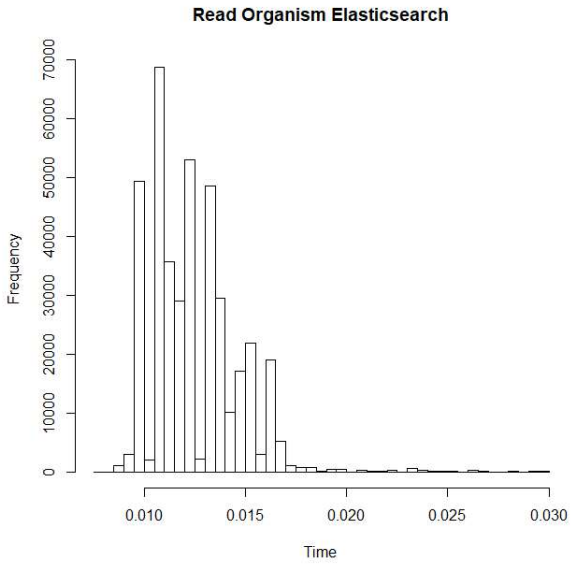

Figura 3.8 Histograma búsqueda de *organism* a través de Elasticsearch con tiempo de respuesta de 0 a 0.030 segundos.

- ***Read Gene***

En la Figura 3.9 se obtiene un diagrama de dispersión con todos los resultados de la operación de búsqueda de un *gene*, los resultados se los puede observar en el ANEXO XII.

En la Figura 3.10 se obtiene un histograma con tiempos de respuesta comprendidos entre 0 y 0.030 segundos en la operación de búsqueda de un *gene*.

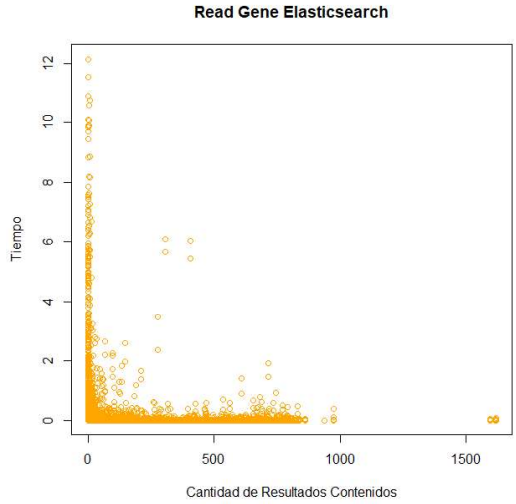

Figura 3.9 Diagrama de dispersión Tiempo De Operación vs Cantidad De Registros con parámetro *gene*

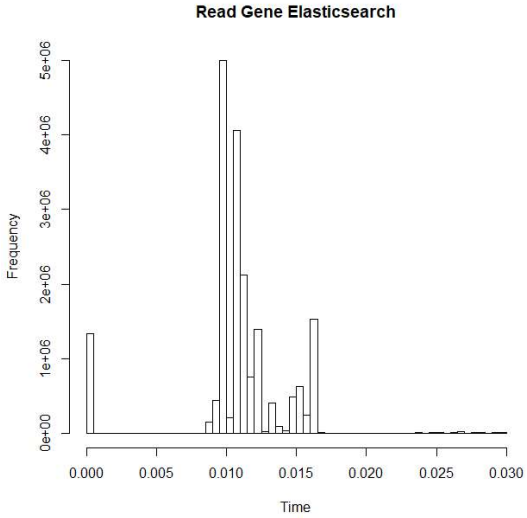

Figura 3.10 Histograma búsqueda de *gene* a través de Elasticsearch con tiempo de respuesta de 0 a 0.030 segundos.

Resultados operación *Read* Gestor de Base de Datos Hadoop

A continuación, se presentan resultados de las búsquedas hechas mediante el gestor Hadoop.

- ***Read Name***

En la Figura 3.11 se obtiene un histograma con tiempos de respuesta comprendidos entre 0 y 10 segundos en la operación de búsqueda de un *name,* los resultados se los puede observar en el ANEXO XVII.

En la

Figura 3.12 se obtiene un histograma con tiempos de respuesta comprendidos entre 0 y 2 segundos en la operación de búsqueda de un *name*.

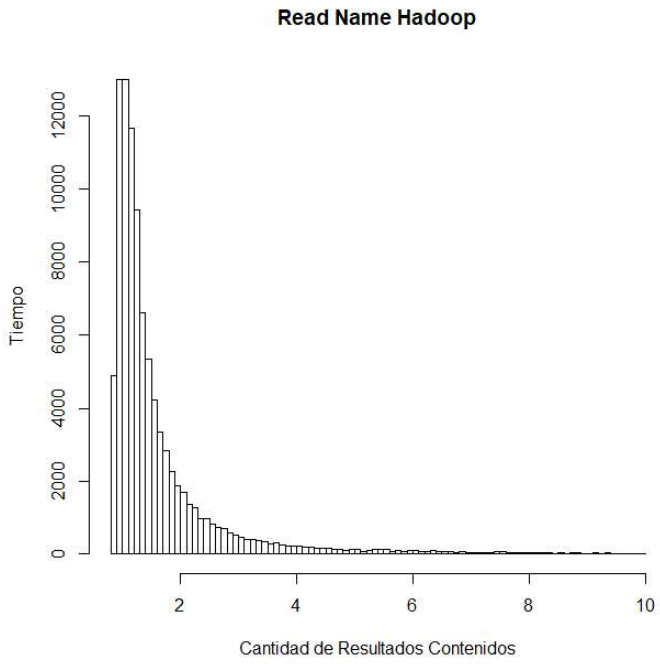

Figura 3.11 Histograma búsqueda de datos con parámetro *name* a través de Hadoop con tiempo de respuesta de 0 a 10 segundos

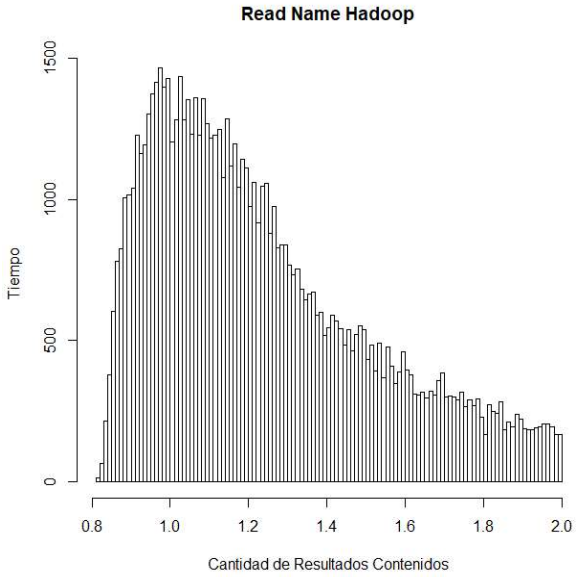

Figura 3.12 Histograma búsqueda de datos con parámetro *name* a través de Hadoop con tiempo de respuesta de 0 a 2 segundos

- ***Read Accession***

En la Figura 3.13 se obtiene un histograma con tiempos de respuesta comprendidos entre 0 y 10 segundos en la operación de búsqueda de un *accession,* los resultados se los puede observar en el ANEXO XVIII.

En la Figura 3.14 se obtiene un histograma parecido a la Figura 3.13 pero con la variante del tiempo de respuesta siendo este comprendido entre 0 y 2 segundos.

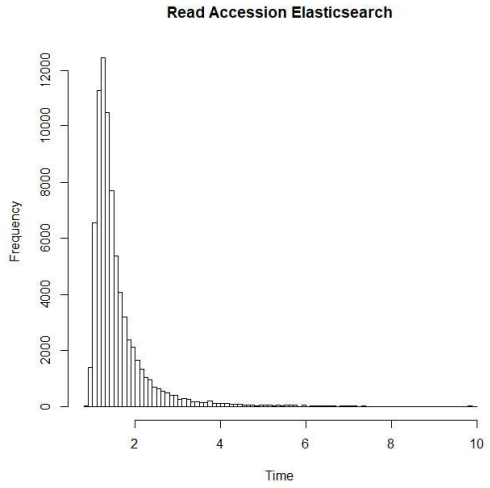

Figura 3.13 Histograma búsqueda de datos con parámetro *accession* a través de Hadoop con tiempo de respuesta de 0 a 10 segundos

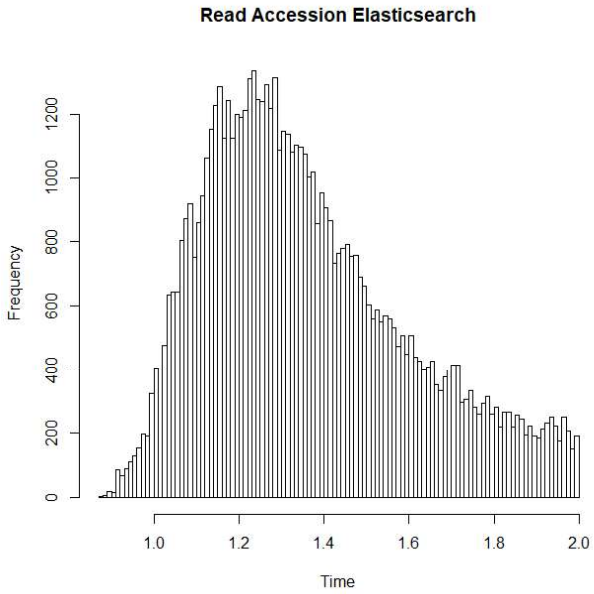

Figura 3.14 Histograma búsqueda de datos con parámetro *accession* a través de Hadoop con tiempo de respuesta de 0 a 2 segundos.

- ***Read Organism***

En la Figura 3.15 se obtiene un diagrama de dispersión con todos los resultados de la operación de búsqueda de *organism,* los resultados se los puede observar en el ANEXO XIX. En la Figura 3.16 se obtiene un histograma con tiempos de respuesta comprendidos entre 0 y 2 segundos en la operación de búsqueda de *organism*.

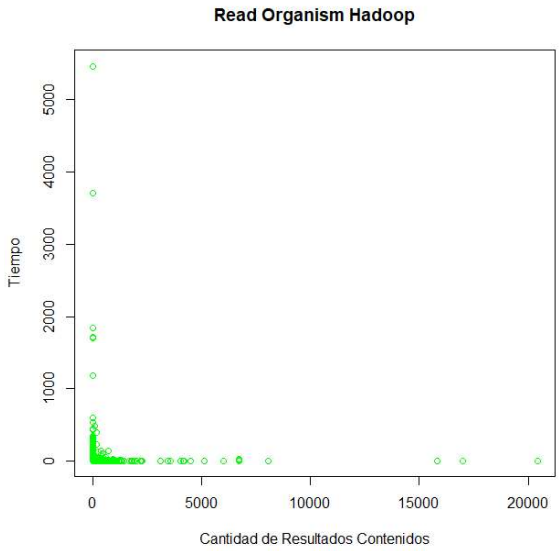

Figura 3.15 Diagrama de dispersión Tiempo De Operación vs Cantidad De Registros con parámetro *organism* Hadoop

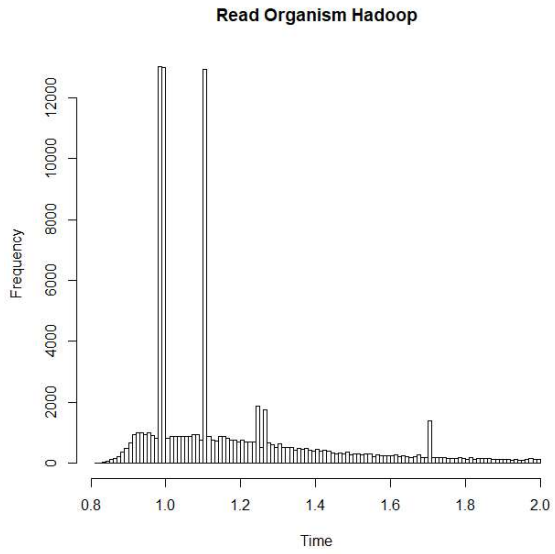

Figura 3.16 Histograma búsqueda de datos con parámetro *organism* a través de Hadoop con tiempo de respuesta de 0 a 2 segundos

- ***Read Gene***

En la Figura 3.17 se obtiene un diagrama de dispersión con todos los resultados de la operación de búsqueda de *gene,* los resultados se los puede observar en el ANEXO XX.

En la Figura 3.18 se obtiene un histograma con tiempos de respuesta comprendidos entre 0 y 2 segundos en la operación de búsqueda de *gene*.

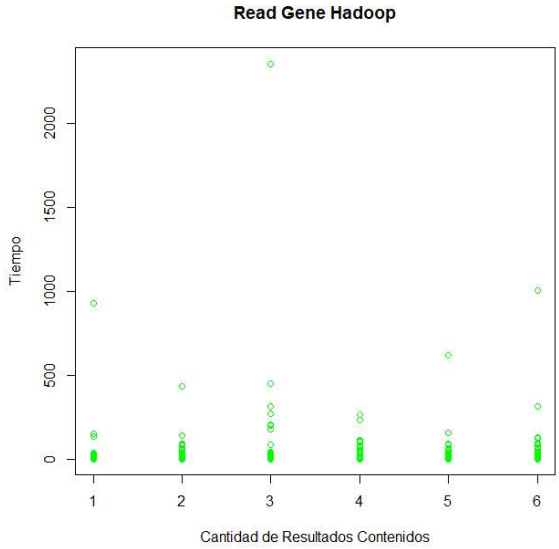

Figura 3.17 Diagrama de dispersión Tiempo De Operación vs Cantidad De Registros con parámetro *gene* Hadoop

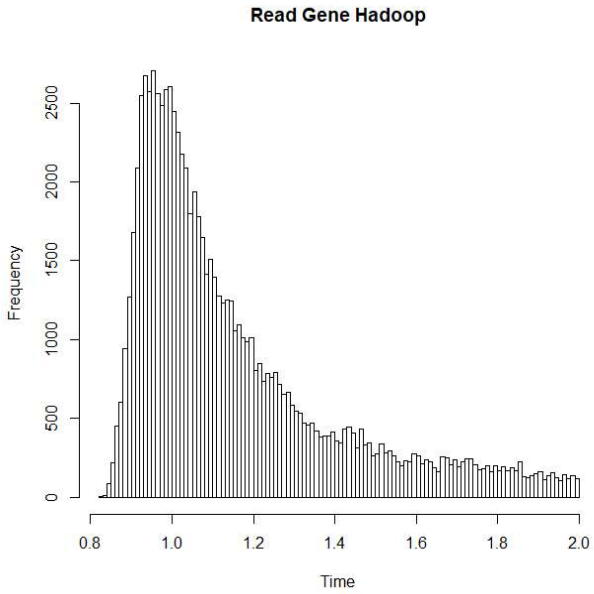

Figura 3.18 Histograma búsqueda de datos con parámetro *gene* a través de Hadoop con tiempo de respuesta de 0 a 2 segundos

UPDATE

Resultados *Update*

Como resultados de la operación se presentan dos tipos de gráficas:

- Gráfica de dispersión: La gráfica de dispersión se representa en dos dimensiones, el Tiempo De Operación, eje *y,* en función la Cantidad De Registros que fueron afectados por dicha operación, eje *x*.
- Histograma: El histograma representa la tendencia de resultados de tiempo que demora la operación en realizar actualizaciones sobre la base de datos UniProt/Swiss-Prot a través de los gestores Elasticsearch y Hadoop con actualizaciones en los campos *name, accession, organism y gene* de los adjuntos en el ANEXO XIV y el ANEXO XXI. El rango de tiempo más común se lo representa en la gráfica en dos dimensiones, eje *x* y eje *y,* siendo el eje *x* los tiempos de respuesta y en el eje *y,* la frecuencia como cantidad de resultados contenidos en los tiempos de respuesta, eje *x*.

Resultados *Update* Gestor de Base de Datos Elasticsearch

En la Figura 3.19 se obtiene un diagrama de dispersión con todos los resultados de la operación de borrado, los resultados se los puede observar en el ANEXO XIV.

En la Figura 3.20 se obtiene un histograma con tiempos de respuesta comprendidos entre 0 y 0.030 segundos en la operación de borrado. En la Figura 3.21 se obtiene un histograma con tiempos de respuesta comprendidos entre 0 y 1 segundo.

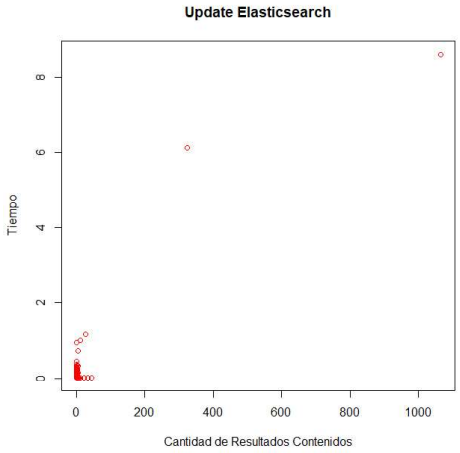

Figura 3.19 Diagrama de dispersión Tiempo De Operación vs Cantidad De Registros afectados *Update* Elasticsearch

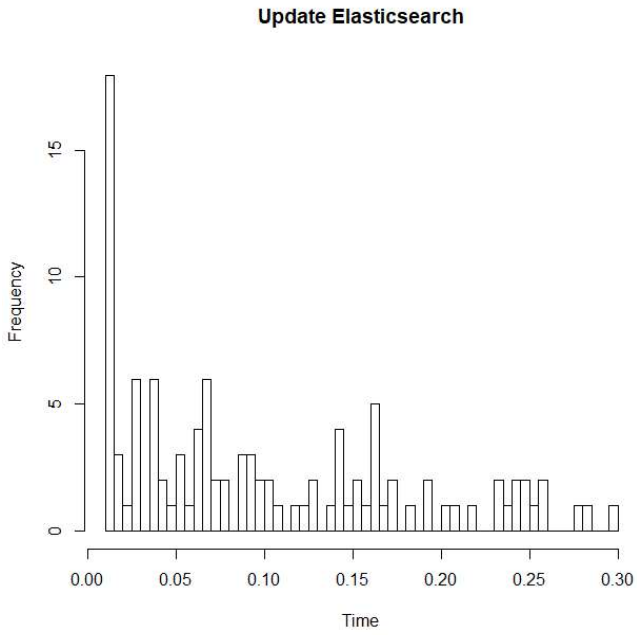

Figura 3.20 Histograma actualizado de valores a través de Elasticsearch con tiempo de respuesta de 0 a 0.030 segundos

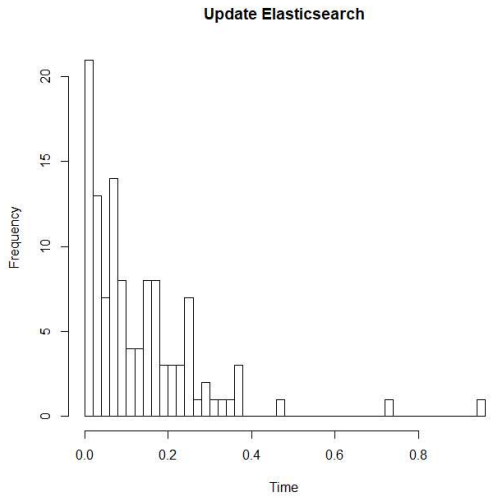

Figura 3.21 Histograma actualizado de valores a través de Elasticsearch con tiempo de respuesta de 0 a 1 segundos.

Resultados de la operación *Update mediante* Gestor de Base de Datos Hadoop

En la Figura 3.22 se obtiene un diagrama de dispersión con todos los resultados de la operación de actualizado, los resultados se los puede observar en el ANEXO XXI.

En la se Figura 3.23 obtiene un histograma con tiempos de respuesta comprendidos entre 0 y 10 segundos en la operación de borrado.

En la se Figura 3.24 obtiene un histograma con tiempos de respuesta comprendidos entre 0 y 2 segundos.

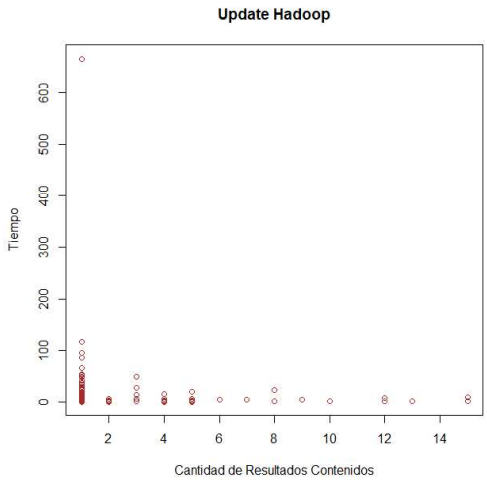

Figura 3.22 Diagrama de dispersión Tiempo De Operación vs Cantidad De Registros afectados *Update* Hadoop

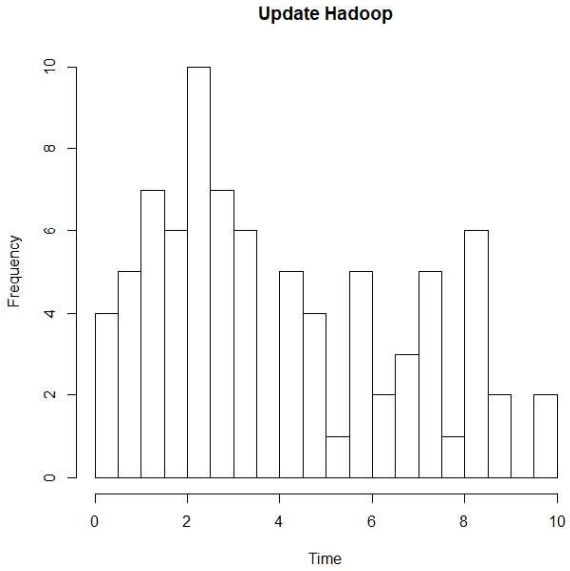

Figura 3.23 Histograma actualizado de valores a través de Hadoop con tiempo de respuesta de 0 a 10 segundos.

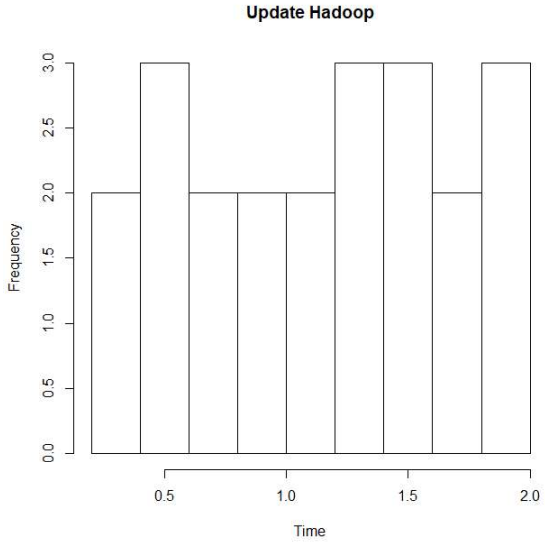

Figura 3.24 Histograma actualizado de valores a través de Hadoop con tiempo de respuesta de 0 a 2 segundos.

DELETE

Se realizó la operación eliminado, DELETE en CRUD sobre la base de datos UniProt\Swiss-Prot mediante cuatro parámetros: *name, accession, organism* y *gene,* de los cuales fueron tomados 30 al azar.

Como resultados de la operación se presentan dos tipos de gráficas:

- Gráfica de dispersión: La gráfica de dispersión se representa en dos dimensiones, el Tiempo De Operación, eje *y,* en función la Cantidad De Registros afectados por dicha operación, eje *x*.

- Histograma: El histograma representa la tendencia de resultados en un tiempo de tiempo que demora la operación en realizar eliminado de los parámetros adjuntos en el ANEXO XV y el ANEXO XXII, sobre la base de datos UniProt/Swiss-Prot a través de los gestores Elasticsearch y Hadoop. El rango de tiempo más común se lo representa en la gráfica en dos dimensiones, eje *x* y eje *y,* siendo el eje *x* los tiempos de respuesta y en el eje *y* la frecuencia como cantidad de resultados que contienen dichos tiempos de respuesta.

Resultados *Delete* Gestor de Base de Datos Elasticsearch

En la Figura 3.25 se obtiene un diagrama de dispersión con todos los resultados de la operación de actualizado, los resultados se los puede observar en el ANEXO XV.

En la Figura 3.26 se obtiene un histograma de la operación de borrado.

En la Figura 3.27 se obtiene un histograma con tiempos de respuesta comprendidos entre 0 y 0.030 segundos.

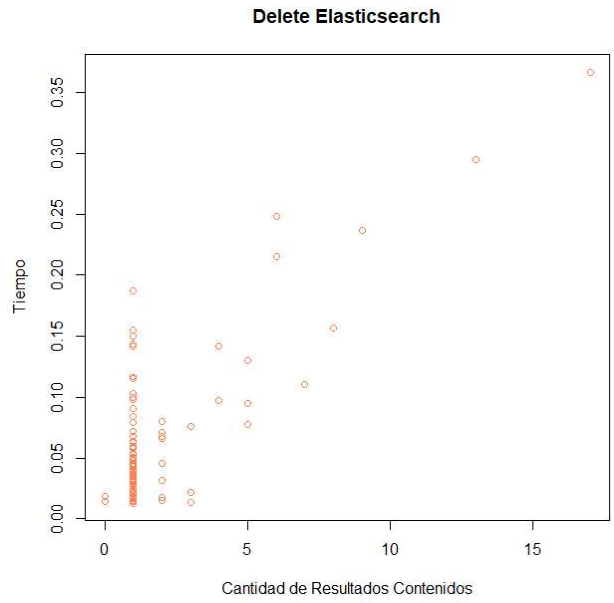

Figura 3.25 Diagrama de dispersión Tiempo De Operación vs Cantidad De Registros *Delete* Elasticsearch

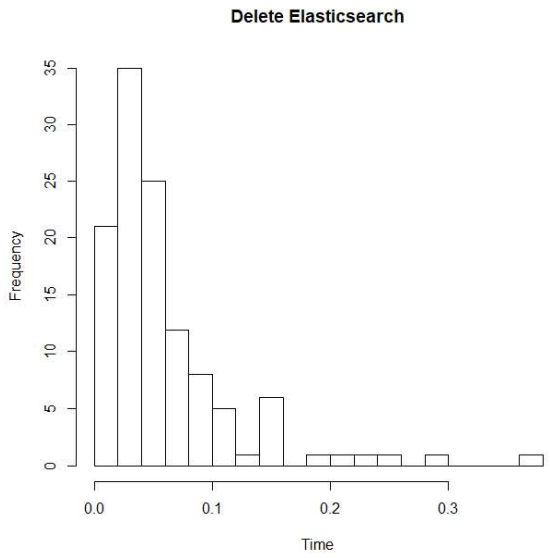

Figura 3.26 Histograma borrado de valores a través de Elasticsearch con tiempo de respuesta de 0 a 2 segundos.

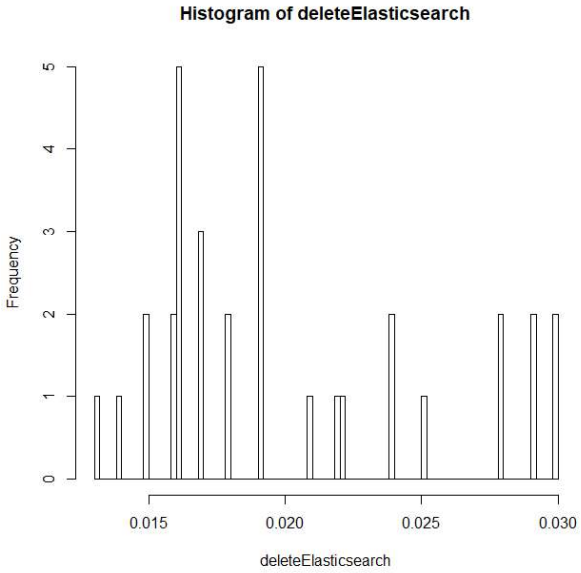

Figura 3.27 Histograma actualizado de valores a través de Elasticsearch con tiempo de respuesta de 0 a 0.30 segundos.

Resultados *Delete* Gestor de Base de Datos Hadoop

En la Figura 3.28 se obtiene un diagrama de dispersión con todos los resultados de la operación de actualizado, los resultados se los puede observar en el ANEXO XXII.

En la Figura 3.29 se obtiene un histograma de la operación de borrado con tiempos de respuesta comprendidos entre 0 y 2 segundos.

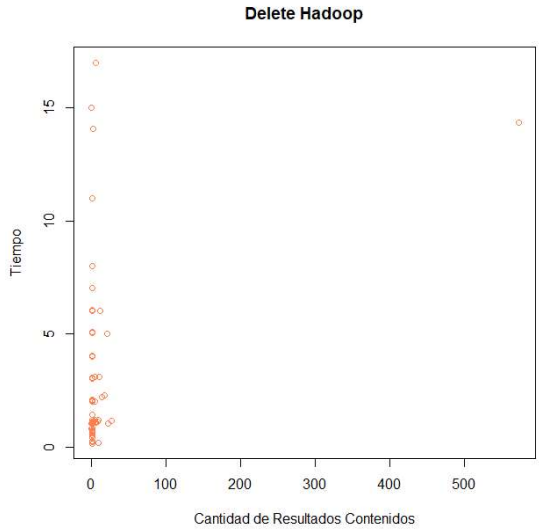

Figura 3.28 Diagrama de dispersión Tiempo De Operación vs Cantidad De Registros *Delete* Hadoop

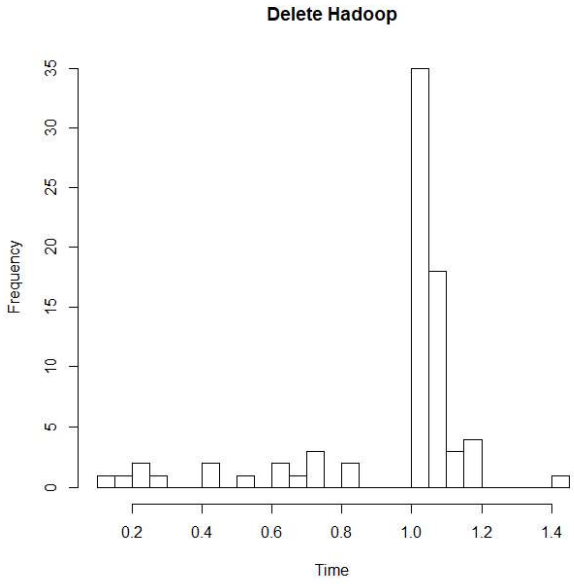

Figura 3.29 Histograma borrado de valores a través de Hadoop con tiempo de respuesta de 0 a 2 segundos.

3.2. Discusión

En la Tabla 3.1 se detallan los valores media, desviación estándar y la cantidad de mediciones que fueron obtenidas en cada uno de los procesamientos de las operaciones CRUD.

$\bar{x}$ = Valor medio en segundos

SD = Desvíación estándar

N = Número de mediciones

Tabla 3.1 Cuadro comparativo entre Elasticsearch y Hadoop

	Elasticsearch			Hadoop		
	$\bar{x}$ [s]	SD [s]	N	$\bar{x}$ [s]	SD [s]	N
Inserción	0.01055172	0.00395135	30	78.44983	49.97379	30
Búsqueda	0.01106615	0.01330013	60,003,06	2.889841	16.40431	484,029
Actualización	0.2676303	0.9628549	120	18.63032	63.06181	120
Borrado	0.06246218	0.05840796	120	2.428767	3.015779	120

Un ejemplo de la obtención de los resultados que se los observó en la Tabla 3.1 **Cuadro comparativo entre Elasticsearch y Hadoop** se lo puede observar en el ANEXO XXIII.

A continuación, se detallan los resultados obtenidos para los gestores de base de datos y que se presentaron en los gráficos de la sección Resultados:

Elasticsearch *Create*

Dentro de las pruebas se obtuvieron 30 resultados de inserciones sobre la base de datos UniProt/Swiss-Prot, de los 30 resultados obtenidos el 100% de estos están entre 0 y 0.03 segundos, Figura 3.1.

Hadoop *Create*

Dentro de las pruebas se obtuvieron 30 resultados de inserciones sobre la base de datos UniProt/Swiss-Prot, Figura 3.2, de los 30 resultados obtenidos el 40% de estos están entre 0 y 50 segundos, Figura 3.3, y el 76.67% están dentro de los 0 a 100 segundos, Figura 3.4.

Elasticsearch *Read*

Para observar el comportamiento de la búsqueda sobre la base de datos UniProt/Swiss-Prot se realizó un histograma de los tiempos obtenidos para observar cual es el tiempo que

tienden a demorarse las búsquedas sobre la base de datos UniProt/Swiss-Prot a través del gestor Elasticsearch. De lo cual se puede observar:

Elasticsearch *Read Name*

- El 100% de los resultados corresponde a 16,771,499 valores de los cuales 16,679,913 corresponde al 99.45% de los valores obtenidos en la búsqueda y los tiempos de respuesta están entre los 0 y 0.03 segundos, Figura 3.5.

Elasticsearch *Read Accession*

- El 100% de los resultados corresponde a 23,593,023 valores de los cuales 23,415,097 corresponde al 99.24% de los valores obtenidos en la búsqueda y los tiempos de respuesta están entre los 0 y 0.03 segundos, Figura 3.6.

En la Figura 3.7 y la Figura 3.9 se puede observar que la herramienta gestora de base de datos Hadoop en consultas con el nodo *organismo y gen,* respectivamente, como parámetro, no varía el tiempo en función a la cantidad de resultados que esta retorna.

Por ello se realizó un histograma de los tiempos obtenidos para observar cual es el tiempo que tienden a demorarse las búsquedas sobre la base de datos UniProt/Swiss-Prot. De lo cual se puede observar:

Elasticsearch *Read Organism*

- El 100% de los resultados corresponde a 413,609 valores de los cuales 407,547 corresponde al 98.53% de los valores obtenidos en la búsqueda están entre los 0 y 0.03 segundos, Figura 3.8.

Elasticsearch *Read Gene*

- El 100% de los resultados corresponde a 19,224,929 valores de los cuales 19,122,554 corresponde al 99.47% de los valores obtenidos en la búsqueda están entre los 0 y 0.03 segundos, Figura 3.10.

Hadoop *Read*

Para observar el comportamiento de la búsqueda sobre la base de datos UniProt/Swiss-Prot se realizó un histograma de los tiempos obtenidos para observar cual es el tiempo que tienden a demorarse las búsquedas sobre la base de datos UniProt/Swiss-Prot a través del gestor Hadoop. De lo cual se puede observar:

Hadoop *Read Name*

- El 100% de los resultados corresponde a 96,341 valores de los cuales 95296 corresponde al 98.92% de los valores obtenidos en la búsqueda.
- El 98.92% de los datos demoran entre 0 a 10 segundos en la búsqueda de registros mediante el organismo, Figura 3.11.
- El 100% de los resultados corresponde a 96,341 valores de los cuales 78,458 corresponde al 81.44% de los valores obtenidos en la búsqueda.
- El 81.44% de los datos demoran entre 0 a 2 segundos en la búsqueda de registros mediante el organismo, Figura 3.12.

Hadoop *Read Accession*

- El 100% de los resultados corresponde a 79,679 valores de los cuales 79,055 corresponde al 99.22% de los valores obtenidos en la búsqueda.
- El 99.22% de los datos demoran entre 0 a 10 segundos en la búsqueda de registros mediante el organismo, Figura 3.13.
- El 100% de los resultados corresponde a 79,679 valores de los cuales 66,923 corresponde al 83.99% de los valores obtenidos en la búsqueda.
- El 83.99% de los datos demoran entre 0 a 2 segundos en la búsqueda de registros mediante el organismo, Figura 3.14.

En la Figura 3.15 y la Figura 3.17 se puede observar que la herramienta gestora de base de datos Hadoop en consultas con el nodo *organismo y gen* como parámetro, no varía el tiempo en función a la cantidad de resultados que esta retorna.

Por ello se realizó un histograma de los tiempos obtenidos para observar cual es el tiempo que tienden a demorarse las búsquedas sobre la base de datos UniProt/Swiss-Prot. De lo cual se puede observar:

Hadoop *Read Organism*

- El 100% de los resultados corresponde a 106,571 valores de los cuales 91.545 corresponde al 85.90% de los valores obtenidos en la búsqueda.
- El 85.90% de los datos demoran entre 0 a 2 segundos en la búsqueda de registros mediante el organismo, Figura 3.16.

Hadoop Read Gene

- El 100% de los resultados corresponde a 100,719 valores de los cuales 84,198 corresponde al 83.60% de los valores obtenidos en la búsqueda.
- El 83.60% de los datos demoran entre 0 a 2 segundos en la búsqueda de registros mediante el organismo, Figura 3.18.

Elasticsearch *Update*

Dentro de las pruebas se obtuvieron 120 resultados de actualizaciones sobre la base de datos UniProt/Swiss-Prot, de los resultados obtenidos el 24.17% de estos están entre 0 y 0.03 segundos, Figura 3.20, y el 96.67% están dentro de los 0 a 1 segundos, Figura 3.21.

Hadoop *Update*

- El 100% de los resultados corresponde a 120 valores de los cuales 82 corresponde al 68.33% de los valores obtenidos en la actualización, los mismos que demoran entre 0 a 10 segundos, Figura 3.23.
- El 19.16% de los datos de tiempo están entre 0 y 2 segundos, Figura 3.24.

Elasticsearch *Delete*

- Dentro de las pruebas se obtuvieron 120 resultados de la eliminación de datos sobre la base de datos UniProt/Swiss-Prot, de los resultados obtenidos el 28.33% de estos están entre 0 y 0.03 segundos, Figura 3.27 y el 100% están dentro de los 0 a 2 segundos, Figura 3.26.

Hadoop *Delete*

- El 100% de los resultados corresponde a 120 valores de los cuales 115 corresponde al 95.83% de los valores obtenidos en la eliminación de registros, los mismos que demoran entre 0 a 10 segundos.
- El 65% de los datos de tiempo están entre 0 y 2 segundos, Figura 3.29.

4. CONCLUSIONES

En base al presente proyecto se responde a la pregunta de investigación ¿existe una diferencia significativa entre el desempeño computacional en términos de tiempo de ejecución de consultas entre los gestores de bases de datos NoSQL Elasticsearch y Hadoop, de forma afirmativa. Si, existe una diferencia significativa.

En el presente proyecto, el clúster de Hadoop resultó tener una infraestructura pesada y cara debido a que maneja respaldos de información entre los nodos disponibles. Esto se reduce a que en la planificación de la capacidad de Hadoop es necesario considerar la cantidad de nodos de datos que se han configurado. En el presente proyecto se configuraron dos nodos de datos en donde se observó que el clúster requiere el doble de memoria para procesamiento de su información y el doble de almacenamiento de sus datos. Además, durante la configuración del sistema operativo del nodo es necesario definir una memoria *Swap* equivalente al tamaño de almacenamiento requerido para la información. Si se consideran estos factores, se pueden evitar errores relacionados con *Garbage Collector* y *Out Of Memory* de java.

En el presente proyecto se ha cumplido con el objetivo de estudiar las herramientas Elasticsearch y Hadoop, el mismo que se puede observar en el apartado 2.2.1.

En el presente proyecto se ha cumplido con el objetivo de conocer la base de datos UniProt/Swiss-Prot, el mismo que se puede observar en el apartado 2.1.2.

En el presente proyecto se ha cumplido con el objetivo de levantar el ambiente mediante la creación de APIs de servicios Web, esto mediante el ANEXO VII para la herramienta Hadoop y para la herramienta Elasticsearch el servicio se encuentra integrado.

Es necesario definir el uso de memoria java en la configuración del cluster de Elasticsearch para evitar problemas de tipo *Out Of Memory*.

Se estudió la base de datos como requisito fundamental para su depuración y selección de información relevante para el estudio.

Se hizo uso de la API que provee Elasticsearch, pero para Hadoop se implementó una API propia ya que la aplicación Hive no disponía de una. De ello se puede concluir que la conexión entre la API y el gestor Hive/Hadoop pudo influir en los resultados del desempeño pero no de una manera significativa, lo cual no se tomó en cuenta ya que la evaluación de tiempo se lo hacía a un nivel de sistema con la arquitectura definida. Futuros estudios pueden verificar si este factor influye o no en los resultados.

Se realizó la comparación de los resultados y se lo puede observar en el apartado 3.2. Discusión. Con base en los resultados obtenidos, el gestor de bases de datos más eficiente en términos de tiempo de ejecución en operaciones sobre la base de datos UniProt/Swiss-Prot es Elasticsearch y se lo explica de la siguiente manera:

- El análisis de los resultados, plasmados en los histogramas, permitió observar que para crear un nuevo registro o documento en la base de datos UniProt/Swiss-Prot, Elasticsearch es significativamente más rápido que Hadoop. Ya que el 100% de la inserción de datos demoró entre 0 y 0.030 segundos. Mientras que Hadoop, la mayoría de los resultados de inserción de datos, correspondiente al 76.67% demoró de 0 a100 segundos y el 40% de la inserción de datos demoró de 0 a 50 segundos, que aun así es significativamente más lento que Elasticsearch.

- Para evaluar la operación de búsqueda los resultados se analizaron en dos tipos de gráficos: 1) el diagrama de dispersión que permitió observar que en la búsqueda de los valores de los nodos *name, accession* y *organism* en la base de datos UniProt/Swiss-Prot, no tiene ninguna relación con la cantidad de resultados encontrados; y por otra parte 2) un histograma que permitió observar que para buscar registros o documentos la diferencia de los tiempos de respuesta entre Hadoop y Elasticsearch no son tan grandes como en la operación de creación, siendo la mínima variación de tiempo entre los dos gestores desde milésimas de segundos hasta 2 segundos. Que, si bien es cierto, Elasticsearch es mucho más rápido que Hadoop, a nivel de sistema no hubo una demora significativa en espera de los resultados de Hadoop con respecto a Elasticsearch. De ello se concluye que para la búsqueda con pocos datos de entrada Hadoop y Elasticsearch no tiene una diferencia significativa en tiempos de respuesta de búsqueda.

- Para evaluar la operación de actualizado los resultados se analizaron en dos tipos de gráficos: 1) el diagrama de dispersión que permitió observar que en la actualización de los valores de los nodos *name, accession, gene* y *organism* en la base de datos UniProt/Swiss-Prot, no tiene ninguna relación con la cantidad de resultados afectados; y por otra parte 2) un histograma que permitió observar que para actualizar registros o documentos en la base de datos el 96.64% de los resultados obtenidos mediante Elasticsearch demoró entre 0 y 1 segundos. Mientras que en Hadoop la mayoría de los resultados de actualización de datos, correspondiente al 68.33% demoró de 0 a10 segundos. Que a nivel de sistema en el cual se hicieron las pruebas si tiende a ser significativo frente a la cantidad de peticiones que se hicieron. Además de ello se concluye que para la actualización

con pocos datos de entrada Hadoop y Elasticsearch podrían no tener una diferencia significativa en tiempos de respuesta.

- Para la operación de borrado los resultados se analizaron mediante dos tipos de gráficos: 1) el diagrama de dispersión permitió observar que en la eliminación de los valores de los nodos *name, accession, gene* y *organism* en la base de datos UniProt/Swiss-Prot, no tiene ninguna relación con la cantidad de resultados afectados; y por otra parte 2) histograma permitió observar que la eliminación de registros o documentos el 100% de las peticiones demoraron entre 0 a 1 segundos en Elasticsearch. Mientras que el 64.70% en Hadoop demoraron entre 0 y 2 segundos, que a nivel de sistema que se probó si tiende a ser significativo frente a la cantidad de peticiones que se hicieron. Además de ello se concluye que para la eliminación con pocos datos de entrada Hadoop y Elasticsearch podrían no tener una diferencia significativa en tiempos de respuesta.

Los histogramas permitieron observar que, obteniendo una pequeña cantidad de valores a graficar, estos valores tienden a ser variables discretas para los gráficos, esto ya que hubo varios casos en la que la variación del tiempo entre una petición y otra no tuvieron puntos intermedios.

Una propiedad intrínseca de Elasticsearch es tener una desviación estándar insignificante.

Los resultados obtenidos en el cálculo de la media y desviación estándar de las medidas de tiempo de la operación inserción indica que las 30 peticiones hechas al gestor Elasticsearch en promedio demoran 0.0105 segundos en ejecutarse, desviados de este dato 0.0039 unidades de la escala de segundos; evidenciando que no existe una alta dispersión de resultado de tiempo de respuesta a la petición de inserción.

Los resultados obtenidos en el cálculo de la media y desviación estándar de las medidas de tiempo de la operación búsqueda indica que las 60,003,061 peticiones hechas al gestor Elasticsearch en promedio demoran 0.0111 segundos en ejecutarse, desviados de este dato 0.0133 unidades de la escala de segundos; evidenciando que no existe una alta dispersión de resultado de tiempo de respuesta a la petición de búsqueda.

Los resultados obtenidos en el cálculo de la media y desviación estándar de las medidas de tiempo de la operación actualizado indica que las 120 peticiones hechas al gestor Elasticsearch en promedio demoran 0.2676 segundos en ejecutarse, desviados de este dato 0.9628 unidades de la escala de segundos; evidenciando que no existe una alta dispersión de resultado de tiempo de respuesta a la petición de actualización.

Los resultados obtenidos en el cálculo de la media y desviación estándar de las medidas de tiempo de la operación eliminado indica que las 120 peticiones hechas al gestor Elasticsearch en promedio demoran 0.0624 segundos en ejecutarse, desviados de este dato 0.0584 unidades de la escala de segundos; evidenciando que no existe una alta dispersión de resultado de tiempo de respuesta a la petición de eliminación.

Los resultados obtenidos en el cálculo de la media y desviación estándar de las medidas de tiempo de la operación inserción indica que las 30 peticiones hechas al gestor Hadoop en promedio demoran 78.45 segundos en ejecutarse, desviados de este dato 49.97 unidades de la escala de segundos; evidenciando que existe una alta dispersión de resultado de tiempo de respuesta a la petición de inserción.

Los resultados obtenidos en el cálculo de la media y desviación estándar de las medidas de tiempo de la operación búsqueda indica que las 484,029 peticiones hechas al gestor Elasticsearch en promedio demoran 2.89 segundos en ejecutarse, desviados de este dato 16.40 unidades de la escala de segundos; evidenciando que existe una alta dispersión de resultado de tiempo de respuesta a la petición de búsqueda.

Los resultados obtenidos en el cálculo de la media y desviación estándar de las medidas de tiempo de la operación actualizado indica que las 120 peticiones hechas al gestor Elasticsearch en promedio demoran 18.63 segundos en ejecutarse, desviados de este dato 63.06 unidades de la escala de segundos; evidenciando que existe una alta dispersión de resultado de tiempo de respuesta a la petición de actualización.

Los resultados obtenidos en el cálculo de la media y desviación estándar de las medidas de tiempo de la operación eliminado indica que las 120 peticiones hechas al gestor Elasticsearch en promedio demoran 2.43 segundos en ejecutarse, desviados de este dato 3.02 unidades de la escala de segundos; evidenciando que no existe una alta dispersión de resultado de tiempo de respuesta a la petición de eliminación.

La desviación estándar en el gestor Hadoop es alta debido a la alta dispersión de valores en tiempo de respuesta del gestor.

Trabajos Futuros

En el presente proyecto la aplicación mediante la cual se analizan los tiempos de respuesta de Hadoop es Apache Hive. Hive es un *data warehouse* a nivel de aplicación del ecosistema Hadoop que transforma las peticiones de un lenguaje like SQL a Jobs MapReduce. Durante los últimos años se ha popularizados Apache Spark y se ha convertido en un framework potente para el procesamiento de datos para Apache Hadoop. Su popularidad se deriva a que contiene APIs flexibles para grandes cargas de trabajo, un desarrollo fácil y un rápido procesamiento por lotes lo que lo convierte en un motor muy potente subyacente a Hive. El paradigma Hive-on-Spark (HoS) permite a Hive ejecutarse en Spark como un motor de cómputo mientras mantiene la compatibilidad completa con las cargas actuales de Hive-on-MapReduce. Se podría hacer uso de Spark como *engine* de Hive y junto con ello un análisis sobre la base de datos UniProt/Swiss-Prot y verificar si el rendimiento del sistema obtiene diferentes tiempos de respuesta.

Por otro lado, se podría hacer la implementación de los clústeres definidos en el proyecto de una manera ya no pseudo-distribuida. Es decir, que podrían emplearse maquinas reales como nodos y verificar si el tiempo de respuesta es el mismo que en clústeres pseudo-distribuidos, que fue el entorno planteado en este proyecto.

Además, Hadoop presenta HCatalog que es una capa de administración de tablas y almacenamiento para Hadoop que permite a los usuarios con diferentes herramientas de procesamiento de datos como Hive leer y escribir datos más fácilmente con HDFS. Además de que HCatalog, contiene un servicio integrado de nombre WebHcat. Podría implementarse como una capa intermedia en el entorno planteado en el proyecto, para verificar si el uso de HCatalog con su servicio integrado, mejora el rendimiento su trabajo de la mano con Hive.

REFERENCIAS BIBLIOGRÁFICAS

[1] H. P.-S. S. G. a. D. D. Ivan Merelli, «Managing, Analysing, and Integrating Big Data in Medical Bioinformatics: Open Problems and Future Perspectives,» *Hindawi,* 2014.

[2] D. G. M. G. N. M. Luscombe, «What is Bioinformatics? A Proposed Definition and Overview of the Field,» *Department of Molecular Biophysics and Biochemistry Yale University, New Haven, USA,* 2001.

[3] ExPASy, «ExPASy Bioinformatics Resource Portal,» 2019. [En línea]. Available: https://web.expasy.org/docs/relnotes/relstat.html. [Último acceso: 05 03 2019].

[4] R. Jain, Art of Computer Systems Performance Analysis Techniques For Experimental Design Measurements Simulation And Modeling, Wiley Computer Publishing, John Wiley & Sons, Inc., 1991.

[5] UniProt, «UniProt,» 2018 01 31. [En línea]. Available: http://www.uniprot.org/docs/userman#what_is_sprot. [Último acceso: 15 02 2018].

[6] E. A. I. V. V. K. D. V. P. A. A. M. A. A. G. M. N. U. N. N. Nazipova, «Big Data in Bioinformatics,» *Mathematical Biology and Bioinformatics,* pp. 102-119, 2017.

[7] A. V. Davide Cirillo, «Big data analytics for personalized medicine,» *ELSEVIER,* 2019.

[8] A. M. A. L. C. O. M. M. P. Ricardo Sánchez-de-Madariaga, «Executing Complexity-Increasing Queries in Relational (MySQL) and NoSQL,» *Journal of Visualized Experiments,* 2018.

[9] EcuRed, «EcuRed,» [En línea]. Available: https://www.ecured.cu/SGBD. [Último acceso: 18 05 2019].

[10] A. R. F. M. MARÍA JESÚS RAMOS, Sistemas gestores de bases de datos, Madrid-España: Mc Graw Hill, 2006.

[11] J. S. G. S. M. C. C. Alexander Castro Romero, «Utilidad y funcionamiento de las bases de datos NoSQL,» *Revista Facultad de Ingeniería, UPTC,* vol. 21, nº 33, pp. 21-32, 2012.

[12] R. Cattell, «Scalable SQL and NoSQL Data Stores,» 2010.

[13] D. d. F. T. Álvaro Diez del Valle Medrano, «Comparativa Del Rendimiento de Consultas Entre Sistemas Relacionales (ORACLE Y MySQL),» 2016.

[14] C.-Y. Z. Philip Chen, «Data-intensitive applications, challenges, techniques and technologies: A survey on Big Data,» 2014.

[15] G. I. Jeffrey Dean and Sanjay Ghemawat, «MapReduce: Simplified Data Processing on Large Clusters,» *USENIX Association.*

[16] The Apache Software Foundation, «Hadoop,» 09 07 2018. [En línea]. Available: https://hadoop.apache.org/docs/r1.2.1/mapred_tutorial.html. [Último acceso: 07 04 2019].

[17] F. R. NAVIA, «ITSoftware,» 24 06 2016. [En línea]. Available: https://itsoftware.com.co/content/mapreduce-procesamiento-paralelo/. [Último acceso: 19 05 2019].

[18] W. K. a. P. B. Thomas Erl, Big Data Fundamentals, Indiana, USA: PRENTICE HALL, 2015.

[19] «UniProt,» [En línea]. Available: https://www.uniprot.org/uniprot/?query=*&fil=reviewed%3Ayes.

[20] UniProt, «UniProt Knowledgebase, Swiss-Prot Protein Knowledgebase, TrEMBL Protein Database-User Manual,» 08 05 2019. [En línea]. Available: https://www.uniprot.org/docs/userman#diffEMBL. [Último acceso: 19 05 2019].

[21] [En línea]. Available: https://www.abbexa.com/enzyme-commission-number.

[22] «Hadoop,» 09 07 2018. [En línea]. Available: https://hadoop.apache.org/docs/r1.2.1/hdfs_design.html. [Último acceso: 19 05 2019].

[23] Siva, «Hadoop Online Tutorial,» 09 04 2014. [En línea]. Available: http://hadooptutorial.info/secondary-namenode-in-hadoop/. [Último acceso: 19 05 2019].

[24] V. K. Vavilapalli, «Hortonworks,» 31 08 2012. [En línea]. Available: https://es.hortonworks.com/blog/apache-hadoop-yarn-resourcemanager/. [Último acceso: 19 05 2019].

[25] O'Reilly Media, Inc., Big Data Now: 2012 Edition, Boston: O'Reilly, 2012.

[26] F. Ugalde, «Francisco Ugalde Blog sobre tecnología, desarrollo y más,» 09 02 2018. [En línea]. Available: https://www.franciscougalde.com/2018/02/19/elasticsearch-arquitectura-proyectos-big-data/. [Último acceso: 10 04 2019].

[27] J. Vanderzyden, «QBOX,» 01 09 2015. [En línea]. Available: https://qbox.io/blog/what-is-Elasticsearch. [Último acceso: 10 04 2019].

[28] «Stats With Cats Blog,» 11 07 2010. [En línea]. Available: https://statswithcats.wordpress.com/2010/07/11/30-samples-standard-suggestion-or-superstition/. [Último acceso: 01 03 2019].

[29] L. Carvajal, Metodología de la Investgación Científica. Curso general y aplicado, 28 ed., Santiago de Cali: U.S.C., 2006, p. 139.

[30] Talend Team, «talend,» 12 03 2019. [En línea]. Available: https://www.talend.com/resources/what-is-mapreduce/. [Último acceso: 07 04 2019].

[31] Universidad Complutense de Madrid, «Universidad Complutense de Madrid,» [En línea]. Available: https://www.ucm.es/gyp/proteomica. [Último acceso: 08 03 2019].

[32] *. A. E. L. B. S. P. N. R. I. X. A. B. a. t. U. C. M.L. Famiglietti, «An enhanced workflow for variant interpretation in UniProtKB/Swiss-Prot improves consistency and reuse in ClinVar,» *DATABASE,* 2018.

[33] PowerData, «PowerData,» [En línea]. Available: https://www.powerdata.es/big-data. [Último acceso: 12 05 2019].

[34] D. G. M. G. N. M. Luscombe, «What is Bioinformatics?A Proposed Definition and Overview of the Field,» *Department of Molecular Biophysics and Biochemistry Yale University, New Haven, USA.*

[35] E. B. Hernández, «Bioinformática: una oportunidad y un desafío,» *Revista Colombiana de Biotecnología,* pp. 132-138, 2008.

[36] v. mdnwebdocs-bot, «MDN web docs,» 23 03 2019. [En línea]. Available: https://developer.mozilla.org/es/docs/Glossary/CRUD. [Último acceso: 18 05 2019].

[37] S. Kumar, «Cloudera,» 09 05 2016. [En línea]. Available: http://vision.cloudera.com/faster-batch-processing-with-hive-on-spark/. [Último acceso: 25 05 2019].

[38] L. Leverenz, 16 12 2018. [En línea]. Available: https://cwiki.apache.org/confluence/display/Hive/HCatalog+UsingHCat. [Último acceso: 25 05 2019].

5. ANEXOS

ANEXO I

El código para transformar datos de formato XML a JSON, proponer un mismo formato para los datos *name, gene, accession, organismo* y *protein* y eliminación de caracteres corruptos en las claves y valores del documento JSON se encuentra anexo en el CD adjunto con el nombre “PruneScript.py”

ANEXO II

La representación de la proteína 1 en formato XML se encuentra adjunto en el CD adjunto, con el nombre “ejemploProteinXML1.txt”

ANEXO III

Código para ejecutar las operaciones *create, update, delete* sobre el gestor Elasticsearch se encuentra adjunto en el CD adjunto, en la carpeta “Scripts-CRUD Elasticsearch”, con el nombre “ESCUDPerformanceTest.py”.

ANEXO IV

Código para ejecutar la operación *read* sobre el gestor Elasticsearch se encuentra adjunto en el CD adjunto, en la carpeta “Scripts-CRUD Elasticsearch”, con el nombre “ESGetPerformanceTest.py”.

ANEXO V

Código para ejecutar las operaciones *create, update, delete* sobre el gestor Elasticsearch se encuentra adjunto en el CD adjunto, en la carpeta “Scripts-CRUD Hadoop”, con el nombre “HDCUDPerformance.py”.

ANEXO VI

Código para ejecutar la operación *read* sobre el gestor Hadoop se encuentra adjunto en el CD adjunto, en la carpeta “Scripts-CRUD Hadoop”, con el nombre “HDGetPerformance.py”.

ANEXO VII

Código del servicio Web levantado para el gestor Hadoop se encuentra adjunto en el CD entregado con el nombre “APIService.py”.

ANEXO VIII

Los datos de entrada de accession, name, gene y organism se encuentra anexado en el CD adjunto, en la carpeta “Datos de Entrada”, con el nombre “accession.txt”, “name.txt”, “gene.txt” y “organismo.txt” respectivamente.

ANEXO IX

El log correspondiente a la operación create en el gestor Elasticsearch se encuentra anexado en el CD adjunto, en la carpeta “Logs Elasticsearch”, con el nombre “create_protein.log”.

ANEXO X

El log correspondiente a la operación read de *name* en el gestor Elasticsearch se encuentra anexado en el CD adjunto, en la carpeta “Logs Elasticsearch”, con el nombre “name-Elasticsearch.log”.

ANEXO XI

El log correspondiente a la operación read de *accession* en el gestor Elasticsearch se encuentra anexado en el CD adjunto, en la carpeta “Logs Elasticsearch”, con el nombre “accession-Elasticsearch.log”.

ANEXO XII

El log correspondiente a la operación read de *organism* en el gestor Elasticsearch se encuentra anexado en el CD adjunto, en la carpeta “Logs Elasticsearch”, con el nombre “organism-Elasticsearch.log”.

ANEXO XIII

El log correspondiente a la operación read de *gene* en el gestor Elasticsearch se encuentra anexado en el CD adjunto, en la carpeta “Logs Elasticsearch”, con el nombre “gene-Elasticsearch.log”

ANEXO XIV

El log correspondiente a la operación update en el gestor Elasticsearch se encuentra anexado en el CD adjunto, en la carpeta “Logs Elasticsearch”, con el nombre “update_protein.log”.

ANEXO XV

El log correspondiente a la operación delete en el gestor Elasticsearch se encuentra anexado en el CD adjunto, en la carpeta “Logs Elasticsearch”, con el nombre “delete_protein.log”

ANEXO XVI

El log correspondiente a la operación create en el gestor Hadoop se encuentra anexado en el CD adjunto, en la carpeta “Logs Hadoop”, con el nombre “create_protein.log”.

ANEXO XVII

El log correspondiente a la operación read de *name* en el gestor Hadoop se encuentra anexado en el CD adjunto, en la carpeta "Logs Hadoop", con el nombre "name-Hadoop.log".

ANEXO XVIII

El log correspondiente a la operación read de *accession* en el gestor Hadoop se encuentra anexado en el CD adjunto, en la carpeta "Logs Hadoop", con el nombre "accession-Hadoop.log".

ANEXO XIX

El log correspondiente a la operación read de *organism* en el gestor Hadoop se encuentra anexado en el CD adjunto, en la carpeta "Logs Hadoop", con el nombre "organism-Hadoop.log".

ANEXO XX

El log correspondiente a la operación read de *gene* en el gestor Hadoop se encuentra anexado en el CD adjunto, en la carpeta "Logs Hadoop", con el nombre "gene- Hadoop.log".

ANEXO XXI

El log correspondiente a la operación update en el gestor Hadoop se encuentra anexado en el CD adjunto, en la carpeta "Logs Hadoop", con el nombre "update_protein.log".

ANEXO XXII

El log correspondiente a la operación delete en el gestor Hadoop se encuentra anexado en el CD adjunto, en la carpeta "Logs Hadoop", con el nombre "delete_protein.log".

ANEXO XXIII

El procedimiento para obtener los resultados de la media y desviación estándar de las operaciones CRUD sobre los gestores Elasticsearch y Hadoop es el siguiente:

- Uso de la herramienta RStudio
- Cargar en una variable todos los valores para obtener la media.
- Utilizar la fórmula propuesta por R para obtener la media de un conjunto de valores, la misma que es *man.* Como se observa a continuación en la figura 1.
- Utilizar la fórmula propuesta por R para obtener la desviación estándar de un conjunto de valores, la misma que es *sd.* Como se observa a continuación en la figura 1.

```
> datosoe= read.csv('C:\\elasticsearch\\update-elasticsearch-hist.csv', dec = '.')
> mean(as.numeric(unlist(datosoe, use.names = FALSE)),na.rm=FALSE)
[1] 0.2676303
>
> sd(as.numeric(unlist(datosoe, use.names = FALSE)),na.rm=FALSE)
[1] 0.9628549
```

Figura 1. Ejemplo de obtención de los valores media y desviación estándar de los resultados obtenidos en actualizado con Elasticsearch

De la misma manera, como se han obtenido los valores de la media y desviación estándar de los resultados obtenidos en el actualizado mediante la herramienta gestora Elasticsearch y Hadoop, se han obtenido los resultados que se han presentado en la Tabla 3.1 Cuadro comparativo entre Elasticsearch y Hadoop.

ANEXO XXIV

El procedimiento para obtener los gráficos presentes en el apartado Resultados se detalla a continuación:

- Uso de la herramienta RStudio.
- Cargar en una variable todos los valores que se desean graficar.
- Graficar histograma, se puede observar su código y gráfico en la Figura 2.
- Graficar diagrama de dispersión, se puede observar su código y gráfico en la Figura 3.

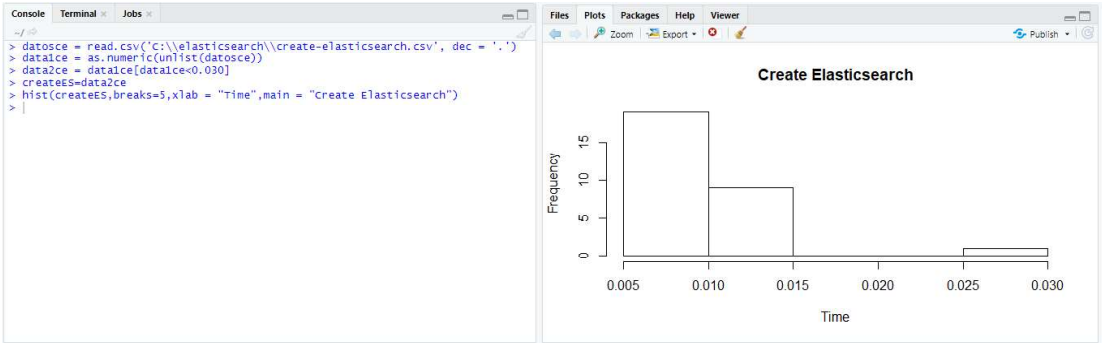

Figura 2. Código y gráfico de histograma de creado de datos con Elasticsearch

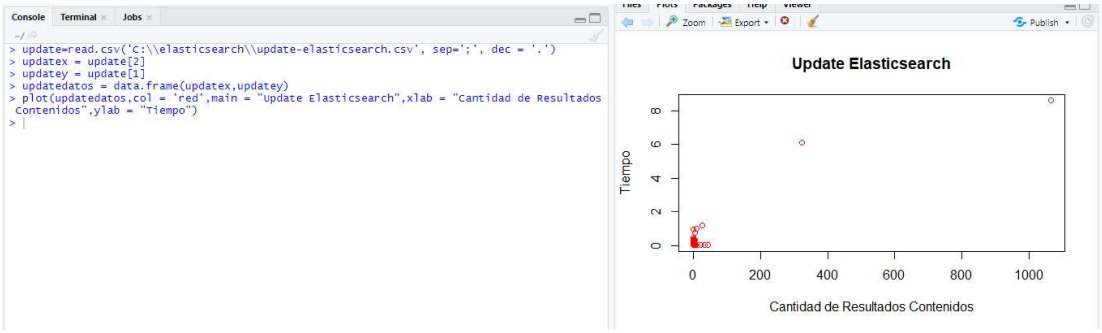

Figura 3. Código y gráfico de diagrama de dispersión de actualizado de datos con Elasticsearch

En los figuras 3 y 4 se observan ejemplos de la generación de los resultados de creado y actualización observados mediante histograma y diagrama de dispersión respectivamente, de la misma manera se fueron generando las demás figuras presentes en el apartado de Resultados