Universidad Politécnica
de Madrid

Escuela Técnica Superior de
Ingenieros Informáticos

Grado en Ingeniería Informática

Trabajo Fin de Grado

Análisis de Grafos en Big Data Sobre Datos de Vuelos

Autor: Juan Felipe López Peternel

Tutor(a): Fernando Pérez Costoya

Madrid, mayo de 2024

Este Trabajo Fin de Grado se ha depositado en la ETSI Informáticos de la Universidad Politécnica de Madrid para su defensa.

Trabajo Fin de Grado
Grado en Ingeniería Informática
Título: Análisis de Grafos en Big Data Sobre Datos de Vuelos
Mayo 2024

Autor: Juan Felipe López Peternel

Tutor: Fernando Pérez Costoya
Departamento de Arquitectura y Tecnología de Sistemas Informáticos (DATSI)
ETSI Informáticos
Universidad Politécnica de Madrid

Resumen

Este Trabajo de Fin de Grado (TFG) se centrará en el estudio de los grafos en el ámbito del Big Data, específicamente en relación con los datos de vuelos. La organización de los datos en forma de grafo se ha vuelto esencial en el campo del Big Data. Aunque existen herramientas clásicas como MapReduce y Spark RDD/DataFrame que se utilizan ampliamente, no ofrecen un soporte directo para el modelo de grafos, lo que obliga a los programadores a adaptarse a las características de cada herramienta.

Para superar esta limitación, han surgido herramientas específicas como Pregel, Apache Giraph, GraphX y GraphFrames de Spark. También se ha observado un cambio en el ámbito de las bases de datos hacia bases de datos orientadas a grafos, como Neo4j.

El objetivo principal de este trabajo es realizar un análisis práctico de estas herramientas, aplicándolas al procesamiento de grandes volúmenes de datos de vuelos. Para ello, se diseñará una aplicación que procese un repositorio de datos de vuelos utilizando estas plataformas especializadas. Se plantean los siguientes objetivos:

- Realizar un análisis de las herramientas de Big Data orientadas al procesamiento de grafos.
- Desarrollo de una aplicación que utilice estas tecnologías para procesar un repositorio de datos de vuelo.

Abstract

This Final Degree Project (TFG) will focus on the study of graphs on Big Data, specifically about flight data. The organization of the data in the form of graphs has become essential in the field of Big Data. Although classic tools such as MapReduce and Spark RDD/DataFrame are widely used, they do not offer direct support for the graph model, leading to programmers to adapt to the features of each tool.

Specific tools such as Pregel, Apache Giraph, GraphX, and Spark's GraphFrames have emerged to overcome this limitation. There has also been a change in database technology towards graph-oriented databases, such as Neo4j.

The main objective of this work is to make a practical analysis of these tools by applying them to the processing of large volumes of flight data. In order to achieve this, an application will be designed to process a flight data repository using these specialized platforms. The following objectives are proposed:

- Analyse the Big Data tools oriented towards graph processing.
- Develop an application that uses these technologies to process a flight data repository.

Tabla de contenidos

Tabla de Ilustraciones

1 Introducción

En este capítulo se expondrán los problemas que representa el Big Data actualmente y qué soluciones se han ido desarrollando para afrontarlos. Se dará la motivación para este TFG, los objetivos a cumplir y su planificación

1.1 Marco Teórico

1.1.1 Introducción al Big Data

El desarrollo tecnológico del BigData es reconocido por una de las áreas de la Tecnología de la Información más importantes, y está evolucionando a velocidades muy altas debido en parte a las redes sociales y al Internet Of Things (IoT). Esto está provocando que las compañías de todo el mundo tengan que adoptar medidas contra esto para poder mantenerse en el mercado [1].

Se crean 2.5 exabytes por día, siendo el 90% de los datos generados en los últimos dos años [2]. Para entender un poco qué significa esa cantidad se conocen los siguientes datos:

- Se hacen 40.000 búsquedas en Google por segundo.
- Se envían 16 millones de mensajes de texto por minuto.
- 156 millones de correos por minuto, contando con 9 mil millones de usuarios.
- Para el final de 2017, se calcula que se sacaron 1,2 billones de fotos.

Por este tipo de razones es que es tan importante encontrar y mejorar las tecnologías existentes para poder analizar estos datos y conseguir información.

Actualmente el Big Data se puede separar en las 5Vs, que son Volumen, Velocidad, Variedad, Veracidad y Valor. [1]

El Volumen se refiere a la cantidad masiva de datos generados cada segundo, desde las redes sociales hasta las transacciones comerciales. Un volumen tan grande significa que no es posible tenerlo solamente en un equipo y es necesario el uso de más de uno para poder tratarlos.

La Velocidad se refiere a la rapidez con la que se generan y procesan estos datos. Como ya vimos anteriormente, la cantidad de datos que se generan por cada día es gigantesca. Para mantener una alta velocidad se requiere de una infraestructura robusta y tecnologías que vayan a la par.

La Variedad explica que hay diferentes tipos de datos, estructurados, no estructurados y semiestructurados. Esto implica que las tecnologías tengan la capacidad de procesar diferentes tipos de datos.

La Veracidad tiene relación con la calidad y precisión de los datos. Es crucial que los datos sean precisos y fiables para obtener insights válidos. Con un conjunto de datos sin tratar, es complicado evitar conclusiones erróneas que puedan resultar costosas.

Y la última Valor, se refiere a la utilidad e importancia de los datos, cuáles pueden ser más beneficiosos de todos los que hay disponibles. El valor de los datos radica en cómo pueden ser utilizados para mejorar la toma de decisiones, optimizar procesos y generar nuevas oportunidades.

1.1.2 Creación de Hadoop

Habiendo visto los problemas y las complejidades del Big Data, nos podemos remontar a 2005 cuando se desarrolló Hadoop[3] como una de las primeras soluciones. Es un marco desarrollado por Doug Cutting, escrito en Java y distribuido bajo la Licencia de Apache que se encargó principalmente del análisis y procesamiento en Big Data. Aborda tres desafíos:

- **Volumen:** Se proporciona un marco en el que se puede escalar horizontalmente para abordar un gran volumen de datos.
- **Velocidad:** Maneja la muy alta velocidad de los datos entrantes desde Sistemas muy grandes.
- **Variedad:** También admite cualquier variedad de datos no estructurados.

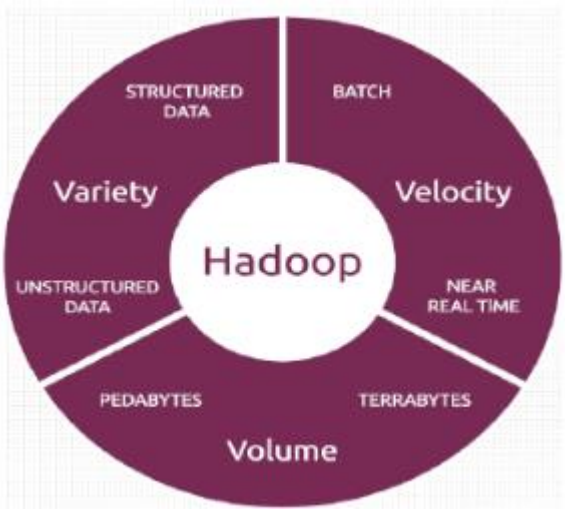

Figure 1: Hadoop en Big Data

Hadoop proporciona un marco de programación llamado MapReduce y un sistema de archivos distribuido, HDFS. Transforma hardware de baja calidad en un servicio que almacena grandes cantidades de datos de manera confiable y también los procesa de manera eficiente a través de una distribución masiva.

Una de las características clave de Hadoop es la partición tanto de datos como de computación en varios hosts, pudiendo ejecutar aplicaciones en paralelo cerca de los datos necesarios. Brinda también redundancia y confiabilidad, lo que significa que, al perderse una máquina debido a alguna falla, automáticamente replica los datos sin que el operador lo note.

Nos centraremos ahora más en la parte de la computación de hadoop, que introdujo este nuevo modelo de programación llamado MapReduce. Funciona en tres pasos principales: Map, Shuffle y Reduce. En el paso de Map, los datos

de entrada se dividen en fragmentos más pequeños y se procesan independientemente en los distintos nodos. Luego, en el paso Shuffle, los resultados intermedios se ordenan y se transfieren entre los nodos. Finalmente, en la etapa Reduce, los resultados intermedios se combinan y se generan los resultados finales.

Esta nueva manera de poder usar varios nodos para poder procesar datos llevó a que se creen varias herramientas similares, como puede ser Apache Spark.

1.1.3 Grafos en Big Data

Otra manera de poder representar los datos y sus relaciones entre sí es mediante grafos. Es una abstracción flexible en la que es posible describir las relaciones entre objetos discretos. Esta manera de pensar que parece ser correcto en varios casos de uso presenta varios problemas y desafíos.[4]

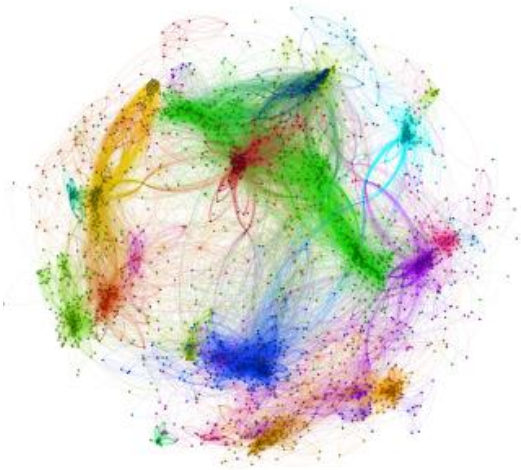

Figure 2: Representación de un grafo en Big Data

El principal problema que nos encontramos es cóunmo hacer la partición en varios equipos para un grafo, ya que no es tan fácil "agrupar" los datos, ya que ahora presentan relaciones entre ellos. Este problema se divide en dos, la partición de los datos almacenados, y la partición del cómputo en esos datos.

Para el primer problema hay dos soluciones principales:

- **Edge-Cut: En** esta solución se agrupan los vértices en grupos distintos de aproximadamente el mismo tamaño. Lo que se busca al separarlos es que haya la mínima cantidad de relaciones entre los equipos separados.

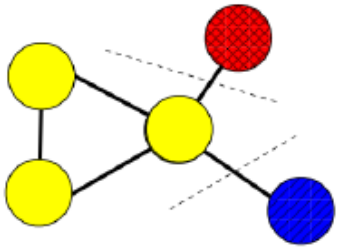

Figure 3: Edge-Cut

- Vertex-Cut: En esta otra solución se hace lo opuesto, separar las relaciones en distintos grupos de casi el mismo tamaño. El objetivo ahora sería encontrar el mínimo número de vértices replicados.

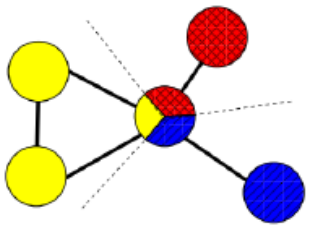

Figure 4: Vertex-Cut

Para el problema de cómputo, primero se evaluó el uso de las herramientas anteriores como MapReduce, pero presentan muchas limitaciones:

- Está diseñado para trabajar con datos en forma de pares clave-valor y seguir un patrón de procesamiento de dos fases: Map y Reduce. Sin embargo, el procesamiento de grafos requiere de operaciones más complejas y flexibles, como la exploración de vecinos, cálculo de caminos más cortos y la detección de comunidades, que no se adaptan a este modelo de programación.
- Generalmente los algoritmos de procesamiento de grafos son iterativos, esto significa que al intentar usarlo en MapReduce hay una comunicación excesiva entre nodos, ya que cada nodo debe comunicarse entre sí para intercambiar información.
- Siguiendo en la misma línea de que los algoritmos son iterativos, MapReduce procesa los datos en lotes, lo que puede suponer un uso ineficiente de recursos del sistema. Esto es debido a que los recursos pueden quedar inactivos mientras se espera la finalización de un lote de procesamiento.

Por abordar estas limitaciones se crearon otras herramientas específicas para el procesamiento de grafos. Se centran en optimizar la eficiencia del procesamiento de grafos, mientras se mantiene la escalabilidad y tolerancia a fallos. En este proyecto se analizarán estas herramientas y se usarán en un caso de uso específico.

1.2 Motivación

La motivación detrás de este proyecto es la importancia cada vez mayor del Big Data en el mundo de la tecnología y los negocios. El Big Data se ha convertido en un campo fundamental que está cambiando la forma en que las empresas recopilan, gestionan y analizan datos para tomar decisiones estratégicas.

El Big Data está en constante crecimiento y evolución debido a la gran cantidad y variedad de datos que se generan en todos los sectores de la sociedad. Desde las redes sociales hasta el Internet Of Things, el Big Data está presente en todas partes y tiene un impacto en todos los aspectos de nuestra vida diaria. En el mundo empresarial, aquellas empresas que pueden aprovechar al máximo el

potencial del Big Data tienen una ventaja competitiva. El análisis de datos avanzado les permite comprender mejor a sus clientes, mejorar sus operaciones, encontrar nuevas oportunidades de mercado y tomar decisiones informadas en tiempo real.

Realizar este proyecto de Fin de Grado en Big Data me daría las habilidades y el conocimiento necesarios para destacar en un campo altamente demandado como el análisis de datos a gran escala. Con una comprensión profunda de las tecnologías y metodologías utilizadas en el procesamiento y análisis de Big Data, estaría preparado para enfrentar los desafíos del mundo laboral actual y futuro, y contribuir al éxito de cualquier organización en la que trabaje.

1.3 Objetivos

El objetivo principal de este Trabajo de Fin de Grado (TFG) es realizar un análisis práctico de herramientas de Big Data orientadas al procesamiento de grafos, aplicándolas al análisis de datos masivos de vuelos. Este proyecto busca abordar la limitación que presentan herramientas clásicas como MapReduce y Spark RDD/DataFrame, las cuales no soportan de manera directa el modelo de grafos, forzando a los programadores a adaptar sus métodos al modelo específico de cada herramienta.

Para superar esta limitación, el TFG se enfocará en el uso de herramientas especializadas que ofrecen soporte directo para el procesamiento de grafos. Además, se explorará el uso de bases de datos orientadas a grafos, que se han desarrollado para abordar las complicaciones que se presentan cuando se usa un modelo relacional para el manejo de datos.

En resumen, los objetivos específicos del TFG son dos: primero, realizar un análisis de las herramientas de Big Data orientadas al procesamiento de grafos, y segundo, desarrollar una aplicación que utilice estas tecnologías para procesar un repositorio de datos de vuelos.

1.4 Planificación

El trabajo se estructurará en diferentes fases, dedicando 20 horas a la familiarización con las tecnologías, 100 horas al diseño de la aplicación, 140 horas a la implementación de esta y 37 horas a la confección de la documentación. El desglose se puede ver representado en el siguiente diagrama de Gantt separado por semanas.

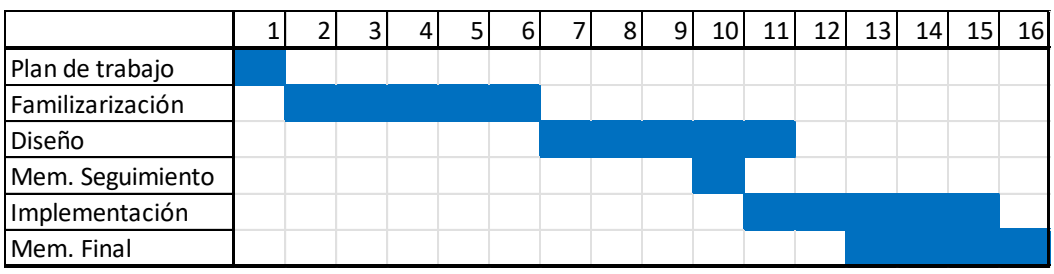

Figure 5: Diagrama de Gantt

1.5 Estructura del trabajo

Este documento constará de las siguientes secciones:

- **Estado del Arte:** En esta sección se describirán las tecnologías utilizadas para el desarrollo de la aplicación.

- **Arquitectura:** Se describirá la arquitectura de la solución propuesta, abarcando desde la Extracción, Transformación y Carga (ETL), hasta la visualización de los datos.

- **Implementación:** En esta parte se detallará el caso de uso específico, las fuentes de datos, configuración del entorno y los resultados de las pruebas realizadas sobre la aplicación.

- **Conclusiones:** Se resumirá el trabajo realizado, incluyendo los problemas encontrados, el cumplimiento de los objetivos y las posibles líneas futuras de desarrollo.

- **Análisis de Impacto:** Se evaluará el impacto potencial de los resultados y desarrollo del TFG en los diferentes Objetivos de Desarrollo Sostenible (ODS).

2 Estado del Arte

2.1 Neo4j

Neo4j es una base de datos orientada a grafos, diseñada específicamente para almacenar, manejar y consultar datos organizados en estructuras de grafos. A diferencia de las bases de datos relacionales tradicionales, brinda una mayor eficacia a la hora de realizar consultar y modelar relaciones entre los datos. [5]

En esta herramienta, la información se organiza a base de nodos, relaciones y propiedades.

Los nodos son las entidades en el grafo. A estos se les puede poner un nombre, llamado "Label", representando los diferentes roles en su dominio. Las características del nodo se representan mediante propiedades, esto incluye desde el nombre hasta los metadatos.

Las relaciones son unidireccionales, y hacen la conexión entre dos nodos. Siempre tiene que haber un nodo inicial y un nodo final, y estos también pueden tener propiedades.

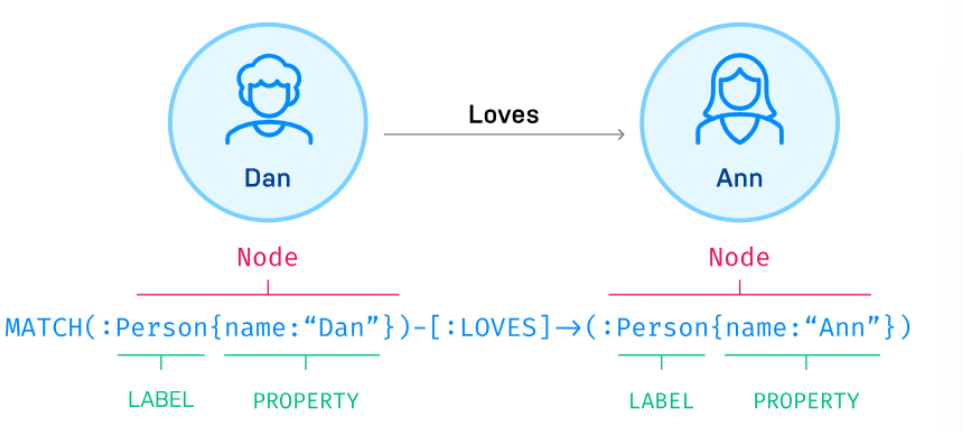

Figure 6: Lenguaje de Cypher

Neo4j está disponible como una aplicación en la nube a través de AuraDB, o se puede desplegar por el propio desarrollador. Provee transacciones ACID, clustering y tolerancia a fallos, además está desarrollado en Java y Scala, y es de código abierto.

Para la consulta de Grafos utiliza un lenguaje de consulta específico llamado Cypher, diseñado para trabajar con grafos. Este permite expresar consultas complejas de una manera muy inteligente, aprovechando la estructura del grafo para realizar las operaciones. Este lenguaje sigue la siguiente sintaxis:

```
(nodes)-[:CONNECT_TO]→(otherNodes)
```

Normalmente las consultas en Cypher son más concisas que su equivalente en SQL. Por ejemplo, se considera las siguientes dos consultas, donde se quiere buscar el nombre del actor que actuó en la película “The Matrix”

```
SELECT actors.name
FROM actors
     LEFT JOIN acted_in ON acted_in.actor_id = actors.id
     LEFT JOIN movies ON movies.id = acted_in.movie_id
WHERE movies.title = "The Matrix"
```

Y su contraparte en Cypher:

```
MATCH (actor:Actor)-[:ACTED_IN]->(movie:Movie {title: 'The
Matrix'})
RETURN actor.name
```

Por último, este permite mediante la librería Neo4j Browser hacer consultas interactivas y visualización de la base de datos. Esto es muy útil a la hora de realizar pruebas para comprobar que los datos se hayan guardado de manera correcta.

2.2 Apache Spark

Según IBM, “Apache Spark es un motor de procesamiento de datos de código abierto y de alta velocidad para aplicaciones de machine learning e inteligencia artificial, respaldado por la mayor comunidad de código abierto en el ámbito del Big Data” [6].

Este motor de procesamiento de datos fue diseñado para el análisis de grandes volúmenes de datos. Se ha convertido en una herramienta unificada popular en el campo del Big Data debido a su alta velocidad y facilidad de uso.

La arquitectura de una aplicación de Spark consiste en un programa principal y un conjunto de procesos ejecutores, como se muestra en la Figura 4.

El programa principal es el corazón de la aplicación, es quien ejecuta la función main(). Es el responsable de mantener la información de la aplicación, responder a los datos de entrada y de la distribución de la carga de trabajado a través de los ejecutores

Luego están los ejecutores, quienes son los que realizan el trabajo. Estos son manejados por el driver y por el Cluster Manager.

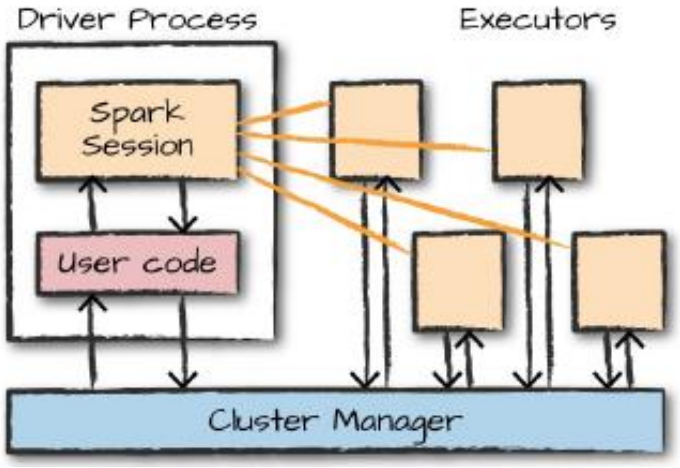

Figure 7: Distribución de Spark

La abstracción fundamental en Spark son los Resilient Distributed Dataset (RDD), que representan una colección distribuida de elementos que se pueden procesar en paralelo. Este está dividido en un número de particiones atómicas que son guardadas en diferentes nodos del clúster. Aunque en las últimas versiones de Spark no son comúnmente usadas, todo código que utiliza Spark se termina compilando como si fuera una RDD.

En Apache Spark, todo flujo de datos (llamado Job), se puede representar mediante un grafo directo acíclico DAG. Este está compuesto por numerosas fuentes de datos, operaciones y conexiones entre sí como se muestra en la Figura 6.

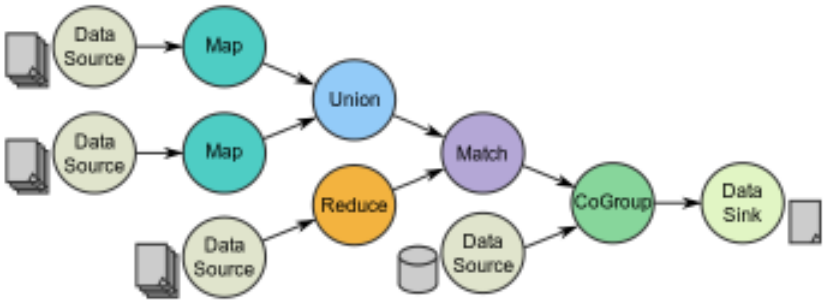

Figure 8: DAG (grafo directo acíclico)

Hay dos tipos diferentes de operaciones, transformaciones y acciones. Las transformaciones son aquellas con construyen la lógica del Job. Toda transformación es retardada, es decir, que no se ejecuta directamente. Las acciones son las que disparan/ejecutan el plan. Se puede decir que le dicen a Spark que compute un resultado con una serie de transformaciones. Estas puedes ser para visualizar, coleccionar en un objeto del lenguaje que se esté usando, o para escribir como salida.

En adición a todo esto, Spark ofrece varias APIs o librerías que extiendes sus capacidades a otro nivel. Spark en su conjunto se compone de Spark Core, Spark SQL, Spark Streaming para datos en movimiento, Spark MLib que brinda habilidades de machine learning y por último Spark GraphX para

procesamiento y análisis de grafos. Spark está diseñado para utilizarse con el lenguaje de programación Scala, pero también se puede desarrollar con Java, Python y R.

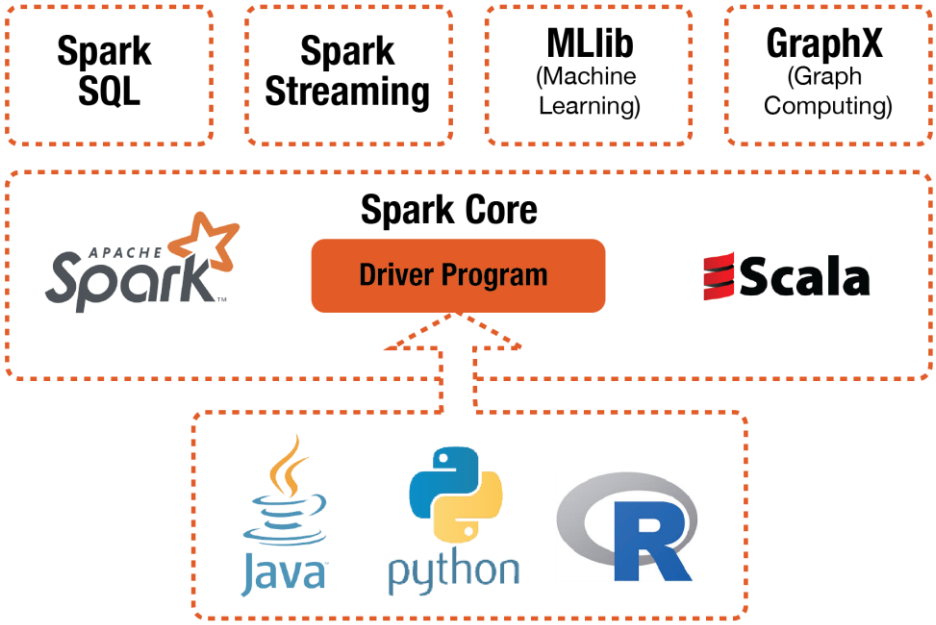

Figure 9: APIs de Spark

2.2.1 SparkSQL

Spark SQL es un módulo de Apache Spark diseñado para trabajar con datos estructurados y semiestructurados. Combina las capacidades de procesamientos distribuidos de Spark con un motor de consultas SQL. Esto permite a los usuarios interactuar con los datos usando tanto lenguaje de consulta SQL como Scala, Java, Python y R.

Esta solución nació porque los RDDs no sabes nada sobre el esquema que tienen los datos. Por ello, se crearon dos nuevas colecciones estructuradas, DataFrames y Datasets [7]. Estas básicamente son colecciones distribuidas parecidas a tablas con columnas y files bien definidas.

Un DataFrame es el equivalente a una tabla en una base de datos relacional. Esta tiene un esquema, en la cual están definidos los nombres de las columnas y sus tipos.

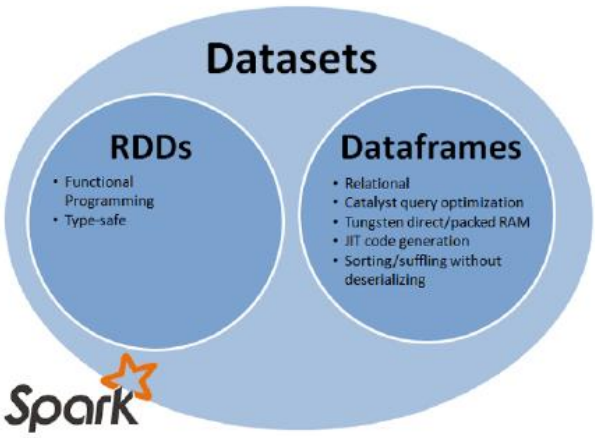

Figure 10: Diferencia entre Datasets, Dataframes y RDDs

Los DataSets son una extensión de los DataFrames que combinan las ventajas de la seguridad en los tipos de datos en un lenguaje de ejecución con las optimizaciones del motor de ejecución de Spark. Esto significa que cuando hacemos el casteo de un tipo de datos, los DataSets puedes ver el error en tiempo de compilación, mientras que los DataFrames en tiempo de ejecución. Para poder usarlos, se tiene que crear un objeto con los tipos de datos que necesites, y esos objetos serán las filas de la tabla de un DataSet.

Spark SQL puede leer datos de muchas fuentes. Algunas de ellas son archivos JSON, Parquet, HDFS, Apache Hive, bases de datos JDBS, archivos CSV, y muchas más. Esto permite que se puede integrar de manera muy flexible con diferentes sistemas de almacenamiento. También incluye integraciones para Kafka, que es una herramienta muy útil para una aplicación en Big Data.

2.2.2 GraphX

GraphX [8] se creó para proporcionar una herramienta unificada dentro del ecosistema de Spark, permitiendo a los usuarios realizar tanto análisis de grafos como el procesamiento de datos general en una sola plataforma.

Esta herramienta utiliza la abstracción de los RDD de Spark, combinando así operaciones de grafos con las transformaciones y acciones de Spark. Esto permite que se pueda realizar un análisis de los datos más completo y sin problemas de integración con otras herramientas.

La abstracción principal de GraphX es el "Property Graph", que consiste en un par de RDDs, uno para los vértices y otro para las aristas. Cada una de estas colecciones puede tener atributos asociados, pudiendo representar un grafo de una manera muy sencilla.

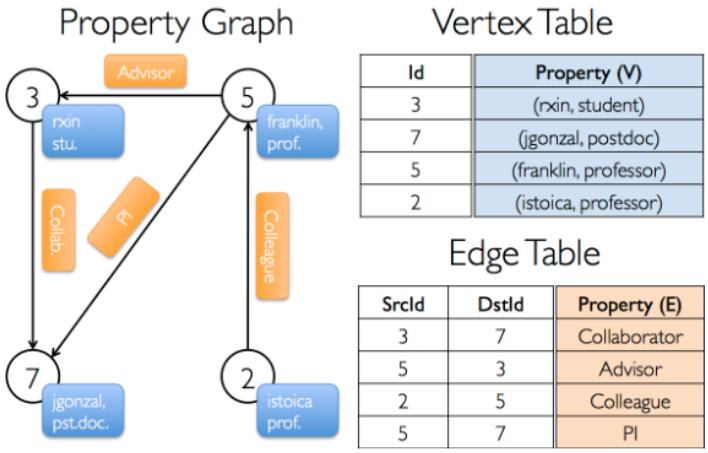

Figure 11: Visualización de un property Graph

Junto con esta abstracción, este incluye una implementación de Pregel, un modelo de computación diseñado por Google, que permite realizar algoritmos iterativos en grafos. Pregel facilita la escritura de algoritmos de grafos mediante lo que se llama "paso de mensajes".

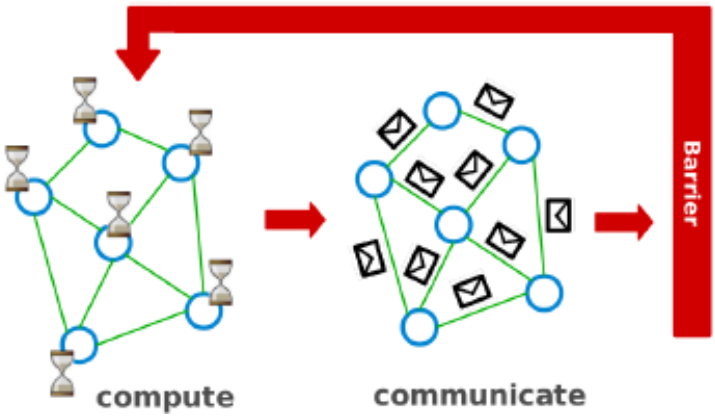

Figure 12: Representación de algoritmo de paso de mensajes

Los algoritmos de paso de mensajes se ejecutan en una serie de iteraciones llamadas supersteps. Se realizan dos pasos, primero los vértices leen los mensajes de entrada, hacen el cómputo y cambian su estado, y luego estos se comunican con el resto enviando un mensaje como se muestra en la Figura 11.

GraphX soporta una variedad de algoritmos que utilizan el paso de mensajes, algunos de estos son el PageRank, Connected Components, Shortest Paths, Triangle Count y más.

2.3 GraphFrames

GraphFrames es una API de Apache Spark desarrollado por DataBricks. Este fue diseñado para extender las capacidades de GraphX, integrando las ventajas de los DataFrames y DataSets. Análogamente es como la diferencia entre RDDs y DataFrames, permite una mayor flexibilidad, facilidad de uso y más casos de uso.

A diferencia de GraphX, este permite realizar Consultas SQL directamente sobre los grafos, permitiendo el uso de cualquier tipo de datos compatibles con los DataFrames.

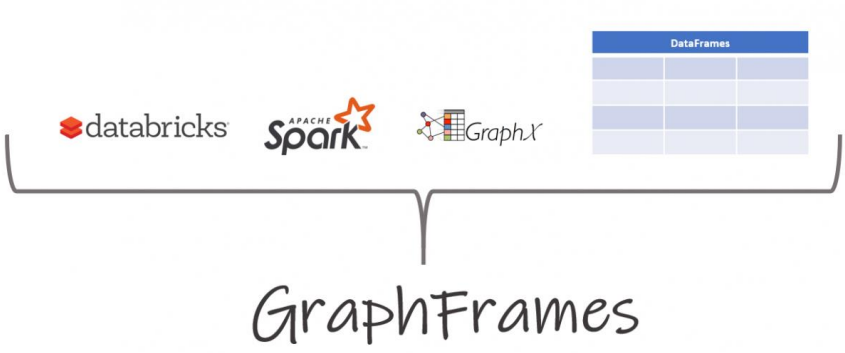

Figure 13: GraphFrames [9]

Este también ofrece nuevas funcionalidades, como las operaciones “motif finding” y “Búsqueda en anchuras”. Además de permitir la integración con MLib y una mejor manera de manejar los valores Nulos.

Similarmente a GraphX, este utiliza una nueva abstracción para representar a un grafo llamada “GraphFrame”. Este se compone de dos DataFrames, uno para los vértices y otro para las aristas.

En conclusión, GraphFrames con su capacidad de realizar consultas SQL y un mejor manejo de los datos hacen de este una de las mejores opciones para el análisis de grafos en el ecosistema de Apache Spark.

3 Arquitectura

La arquitectura de la aplicación planteada se muestra en la Figura 14. Está dividida en tres partes fundamentales:

- ETL (Extract, Transform, Load).
- Análisis
- Visualización

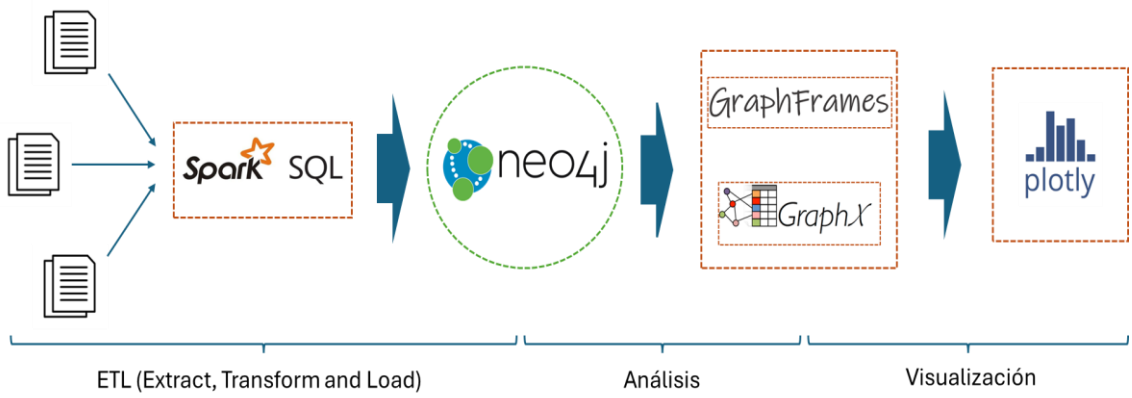

Figure 14: Arquitectura de la aplicación

3.1 ETL

La arquitectura muestra varios documentos que representan las fuentes de datos iniciales. Estos pueden ser archivos de formatos diferentes, como CSV, JSON, y más que contienen datos principalmente estructurados.

La primera parte es la extracción de los datos, en esta se utiliza Spark SQL. Este es muy útil para datos estructurados y puede inferir el esquema de los archivos.

La segunda parte es la transformación de los datos. Esta parte es necesaria para realizar diversas operaciones como limpieza de datos, filtrado, agregación y unción de tablas. Es necesario hacer esta limpieza para que el análisis posterior.

Finalmente, la carga de datos, luego de que se haga la transformación, los datos procesados se cargarán en una base de datos orientada a grafos llamada Neo4j.

En conclusión, se utiliza un intermediario entre la base de datos y los datos en sí por dos principales razones. La primera para que el proceso sea más escalable y flexible, y la segunda para que poder limpiar los datos antes de ingresarlos directamente a la base de datos.

3.2 Análisis

En esta etapa se utilizan dos herramientas principales, GraphX y GraphFrames. Normalmente se utilizaría únicamente GraphFrames ya que incluye los algoritmos de grafos de GraphX. El problema que ocurre es que la versión que se requiere de Spark para conectarse a Neo4j no es compatible con parte de la API de GraphFrames. Se podría haber utilizado GraphX solamente, pero hay varias funciones de GraphFrames que, sí son compatibles y son más eficaces, por lo que se le dio un uso distinto a cada una.

Entonces, como primer paso, la aplicación extrae los datos de Neo4j y se lo envía a estas dos APIs para que realicen los algoritmos que les pida el usuario.

GraphX se utilizará para realizar todos los algoritmos que sean por paso de mensajes. Como se dijo anteriormente, estos algoritmos como el PageRank y detección de comunidades GraphFrames no puede realizarlos en esta versión.

GraphFrames en cambio realizará todos los análisis que se asemejen a consultas SQL. Por ejemplo, puede ser buscar el vuelo más largo, y también que rutas de vuelo no son directas.

3.3 Visualización

Plotly es una biblioteca de visualización de datos que permite crear gráficos interactivos y personalizados. En esta etapa, los resultados de los algoritmos realizados en la anterior etapa de análisis se verán representados.

Se permiten crear una gran cantidad de gráficos, que incluye gráficos de líneas, de dispersión, barras, de red, etc.

Esta etapa es útil para facilitar la interpretación de los resultados para los usuarios, permitiendo una mejor toma de decisiones.

Además de Plotly, aunque no esté representado en la arquitectura, Neo4j presenta una parte para la visualización de datos en los que se pueden ver en forma de grafos. Permite la interacción con los grafos y es muy útil para ver la imagen completa de cómo está representada la información.

3.4 Beneficios

Esta arquitectura se ha hecho de esta manera luego de considerar ciertos aspectos muy importantes para cualquier aplicación en el ámbito del Big Data

Al utilizar Apache Spark y sus componentes, la arquitectura presenta una gran capacidad para manejar grandes volúmenes de datos de manera eficiente. Es una solución que se puede escalar para cuando se requiere el uso de otras fuentes de información o se amplíe la base de datos.

Esta solución es muy flexible, fundamentalmente en la primera parte de ETL. En esa etapa es posible aumentar el número de fuentes de datos, y se puede cambiar el tipo sin problema alguno. Solamente haría falta cambiar algunos detalles para que los acepte.

Además, al utilizar Plotly y Neo4j permite que la interacción con la aplicación sea muy interactiva y fácil para comprender los resultados. También es bastante flexible a la hora de poder compartir estos resultados a otra persona.

En resumen, esta arquitectura integra varias tecnologías de Big Data y análisis de grafos, proporcionando una solución robusta y escalable para el procesamiento y análisis de datos estructurados.

4 Implementación

4.1 Caso de Uso

El objetivo de esta aplicación es realizar un análisis de los datos de vuelo utilizando la arquitectura previamente descrita. Esta aplicación tiene como fin manejar y analizar datos de manera eficiente, y extraer información que puede ser útil principalmente a los aeropuertos para mejorar sus operaciones y servicios.

La elección de este caso de uso se debió a dos principales factores, que pueda generar beneficios y un impacto en la sociedad, y que se justifique el uso de tecnologías de Big Data.

Los beneficios que puede llegar a brindar esta solución son varios:

- Optimización de rutas: Identificación de rutas de vuelo más eficientes, reduciendo el tiempo de vuelo y consumo de combustible. Puede ayudar también a reducir costos operativos y mejorar la puntualidad.
- Análisis de Conexiones: Evaluación de la efectividad de la red de vuelos y aeropuertos que existe, identificando puntos de congestión y oportunidades de mejora. De esta manera se puede reducir los tiempos de conexión para ser más conveniente para los pasajeros.
- Reducción de Impacto Ambiental: Al ser más eficiente y optimizado, se reduce el impacto que se tiene en el ambiente.
- Mejora en la experiencia del pasajero.

Este caso de uso es bueno para probar en Big Data por varias razones. Los datos de vuelo son generados en grandes volúmenes diariamente, incluyendo registros de vuelos, datos meteorológicos y más. También resultan ser de diversas fuentes y diferentes formatos.

4.2 Fuente de datos

Se han recopilado los datos de la página de Bureau of Transportation and Statistics (BTS). Es una agencia del Departamento de Transporte de los Estados Unidos que se encarga de recopilar y hacer análisis sobre datos sobre el transporte para mejorar el sector.

Los datos de vuelos se han recogido en un archivo CSV de 2018 que contiene todos los vuelos de Estados Unidos. Este archivo de datos incluye una gran variedad de datos como se mostrará en el capítulo 4.4.

4.3 Configuración del entorno

Para configurar el entorno habrá primero que instalar las herramientas necesarias, para luego poder configurar dos partes principales, del lado de la aplicación y del lado de la base de datos. Cabe destacar que la configuración y la implementación está hecha toda en su conjunto en local.

4.3.1 Configuración de la aplicación

Como prerrequisito para esta sección primero se debe instalar Java, Scala y Spark. Para Java se ha utilizado la versión Java 1.11 JDK, para Scala la 2.12, y para Spark la 3.5.

Luego de tener las correspondientes instalaciones hechas, para poder ejecutar aplicaciones autocontenidas en Scala se necesitará de un archivo SBT llamado build.sbt. Para poder utilizarlo también se tendrá que realizar la instalación de ella que es un poco más complicado que lo anterior, por lo que en la figura 15 se muestra cómo.

```
echo "deb https://repo.scala-sbt.org/scalasbt/debian all main" | sudo tee /etc/apt/sources.list.d/sbt.list
echo "deb https://repo.scala-sbt.org/scalasbt/debian /" | sudo tee /etc/apt/sources.list.d/sbt_old.list
curl -sL "https://keyserver.ubuntu.com/pks/lookup?op=get&search=0x2EE0EA64E40A89B84B2DF73499E82A75642AC823"
     | sudo apt-key add
sudo apt-get update
sudo apt-get install sbt
```

Figure 15: Instalación de SBT

Ahora se deberá configurar el archivo build.sbt, en esta tendrá que figurar la versión de Scala:

```
3   ThisBuild / scalaVersion := "2.12.12"
```

Así como también todas las librerías que se tendrás que utilizar:

```
11  resolvers += "SparkPackages" at "https://repos.spark-packages.org/"
12
13  val sparkVersion = "3.5.1"
14  libraryDependencies ++= Seq(
15    "org.apache.spark" %% "spark-core" % sparkVersion,
16    "org.apache.spark" %% "spark-sql" % sparkVersion,
17    "org.apache.spark" %% "spark-graphx" % sparkVersion,
18    "graphframes" % "graphframes" % "0.8.1-spark3.0-s_2.12",
19    "org.neo4j" %% "neo4j-connector-apache-spark" % "5.3.0_for_spark_3",
20    "org.plotly-scala" %% "plotly-almond" % "0.8.2"
21  )
```

En ella podemos ver que la versión de Spark es la “3.5.1”, y en la variable libraryDependencies se incluyen todas las librerías necesarias para el uso de esta aplicación.

Destaca una variable llamada resolvers que es utilizada para indicarle al archivo SBT otra fuente para buscar las librerías. Esto sucede porque algunas no se encuentras en las fuentes preestablecidas.

4.3.2 Levantamiento de Neo4j

Primero habrá que descargar la aplicación de Neo4j Desktop de la página principal. Se descargará un archivo que será el que se ejecute y se vea una pantalla similar a la figura 16.

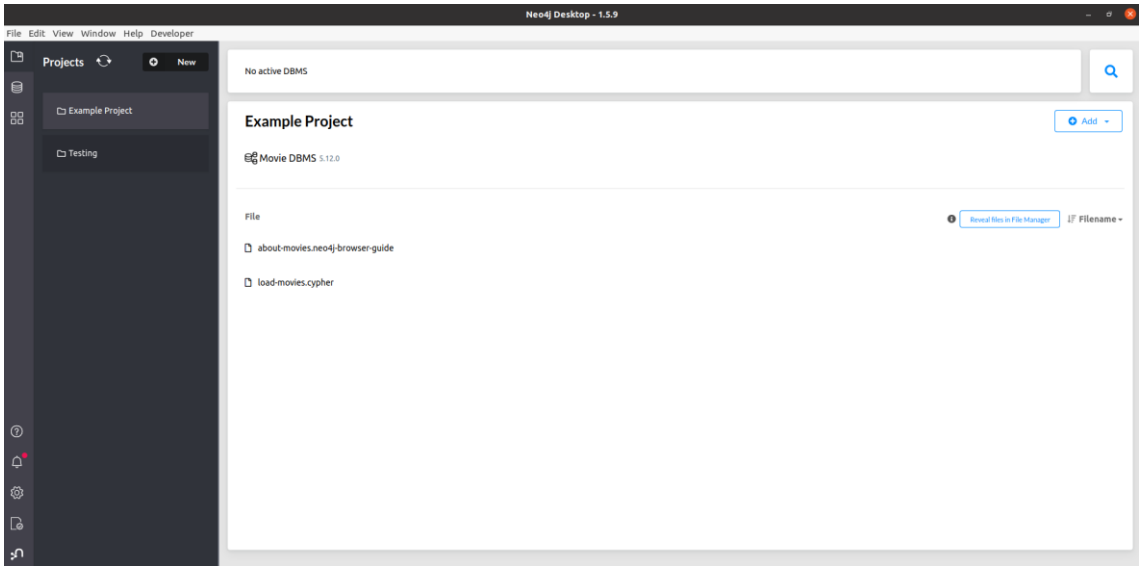

Figure 16: Home de Neo4j Desktop

Lo que se requiere para poder empezar a utilizarla es primero crear un proyecto nuevo, que es yendo a **New => Create Proyect.**

Luego se deberá crear una base de datos en ese proyecto para poder utilizarla, esto se hará en **Add => Local DBMS**, en donde se elegirá el nombre, contraseña y versión de esta.

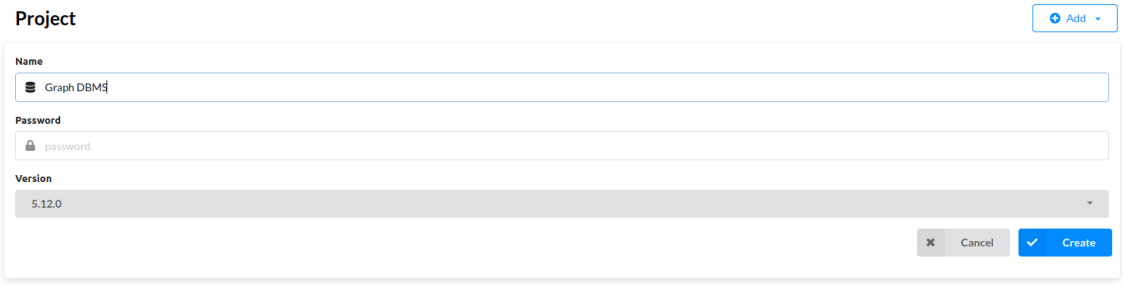

Figure 17: Creación de DMBS en Neo4j

Finalmente, para que la base de datos creada se puede iniciar, simplemente se toca el botón **Start**, y por defecto estará iniciada en Localhost:7687.

4.4 Transformación de los datos

Como primer paso hay que extraer los datos, que eso se realizó mediante Spark SQL el cual puede inferir el esquema de la tabla. Luego de haber extraídos los datos de vuelos se procedió a hacer un análisis de estos.

Se empezó por ver que columnas tenía:

```
root
 |-- FL_DATE: string (nullable = true)
 |-- OP_CARRIER: string (nullable = true)
 |-- OP_CARRIER_FL_NUM: integer (nullable = true)
 |-- ORIGIN: string (nullable = true)
 |-- DEST: string (nullable = true)
 |-- CRS_DEP_TIME: integer (nullable = true)
 |-- DEP_TIME: integer (nullable = true)
 |-- DEP_DELAY: integer (nullable = true)
 |-- TAXI_OUT: integer (nullable = true)
 |-- WHEELS_OFF: integer (nullable = true)
 |-- WHEELS_ON: integer (nullable = true)
 |-- TAXI_IN: integer (nullable = true)
 |-- CRS_ARR_TIME: integer (nullable = true)
 |-- ARR_TIME: integer (nullable = true)
 |-- ARR_DELAY: integer (nullable = true)
 |-- CANCELLED: integer (nullable = true)
 |-- CANCELLATION_CODE: string (nullable = true)
 |-- DIVERTED: integer (nullable = true)
 |-- CRS_ELAPSED_TIME: integer (nullable = true)
 |-- ACTUAL_ELAPSED_TIME: integer (nullable = true)
 |-- AIR_TIME: integer (nullable = true)
 |-- DISTANCE: integer (nullable = true)
 |-- CARRIER_DELAY: integer (nullable = true)
 |-- WEATHER_DELAY: integer (nullable = true)
 |-- NAS_DELAY: integer (nullable = true)
 |-- SECURITY_DELAY: integer (nullable = true)
 |-- LATE_AIRCRAFT_DELAY: integer (nullable = true)
 |-- Unnamed: 27: string (nullable = true)
```

Figure 18: Columnas de los datos de vuelos

Al entender un poco más qué datos tenemos, podemos decidir cuáles son los que nos interesan. Para esta primera versión de la aplicación solo se dejaron las columnas de “OP_CARRIER”, “ORIGIN”, “DEST” y “DISTANCE”, pero se puede ver que puede usarse varios atributos más.

Luego se revisó si las columnas restantes limpiadas tenían valores nulos, pero no tenían.

4.5 Carga de los datos en Neo4j

Para realizar la integración de Neo4j con Spark se debe utilizar un conector tanto para carga como descarga de datos en la base de datos. A este conector se le deben pasar los datos de la base de datos, como usuario, contraseña y dirección IP para que pueda realizar la conexión.

```
8   val url = "neo4j://localhost:7687"
9   val username = "neo4j"
10  val password = "contrase;a"
11
12  val spark = SparkSession.builder()
13    .config("neo4j.url",url)
14    .config("neo4j.authentication.basic.username", username)
15    .config("neo4j.authentication.basic.password", password)
16    .appName("Spark Input")
17    .master("local[*]")
18    .getOrCreate()
```

Figure 19: Conector a Neo4j

Ahora ya configurado el conector en Spark, se deberán extraer y transformar los datos como se explicó en el anterior capítulo. Luego se procederá a la carga de estos.

Para que la subida sea más rápida se tienen que subir primero los nodos y luego las relaciones. La otra posibilidad de hacerlo todo al mismo tiempo es posible, pero tiene un principal problema, y es que al hacerlo de esa manera no solo es más lenta la subida, sino que el código resulta ser más complicado y no sigue las pautas de buena programación.

Para cargar primero los nodos se tienen que extraer de los datos todos los aeropuertos existentes de las columnas ORIGIN y DEST y juntarlos todos en un solo DataFrame. Cuando se haya conseguido eso, se envían a través del conector utilizando la opción “labels” y “Overwrite”. Esto se hace de esa manera para que cuando se utilice de nuevo la carga de datos a la base de datos no se repitan aeropuertos.

```
    airports.write
      .format("org.neo4j.spark.DataSource")
      .mode("Overwrite")
      .option("labels","Airport")
      .option("node.keys","AIRPORT:Name")
      .save()
```

Figure 20: Cargando aeropurtos en Neo4j

Ahora se tendrán que subir las relaciones, las cuales parten del DataFrame principal y se utilizará la opción de “relationship”.

```
    flights.write
      .mode("Append")
      .format("org.neo4j.spark.DataSource")
      .option("batch.size","20000")
      .option("relationship", "Flight")
      // Use `keys` strategy
      .option("relationship.save.strategy", "keys")
      // Create source nodes and assign them a label
      .option("relationship.source.save.mode", "Overwrite")
      .option("relationship.source.labels", ":Airport")
      .option("relationship.source.node.keys", "ORIGIN:Name")
      // Create target nodes and assign them a label
      .option("relationship.target.save.mode", "Overwrite")
      .option("relationship.target.labels", ":Airport")
      .option("relationship.target.node.keys", "DEST:Name")
      // Map the DataFrame columns to relationship properties
      .option("relationship.properties", "OP_CARRIER:Carrier,DISTANCE:Distance")
      .save()
  }
}
```

Figure 21: Cargando vuelos en Neo4j

Luego de haber hecho la carga de datos se puede observar en Neo4j Browser como se subieron los datos. En la Figura 22 se puede observar cómo se visualizarían los aeropuertos y los vuelos en un subgrafo de toda la base de datos.

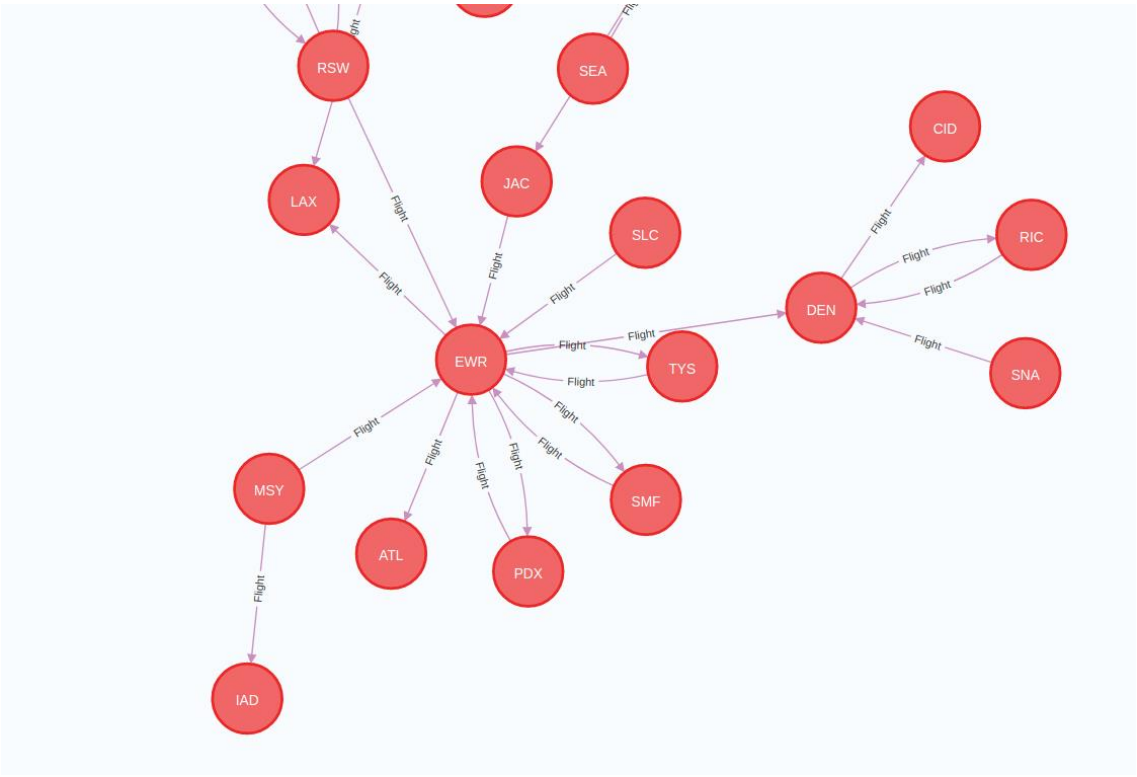

Figure 22: Subgrafo en Neo4j

También se puede seleccionar un Vuelo y ver qué propiedades tiene:

Relationship properties

Flight

<id>	0
Carrier	UA
DEST	DEN
Distance	1605.0
ORIGIN	EWR

Figure 23: Propiedades de un vuelo en la BBDD

4.6 Pruebas funcionales

Antes de empezar, se debe iniciar la aplicación en donde se podrá elegir qué análisis sobre vuelos se quieren ejecutar y visualizar. Cuando se ejecuta se puede observar en localhost:4040/jobs/ los Jobs que se van ejecutando, y se puede visualizar los DAG de los mismos. Se puede ver en la figura 24 que no hay todavía ningún Job iniciado.

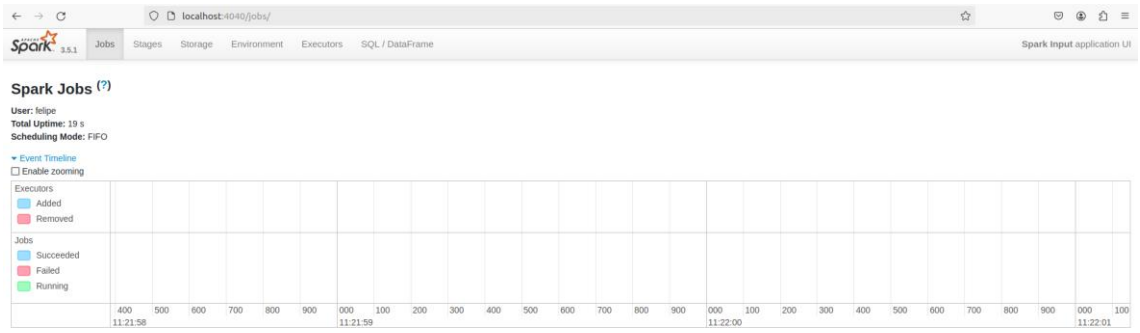

Figure 24: Página de Jobs en Spark

4.6.1 Rutas de vuelo más comunes

En esta prueba se ve cuáles son las rutas de vuelo más comunes entre dos aeropuertos. Esta función se hizo mediante GraphFrames, primero se debe crear la conexión con la base de datos y luego se leen los datos necesarios.

```
29    val flights = spark.read
30      .format("org.neo4j.spark.DataSource")
31      .option("relationship", "Flight")
32      .option("relationship.source.labels", ":Airport")
33      .option("relationship.target.labels", ":Airport")
34      .load()
35      .selectExpr("`source.Name` AS src","`target.Name` AS dst","`rel.Carrier` AS carrier","`rel.Distance` AS distance")
```

Figure 25: Lectura de datos de la BBDD

Luego de ello se crea otro DataFrame que contenga todos los aeropuertos contenidos en el primer DataFrame flights. Con estos dos DF se puede crear un grafo de GraphFrames, pudiendo ejecutar la prueba.

Cuando se ejecuta se envía el job a Spark y este hace la ejecución con los workers que tenga disponibles, que en este caso es solo uno por ser en local.

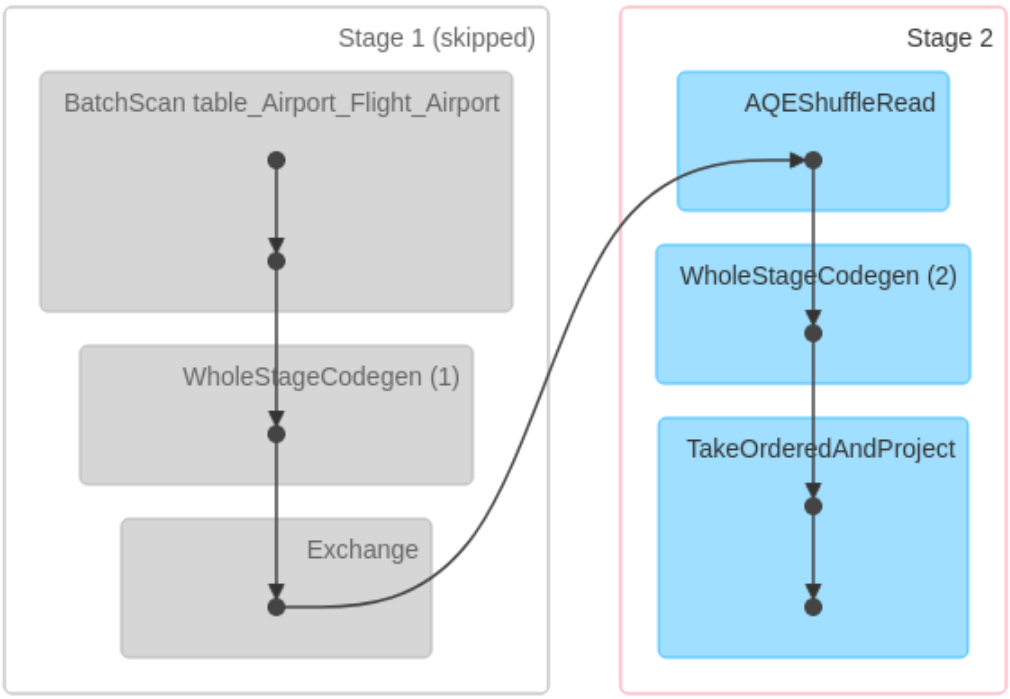

Figure 26: DAG de la primera prueba

Se puede observar cómo funciona Spark, donde separa el Job en dos Stages y hace la ejecución. El Stage de la izquierda fue salteado porque al inicio de la aplicación fue ejecutado, por lo que no es necesario repetirlo. Abajo en la figura 27 se muestran los resultados.

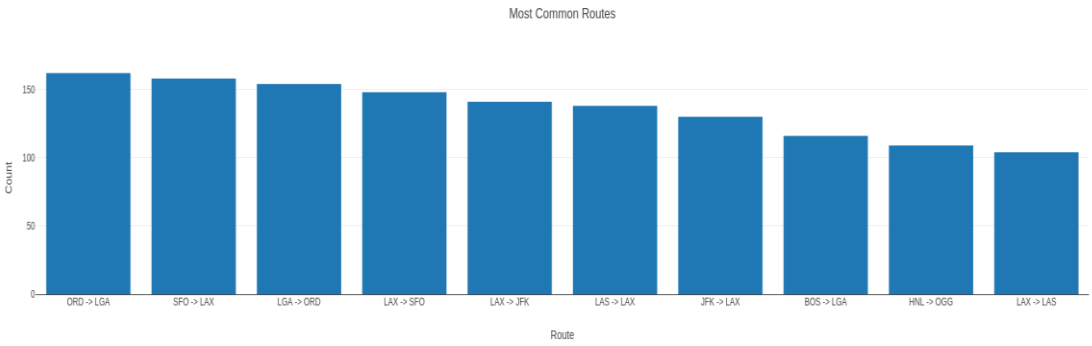

Figure 27: Resultado prueba 1

4.6.2 Aeropuertos con más conexiones

Este análisis también es realizado con GraphFrames, y sigue los mismos pasos que la anterior.

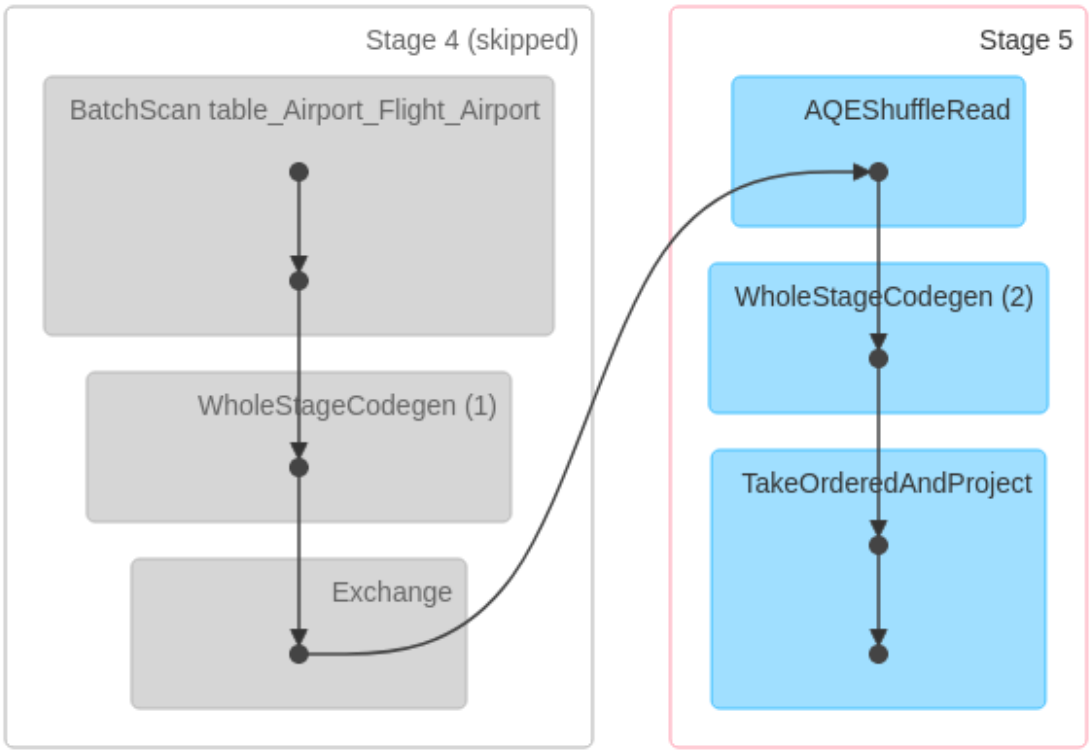

Figure 28: DAG prueba 2

Se puede ver que el DAG en este es idéntico al anterior, esto es porque las funciones para realizarlo se basan en lo mismo. Los resultados son diferentes en cambio:

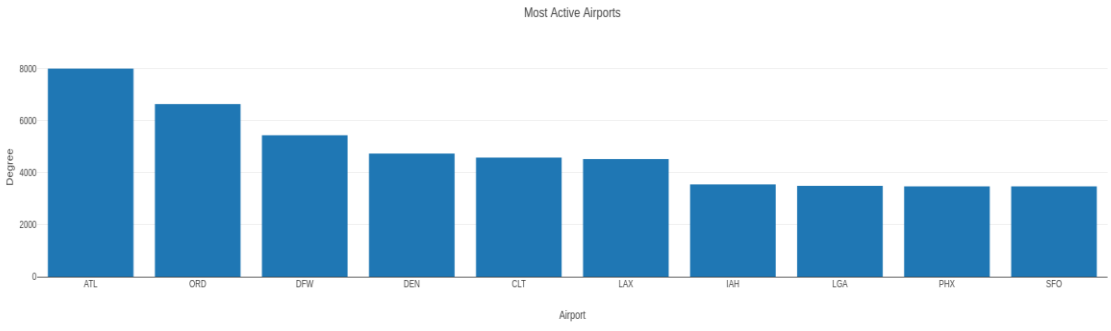

Figure 29: Resultados prueba 2

4.6.3 Aeropuertos más importantes por PageRank

Este análisis es distinto a los anteriores, utiliza GraphX debido que es un algoritmo de paso de mensajes. La única diferencia para ejecutarlo es que hay que crear de manera distinta el grafo, ya que se debe hacer con dos RDD en vez de DataFrames.

```
// Convertir el DataFrame de vertices a un RDD de tuplas (id, name)
val verticesRDD = airportsDF.rdd.map(row => (row.getAs[Long]("<id>"), row.getAs[String]("Name")))

// Convertir el DataFrame de edges a un RDD de Edge con los atributos que necesites
val edgesRDD = flightsDF.rdd.map(row => Edge(
  row.getAs[Long]("<source.id>"),
  row.getAs[Long]("<target.id>"),
  (row.getAs[String]("rel.Carrier"), row.getAs[String]("rel.Distance"))
))

val graph = Graph(verticesRDD,edgesRDD)
```

Figure 30: Crear grafo en GraphX

Como es un algoritmo iterativo y no de tipo consulta como lo son los que ejecutamos antes, se requieren de muchos Jobs como se ve en la figura 31.

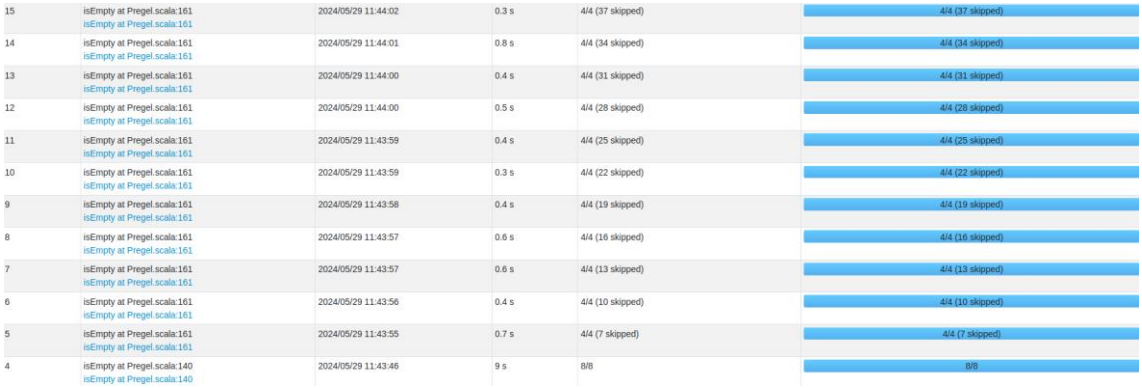

Figure 31: Jobs para ejecutar un PageRank

Y al ser tantos Jobs dificulta la visualización del DAG, por lo que solo se puede enseñar una parte en la figura 32.

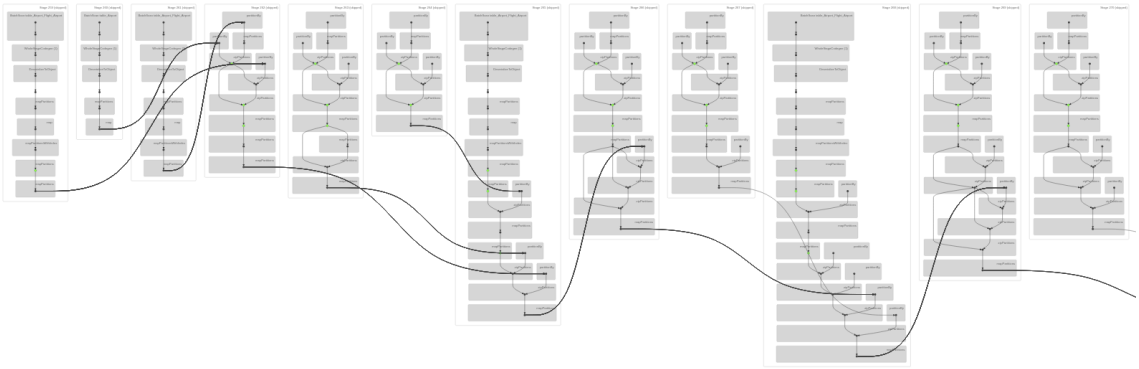

Figure 32: DAG prueba 3

Y como resultados de todo esto tenemos los rankings de los 10 mejores aeropuertos:

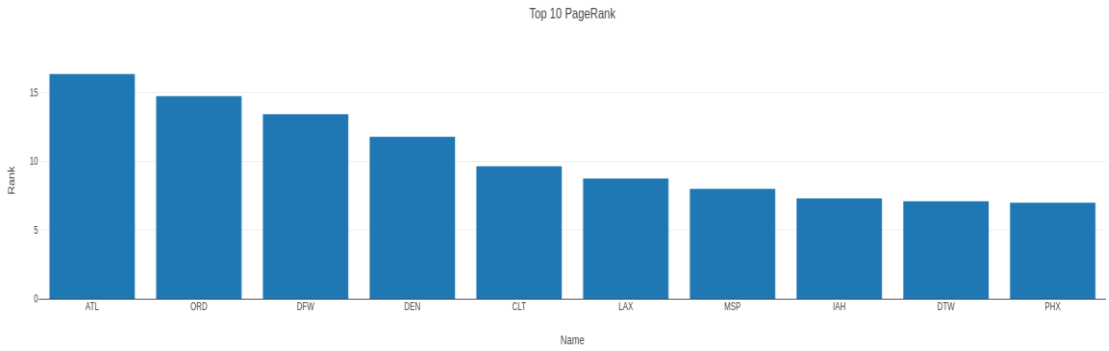

Figure 33: Resultados prueba 3

4.6.4 Aeropuertos más importantes según Triangle Count

Este es otro algoritmo realizado con GraphX en donde se calcula la cantidad de triángulos (ciclos de tres vértices) que incluyen un vértice particular. Un triángulo en este contexto sería un conjunto de tres aeropuertos que están interconectados.

A diferencia del anterior, solamente utiliza un Spark Job para ejecutar este algoritmo ya que necesita menos recursos, y se puede ver el DAG completo.

Job Id ▾	Description	Submitted	Duration	Stages: Succeeded/Total	Tasks (for all stages): Succeeded/Total
0	take at Main.scala:190 take at Main.scala:190	2024/06/01 08:17:01	15 s	11/11	11/11

Figure 34: Jobs Prueba 4

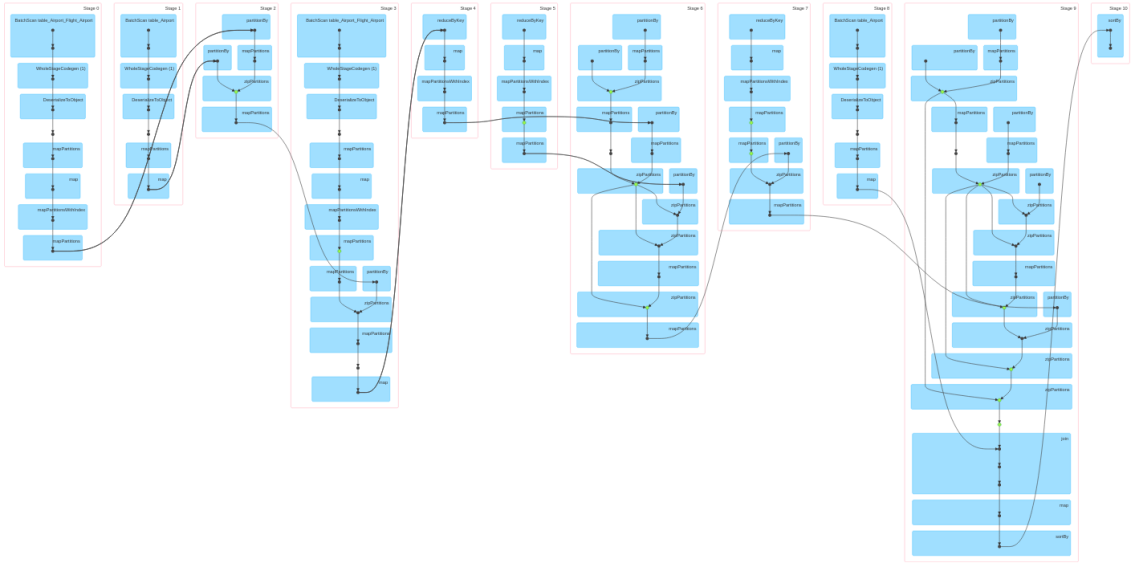

Figure 35: DAG Prueba 4

Este algoritmo se utiliza en dos aplicaciones distintas. Primero para ver la densidad, es decir, para identificar grupos de aeropuertos que estén altamente interconectados. Y segundo para medir la conectividad local y la cohesión de la red.

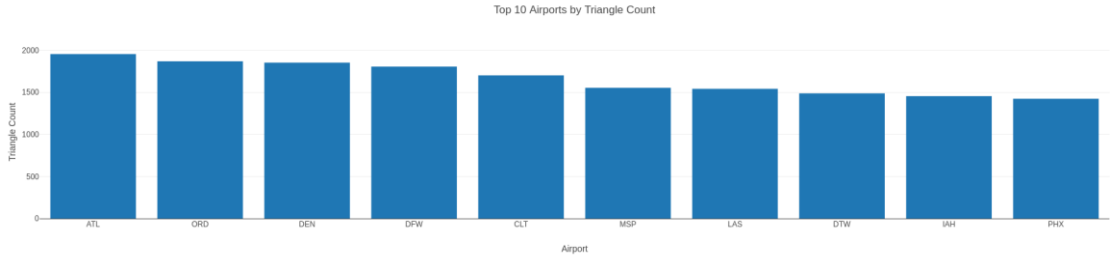

Figure 36: Resultado Prueba 4

4.6.5 Prueba del algoritmo Strongly Connected Components

Este último algoritmo también se ejecuta con GraphX, y como PageRank también requiere de una gran cantidad de Jobs para poder ejecutarse.

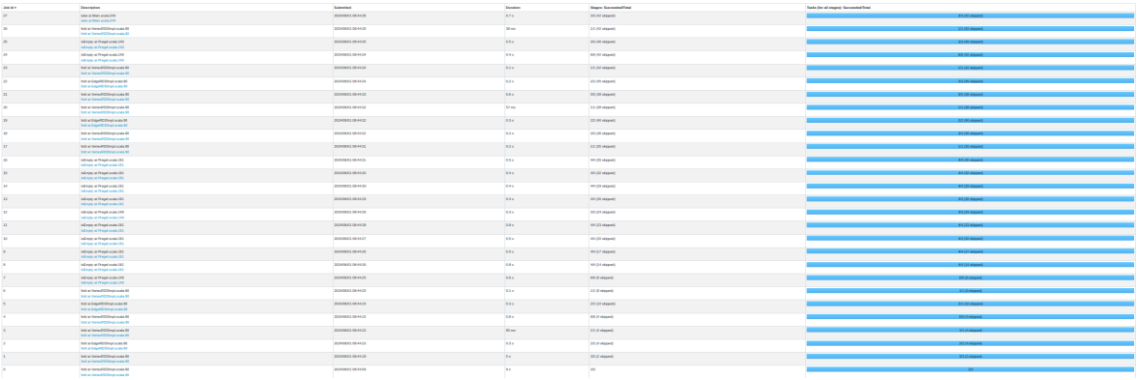

Figure 37: Jobs Prueba 5

Los componentes conectados son subgrafos en los que todos los vértices están conectados entre sí.

Es utilizado principalmente para identificar subgrupos de aeropuertos que están aislados del resto de la red. También tiene como utilidad entender la fragmentación de la red.

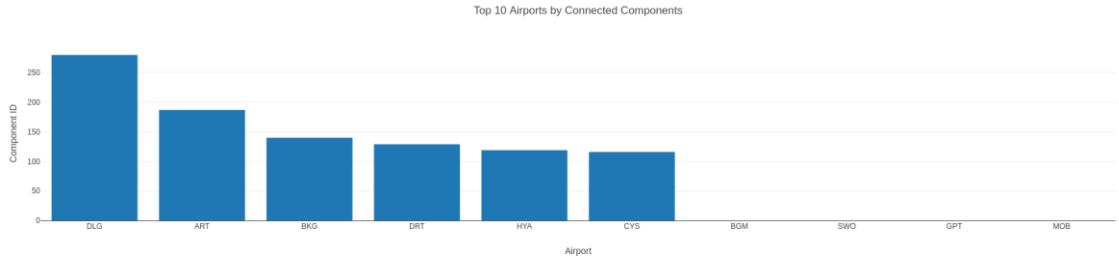

Figure 38: Resultado Prueba 5

5 Conclusiones

5.1 Conclusión

En este capítulo me gustaría resumir el aprendizaje y los resultados que se han logrado al realizar este trabajo.

He conseguido aprender mucho sobre las tecnologías existentes para el desarrollo en Big Data, principalmente para grafos. Previamente conocía herramientas para Big Data, pero no para grafos, así que fue un nuevo mundo abierto para mí.

Además de aprender sobre estas tecnologías, pude aplicar varias de ellas en la aplicación desarrollada, pudiendo tener un conocimiento más profundo.

Ha sido un proyecto muy satisfactorio, no solo porque he aprendido mucho, sino también porque sé que puede ser útil si se sigue desarrollando esta aplicación.

Además de todo esto, agradezco a mi tutor por permitirme realizar un proyecto que me guste tanto y que me ayude cuando lo requiera.

5.2 Problemas encontrados

En general no se encontraron muchos problemas en el desarrollo de la aplicación. Principalmente fueron dos que se puede resumir en uno:

- Problemas de versiones: Al principio fue muy difícil controlar las versionas de todas las herramientas. Principalmente porque se utilizaban herramientas muy distintas y que requerían de un ajuste fino. Luego de haber encontrado que en la página principal de Maven se podían ver cómo importar las librerías y haber encontrado un IDE especialmente desarrollado para Scala, se puede lograr que todo funcione.
- El segundo problema que tuve, que también es parte del primero, es que la versión que necesitaba de Spark para que funcione el conector con Neo4j no era del todo compatible con GraphFrames. Esto provocaba que algunas funciones de esta API no estén disponibles. Como solución se tuvo que rever la arquitectura y utilizar GraphX para ejecutar algoritmos de paso de mensajes.

5.3 Líneas futuras

Hay tres principales líneas futuras para poder continuar con el desarrollo de esta aplicación.

La primera es mejorar la entrada de datos. Esto se puede realizar de varias maneras, como por ejemplo que se puedan manejar varios tipos de archivos sin necesidad de cambiar el código. La mejor solución para esto yo creo que es utilizar otra herramienta llamada Kafka. Esta aplicación distribuida es ampliamente utilizada en entornos Big Data y tiene como función separar la ingesta de datos con la recuperación de estos. Esto sirve principalmente para que distintas personas en los aeropuertos ingresen los datos de vuelos del día

a la aplicación, y que estos se guarden en Kafka, y luego cada cierto tiempo, se actualizara la base de datos descargando todos los datos de este mismo.

La segunda línea futura es levantar la aplicación en un clúster de máquinas y no en local, y empezar a hacer pruebas de rendimiento de verdad en un entorno distribuido.

La última es mejorar lo que es la Interfaz del Usuario. En este proyecto se dio más prioridad a lo que era la funcionalidad de la aplicación, por lo que sería un gran avance mejorar ese aspecto.

6 Análisis de Impacto

En este capítulo se realizará un análisis del impacto potencial de los resultados obtenidos durante la realización del TFG en diferentes contextos.

La solución propuesta en este TFG está alineada con varios de los Objetivos de Desarrollo Sostenible (ODS) de la Agenda 2030 [10]. A continuación, se detallan los ODS más relevantes y cómo la solución contribuye a cada uno de ellos:

6.1 ODS 9: Industria, Innovación e Infraestructura

La propuesta promueve la innovación tecnológica y la mejora de la infraestructura del transporte aéreo mediante el uso de herramientas de análisis de datos. Los grafos permiten una representación más eficiente y precisa de los datos de vuelos, los beneficios incluyen:

- Innovación en el transporte aéreo: El uso de grafos para analizar y optimizar rutas de vuelo representa una innovación significativa en la industria.

6.2 ODS 13: Acción por el Clima

La optimización de rutas de vuelo y la reducción de retrasos pueden resultar en una disminución del consumo de combustible, significando una reducción de las emisiones de gases de efecto invernadero. Los beneficios incluyen:

- Mitigación del cambio climático: La reducción de emisiones de gases de efecto invernadero contribuye a los esfuerzos globales para combatir el cambio climático.

6.3 ODS 8: Trabajo Decente y Crecimiento Económico

Al mejorar la eficiencia y reducir costos en la industria del transporte aéreo, el proyecto puede contribuir al crecimiento económico sostenible y a la creación de empleo. Los beneficios incluyen:

- Aumento de la productividad: La optimización de la gestión de vuelos y la reducción de costos operativos aumentan la productividad y la eficiencia.
- Creación de empleo: La innovación tecnológica y la implementación de nuevas herramientas pueden generar nuevas oportunidades de empleo en la industria.

- Crecimiento económico: La mejora en la eficiencia y la reducción de costos contribuyen al crecimiento económico sostenible en el sector del transporte aéreo.

7 Bibliografía

[1] In Lee. (2017). *Big data: Dimensions, evolution, impacts, and challenges, Business Horizons* [Online]. Available: https://www.sciencedirect.com/science/article/abs/pii/S0007681317300046

[2] Bernard Marr. *How Much Data Do We Create Every Day? The Mind-Blowing Stats Everyone Should Read. (2024).* [Online]. Available: https://bernardmarr.com/how-much-data-do-we-create-every-day-the-mind-blowing-stats-everyone-should-read/

[3] Singh, Kamalpreet & Kaur, Ravinder. (2014). *Hadoop: Addressing Challenges of Big Data* [Online]. Available: https://www.researchgate.net/publication/265848933_Hadoop_Addressing_Challenges_of_Big_Data

[4] Adoni, Hamilton & Nahhal, Tarik & Krichen, Moez & Aghezzaf, Brahim & Elbyed, Abdeltif. (2020). *A survey of current challenges in partitioning and processing of graph-structured data in parallel and distributed systems. Distributed and Parallel Databases* [Online]. Available: https://www.researchgate.net/publication/337197555_A_survey_of_current_challenges_in_partitioning_and_processing_of_graph-structured_data_in_parallel_and_distributed_systems

[5] Neo4j. (2024). *Getting started with neo4j* [Online]. Available: https://neo4j.com/docs/getting-started/get-started-with-neo4j/graph-database/

[6] IBM. (2024). *Apache Spark* [Online]. Available: https://www.ibm.com/topics/apache-spark

[7] Chandan. (2016). *Apache spark rdd vs dataframe vs dataset* [Online]. Available: http://why-not-learn-something.blogspot.com/2016/07/apache-spark-rdd-vs-dataframe-vs-dataset.html

[8] Apache Spark. (2024). *GraphX Programming Guide* [Online]. Available: https://spark.apache.org/docs/0.9.2/graphx-programming-guide.html#background-on-graph-parallel-computation

[9] Alex Lai. (2020). *GraphFrames: An Overview* [Online]. Available: https://adatis.co.uk/graphframes/

[10] Naciones Unidas. (2015). *Objetivos de desarrollo sostenible* [Online]. Available: https://www.un.org/sustainabledevelopment/es/objetivos-de-desarrollo-sostenible/

8 Anexo

TFG_JUAN_FELIPE_LOPEZ_PETERNEL.pdf

ORIGINALITY REPORT

13%	12%	%	7%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

PRIMARY SOURCES

1	Submitted to Universidad Politécnica de Madrid Student Paper	2%
2	oa.upm.es Internet Source	1%
3	hdl.handle.net Internet Source	1%
4	neo4j.com Internet Source	1%
5	www.coursehero.com Internet Source	1%
6	Submitted to Universidad Manuela Beltrán Student Paper	<1%
7	Submitted to University of Strathclyde Student Paper	<1%
8	catalonica.bnc.cat Internet Source	<1%
9	repositorio.ufsc.br Internet Source	<1%

10	ia801006.us.archive.org Internet Source	<1 %
11	Submitted to Monash University Student Paper	<1 %
12	Submitted to University of Hertfordshire Student Paper	<1 %
13	Submitted to University of Malta Student Paper	<1 %
14	es.scribd.com Internet Source	<1 %
15	fr.slideshare.net Internet Source	<1 %
16	Submitted to Infile Student Paper	<1 %
17	Submitted to Universidad Rey Juan Carlos Student Paper	<1 %
18	www.researchgate.net Internet Source	<1 %
19	Submitted to Obudai Egyetem Student Paper	<1 %
20	rural-digital-europe.openaire.eu Internet Source	<1 %
21	Submitted to Corporación Universitaria Minuto de Dios, UNIMINUTO	<1 %

Student Paper

22	Submitted to National College of Ireland Student Paper	<1 %
23	upc.aws.openrepository.com Internet Source	<1 %
24	Submitted to Universidad Estatal a Distancia Student Paper	<1 %
25	icelisting.com Internet Source	<1 %
26	Submitted to PSB Academy (ACP eSolutions) Student Paper	<1 %
27	les7duquebec.net Internet Source	<1 %
28	spaceanalytics.blogspot.com Internet Source	<1 %
29	www.eib.org Internet Source	<1 %
30	www.lotrading.com Internet Source	<1 %
31	cdn.www.gob.pe Internet Source	<1 %
32	enlarge-project.eu Internet Source	<1 %
33	isteonline.in	

	Internet Source	<1%
34	repositorio.comillas.edu Internet Source	<1%
35	techitkvm12.blogspot.com Internet Source	<1%
36	www.auditool.org Internet Source	<1%
37	www.efenet.com Internet Source	<1%
38	www.grapheverywhere.com Internet Source	<1%
39	www.lhssproject.org Internet Source	<1%
40	blog.seidor.com Internet Source	<1%
41	patents.google.com Internet Source	<1%
42	preguntas.dev-co.org Internet Source	<1%
43	upcommons.upc.edu Internet Source	<1%
44	www.proceedings.com Internet Source	<1%

45	www.resourcepanel.org Internet Source	<1 %
46	cloud.google.com Internet Source	<1 %
47	ebin.pub Internet Source	<1 %
48	qdoc.tips Internet Source	<1 %
49	repositorio.uta.edu.ec Internet Source	<1 %
50	repository.eafit.edu.co Internet Source	<1 %
51	empiezoinformatica.wordpress.com Internet Source	<1 %
52	www.cacic2016.unsl.edu.ar Internet Source	<1 %

Exclude quotes	Off	Exclude matches	Off
Exclude bibliography	Off		

Este documento esta firmado por

	**Firmante**	CN=tfgm.fi.upm.es, OU=CCFI, O=ETS Ingenieros Informaticos - UPM, C=ES
	Fecha/Hora	Mon Jun 03 19:41:47 CEST 2024
	Emisor del Certificado	EMAILADDRESS=camanager@etsiinf.upm.es, CN=CA ETS Ingenieros Informaticos, O=ETS Ingenieros Informaticos - UPM, C=ES
	Numero de Serie	561
	Metodo	urn:adobe.com:Adobe.PPKLite:adbe.pkcs7.sha1 (Adobe Signature)