UNIVERSIDAD CENTRAL DE VENEZUELA
FACULTAD DE CIENCIAS
ESCUELA COMPUTACIÓN
CENTRO DE COMPUTACIÓN DISTRIBUIDA Y PARALELA

MANEJO DE GRAFOS DE GRAN ESCALA (LARGE GRAPH BIG DATA)

Trabajo Especial de Grado presentado ante la ilustre

Universidad Central de Venezuela por el

Br. Eric Gabriel Bellet Locker.

C.I. 24.463.483.

Tutores: Jesús Lares y José Sosa.

Caracas, Octubre 2016.

Acta

Agradecimientos

A mi padre Alain, madre Gabrielle y hermana Isabelle gracias por su apoyo incondicional.

A mis tutores Jesús Lares y José Sosa, gracias por su apoyo durante todo el proyecto y por siempre estar al tanto de este gran trabajo.

A todos mis amigos de la universidad Andrés Álvarez, Carlos Zapata, Karen Moncada, Miguel Del Duca, Rafael Piña, Leonardo Santella, Deyban Pérez, Fernando Crema, Juan Sanchez, Pedro Valdivieso, Sebastian Ziegler e Israel Rodríguez.

A la Universidad Central de Venezuela, a la Escuela de Computación y a todos y cada uno de los profesores que contribuyeron en mi formación académica.

Gracias a todos aquellos que durante esta carrera fueron de gran apoyo y me ayudaron a crecer como profesional y persona.

Resumen

Las organizaciones comúnmente utilizan herramientas básicas para el análisis de datos, sin embargo estas no permiten el procesamiento de grandes volúmenes de datos (Big Data), por lo tanto existen muchos datos que no son tratados, es decir información que no puede ser obtenida. Gracias a los avances tecnológicos actualmente existen herramientas que permiten el manejo de grandes volúmenes de datos, por ejemplo Apache Hadoop. Existen muchas formas de modelar los conjuntos de datos, una de estas es denominada grafo, que permite representar redes de la vida cotidiana. El enfoque de esta investigación es modelar redes de gran tamaño que no pueden ser procesados por herramientas tradicionales como grafos de gran escala utilizando herramientas Big Data, para un posterior análisis.

En consecuencia, el objetivo de este trabajo especial de grado, consiste en estructurar, almacenar, realizar diferentes operaciones y analizar diversas redes de gran escala, mediante la utilización de HDFS, Apache Spark, GraphX y Scala. HDFS en un sistema de archivo distribuido, que permite que los datos no se guarden en una única máquina sino que sea capaz de distribuir la información en distintos dispositivos. Spark es una plataforma de computación de código abierto para análisis y procesos avanzados que posee un framework integrado para implementar análisis avanzados que incluye la librería MLlib, el motor gráfico GraphX, Spark Streaming, y la herramienta de consulta Shark. GraphX es un API para grafos que permite el manejo de estos de forma paralela. Scala es un lenguaje de programación multiparadigma que combina propiedades de lenguajes funcionales con orientados a objetos.

Palabras clave: Big Data, Grafo, Apache Hadoop, Apache Spark, GraphX, Scala.

Índice general

Índice de figuras

Índice de tablas

Introducción

A medida que pasa el tiempo algunos de los problemas en la computación cambian gracias a los avances científicos, en el pasado uno de los problemas era el almacenamiento de grandes volúmenes de datos, en el presente el problema es detectar oportunidades o valor en estos datos, en consecuencia las empresas cada vez se preocupan más de la obtención y recolección de datos. Las propiedades de los datos han evolucionado, donde el volumen de estos ha incrementado, ha incrementado su diversidad en forma y origen, es decir la variedad, ha aumentado la necesidad de obtener resultados en tiempo real, en otras palabras la velocidad, el valor es una propiedad importante para las empresas, igual que la visualización de análisis de los datos, lo que aporta criterios para la toma de decisiones. La gestión de los datos se consigue gracias a los sistemas Big Data.

En la actualidad las redes se encuentran por todas partes, nos rodean y formamos parte de ellas, por lo tanto analizarlas puede ser de gran utilidad. Las redes de comunicación, la World Wide Web (www), las redes de proteínas, el genoma humano, las redes neuronales, las de transportes, las redes sociales, el lenguaje, las redes de colaboración científica o las redes terroristas son algunos ejemplos. En base a una estructura de grafo, se puede representar una red y realizar análisis es relativamente sencillo y permite aplicar conceptos de la teoría de grafos, la cual es una disciplina que posee muchos avances y tiene años siendo estudiada.

Muchas redes son un ejemplo claro de grandes volúmenes de datos, las redes sociales es una de estas y realizar distintos tipos de análisis puede ser de gran utilidad. Muchas organizaciones se han dado cuenta que la gestión y un análisis completo de los datos, puede influir positivamente en la empresa si se realiza de forma adecuada y se toman las decisiones correctas. Combinando una estructura y la teoría de grafos junto a Big Data es posible el análisis sobre grandes cantidades de datos que generan las diversas redes.

CAPÍTULO 1

PROBLEMA DE INVESTIGACIÓN

1.1. Planteamiento del problema

Hoy en día, existen muchas organizaciones que generan muchos datos en corto tiempo y no son capaces de gestionarlos utilizando herramientas tradicionales. Existen empresas que no poseen el problema anteriormente mencionado ya que utilizan herramientas Big Data para analizar datos, pero se encuentran con estructuras que no son sencillas de manipular, por ejemplo las redes.

Gracias a la teoría de grafos, actualmente sabemos que las redes se pueden representar como un grafo, lo cual facilita la realización de análisis. Existen herramientas especializadas que combinan la teoría de grafos y Big Data. El problema es estructurar redes que representadas mediante un grafo posean muchos nodos y aristas, es decir que sean muy grandes o de gran escala, que sean considerados como un problema de grandes volúmenes de datos y utilizando herramientas comunes de Big Data es extremadamente complicado realizar análisis sobre estos.

1.2. Justificación

1.2.1. ¿Por qué es un problema?

Es un problema ya que es complicado manejar tanto volumen de datos utilizando herramientas tradicionales para el análisis de datos y utilizando las herramientas Big Data básicas para realizar análisis sobre redes de gran escala no es sencillo.

1.2.2. ¿Para quién es un problema?

Es un posible problema para cualquier usuario u organización que le interese y/o necesite realizar un análisis de redes que representen un gran volumen de

datos.

1.2.3. ¿Desde cuándo es un problema?

Es un problema desde que existen redes que representadas como grafos contienen muchos nodos y aristas.

1.3. Objetivos de la investigación

1.3.1. General

Manipular, almacenar, estructurar, calcular y realizar análisis sobre diversas redes representadas como grafos a partir de grandes volúmenes de datos, utilizando herramientas especializadas en procesamiento paralelo y distribuido para grafos de gran escala.

1.3.2. Específicos

- Definir las herramientas a utilizar, HDFS, Apache Spark, GraphX y Scala.
- Estructurar redes que representen un gran volumen de datos mediante un grafo de gran escala.
- Almacenar los grafos en HDFS.
- Utilizar Apache Spark para establecer una conexión de HDFS con GraphX.
- Transformar los diversos grafos al formato GraphX.
- Implementar diversos cálculos para obtener información sobre los grafos.
- Implementar algoritmos que provean información compleja sobre los grafos.
- Realizar pruebas de la aplicación.
- Visualizar los resultados obtenidos.
- Visualizar gráficamente los grafos.

1.4. Antecedentes

Duda... Giraph GraphLab

1.5. Alcance

Duda... Desarrollar frameworks. Realizar análisis mas específicos. Crear algoritmos de detección de comunidades y medidas de centralidad. Etc

CAPÍTULO 2

MARCO TEÓRICO

2.1. Dato

Los datos son números, letras o símbolos que describen objetos, condiciones o situaciones. Son el conjunto básico de hechos referentes a una persona, cosa o transacción de interés para distintos objetivos, entre los cuales se encuentra la toma de decisiones. [1]

2.2. Información

La información es un sistema de control, en tanto que es la propagación de consignas que deberíamos de creer. En tal sentido la información es un conjunto organizado de datos capaz de cambiar el estado de conocimiento. Un grupo de datos ordenados y supervisados, que sirven para construir un mensaje basado en un cierto fenómeno o ente, la cual permite resolver problemas y tomar decisiones, es información, ya que su aprovechamiento racional es la base del conocimiento. [2]

2.3. Conocimiento

Fidias Arias (2004), define el conocimiento como un "proceso en el cual se relacionan el sujeto que conoce, que percibe mediante sus sentidos, y el objeto conocido y percibido". El conocimiento es el acto o efecto de conocer. Es la capacidad del hombre para comprender por medio de la razón la naturaleza, cualidades y relaciones de las cosas.
La ciencia considera que, para alcanzar el conocimiento, es necesario seguir un método. El conocimiento científico no sólo debe ser válido y consistente desde el punto de vista lógico, sino que también debe ser probado mediante el método científico o experimental. [3]

2.4. Minería de datos (Data Mining)

Es un campo de la estadística y las ciencias de la computación detectar la información procesable o patrones sobre conjuntos grandes de datos. Normalmente, estos patrones no se pueden detectar mediante la exploración tradicional de los datos porque las relaciones son demasiado complejas o porque hay demasiado datos. [4]

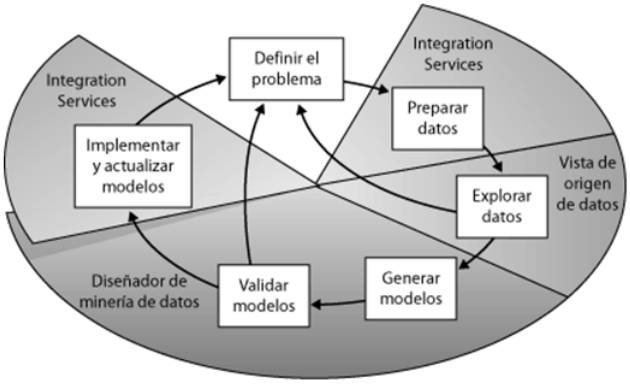

Figura 2.1: Proceso para la minería de datos.

La generación de un modelo de minería de datos forma parte de un proceso mayor que incluye desde la formulación de preguntas acerca de los datos y la creación de un modelo para responderlas, hasta la implementación del modelo en un entorno de trabajo. Este proceso se puede definir mediante los seis pasos básicos siguientes:

- Definir el problema: Este paso incluye analizar los requisitos organizacionales, definir el ámbito del problema, definir las métricas por las que se evaluará el modelo y definir los objetivos concretos del proyecto de minería de datos.
- Preparar los datos o preprocesamiento: Los datos pueden estar dispersos en la empresa y almacenados en formatos distintos; también pueden contener incoherencias como entradas que faltan o incorrectas, la finalidad de este paso es consolidar y limpiar los datos.
- Explorar los datos o análisis exploratorio: Entre las técnicas de exploración se incluyen calcular los valores mínimos y máximos, calcular la media y las desviaciones estándar, y examinar la distribución de los datos. Al explorar los datos se puede decidir si el conjunto de datos contiene datos defectuosos y, a continuación, puede proponer una estrategia para corregir los problemas u obtener una descripción más profunda de los comportamientos que son típicos de su negocio.

- Generar modelos: Consiste en generar el modelo o modelos de minería de datos. Antes de procesar la estructura y el modelo, un modelo de minería de datos simplemente es un contenedor que especifica las columnas que se usan para la entrada, el atributo que está prediciendo y parámetros que indican al algoritmo cómo procesar los datos. El procesamiento de un modelo a menudo se denomina entrenamiento.
- Validar modelos: Consiste en explorar los modelos de minería de datos que ha generado y comprobar su eficacia. Antes de implementar un modelo en un entorno de producción, es aconsejable probar si funciona correctamente.
- Implementar y actualizar los modelos: Consiste en implementar los modelos que funcionan mejor en un entorno de producción.

2.5. Aprendizaje automático (Machine Learning)

En ciencias de la computación el aprendizaje automático es una rama de la inteligencia artificial cuyo objetivo es desarrollar técnicas que permitan a las computadoras aprender. De forma más concreta, se trata de crear programas capaces de generalizar comportamientos a partir de una información no estructurada suministrada en forma de ejemplos. El aprendizaje automático se divide en dos áreas principales: aprendizaje supervisado y aprendizaje no supervisado.[5]

El objetivo del aprendizaje supervisado es hacer predicciones a futuro basadas en comportamientos o caracterısticas que se han visto en los datos ya almacenados. El aprendizaje supervisado permite buscar patrones en datos históricos relacionando los todos campos con el campo objetivo. Por otro lado, el aprendizaje no supervisado usa datos históricos que no están etiquetados. El fin es explorarlos para encontrar alguna estructura o forma de organizarlos.

2.6. Inteligencia artificial (Artificial Intelligence)

Inteligencia artificial es considerada una rama de la computación que relaciona un fenómeno natural con una analogía artificial mediante programas de computador. Consiste en el diseño de procesos que, al ejecutarse sobre una arquitectura física, producen resultados que maximizan una cierta medida de rendimiento denominado comúnmente comportamiento inteligente. [6]

2.7. Inteligencia de negocios (Business Intelligence)

Inteligencia de negocios es el conjunto de conceptos y métodos para mejorar la toma de decisiones en los negocios, utilizando sistemas de apoyo basados en

hechos (Howard Dresner, 1989). Sin embargo, en la actualidad este concepto incluye una amplia categoría de metodologías, aplicaciones y tecnologías que permiten reunir, acceder, transformar y analizar los datos con el propósito de ayudar a los usuarios de una organización a tomar mejores decisiones de negocio [7]

2.8. Ciencia de datos

Ciencias de los datos es un campo interdisciplinario, el cual abarca un conjunto de procesos y sistemas para extraer información de un conjunto de datos estructurados o no estructurados con el objetivo de generar conocimiento. Esta área esta estrechamente relacionada con las estadísticas, la minería de datos, análisis predictivo, entre otros. La ciencia de datos utiliza la preparación de datos, estadísticas, modelos predictivos y de aprendizaje automático para investigar problemas en diversos ámbitos. En la actualidad el concepto Big Data es equivalente al de Ciencia de datos pero haciendo énfasis a grandes volúmenes de datos concepto.[8]

La ciencia de datos permite la generación de conocimiento a partir de grandes volúmenes de datos, aplicando técnicas de procesamiento paralelo y distribuido, para implementar algoritmos que permitan predecir o detectar patrones sobre los datos almacenados. A partir de los resultados obtenidos se podrán construir herramientas que permitan analizar los resultados y apoyar los procesos de toma de decisiones.

2.8.1. Big data

Es una plataforma tecnológica (hardware y software) que permite almacenar (de manera distribuida) y procesar (en forma paralela y distribuida) conjuntos de datos, que por su gran volumen, superan las capacidades de las plataformas TI tradicionales, ya sea porque tomaría demasiado tiempo procesar dichos datos o porque sería muy costoso implementar una arquitectura que soporte tal cantidad de datos.

En otras palabras, Big data es un concepto que hace referencia a la acumulación de grandes cantidades de datos y a los procedimientos usados para encontrar patrones repetitivos dentro de esos datos. Se define como un conjunto de herramientas informáticas destinadas a la manipulación, gestión y análisis de grandes volúmenes de datos de todo tipo, los cuales no pueden ser gestionados por las herramientas informáticas tradicionales, que tiene por objetivo analizar datos e información de manera inteligente que ayuden a una correcta toma de decisión.

En la actualidad la cantidad de información digital que se genera diariamente en nuestro planeta crece exponencialmente, lo cual ha desencadenado que

muchas empresas y organizaciones, desean utilizar esta información con el objetivo de mejorar las prestaciones de sus servicios o negocios. Por lo tanto, el objetivo fundamental de los sistemas Big Data es dotar de una infraestructura tecnológica a las empresas y organizaciones con la finalidad de poder almacenar, tratar y analizar de manera económica, rápida y flexible la gran cantidad de datos que se generan diariamente, para ello es necesario el desarrollo y la implantación tanto de hardware como de software específicos que gestionen esta explosión de datos con el objetivo de extraer valor para obtener información útil para nuestros objetivos o negocios. [9]

El término de Big Data, en general se definen con 8 dimensiones llamada las 8V [10], Velocidad, Veracidad, Valor, Volumen, Variedad, Variabilidad, Visualización y Visión:

- Velocidad: La tecnología Big Data ha de ser capaz de almacenar y trabajar en tiempo real con las fuentes generadoras de información como sensores, cámaras de vídeos, redes sociales, blogs, páginas webs, etc, fuentes que generan millones y millones de datos al segundo, por otro lado la capacidad de análisis de dichos datos han de ser rápidos reduciendo los largos tiempos de procesamiento que presentaban las herramientas tradicionales de análisis.

- Veracidad: Big Data ha de ser capaz de tratar y analizar inteligentemente este vasto volumen de datos con la finalidad de obtener una información verídica y útil que nos permita mejorar nuestra toma de decisiones.

- Volumen de datos: Como su propio nombre indica la tecnología Big Data (datos masivos) ha de ser capaz de gestionar un gran volumen de datos que se generan diariamente por las empresas y organizaciones de todo el mundo.

- Variedad de datos: Big data ha de tener la capacidad de combinar una gran variedad de información digital en los diferentes formatos en las que se puedan presentar ya sean en formato vídeo, audio o texto.

- Valor: El valor se refiere a nuestra capacidad de convertir los datos en valor. Es importante que las empresas realizar cualquier intento de recoger y aprovechar los datos grandes. Todos los que los datos disponibles crearán mucho valor para las organizaciones, sociedades y consumidores. Grandes datos significa un gran negocio y todas las industrias cosecharán los beneficios de los grandes datos. Por supuesto, los datos en sí mismo no es valioso en absoluto. El valor está en los análisis realizados en esos datos

y cómo los datos se convierte en información y, finalmente, convirtiéndola en conocimiento. El valor está en cómo las organizaciones usarán esos datos y convertir su organización en una empresa centrada en la información que se basa en ideas derivadas de los análisis de datos para la toma de decisiones.

- Variabilidad: La variabilidad se refiere a los datos cuyo significado está en constante cambio. Este es particularmente el caso cuando la recolección de datos se basa en el procesamiento del lenguaje.
- Visualización: Esta es la parte dura de grandes volúmenes de datos, hacer que la gran cantidad de datos sea comprensible de manera que sea fácil de entender y leer. Con los análisis y visualizaciones adecuadas, los datos brutos se pueden utilizar para tomas de decisiones adecuadas. Las visualizaciones por supuesto no significan gráficas ordinarias o gráficos circulares. Significan gráficos complejos que pueden incluir muchas variables de datos sin dejar de ser comprensible y legible. La visualización puede no ser la parte más difícil con respecto al uso de la tecnología, pero lograr buenos resultados seguro que es la parte más difícil. Contar una historia compleja en un gráfico es muy difícil, pero también es extremadamente crucial.
- Visión: todas las empresas, que se inician con grandes volúmenes de datos deben tener una visión, qué hacer con ellos. La visión define las metas que se pretenden conseguir en el futuro. Estas metas tienen que ser realistas y alcanzables, puesto que la propuesta de visión tiene un carácter inspirador y motivador. La descarga de Hadoop, la instalación de él y la alimentación con algunos datos no ayudarán. La empresa tiene que estar lista para la transformación digital. Si la organización no entiende lo que puede ofrecer grandes volúmenes de datos, no habrá ningún éxito.

2.8.2. Campos en los cuales es utilizado

El manejo de grandes volúmenes de datos es utilizado y se puede utilizar en múltiples áreas, por ejemplo:

- Seguridad: Su potencial reside en la capacidad de análisis de volúmenes de datos antes impensable de una manera óptima y ágil. Existen, por ejemplo, modelos de análisis del comportamiento humano para prevenir atentados terroristas obtenidos mediante un análisis permanente de las cámaras, sensores y accesos secuenciales a un sistema.

- Investigación médica: La investigación médica puede mejorar muchísimo si es capaz de asimilar una enorme cantidad de datos (monitorización, historiales, tratamientos, etc.) y estructurarlos para el establecimiento de diagnósticos o la síntesis de medicamentos.

- Gobierno y toma de decisiones: Big Data ofrece una mejora y optimización en los procesos de toma de decisiones de empresas y gobiernos, permitiendo entre muchas otras, el soporte a la toma decisiones, siendo complementario a las plataformas de "Business Intelligence" (BI).

- Internet 2.0: genera una gran multitud de datos que difícilmente se podrían gestionar sin un Big Data. Las redes sociales cada vez se extienden a más ámbitos de nuestra sociedad

- CRM: La gestión de la relación de una empresa con sus clientes suele implicar la gestión de Almacén de Datos y la interrelación de diversidad de datos (comercial, operaciones, marketing,...), diversos canales (web, redes sociales, correo,...) y formatos. Big Data facilita las operaciones de análisis y seguimiento, favoreciendo la fidelidad y descubrimiento de nuevos mercados.

- Logística: El sector logístico mejora notablemente gracias a las posibilidades analíticas de un Big Data y su potencial para el despliegue de servicios específicos (movilidad, tracking, seguridad, etc.). El ejemplo más popular se encuentra en el control de flotas (la ruta óptima permite a los vehículos circular con la máxima capacidad de carga, pudiendo recorrer rutas mejorando tiempos, consumos y contaminación).

2.9. Base de datos

Se define una base de datos como una serie de datos organizados y relacionados entre sí, los cuales son recolectados y explotados por los sistemas de información de una empresa o negocio en particular. [11] Entre las principales características de los sistemas de base de datos podemos mencionar:

- Independencia lógica y física de los datos.
- Redundancia mínima.
- Acceso concurrente por parte de múltiples usuarios.
- Integridad de los datos.
- Consultas complejas optimizadas.
- Seguridad de acceso y auditoría.
- Respaldo y recuperación.
- Acceso a través de lenguajes de programación estándar.

2.9.1. Base de datos relacionales

Una base de datos relacional es una colección de elementos de datos organizados en un conjunto de tablas formalmente descritas desde la que se puede acceder a los datos o volver a montarlos de muchas maneras diferentes sin tener que reorganizar las tablas de la base.

Estas bases de datos poseen un conjunto de tablas que contienen datos provistos en categorías predefinidas. Cada tabla (que a veces se llaman 'relación') contiene una o más categorías de datos en columnas. Cada fila contiene una instancia única de datos para las categorías definidas por las columnas. [12]

2.9.1.1. Ventajas de las base de datos relacionales

- Está más adaptado su uso y los perfiles que los conocen son mayoritarios y más baratos.
- Debido al largo tiempo que llevan en el mercado, estas herramientas tienen un mayor soporte y mejores suites de productos y add-ons para gestionar estas bases de datos.
- La atomicidad de las operaciones en la base de datos. Esto es, que en estas bases de datos o se hace la operación entera o no se hace utilizando la famosa técnica del rollback.
- Los datos deben cumplir requisitos de integridad tanto en tipo de dato como en compatibilidad.

2.9.1.2. Desventajas de las base de datos relacionales

- La atomicidad de las operaciones juegan un papel crucial en el rendimiento de las bases de datos.
- Escalabilidad, que aunque probada en muchos entornos productivos suele, por norma, ser inferior a las bases de datos NoSQL.

2.9.2. Base de datos no relacionales

Son un enfoque hacia la gestión de datos y el diseño de base de datos que es útil para grandes conjuntos de datos distribuidos. Los datos almacenados no requieren estructuras fijas como tablas, normalmente no soportan operaciones JOIN, ni garantizan completamente ACID (atomicidad, consistencia, aislamiento y durabilidad), y habitualmente escalan bien horizontalmente.

Son especialmente útil cuando una empresa necesita acceder y analizar grandes cantidades de datos no estructurados o datos que se almacenan de forma remota en varios servidores virtuales en la nube. [13]

2.9.2.1. Ventajas de las base de datos no relacionales

- La escalabilidad y su carácter descentralizado. Soportan estructuras distribuidas.
- Suelen ser bases de datos mucho más abiertos y flexibles. Permiten adaptarse a necesidades de proyectos mucho más fácilmente que los modelos de Entidad Relación.
- Se pueden hacer cambios de los esquemas sin tener que parar bases de datos.
- Escalabilidad horizontal: son capaces de crecer en número de máquinas, en lugar de tener que residir en grandes máquinas.
- Se pueden ejecutar en máquinas con pocos recursos.
- Optimización de consultas en base de datos para grandes cantidades de datos.

2.9.2.2. Desventajas de las base de datos no relacionales

- No todas las bases de datos NoSQL contemplan la atomicidad de las instrucciones y la integridad de los datos. Soportan lo que se llama consistencia eventual.
- Problemas de compatibilidad entre instrucciones SQL. Las nuevas bases de datos utilizan sus propias características en el lenguaje de consulta y no son 100 % compatibles con el SQL de las bases de datos relacionales. El soporte a problemas con las queries de trabajo en una base de datos NoSQL es más complicado.
- Falta de estandarización. Hay muchas bases de datos NoSQL y aún no hay un estándar como si lo hay en las bases de datos relacionales. Se presume un futuro incierto en estas bases de datos.
- Soporte multiplataforma. Aún quedan muchas mejoras en algunos sistemas para que soporten sistemas operativos que no sean Linux.
- Suelen tener herramientas de administración no muy usables o se accede por consola.

2.9.2.3. Tipos de base de datos no relacionales

MODELO DE DATOS	FORMATO	CARACTERÍSTICAS	APLICACIONES
Documento.	Similar a JSON (JavaScript Object Notation).	- Intuitivo. - Manera natural de modelar datos cercana a la programación orientada a objetos. - Flexibles, con esquemas dinámicos. - Reducen la complejidad de acceso a los datos.	Se pueden utilizar en diferentes tipos de aplicaciones debido a la flexibilidad que ofrecen.
Grafo.	Nodos con propiedades (atributos) y relaciones (aristas).	- Los datos se modelan como un conjunto de relaciones entre elementos específicos. - Flexibles, atributos y longitud de registros variables. - Permite consultas más amplias y jerárquicas.	Redes sociales, software de recomendación, geolocalización, topologías de red, etc.
Clave-Valor y Columna.	Clave-valor: una clave y su valor correspondiente Columnas: variante que permite más de un valor (columna) por clave.	- Rendimiento muy alto. - Alta curva de escalabilidad. - Útil para representar datos no estructurados.	Aplicaciones que solo utilizan consulta de datos por un solo valor de la clave.

Tabla 2.1: Tipos de Base de Datos NoSql.

2.9.2.3.1. Orientada a columnas

Este tipo de bases de datos están pensadas para realizar consultas y agregaciones sobre grandes cantidades de datos. Funcionan de forma parecida a las bases de datos relacionales, pero almacenando columnas de datos en lugar de registros. Algunos ejemplos de base de datos orientada a columnas: Cassandra, HBase.

- Cassandra: incluida en esta sección, aunque en realidad sigue un modelo híbrido entre orientada a columnas y clave-valor. Es utilizada por Facebook y Twitter (aunque dejaron de usarla para almacenar tweets).

- HBase. Escrita en Java y mantenida por el Projecto Hadoop de Apache, se utiliza para procesar grandes cantidades de datos. La utilizan Facebook, Twitter o Yahoo.

2.9.2.3.2. Orientada a clave/valor

Son sencillas de entender. Simplemente guardan tuplas que contienen una clave y su valor. Cuándo se quiere recuperar un dato, simplemente se busca por su clave y se recupera el valor. Algunos ejemplos de base de datos clave/valor: DynamoDB, Redis.

- DynamoDB: desarrollada por Amazon, es una opción de almacenaje que puedemos usar desde los Amazon Web Services. La utilizan el Washington Post y Scopely.
- Redis: desarrollada en C y de código abierto, es utilizada por Craiglist y Stack Overflow (a modo de caché).

2.9.2.3.3. Orientada a documentos

Son aquellas que gestionan datos semi estructurados. Es decir documentos. Estos datos son almacenados en algún formato estándar como puede ser XML, JSON o BSON. Son las bases de datos NoSQL más versátiles. Se pueden utilizar en gran cantidad de proyectos, incluyendo muchos que tradicionalmente funcionarían sobre bases de datos relacionales. Algunos ejemplos de base de datos orientada a documentos: MongoDB, CouchDB.

- MongoDB: probablemente la base de datos NoSQL más famosa del momento. En octubre del año pasado, MongoDB conseguía 150 millones de dólares en financiación, convirtiéndose en una da las startups más prometedoras. Algunas compañías que actualmente utilizan MongoDB son Foursquare o eBay.
- CouchDB: es la base de datos orientada a documentos de Apache. Una de sus interesantes características es que los datos son accesibles a través de una API Rest. Este sistema es utilizado por compañías como Credit Suisse y la BBC.

2.9.2.3.4. Orientada a grafo

Basadas en la teoría de grafos utilizan nodos y aristas para representar los datos almacenados. Son muy útiles para guardar información en modelos con muchas relaciones, como redes y conexiones sociales. Algunos ejemplos de base de datos orientada a grafos: Infinite Graph, Neo4j.

- Infinite Graph: escrita en Java y C++ por la compañía Objectivity. Tiene dos modelos de licenciamiento: uno gratuito y otro de pago.

- Neo4j: base de datos de código abierto, escrita en Java por la compañía Neo Technology. Utilizada por compañías como HP, Infojobs o Cisco.

2.10. Almacén de datos (Data Warehouse)

Un almacén de datos es una base de datos corporativa o repositorio de datos, que se caracteriza por integrar y depurar información de una o más fuentes distintas, para luego procesarla permitiendo su análisis desde infinidad de perspectivas y con grandes velocidades de respuesta. La creación de un almacén de datos colecciona datos orientado a temas, integrado, no volátil, de tiempo variante el cual representa en la mayoría de las ocasiones el primer paso, desde el punto de vista técnico, para implantar una solución completa y fiable de inteligencia de negocios. Existen 2 investigadores muy famosos, Bill Inmon y Ralph Kimball relacionados con este concepto de almacén de datos. [14]

Para Bill Inmon (1992), quien acuño el término por primera vez, "el Almacén de Datos es una colección de datos orientados al tema, integrados, no volátiles e históricos, organizados para el apoyo de un proceso de ayuda a la decisión. No obstante, y como cabe suponer, es mucho más que eso".

Para Ralph Kimball (1997) "el Almacén de Datos es una copia de las transacciones de datos específicamente estructurada para la consulta y el análisis; es la unión de todos las Bodegas de Datos de una entidad".

2.11. Lenguajes de programación

En computación, un lenguaje de programación es cualquier lenguaje artificial ya que intenta conservar una similitud con el lenguaje humano, el cual, se utiliza para definir adecuadamente una secuencia de instrucciones que puedan ser interpretadas y ejecutadas en una computadora. [15]

Establecen un conjunto de símbolos, reglas sintácticas y semánticas, las cuales rigen la estructura y el significado del programa, junto con sus elementos y expresiones. De esta forma, permiten a los programadores o desarrolladores, poder especificar de forma precisa los datos sobre los que se va a actuar, su almacenamiento, transmisión y demás acciones a realizar bajo las distintas circunstancias consideradas. Usualmente se clasifican en interpretados y compilados, en el cual los compilados tienen un compilador específico que obtiene como entrada un programa y traduce las instrucciones las cuales pueden servir de entrada para otro interprete o compilado y los interpretados tienen un intérprete especifico que obtiene como entrada un programa y ejecuta las acciones escritas a medida que las va procesando.

2.11.1. R

R es un lenguaje y entorno de programación para análisis estadístico y gráfico, el cual proporciona un amplio abanico de herramientas estadísticas (modelos lineales y no lineales, tests estadísticos, análisis de series temporales, algoritmos de clasificación y agrupamiento, etc.) y gráficas. Se trata de un proyecto de software libre, resultado de la implementación GNU del premiado lenguaje S y se distribuye bajo la licencia GNU GPL y está disponible para los sistemas operativos Windows, Macintosh, Unix y GNU/Linux. Puede integrarse con distintas bases de datos y existen bibliotecas que facilitan su utilización desde lenguajes de programación interpretados como Perl y Python. [16]

R al estar orientado a las estadísticas, proporciona un amplio abanico de herramientas de cálculo numérico y a su vez para minería de datos, y posee una capacidad gráfica, que permite generar gráficos con alta calidad, con sólo utilizar las funciones de graficación.

2.11.1.1. R Studio

RStudio es un entorno de desarrollo integrado (IDE) construido exclusivo para R, para computación estadística y gráficos . Incluye una consola, editor de sintaxis que apoya la ejecución de código, así como herramientas para el trazado, la depuración y la gestión del espacio de trabajo. [17]

Algunas de sus características son:

- Ejecutar código R directamente desde el editor de código fuente.
- Salto rápido a las funciones definidas.
- Colaborativo.
- Documentación y soporte integrado.
- Administración sencilla de múltiples directorios de trabajo mediante proyectos.
- Navegación en espacios de trabajo y visor de datos.
- Potente autoría y depuración.
- Depurador interactivo para diagnosticar y corregir los errores rápidamente.
- Herramientas de desarrollo extensas.
- Autoría con Sweave y R Markdown.

2.11.1.2. Igraph

Es un paquete que provee R que permite realizar análisis de redes. Proporciona rutinas y funciones para crear y manipular grafos con facilidad. En otras palabras, es una colección de herramientas para análisis de redes de forma eficiente, portable y de sencillo uso. Igraph es de código abierto y libre, y permite la utilización de los lenguajes R, Python, C y C++. [18]

2.11.2. Java

Java es un lenguaje de programación de propósito general, concurrente, orientado a objetos que fue diseñado específicamente para tener tan pocas dependencias de implementación como fuera posible. Su sintaxis deriva en gran medida de C y C++, pero tiene menos utilidades de bajo nivel que cualquiera de ellos. Las aplicaciones de Java son generalmente compiladas a bytecode (clase Java) que puede ejecutarse en cualquier máquina virtual Java (JVM) sin importar la arquitectura de la computadora subyacente.

Una de las principales características por las que Java se ha hecho muy famoso es que es un lenguaje independiente de la plataforma. Es una ventaja significativa para los desarrolladores de software, pues antes tenían que hacer un programa para cada sistema operativo, por ejemplo Windows, Linux, Apple, etc. Esto lo consigue porque se ha creado una Máquina de Java para cada sistema que hace de puente entre el sistema operativo y el programa de Java y posibilita que este último se entienda perfectamente. En la actualidad es un lenguaje muy extendido y cada vez cobra más importancia tanto en el ámbito de Internet como en la informática en general. [19]

El lenguaje Java se creó con cinco objetivos principales:

- Debería incluir por defecto soporte para trabajo en red.
- Debería usar el paradigma de la programación orientada a objetos.
- Debería permitir la ejecución de un mismo programa en múltiples sistemas operativos.
- Debería ser fácil de usar y tomar lo mejor de otros lenguajes orientados a objetos, como C++.
- Debería diseñarse para ejecutar código en sistemas remotos de forma segura.

2.11.3. Scala

El nombre de Scala significa "Scalable Languaje", se llama así ya que fue diseñado para poder crecer según la demanda de los usuarios, se puede usar Scala para crear pequeños scripts hasta para desarrollar grandes sistemas muy

sofisticados. Scala es un lenguaje de programación multi-paradigma que une lo programación orientado a objetos y la programación funcional, que promueven la escalabilidad desde lo más pequeño. [20] Las principales características de Scala son:

- Una sintaxis concisa, elegante y flexibles: Scala utiliza una serie de técnicas para minimizar la sintaxis innecesarios. La inferencia de tipos minimiza la necesidad de información de tipo explícito en muchos contextos. Las declaraciones de tipos y funciones son muy concisas. Scala permite incluir caracteres no alfanuméricos en el nombres de las funciones. Combinado con un poco de azúcar sintáctica, esta característica permite al usuario definir métodos que se parezcan y se comporten como operadores.
- Tipado estático: Scala es un lenguaje de tipado estático, lo que quiere decir que une el tipo a una variable durante toda la vida de esa variable.
- Orientado a Objetos: Scala es un lenguaje orientado a objetos puro en el sentido de que cada valor es un objeto y cada operación es una llamada de método.
- Funcional: A pesar de que su sintaxis es bastante convencional, Scala es también un lenguaje funcional en toda regla. Tiene todo lo que se puede esperar, incluyendo funciones de primera clase, funciones anónimas, funciones de orden superior, funciones anidadas, una biblioteca con estructuras de datos inmutables eficientes, y una preferencia general de inmutabilidad sobre la mutabilidad.
- JVM (Java Virtual Machine): Scala se ejecuta en todas las máquinas virtuales de Java y también en Android. Las clases de Java y Scala pueden combinarse libremente, sin importar si residen en diferentes proyectos o en el mismo. Incluso pueden referirse mutuamente entre sí.
- Arquitecturas Escalables: Scala está diseñado para escalar desde pequeños scripts interpretado hasta grandes aplicaciones.

2.11.4. Python

Python es un lenguaje de programación multiparadigma, ya que soporta orientación a objetos, programación imperativa y, en menor medida, programación funcional. Es un lenguaje interpretado, usa tipado dinámico y es multiplataforma, cuya filosofía hace hincapié en una sintaxis que favorezca un código legible. Es administrado por la Python Software Foundation. Posee una licencia de código abierto, denominada Python Software Foundation License. [21] Algunas de las características de Python son:

- Orientado a Objetos : La programación orientada a objetos está soportada en Python y ofrece en muchos casos una manera sencilla de crear programas con componentes reutilizables.

- Funciones y librerías: Dispone de muchas funciones incorporadas en el propio lenguaje, para el tratamiento de strings, números, archivos, etc. Además, existen muchas librerías que podemos importar en los programas para tratar temas específicos como la programación de ventanas o sistemas en red o cosas tan interesantes como crear archivos comprimidos en .zip.
- Propósito general: Se pueden crear todo tipo de programas. No es un lenguaje creado específicamente para la web, aunque entre sus posibilidades sí se encuentra el desarrollo de páginas.
- Multiplataforma: Hay versiones disponibles de Python en muchos sistemas informáticos distintos. Originalmente se desarrolló para Unix, aunque cualquier sistema es compatible con el lenguaje siempre y cuando exista un intérprete programado para él.
- Interpretado: Quiere decir que no se debe compilar el código antes de su ejecución. En realidad sí que se realiza una compilación, pero esta se realiza de manera transparente para el programador. En ciertos casos, cuando se ejecuta por primera vez un código, se producen unos bytecodes que se guardan en el sistema y que sirven para acelerar la compilación implícita que realiza el intérprete cada vez que se ejecuta el mismo código.
- Interactivo: Python dispone de un intérprete por línea de comandos en el que se pueden introducir sentencias. Cada sentencia se ejecuta y produce un resultado visible, que puede ayudarnos a entender mejor el lenguaje y probar los resultados de la ejecución de porciones de código rápidamente.
- Sintaxis clara: Por último, destacar que Python tiene una sintaxis muy visual, gracias a una notación identada (con márgenes) de obligado cumplimiento. En muchos lenguajes, para separar porciones de código, se utilizan elementos como las llaves o las palabras clave begin y end. Para separar las porciones de código en Python se debe tabular hacia dentro, colocando un margen al código que iría dentro de una función o un bucle. Esto ayuda a que todos los programadores adopten unas mismas notaciones y que los programas de cualquier persona tengan un aspecto muy similar.

2.12. Apache Hadoop

Apache Hadoop es un framework que permite el procesamiento de grandes volúmenes de datos a través de clusters, usando un modelo simple de programación. [22] En otras palabras, es un framework de software que soporta aplicaciones distribuidas bajo una licencia libre de la comunidad de Apache. Algunas de sus ventajas se basan en algunas de sus principales cualidades:

- Velocidad: garantiza una alta eficiencia de procesamiento.
- Flexibilidad: se adapta a las necesidades del negocio, permite la utilización de diversas fuentes de datos y distintos tipos de datos.

- Escalabilidad: permite almacenar y distribuir conjuntos de datos inmensos en sus cientos de servidores que operan en paralelo, permitiendo olvidarse de los límites que otras alternativas imponen.
- Resistencia al fracaso: su tolerancia a errores es uno de sus atributos mejor valorados por los usuarios ya que toda la información contenida en cada nodo tiene su réplica en otros nodos del cluster. En caso de producirse un fallo siempre existirá una copia lista para ser usada.

Hadoop usa una arquitectura maestro-esclavo (Master-Slave), usando para almacenar datos Hadoop Distributed File System (HDFS) y algoritmos de MapReduce para hacer cálculos. Permite a las aplicaciones trabajar con miles de nodos y petabytes de datos. Hadoop trata de ser confiable, proveer alta disponibilidad y manejo de fallos.

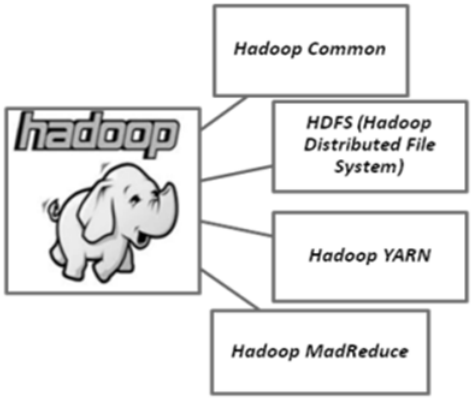

Figura 2.2: Apache Hadoop.

Los servicios de Hadoop proporcionan almacenamiento de datos, procesamiento de datos, acceso a datos, la gestión de datos, seguridad y operaciones. Posee 4 módulos esenciales, Common, HDFS, YARN y MapReduce.

2.12.1. Common

Contiene bibliotecas y otras utilidades que necesitan los otros módulos de Hadoop. Provee un conjunto de utilidades que permiten el soporte a otros proyectos de Hadoop, estas utilidades son consideradas como el núcleo del framework que provee servicios esenciales para otras aplicaciones basadas en Hadoop. [23] Proporciona abstracciones a nivel de sistema de archivos o sistema operativo. Contiene los archivos .jar (Java ARchive) y scripts necesarios para iniciar Hadoop. Al igual que todos los demás módulos, Hadoop Common maneja los fallos de hardware comunes de forma automática. El proyecto Apache Commons se compone de tres partes:

- Commons Proper: Un repositorio de componentes Java reutilizables.
- Commons Sandbox: Un espacio de trabajo para el desarrollo de componentes de Java.
- Commons Dormant: Un repositorio de componentes que se encuentran actualmente inactivas.

2.12.2. Hadoop Distributed File System (HDFS)

Es un sistema de archivos distribuido que proporciona acceso de alto rendimiento a datos. Fue creado a partir del Google File System (GFS). HDFS se encuentra optimizado para grandes flujos y trabajar con ficheros grandes en sus lecturas y escrituras. [24] La escalabilidad y disponibilidad son otras de sus claves, gracias a la replicación de los datos y tolerancia a los fallos.

2.12.2.1. Características

HDFS posee características muy útiles para el manejo de grandes volúmenes de datos:

- Es adecuado para el almacenamiento y procesamiento distribuido.
- Permite el manejo de grandes archivos que pueden pesar cientos de Mega-Bytes, Giga-Bytes, Tera-Bytes, Peta-Bytes, etc.
- Proporciona permisos de archivo y autenticación.
- Los servidores ayudan a los usuarios a comprobar fácilmente el estado del clúster.
- Hadoop proporciona una interfaz de comandos para interactuar con HDFS.
- Tolerancia a fallos: para poder tener siempre disponible los datos en caso de ser requeridos, utiliza la replicación de los datos en distintos nodos.
- Los nodos de datos pueden hablar entre ellos para reequilibrar datos, mover copias, y conservar alta la replicación de datos.

2.12.2.2. Arquitectura

HDFS sigue una arquitectura maestro-esclavo y contiene los siguientes elementos:

- Namenode: Actúa como el servidor maestro y se encarga de la administración del espacio de nombres del sistema de archivos, regula el acceso a los ficheros por parte de los clientes, ejecuta las operaciones del sistema de archivos como el cambio de nombre, cierre y apertura de archivos y directorios y realiza el mantenimiento del sistema de archivos y la metadata asociada a todos estos. Regula el acceso a los ficheros por parte de

los clientes. Mantiene en memoria la metadata del sistema de ficheros y control de los bloques de fichero que tiene cada DataNode.

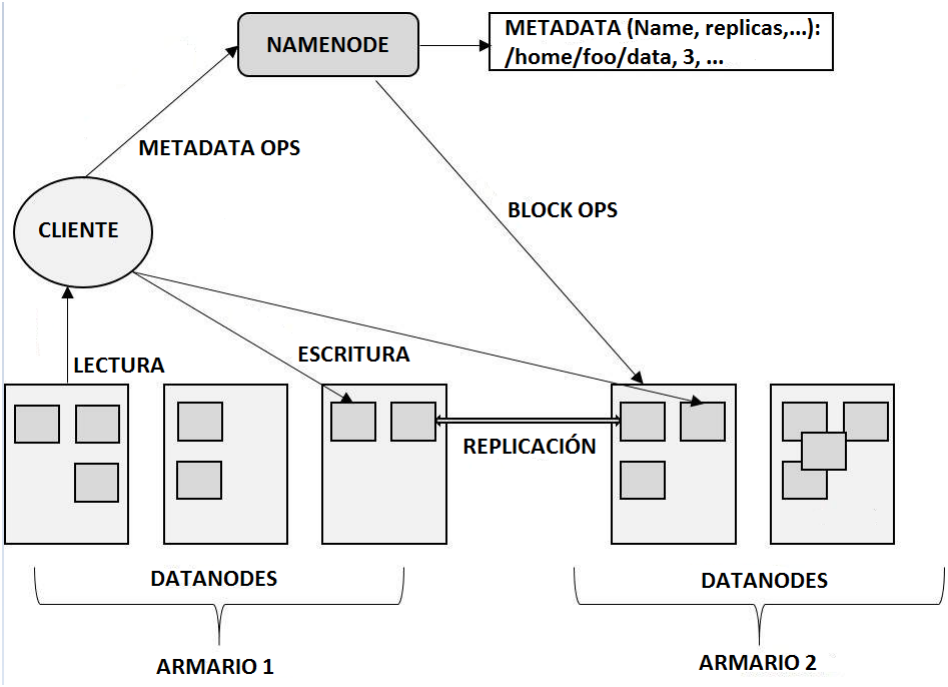

Figura 2.3: Arquitectura de HDFS.

- Datanode: Son los responsables de leer y escribir las peticiones de los clientes. Almacena y distribuye los bloques entre los distintos nodos. Permiten realizar operaciones tales como creación, supresión, con lo que la replicación de acuerdo con las instrucciones del namenode. Los ficheros están formados por bloques, estos se encuentran replicados en diferentes nodos. La distribución de los bloques es realizada cuando reciben un aviso desde un namenode o algún cliente (explicado en el siguiente punto) solicita una distribución. Una vez realizada la distribución, estos realizan un reporte al namenode, este reporte se realiza periódicamente y contiene los bloques los cuales están siendo almacenados en ese nodo en específico.

- Bloque: En general los datos de usuario se almacenan en los archivos de HDFS. El archivo en un sistema de archivos se divide en uno o más segmentos y/o almacenados en los nodos de datos. Estos segmentos se denominan como bloques. En otras palabras, la cantidad mínima de datos que HDFS puede leer o escribir se llama un bloque. El tamaño de bloque por defecto es de 64 MB, pero puede ser aumentado por la necesidad de cambiar de configuración HDFS.

- Clientes: son los usuarios del sistema de archivos

HDFS se gestiona a través de un servidor NameNode dedicado para alojar el índice de sistema de archivos y un NameNode secundario destinado a generar instantáneas de estructuras de memoria del NameNode principal. De esta manera, se evita la corrupción del sistema de archivos y la reducción de pérdida de datos.

2.12.3. Yet Another Resource Negotiator (YARN)

Apache Hadoop YARN (por las siglas en inglés de "otro negociador de recursos") es una tecnología de administración de clústeres. Facilita la planificación de tareas y gestión de recursos de clúster. Nace para dar solución a una idea fundamental: Dividir las dos funciones principales del JobTracker (NameNode), es decir, tener en servicios o demonios totalmente separados e independientes la gestión de recursos por un lado y, por otro, la planificación y monitorización de las tareas o ejecuciones. [25]

Gracias a YARN, Hadoop tiene un entorno de gestión de recursos y aplicaciones distribuidas dónde se pueden implementar múltiples aplicaciones de procesamiento de datos totalmente personalizadas y específicas para realizar una tarea en cuestión. YARN combina un administrador central de recursos que reconcilia la forma en que las aplicaciones utilizan los recursos del sistema de Hadoop con los agentes de administración de nodo que monitorean las operaciones de procesamiento de nodos individuales del clúster. Ejecutándose en clústeres de hardware básicos, Hadoop ha atraído un interés particular como zona de espera y de almacenamiento de datos para grandes volúmenes de datos estructurados y no estructurados destinados al uso en aplicaciones de analítica. Separar HDFS de MapReduce con YARN hace al ambiente Hadoop más adecuado para las aplicaciones operativas que no pueden esperar para que terminen los trabajos por lotes.

2.12.3.1. Características

- Procesamiento: soporta distintos modelos de procesamiento y posee la capacidad de adaptarse a muchos más.
- Compatibilidad: las aplicaciones existentes de MapReduce pueden correr en YARN sin inconvenientes.
- Escalabilidad: YARN evita ciertos problemas de cuello de botella en grandes clúster.

2.12.3.2. Arquitectura

Los elementos que intervienen son:

- Resource Manager: asigna los recursos del clúster de manera abstracta a través de contenedores (Containers) los cuales incorporan elementos como memoria, CPU, disco, red, entre otros.
- Node Manager: hay uno por nodo esclavo, es el responsable de la monitorización y gestión de los recursos. Recoge las directrices del ResourceManager y crea contenedores basado en los requerimientos de la tarea.
- Application master: se despliega junto al NodeManager. Controla la monitorización y la ejecución de las tareas usando el contenedor. Puede ser alguna librería específica de algún framework que es la encargada de negociar recursos con el ResourceManager y coordinar las tareas con el NodeManager. Proporciona la tolerancia a fallos a nivel de tarea.

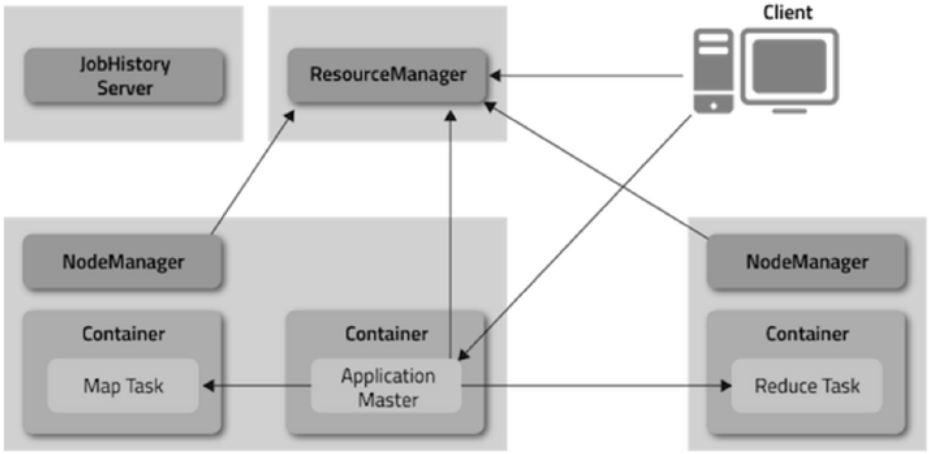

Figura 2.4: Arquitectura de YARN.

- Container: es la unidad básica de la asignación. El contenedor se define con atributos como la memoria, CPU, disco, entre otros, aplicaciones como procesamiento gráfico y MPI.
- History Server: mantiene la historia de todos los Jobs.

2.12.4. MapReduce

MapReduce fue desarrollado por Google y expuesto al resto del mundo en una publicación hecha por ellos mismos en diciembre del 2004, Google usa MapReduce para indexar páginas web. MapReduce es un modelo de programación con una implementación asociada al procesamiento y generación de grandes cantidades de datos. Los usuarios especifican una función de map que procesa pares clave/valor para generar un grupo intermedio de pares clave/valor y una

función de reduce que combina todos los valores intermedios. [26]

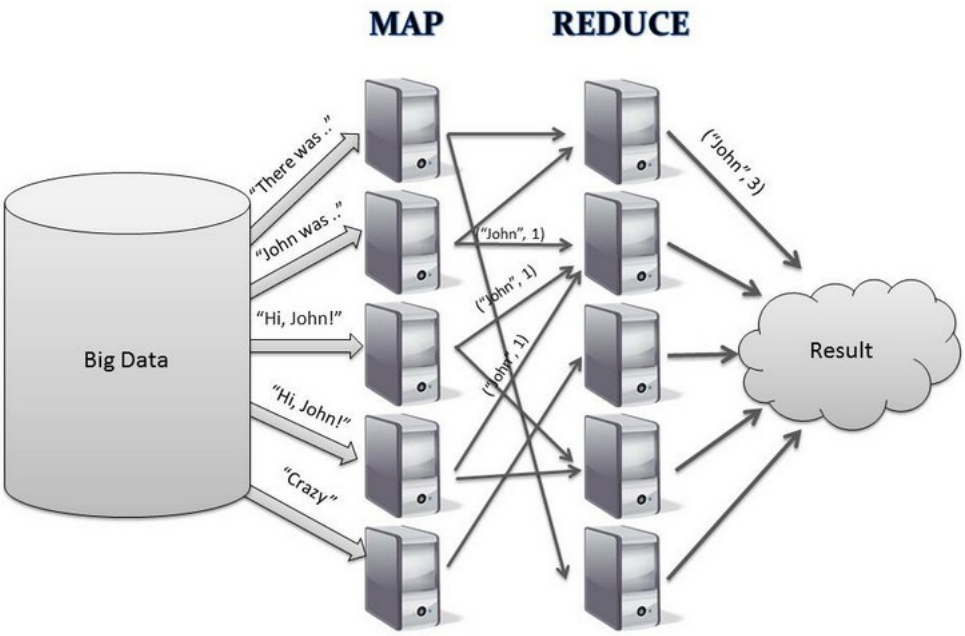

Figura 2.5: Ejemplo de MapReduce.

Es un subproyecto del proyecto Hadoop. Posee una arquitectura maestro-esclavo y es un paradigma de programación que permite la escalabilidad masiva a través de cientos o miles de servidores en un clúster Hadoop. Cuenta con un servidor maestro o JobTracker y varios servidores esclavos o TaskTrackers, uno por cada nodo del clúster. El JobTracker es el punto de interacción entre los usuarios y el framework MapReduce. Los usuarios envían trabajos MapReduce al JobTracker, que los pone en una cola de trabajos pendientes y los ejecuta en el orden de llegada.

El JobTracker gestiona la asignación de tareas y delega las tareas a los TaskTrackers. Los TaskTrackers ejecutan tareas bajo la orden del JobTracker y también manejan el movimiento de datos entre la fase Map y Reduce.

2.12.4.1. JobTracker

- Capacidad para manejar metadatos de trabajos.
- Estado de la petición del trabajo.

- Estado de las tareas que se ejecutan en TaskTracker.
- Decide sobre la programación.
- Hay exactamente un JobTracker por cluster.
- Recibe peticiones de tareas enviadas por el cliente.
- Programa y monitoriza los trabajos MapReduce con TaskTrackers.

2.12.4.2. TaskTracker

- Ejecuta las solicitudes de trabajo de JobTrackers.
- Obtiene el código que se ejecutará.
- Aplica la configuración especifica del trabajo.
- Comunicación con el JobTracker: Envíos de la salida, finalizar tareas, actualización de tareas, etc.

2.12.4.3. Map

La función Map recibe como parámetros un par de (clave, valor) y devuelve una lista de pares. Esta función se encarga del mapeo y se aplica a cada elemento de la entrada de datos, por lo que se obtendrá una lista de pares por cada llamada a la función Map. Después se agrupan todos los pares con la misma clave de todas las listas, creando un grupo por cada una de las diferentes claves generadas. No hay requisito de que el tipo de datos para la entrada coincida con la salida y no es necesario que las claves de salida sean únicas.

2.12.4.4. Reduce

La función Reduce se aplica en paralelo para cada grupo creado por la función Map().La función Reduce se llama una vez para cada clave única de la salida de la función Map. Junto con esta clave, se pasa una lista de todos los valores asociados con la clave para que pueda realizar alguna fusión para producir un conjunto más pequeño de los valores.

Cuando se inicia la tarea Reduce, la entrada se encuentra dispersa en varios archivos a través de los nodos en las tareas de Map. Los datos obtenidos de la fase Map se ordenan para que los pares clave-valor sean contiguos (fase de ordenación, sort fase), esto hace que la operación Reduce se simplifique ya que el archivo se lee secuencialmente.

Si se ejecuta el modo distribuido estos necesitan ser primero copiados al filesystem local en la fase de copia. Una vez que todos los datos están disponibles a nivel local se adjuntan a una fase de adición, el archivo se fusiona (merge) de forma ordenado. Al final, la salida consistirá en un archivo de salida por tarea

reduce ejecutada.

Por lo tanto, N archivos de entrada generará M mapas de tareas para ser ejecutados y cada mapa de tareas generará tantos archivos de salida como tareas Reduce hayan configuradas en el sistema.

2.12.5. Ecosistema Hadoop

El ecosistema de Hadoop posee un conjunto de herramientas muy diverso, que crece día tras día, por lo que es difícil saber de todos los proyectos que interactúan con Hadoop de alguna forma, las cuales a su vez poseen subconjuntos, en este capítulo mencionaremos algunos de ellos y una breve descripción.

2.12.5.1. Administración de los datos

Son herramientas que permiten el almacenamiento y el manejo de datos, como HDFS y YARN que fueron mencionados anteriormente.

2.12.5.2. Acceso a los datos

Son herramientas que se encargan del acceso a los datos por parte de los usuarios y permiten la lectura, escritura y procesamiento de los datos.

2.12.5.2.1. APACHE ACCUMULO

Apache Accumulo es un store clave/valor distribuido, escalable y de alto rendimiento. Se basa en el diseño de Google BigTable y se construye sobre Hadoop, Zookeeper y Thrift. [27]

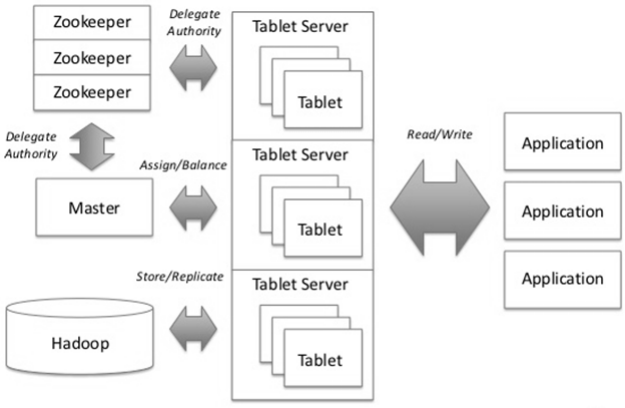

Figura 2.6: Arquitectura Apache Accumulo.

Accumulo permite el manejo de datos a nivel de celdas, lo cual es una funcionalidad muy importante debido a que se puede restringir el acceso a los datos a ciertos usuarios en específico. Permite también la mezcla de distintos datos los cuales pueden estar restringidos o no, las reglas que pueden ser aplicadas a los datos pueden llegar a ser muy específicas.

2.12.5.2.2. APACHE HBASE

HBase, se trata de la base de datos de Hadoop. HBase es el componente de Hadoop a usar cuando se requiere escrituras/lecturas en tiempo real y acceso aleatorio para grandes conjuntos de datos. Es una base de datos distribuida orientada a columnas que funciona sobre HDFS, eso quiere decir que no sigue el esquema relacional. [28]

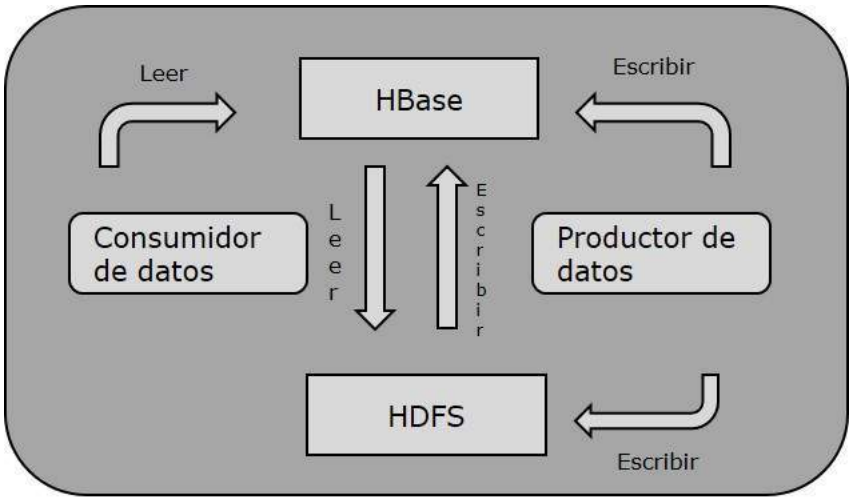

Figura 2.7: Modelo utilizando Apache HBase.

Características de HBase:

- Escalabilidad linear y modular.
- Soporte para caídas de nodos.
- Proporciona lectura coherente y escrituras.
- Se integra con Hadoop, tanto como un origen y un destino.
- Compatibilidad con trabajos MapReduce.
- Proporciona replicación de datos en clústeres.

HDFS	HBase
HDFS es un sistema de ficheros distribuido adecuado para almacenar archivos de gran tamaño.	HBase es una base de datos creada en la parte superior de la HDFS.
HDFS no admite búsquedas rápidas registro individual.	HBase proporciona búsquedas rápidas tablas más grandes.
Proporciona una alta latencia procesamiento por lotes; un concepto de procesamiento por lotes.	Proporciona acceso de baja latencia a filas de miles de millones de registros (acceso aleatorio).
Sólo proporciona acceso secuencial de los datos.	HBase internamente usa tablas Hash y proporciona acceso aleatorio, y que almacena los datos en archivos indexados HDFS búsquedas más rápido.

Tabla 2.2: Comparación entre HBase y HDFS.

Las aplicaciones de HBase:

- Apache HBase se utiliza para tener al azar y en tiempo real de acceso de lectura/escritura a los grandes datos.
- Alberga las tablas de gran tamaño en la parte superior de los grupos de hardware de productos básicos.
- Se usa cuando es necesario escribir aplicaciones pesadas.
- HBase se utiliza cada vez que necesitemos para proporcionar un rápido acceso aleatorio a los datos disponibles.
- Empresas como Facebook, Twitter, Yahoo y Adobe usan HBase internamente.

2.12.5.2.3. APACHE HIVE

Hive es un sistema de Data Warehouse para Hadoop que facilita el uso de la agregación de los datos, ad-hoc queries, y el análisis de grandes datasets almacenados en Hadoop. Hive proporciona métodos de consulta de los datos usando un lenguaje parecido al SQL, llamado HiveQL. Además permite de usar los tradicionales MapReduce cuando el rendimiento no es el correcto. Tiene interfaces JDBC/ODBC, por lo que empieza a funcionar su integración con herramientas de inteligencia de negocios. [29]

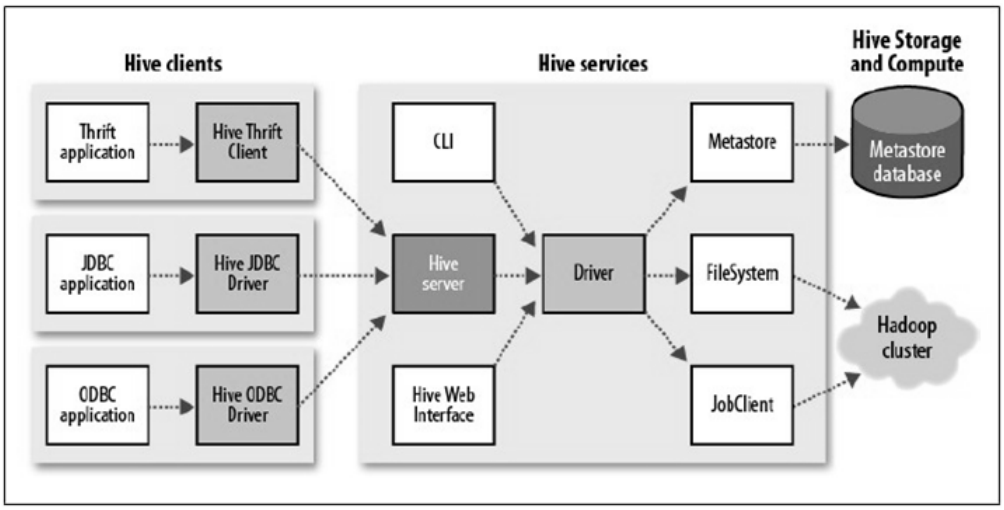

Figura 2.8: Arquitectura Apache Hive.

Características de Hive:

- Esquema que almacena en una base de datos y se procesan los datos en HDFS.
- Está diseñado para OLAP.
- Proporciona un lenguaje de consulta parecido a SQL, HiveQL (HQL).
- Es familiar, rápido, escalable y extensible.

2.12.5.2.4. APACHE STORM

Apache Storm es un sistema que sirve para recuperar streams de datos en tiempo real desde múltiples fuentes de manera distribuida, tolerante a fallos y en alta disponibilidad. Storm está principalmente pensado para trabajar con datos que deben ser analizados en tiempo real, por ejemplo datos de sensores que se emiten con una alta frecuencia o datos que provengan de las redes sociales donde a veces es importante saber qué se está compartiendo en este momento. [30]

Se compone de dos partes principalmente. La primera es la que se denomina Spout y es la encargada de recoger el flujo de datos de entrada. La segunda se denomina Bolt y es la encargada del procesado o transformación de los datos.

2.12.5.2.5. APACHE MAHOUT

Es una librería de los algoritmos más famosos de aprendizaje automático (Machine Learning), fue implementado sobre Hadoop utilizando el paradigma de MapReduce. Con los datos almacenados en HDFS, Mahout provee las herramientas necesarias para poder aplicar ciencia de datos sobre dichos datos. Mahout también proporciona bibliotecas de Java para las operaciones matemáticas comunes (centrado en álgebra lineal y estadística) y las colecciones de Java primitivos. [31]

Algunos de los algoritmos que provee Mahout son los siguientes:

Algoritmo	Descripción breve	Caso de uso
Regresión logística, resuelta por gradiente estocástico descendiente (SGD).	Clasificador brillante, rápido, simple y secuencial, capaz de aprendizaje online en entornos exigentes.	Recomiende publicidad a los usuarios, clasifique texto en categorías.
Modelos ocultos de Markov (HMM).	Implementaciones secuenciales y paralelas del algoritmo clásico de clasificación diseñado para modelar procesos del mundo real cuando el proceso de generación subyacente es desconocido.	Etiquetado de texto parte-del-discurso; reconocimiento del discurso.
Descomposición de valor singular (SVD).	Diseñado para reducir el ruido en matrices grandes, haciendo con esto que sean más pequeñas y que sea más fácil trabajar con ellas.	Como precursor del almacenamiento en clúster, los recomendadores y la clasificación para realizar selección de recursos automáticamente.
Almacenamiento en clúster Dirichlet.	Enfoque de almacenamiento en clúster basado en modelo, que determina la propiedad con base en si los datos se ajustan al modelo subyacente.	Útil cuando los datos tienen sobreposición o jerarquía.
Almacenamiento en clúster espectral.	Es una familia de enfoques similares que usa un enfoque basado en gráficas para determinar la membresía a clúster.	Como todos los algoritmos de almacenamiento en clúster, es útil para explorar conjuntos de datos grandes y no vistos.
Almacenamiento en clúster Minhash.	Utiliza una estrategia de hash para agrupar elementos similares, produciendo así clústeres.	Igual a otros enfoques de clúster.
Numerosas mejoras de recomendador.	Co-ocurrencia distribuida, SVD, mínimos cuadrados alternantes.	Sitios de citas, e-commerce, recomendaciones de películas o de libros.
Colocaciones.	Implementación de colocación reducida por correlacionamiento.	Encontrando frases estadísticamente interesantes en texto.

Tabla 2.3: Algoritmos Mahout.

2.12.5.3. Integración

Aquellas herramientas que facilitan la extracción, transformación, replicación, entre otros, de los datos, son las herramientas de integración.

2.12.5.4. Operaciones

2.12.5.5. Seguridad

2.12.6. Distribuciones Hadoop

2.12.6.1. Cloudera

2.12.6.2. Hortonworks

2.12.6.3. MapR

2.13. Apache Spark

2.13.1. Resilient Distributed Dataset (RDD)

2.14. Grafo

Un grafo es básicamente un conjunto no vacío (al menos contiene un elemento) de puntos llamados vértices y un conjunto de líneas llamadas aristas cada una de las cuales une dos vértices. Se llama lazo a una arista que une un vértice consigo mismo. Se dice que dos vértices son adyacentes si existe una arista que los une. Se dice que un grafo es simple si para cualesquiera dos vértices existe a lo sumo una arista que los une. En otro caso se denomina multigrafo. Si v es un vértice de un grafo, se denomina grado de v al número de aristas que inciden en el mismo (por convenio de considera que un lazo cuenta dos veces al determinar el grado de su vértice).[32]

Algunos ejemplos que pueden ser representado como grafos, son los siguientes:

- Redes tecnológicas (technological networks).
- Redes biológicas (biological networks).
- Redes sociales (social networks).
- Redes de información (information networks).
- Redes de idiomas (language networks).
- Redes de información humana (human information network).
- Ciudades inteligentes (smart cities).

2.14.1. Redes tecnológicas (technological networks)

Son las redes diseñadas para la distribución de electricidad (energía), agua, gas, las redes de transportes (carreteras, ferrocarril, rutas aéreas), las redes

telefónicas e internet (sólo las redes físicas de cables y postes, puesto que las redes de llamadas formarían parte de las denominadas redes sociales).

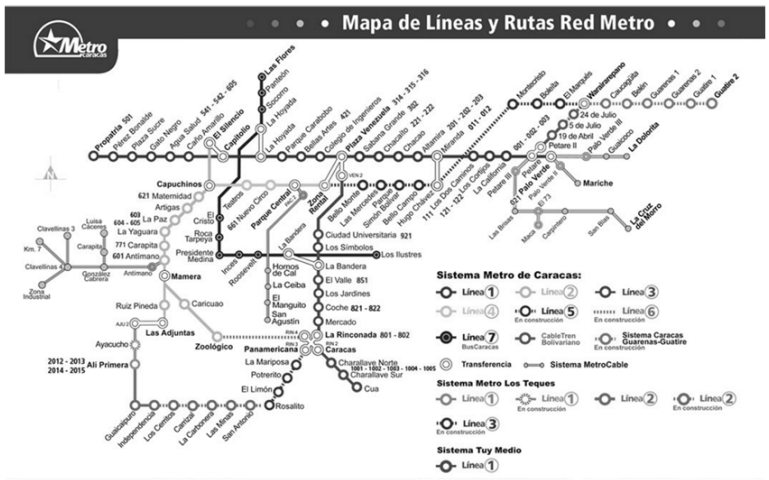

Figura 2.9: Red tecnológica. Red de metro.

2.14.2. Redes de información (information networks)

También denominadas redes de conocimiento. El ejemplo clásico de redes reales de esta categoría son las de citas o referencias de trabajos científicos. Otro ejemplo, ampliamente estudiado de redes de información es la World Wide Web, que es una red que contiene páginas informativas que se enlazan a través de hipervínculos. Al igual que las redes de citas, en la www también influyen aspectos sociales que trascienden el mero interés informativo de los vínculos.

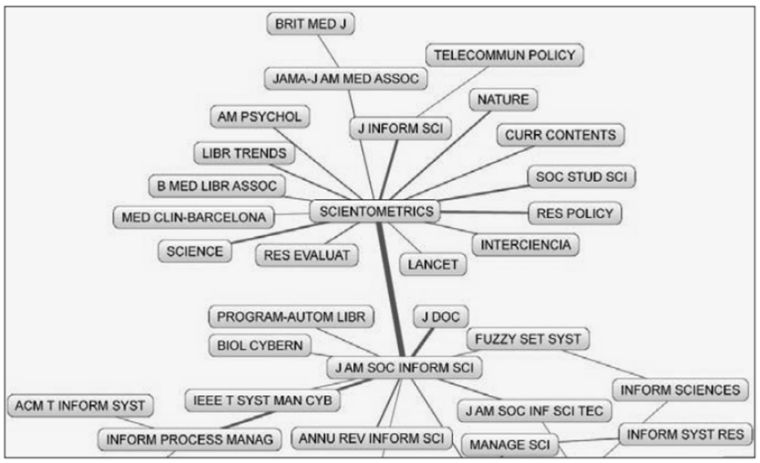

Figura 2.10: Red de información. Cocitación de revistas.

2.14.3. Redes sociales (social networks)

Las redes sociales están compuestas por individuos o grupos de individuos con patrones de contactos o interacciones entre ellos. Ejemplos de este tipo de redes son las relaciones de amistad, de negocios entre directivos de empresas, o entre familias a partir de sus matrimonios y descendencia (genealogías). Al análisis de este tipo de redes se asocian a menudo dificultades de imprecisión y subjetividad, debidas al reducido tamaño de las muestras que emplean y a los métodos utilizados para la recogida de datos: generalmente encuestas, cuestionarios o entrevistas. Para superar estas dificultades los investigadores han probado nuevos métodos de investigación en busca de muestras más numerosas y fiables mediante la utilización de grandes bases de datos.

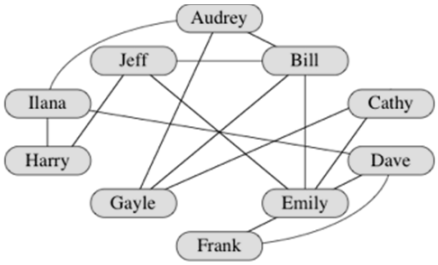

Figura 2.11: Red social. Colaboración científica.

2.14.4. Redes biológicas (biological networks)

Los expertos en biomedicina quieren realizar descubrimientos científicos, un caso es, descubrir relaciones desconocidas . Por ejemplo, tomar dos enfermeda-

des, enfermedades muy diferentes, cáncer colorrectal y la enfermedad de Alzheimer, y puede que tengan algún tipo de relación. Otros ejemplos son, las relaciones entre genes y proteínas, interacciones entre genes, señalización celular y asociaciones entre las enfermedades.

Son diversos los sistemas biológicos susceptibles de representarse en forma de redes. Las redes de reacciones metabólicas, las redes genéticas, los ecosistemas y cadenas tróficas, las redes neuronales o las vasculares son algunos de los ejemplos de redes biológicas analizadas desde la perspectiva de la teoría de redes. Las redes alimentarias, por ejemplo, pueden ser descritas como un grafo con un conjunto finito de nodos (especies) y un conjunto finito de enlaces que asocian cada uno de esos nodos entre sí.

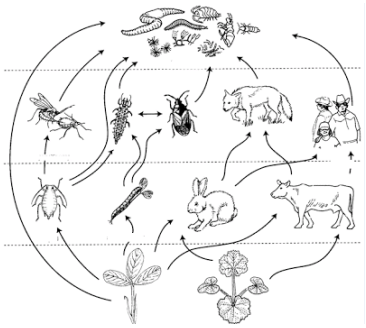

Figura 2.12: Red biológica. Cadena trófica.

2.14.5. Ciudades inteligentes (smart cities)

Una ciudad es un espacio geográficamente limitado, y contiene muchas redes diferentes que operan dentro del mismo dominio espacial. Cuenta con las redes de transporte, agua, cloacas, transmisión de energía, entre otros. Algunas de estas redes tienen múltiples subtipos, por ejemplo, las redes de transporte incluyen las redes de autobuses, metro, carreteras, ferroviaria, y así sucesivamente. Una ciudad inteligente se refiere a un tipo de desarrollo urbano basado en la sostenibilidad que es capaz de responder adecuadamente a las necesidades básicas de instituciones, empresas, y de los propios habitantes, tanto en el plano económico, como en los aspectos operativos, sociales y ambientales. Estas redes forman una infraestructura física y por lo tanto pueden ser representados mediante grafos, donde cada nodo tiene una coordenada geográfica.

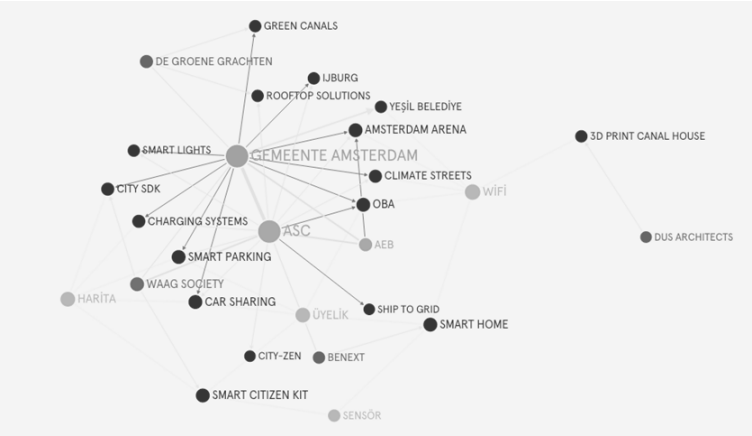

Figura 2.13: Ciudades inteligentes.

2.14.6. Teoría de grafos

La teoría de grafos es un campo de estudio de las matemáticas y las ciencias de la computación, que estudia las propiedades de los grafos. La teoría de grafos es una rama de las matemáticas discretas y de las matemáticas aplicadas, y es un tratado que usa diferentes conceptos de diversas áreas como combinatoria, álgebra, probabilidad, geometría de polígonos, aritmética y topología.

El origen de la teoría de grafos se remonta al siglo XVIII con el problema de los puentes de Königsberg, el cual consistía en encontrar un camino que recorriera los siete puentes del río Pregel en la ciudad de Königsberg, actualmente Kaliningrado, de modo que se recorrieran todos los puentes pasando una sola vez por cada uno de ellos.

La teoría de grafos se usa para la solución de problemas de biología, genética, automatización, entre otros, y ha servido de inspiración para las ciencias sociales, en especial para desarrollar un concepto no metafórico de red social que sustituye los nodos por los actores sociales y verifica la posición, centralidad e importancia de cada actor dentro de la red. Esta medida permite cuantificar y abstraer relaciones complejas, de manera que la estructura social puede representarse gráficamente. Por ejemplo, una red social puede representar la estructura de poder dentro de una sociedad al identificar los vínculos (aristas), su dirección e intensidad y da idea de la manera en que el poder se transmite y a quiénes.

2.14.7. Distancia geodésica

La distancia geodésica entre un par de nodos en un grafo es la longitud del camino más corto entre los dos nodos, y es la base para definir el diámetro de

un grafo. En otras palabras, es la distancia entre dos nodos de un grafo es la longitud del camino más corto

2.14.8. Medidas de centralidad

2.14.8.1. Centralidad de grado o grado nodal (Degree centrality)

Es el número de actores a los cuales un actor esta directamente conectado. Se divide en grado de entrada y salida:

- Grado de salida: es la suma de las relaciones que los actores dicen tener con el resto. Por ejemplo, Pedro dice tener relación con Elohina, Andres y Carlos, por lo cual su grado de salida es 3.
- Grado de entrada: es la suma de las relaciones referidas hacia un actor por otros. Por ejemplo, Israel es mencionado por 4 personas (Rafael, Sebastian, Miguel y Leo), por lo tanto su grado de entrada es de 4.

2.14.8.2. Centralidad de intermediación (Betweenness centrality)

Se interpreta como la posibilidad que tiene un nodo o actor para intermediar las comunicaciones entre pares de nodos. En este análisis se consideran todos los posibles caminos geodésicos entre todos los pares posibles. La medida de intermediación de un nodo se obtiene contando las veces que este aparece en los caminos geodésicos que conectan a todos los pares de nodos de la red, a estos actores se les denomina actores puente.

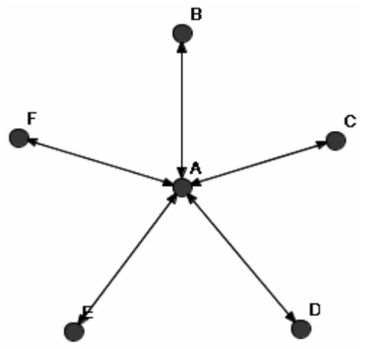

Figura 2.14: Centralidad de intermediación.

En la figura 2.6 podemos visualizar un ejemplo, en donde el nodo A aparece en todos los caminos posibles para que los demás nodos puedan conectarse (F a B, F a C, F a D, F a E, B a C, B a D, B a E, C a D, C a E y D a E), por lo tanto el grado de intermediación de a es 10 y del resto de los actores es 0.

2.14.8.3. Centralidad de cercanía (Closeness centrality)

Es la capacidad que tiene un nodo de llegar a todos los actores de una red en particular, se calcula contando todas las distancias geodésicas de un actor para llegar a los demás.

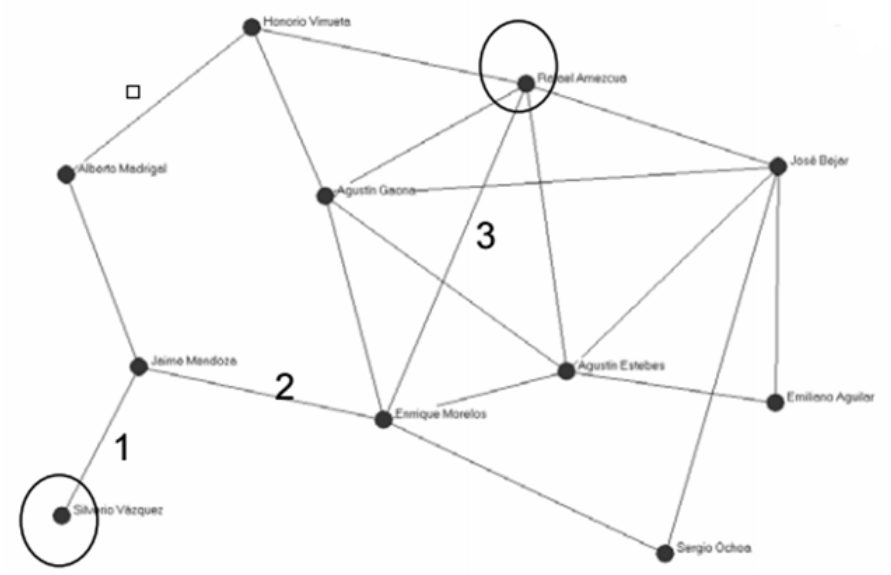

Figura 2.15: Centralidad de cercanía.

En las figuras 2.7 y 2.8 podemos observar que cada nodo posee un valor para cada uno de sus compañeros, este valor es la distancia geodésica, la suma de estas distancias tiene el nombre de lejanía y la cercanía es el inverso de la lejanía, es decir 1/lejanía.

	Jaime Mendoza	Emiliano Aguilar	Rafael Amezcua	Alberto Madrigal	Silverio Vázquez	Honorio Virrueta	Agustín Gaona	Sergio Ochoa	Agustín Estebes	Enrique Morelos	José Bejar	**Lejanía**	**Cercanía**
Jaime Mendoza		3	2	1	1	2	2	2	2	1	3	**19**	**52.6**
Emiliano Aguilar	3		2	4	4	3	2	2	1	2	1	**24**	**41.7**
Rafael Amezcua	2	2		2	3	1	1	2	1	1	1	**16**	**62.5**
Alberto Madrigal	1	4	2		2	1	2	3	3	2	3	**23**	**43.5**
Silverio Vázquez	1	4	3	2		3	3	3	3	2	4	**28**	**35.7**
Honorio Virrueta	2	3	1	1	3		1	3	2	2	2	**20**	**50.0**
Agustín Gaona	2	2	1	2	3	1		2	1	1	1	**16**	**62.5**
Sergio Ochoa	2	2	2	3	3	3	2		2	1	1	**21**	**47.6**
Agustín Estebes	2	1	1	3	3	2	1	2		1	1	**17**	**58.8**
Enrique Morelos	1	2	1	2	2	2	1	1	1		2	**15**	**66.7**
José Bejar	3	1	1	3	4	2	1	1	1	2		**19**	**52.6**

Figura 2.16: Cálculo de cercanía de una red simetrizada.

2.14.8.4. Centralidad de vector propio o autovector (Eigenvector centrality)

2.14.8.5. Centralidad de Bonacich

2.14.8.6. Centralidad armónica

2.14.8.7. Centralidad de Katz

2.14.9. Detección de comunidades

2.14.9.1. Métodos jerárquicos

2.14.9.1.1. Aglomerativos

2.14.9.1.2. Newman-Girvan

2.14.9.1.3. Radicchi

2.14.9.2. Métodos modulares

2.14.9.2.1. Modularidad Q

2.14.9.2.2. Algoritmo greedy

2.14.9.2.3. Algoritmo Fast Greedy

2.14.9.2.4. Algoritmo MultiStep Greedy

2.14.9.3. Métodos particionales

CAPÍTULO 3

MARCO METODOLÓGICO

Para implementar la solución de un problema es importante tener un orden y establecer distintas prioridades, es decir una metodología. Realizar análisis sobre grafos de gran escala puede ser complejo, por lo tanto realizar un conjunto de actividades estructurada y metódica es de gran ayuda.

Específicamente para este trabajo de investigación se decidió utilizar la Metodología Fundamental para la Ciencia de Datos propuesta por la empresa IBM. Consiste en los siguientes pasos:

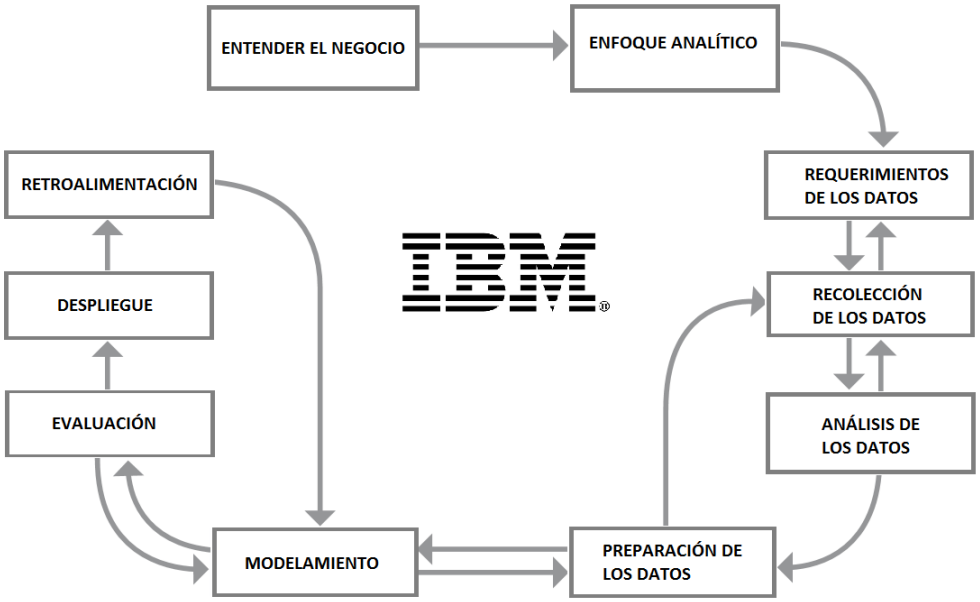

Figura 3.1: Metodología para Ciencia de los Datos.

- Entender el negocio: Todo proyecto, independientemente de su tamaño, comienza con la comprensión del negocio y establecer las bases para una solución exitosa del problema. Los usuarios finales deben plantear los objetivos del proyecto y los requisitos de la solución, por lo tanto es de suma importancia una fase de análisis.

- Enfoque analítico: Una vez que el problema de la empresa se ha establecido claramente, el científico de datos puede definir el enfoque analítico para resolver el problema. Es decir, expresar el problema en un contexto basado en técnicas estadísticas, algoritmos e infraestructura para que el científico de datos puede identificar las técnicas adecuadas para lograr el resultado deseado.

- Requerimientos de los datos: El enfoque analítico elegido determina los requisitos de los datos . En concreto, los métodos analíticos que se utilizan requieren un conjunto de datos, formatos y representaciones, guiados por el conocimiento basados en el dominio.

- Recolección de los datos: En la etapa inicial de la recopilación de datos, los científicos de datos identifican y reúnen los recursos o datos estructurados, no estructurados y semi-estructurados que sean relevantes para el problema. Por lo general, tienen que elegir si desea realizar inversiones adicionales para obtener datos menos accesibles o con características particulares. Si faltan datos en el proceso de recopilación, el científico de datos puede tener que revisar los requerimientos de los datos en consecuencia, imputar, recoger más y/o nuevos datos.

- Análisis de los datos: Utilizando estadística descriptiva y técnicas de visualización puede ser de ayuda para que el científico de datos comprenda el contenido de los datos, evalúe la calidad de los datos y descubra patrones en los datos. Una nueva visita de la etapa anterior, la recopilación de datos, podría ser necesario para completar adecuadamente esta fase.

- Preparación de los datos: La etapa de preparación de datos comprende todas las actividades que se utilizan para construir el conjunto de datos que se para la etapa de modelado. Estos incluyen la limpieza de datos, combinando datos de múltiples fuentes y transformar datos. La etapa de preparación de datos es el que más tiempo consume, aproximadamente 80 % del tiempo total del proyecto.

  Sin embargo, se puede reducir el tiempo si los recursos de datos están bien gestionados, integrados y limpios. Automatizar algunos pasos de preparación de los datos puede reducir el tiempo aún más.

- Modelamiento: A partir de la primera versión del conjunto de datos preparados, la etapa de modelado se centra en el desarrollo de modelos predictivos o descriptivos de acuerdo al enfoque analítico previamente definido. Con los modelos predictivos, los científicos utilizan un conjunto de datos

de entrenamiento (datos históricos en los que se conoce el resultado de interés o se tiene la tiene columna clase) para construir el modelo . El proceso de modelado es típicamente muy iterativo. Para una determinada técnica , los científicos de datos pueden probar varios algoritmos con sus respectivos parámetros para encontrar el mejor modelo para las variables disponibles .

- Evaluación: Durante el desarrollo del modelo y antes de la implementación, el científico de datos evalúa el modelo y asegura que de forma apropiada y completa aborda el problema de negocio. La evaluación implica el cálculo de diversas medidas de diagnóstico, tablas y gráficos, que permiten al científico de datos interpretar la calidad del modelo y su eficacia en la solución del problema. Para un modelo predictivo, los científicos de datos utilizan un conjunto de pruebas, que es independiente del conjunto de entrenamiento, pero sigue la misma distribución de probabilidad y tiene un resultado conocido. El conjunto de prueba se utiliza para evaluar el modelo de lo que puede ser refinado según sea necesario.

- Despliegue: Después de un modelo satisfactorio desarrollado y es aprobado por los usuarios finales, se implementa un entorno de prueba. La implementación puede ser tan simple como la generación de un informe con recomendaciones, o tan complicado como la incorporación del modelo en un complejo proceso de flujo de trabajo.

- Retroalimentación: Mediante la recopilación de los resultados del modelo implementado, se recibe información sobre el desempeño. El análisis de esta información permite a los científicos de datos para refinar el modelo para mejorar su precisión y utilidad. Se pueden automatizar algunos o todos los pasos de retroalimentación, recopilación y evaluación de modelo.

Adaptando esta metodología al presente trabajo especial de grado, los pasos quedan de la siguiente manera:

- Entender el negocio: ...
- Enfoque analítico: ...
- Requerimientos de los datos: ...
- Recolección de los datos: ...
- Análisis de los datos: ...
- Preparación de los datos: ...
- Modelamiento: ...
- Evaluación: ...

- Despliegue: ...
- Retroalimentación: ...

CAPÍTULO 4

MARCO APLICATIVO

4.1. Instalación

El primer paso es instalar una herramienta que proporcione un software de virtualización, en este proyecto se utilizó VMware Workstation Pro 12, el cual fue descargado del siguiente enlace `https://my.vmware.com/web/vmware/info?slug=desktop_end_user_computing/vmware_workstation_pro/12_0`. La instación de VMware se desarrolló en el sistema operativo Windows 10.

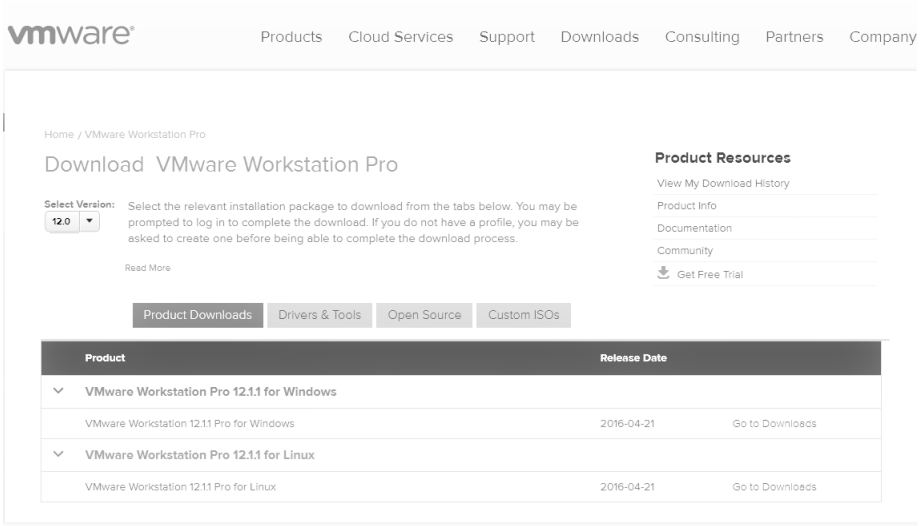

Figura 4.1: Descarga e instalación de VMware Workstation Pro 12.

El siguiente paso fue descargar una máquina virtual que provee la distribución Cloudera que provee todos los componentes y configuraciones necesarias para las diversas implementaciones que se desarrollaron. El enlace es el siguiente

`http://www.cloudera.com/downloads.html`.

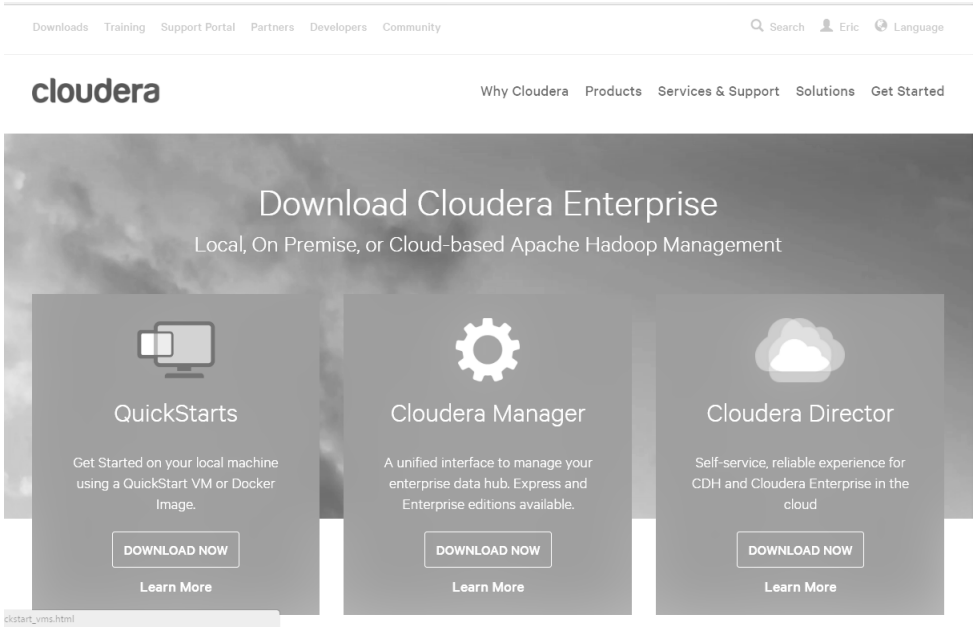

Figura 4.2: Descarga de Cloudera Quickstart VM CDH 5.7 Parte I.

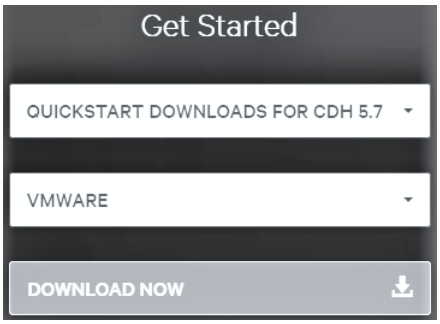

Figura 4.3: Descarga de Cloudera Quickstart VM CDH 5.7 Parte II.

Posteriormente se ejecuta VMware Workstation Pro 12 y se seleccionó la opción de Abrir una máquina virtual (Open a Virtual Machine).

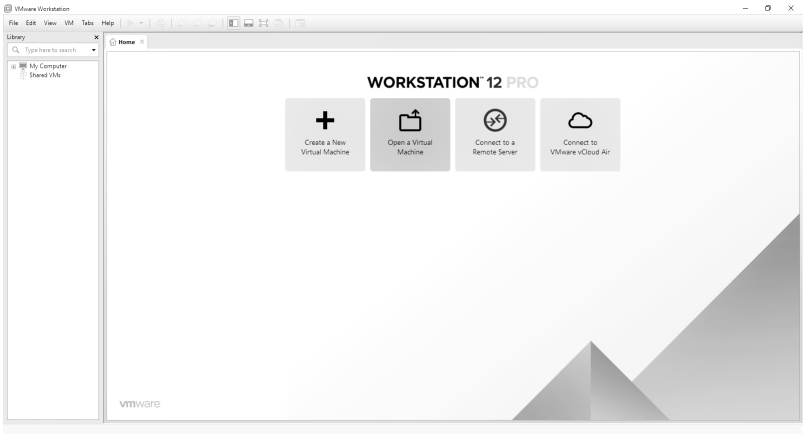

Figura 4.4: Abrir una máquina virtual.

Se utilizó la máquina virtual Cloudera Quickstart VM CDH 5.7, y finalmente se inició.

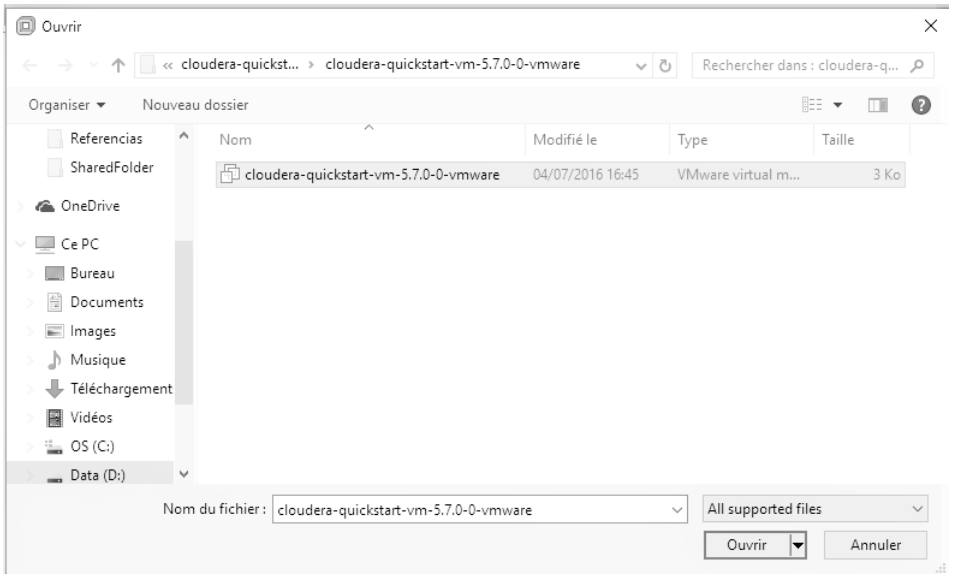

Figura 4.5: Seleccionar una máquina virtual.

4.2. Entorno

Se creo una carpeta con todos los componentes necesarios.

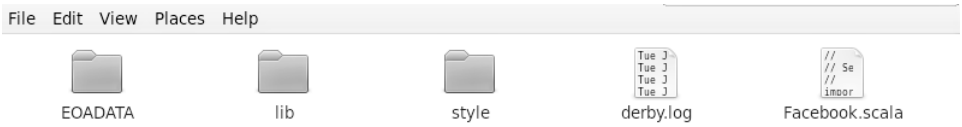

Figura 4.6: Conjunto de archivos utilizados.

En la carpeta EOADATA se encuentran todos los datasets y el script Facebook.scala se encuentra la implementación desarrollada para el análisis de un grafo que representa un subconjunto de la red social, Facebook. En la carpeta lib se encuentran todas las bibliotecas utilizadas.

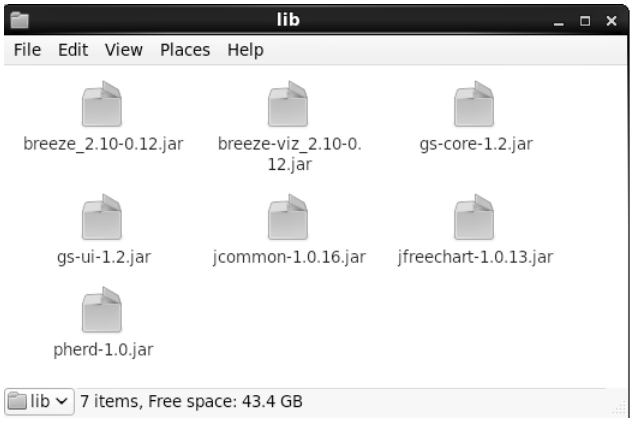

Figura 4.7: Bibliotecas utilizadas.

Breeze es un conjunto básico de bibliotecas para ScalaNLP, incluye algoritmos de álgebra lineal, cálculo numérico y optimización. Permite un enfoque genérico, potente y aún así eficiente para el aprendizaje automático y breeze-viz es un repositorio que permite visualizar diversas figuras.

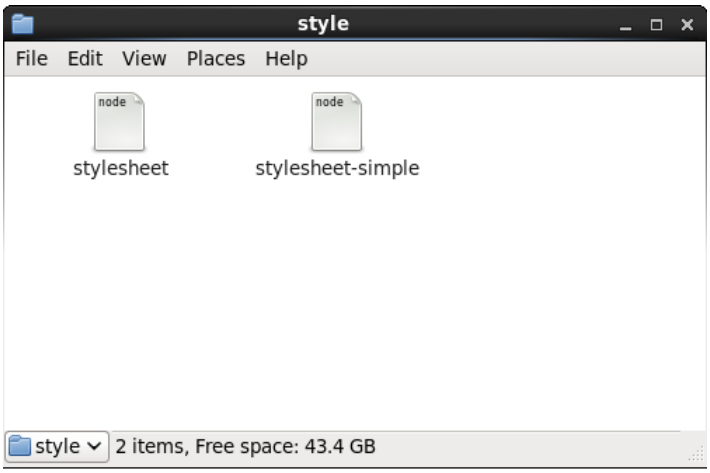

Figura 4.8: Style.

4.3. Conjunto de datos de prueba (datasets)

En la carpeta EOADATA se encuentran todos los datasets utilizados.

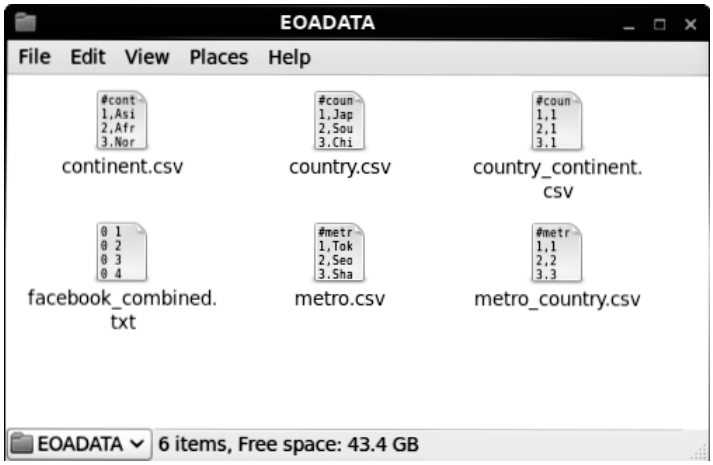

Figura 4.9: Datasets.

4.4. Implementación

En primer lugar almacenamos los datasets en HDFS.

```
[cloudera@quickstart ExamplesOfAnalytics]$ hdfs dfs -put EOADATA
```

Figura 4.10: Almacenar datos en HDFS.

```
[cloudera@quickstart ExamplesOfAnalytics]$ hdfs dfs -ls EOADATA
Found 6 items
-rw-r--r--   1 cloudera cloudera        101 2016-06-21 07:42 EOADATA/continent.csv
-rw-r--r--   1 cloudera cloudera        354 2016-06-21 07:42 EOADATA/country.csv
-rw-r--r--   1 cloudera cloudera        156 2016-06-21 07:42 EOADATA/country_continent.csv
-rw-r--r--   1 cloudera cloudera     854362 2016-06-21 07:42 EOADATA/facebook_combined.txt
-rw-r--r--   1 cloudera cloudera       1402 2016-06-21 07:42 EOADATA/metro.csv
-rw-r--r--   1 cloudera cloudera        430 2016-06-21 07:42 EOADATA/metro_country.csv
```

Figura 4.11: Visualizar datos en HDFS.

4.5. Pruebas

Ejecutamos la consola de Spark junto a las distintas bibliotecas y el script Facebook.scala.

```
[cloudera@quickstart ExamplesOfAnalytics]$ spark-shell --jars lib/gs-core-1.2.jar,lib/gs-ui-1.2.j
ar,lib/jcommon-.0.16.jar,lib/jfreechart-1.0.13.jar,lib/breeze_2.10-0.12.jar,lib/breeze-viz_2.10-0
.12.jar,lib/pherd-1.0.jar -i Facebook.scala
```

Figura 4.12: Ejecución de Apache Spark y otros.

```
SLF4J: Class path contains multiple SLF4J bindings.
SLF4J: Found binding in [jar:file:/usr/lib/zookeeper/lib/slf4j-log4j12-1.7.5.jar!/org/slf4j/impl/
StaticLoggerBinder.class]
SLF4J: Found binding in [jar:file:/usr/jars/slf4j-log4j12-1.7.5.jar!/org/slf4j/impl/StaticLoggerB
inder.class]
SLF4J: See http://www.slf4j.org/codes.html#multiple_bindings for an explanation.
SLF4J: Actual binding is of type [org.slf4j.impl.Log4jLoggerFactory]
Setting default log level to "WARN".
To adjust logging level use sc.setLogLevel(newLevel).
Welcome to
      ____              __
     / __/__  ___ _____/ /__
    _\ \/ _ \/ _ `/ __/  '_/
   /___/ .__/\_,_/_/ /_/\_\   version 1.6.0
      /_/

Using Scala version 2.10.5 (Java HotSpot(TM) 64-Bit Server VM, Java 1.7.0_67)
Type in expressions to have them evaluated.
Type :help for more information.
```

Figura 4.13: Ejecutando Spark.

El script Facebook.scala contiene lo siguiente:

```
//
// Set log level to error, suppress info and warn messages
//
import org.apache.log4j.Logger
import org.apache.log4j.Level

Logger.getLogger("org").setLevel(Level.ERROR)
Logger.getLogger("akka").setLevel(Level.ERROR)


//
// Hands On: Building A Graph
//
import org.apache.spark.graphx._


val facebookGraph = GraphLoader.edgeListFile(sc, "./EOADATA/facebook_combined.txt")


//
// Hands On: Network Connectedness and Clustering Components
//

import org.graphstream.graph.implementations._
```

Figura 4.14: Importando componentes.

Se importa los componentes ha utilizar.

```
val graph: SingleGraph = new SingleGraph("facebookGraph")

// Set up the visual attributes for graph visualization.

graph.addAttribute("ui.stylesheet", styleSheet);


graph.addAttribute("ui.quality")
graph.addAttribute("ui.antialias")

// Given the facebookGraph, load the graphX vertices into GraphStream
for ((id, _) <- facebookGraph.vertices.collect()) {
  graph.addNode(id.toString).asInstanceOf[SingleNode]
}

// Load the graphX edges into GraphStream edges
for ((Edge(x, y, _), count) <- facebookGraph.edges.collect().zipWithIndex) {
  graph.addEdge(count.toString, x.toString, y.toString).asInstanceOf[AbstractEdge]
}

// Display the graph.
graph.display()
```

Figura 4.15: Código implementado.

Se realizan diversos cálculos y algoritmos.

4.6. Resultados

Finalmente se muestran los resultados de los cálculos y algoritmos realizados.

```
facebookGraph.numEdges
facebookGraph.numVertices
```

Figura 4.16: Código para calcular el número de vértices y aristas.

```
Loading Facebook.scala...
import org.apache.log4j.Logger
import org.apache.log4j.Level
import org.apache.spark.graphx._
facebookGraph: org.apache.spark.graphx.Graph[Int,Int] = org.apache.spark.graphx.
impl.GraphImpl@941c64
import org.graphstream.graph.implementations._
graph: org.graphstream.graph.implementations.SingleGraph = facebookGraph
res7: Long = 88234
res8: Long = 4039
```

Figura 4.17: Resultado número de vértices y aristas del dataset de Facebook.

```
def max(a: (VertexId, Int), b: (VertexId, Int)): (VertexId, Int) = {
if (a._2 > b._2) a else b
}

def min(a: (VertexId, Int), b: (VertexId, Int)): (VertexId, Int) = {
if (a._2 <= b._2) a else b
}

facebookGraph.outDegrees.reduce(max)
facebookGraph.inDegrees.reduce(max)
```

Figura 4.18: Algoritmos para calcular el vértice y su total de aristas que posea mayor InDegree y OutDegree.

```
res9: (org.apache.spark.graphx.VertexId, Int) = (107,1043)
res10: (org.apache.spark.graphx.VertexId, Int) = (1888,251)


scala>
```

Figura 4.19: Resultados InDegree y OutDegree del dataset de Facebook.

Se muestra el grafo de forma visual.

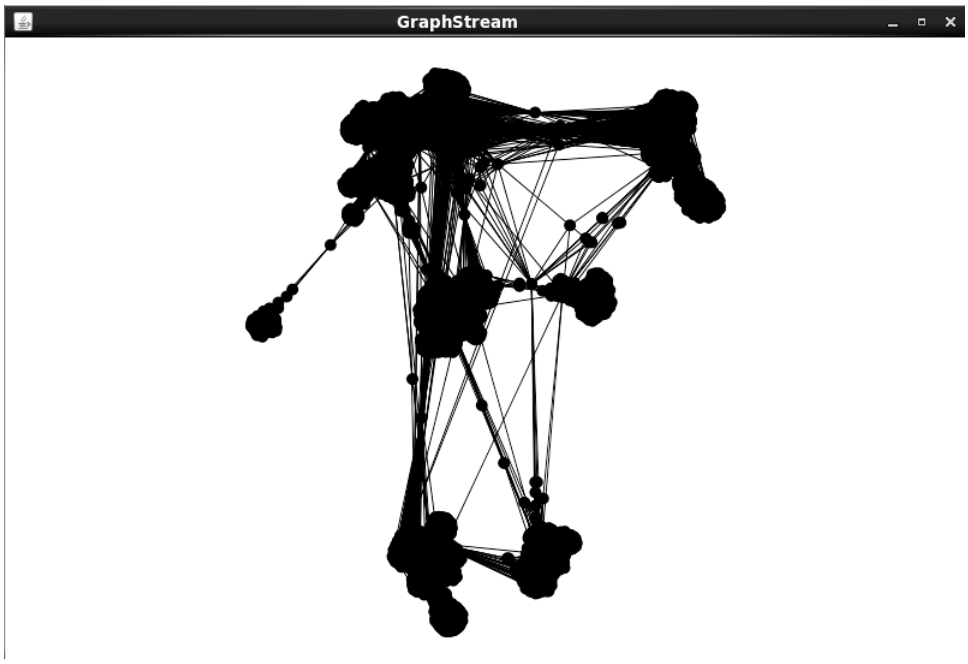

Figura 4.20: Visualización del dataset de Facebook.

CAPÍTULO 5

CONCLUSIONES

Bibliografía

[1] Dato. http://definicion.de/datos/.

[2] Información. http://www.definicionabc.com/tecnologia/informacion.php.

[3] Conocimiento. http://sobreconceptos.com/conocimiento.

[4] Minería de datos. http://infolab.stanford.edu/~ullman/mmds/book.pdf.

[5] Aprendizaje automático. http://online.stanford.edu/course/machine-learning.

[6] Inteligencia artificial. https://www.udacity.com/course/intro-to-artificial-intelligence--cs271.

[7] Inteligencia de Negocios. http://searchdatamanagement.techtarget.com/definition/business-intelligence.

[8] Data Science. https://datascience.berkeley.edu/about/what-is-data-science/.

[9] Big data. http://searchcloudcomputing.techtarget.com/definition/big-data-Big-Data.

[10] 8v. http://www.obs-edu.com/noticias/estudio-obs/en-2020-mas-de-30-mil-millones-de-dispositivos-estaran-conectados-internet/.

[11] Base de Datos. http://searchsqlserver.techtarget.com/definition/database.

[12] Base de datos relacionales. http://searchdatacenter.techtarget.com/es/definicion/Base-de-datos-relacional.

[13] Base de datos no relacionales. http://nosql-database.org/.

[14] Almacén de Datos. http://www.sinnexus.com/business_intelligence/datawarehouse.aspx.

[15] Lenguajes de programación. http://www.lenguajes-de-programacion.com/lenguajes-de-programacion.shtml.

[16] R. https://www.r-project.org/.

[17] RStudio. https://www.rstudio.com/.

[18] Igraph. https://cran.r-project.org/web/packages/igraph/igraph.pdf.

[19] Java. http://searchsoa.techtarget.com/definition/Java.

[20] Scala. http://www.scala-lang.org/.

[21] Python. http://searchenterpriselinux.techtarget.com/definition/Python.

[22] Apache Hadoop. http://hadoop.apache.org/.

[23] Common. https://commons.apache.org/.

[24] HDFS. https://hadoop.apache.org/docs/r1.2.1/hdfs_design.html.

[25] YARN. http://hadoop.apache.org/docs/current/hadoop-yarn/hadoop-yarn-site/YARN.html.

[26] MapReduce. https://hadoop.apache.org/docs/r1.2.1/mapred_tutorial.html.

[27] ACCUMULO. https://accumulo.apache.org/.

[28] HBASE. https://hbase.apache.org/.

[29] HIVE. https://hive.apache.org/.

[30] STORM. http://storm.apache.org/.

[31] MAHOUT. http://mahout.apache.org/.

[32] Grafo. http://gaussianos.com/los-puentes-de-konigsberg-el-comienzo-de-la-teoria-de-grafo