Escuela Técnica Superior de
Ingeniería Informática

TRABAJO FIN DE GRADO

Análisis y estudio del grafo de transacciones de Bitcoin

Realizado por
Fernando Claros Barrero

Para la obtención del título de
Grado en Ingeniería Informática - Ingeniería del Software

Dirigido por
Rocío González Díaz
Nieves Atienza Martínez

Realizado en el departamento de
Matemática Aplicada I

Convocatoria de Junio, curso 2021/22

Agradecimientos

Quiero agradecer a George Laurentiu Bogdan por su ayuda de cara a afrontar el desarrollo de este proyecto y ayudarme a entender sus tecnologías

También quiero agradecer a mis tutoras, Rocío y Nieves por la paciencia que han tenido con mi desatención en ciertos momentos del proyecto y no dejar de confiar en mí en ningún momento.

Resumen

Durante los últimos años, el mundo del Bitcoin y las criptomonedas ha revolucionado por completo el concepto de pago y mercado digital. Este área tan innovadora ha despertado mucho interés entre las grandes empresas y las comunidades de investigadores, que se han aventurado en la búsqueda de herramientas que nos brinden la capacidad de entender e interpretar los fenómenos que suceden dentro de este área.

Es por este motivo, que he decido implementar un sistema de procesamiento y análisis de transacciones Bitcoin. El objetivo del proyecto es la obtención de los datos de la cadena de bloques de Bitcoin, para su posterior conversión en un grafo y el estudio de las propiedades matemáticas de éste.

Para llevar acabo este proyecto, se han aplicado metodologías ágiles y se ha implantado un sistema de planificación de proyectos basadas en las pautas del PMBOK (Project Management Body Of Knowledge [1]), garantizando en todo momento un desarrollo eficiente y una planificación óptima.

Link to the repository: https://github.com/ferclabar/trabajo-fin-grado

Palabras clave: Bitcoin, Blockchain, transacción, criptodivisa, herramientas fintech, acaparamiento de capital, grafo

Abstract

During the last few years, the Bitcoin and crypto world has revolutionized the paying concept and the digital market. This new knowledge area has woken up a lot of interest between big enterprises and investigation comunities, which have started designing and looking for tools that bring us the capacity to understand and analyze the phenomenons that happen in this area.

This was the reason beacause I have decided to design and implement a system that process and analyze de Bitcoin transaction data, and after that converts the transactions map into a graph in order to study the mathematic properties of it.

To succed on this proyect, I have worked with agile methodologies and also have followed a project mangement system based on the rules and sugestions of the PMBOK (Project Management Body Of Knowledge [1]), guarateeing an efficient developement and an optimal planification on every single step of the project.

Link to the repository: https://github.com/ferclabar/trabajo-fin-grado

Keywords: Bitcoin, Blockchain, transaction, cryptocurrency, fintech tools, capital hoarding, graph

Índice general

Índice de figuras

Índice de extractos de código

1. Introducción

1.1. Bitcoin

1.1.1. ¿Qué es Bitcoin?

Bitcoin es una moneda digital y un sistema de pago sin banco central o administrador único. Este sistema fue creado por un grupo de matemáticos anónimos que trabajaron bajo el pseudónimo de Satoshi Nakamoto, a mediados de 2009. En principio, los usuarios de Bitcoin pueden transferir dinero entre sí a través de una red entre iguales usando software libre y de código abierto [20].

Figura 1.1: Busto de Satoshi Nakamoto en Budapest

Este sistema esta teniendo una gran popularidad en la comunidad debido a que implementa un sistema de pago público y a su vez anónimo donde todo el historial de transacciones está disponible para consultar en la red pero, a la misma vez, todos los monederos son protegidos con anonimato. Además, está construido sobre un sistema de datos distribuido, el sistema Blockchain (explicado en la sección 1.1.2), lo que hace que el sistema no dependa de ningún organismo central y sea la propia comunidad de usuarios la que rige la veracidad y oficialidad de cada movimiento monetario [20]. Se caracteriza por estar construido con base en un protocolo criptográfico. Esto significa que cada transacción que quiere realizarse deberá ser validada por la comunidad por medio de la resolución de un problema de criptografía de complejidad muy alta. La resolución de este problema para la confirmación de la transacción la denominamos minado. Con ello, se ofrece un alto nivel de seguridad pues se ha observado resistente a fraude, falsificación, devoluciones y otros ataques.

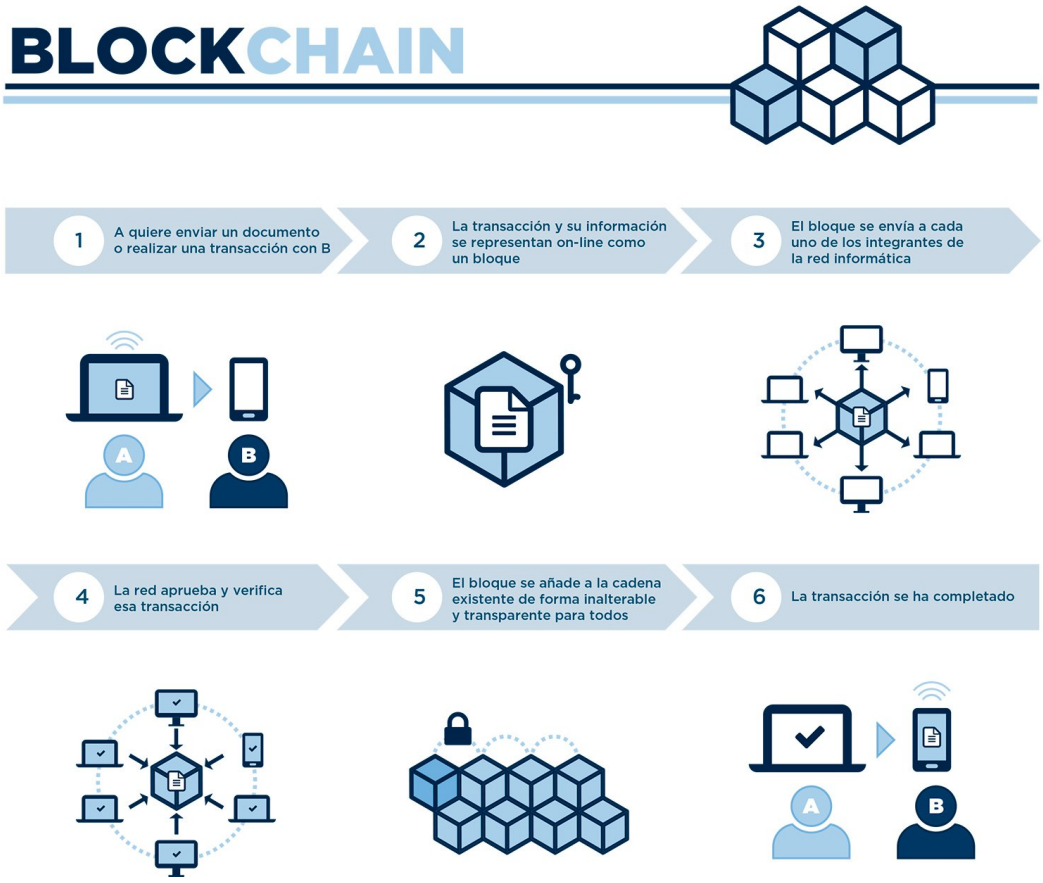


Figura 1.2: Estructura de funcionamiento de bitcoin

1.1.2. Apoyo a Bitcoin: El sistema Blockchain

Gran parte de la popularidad que ha cobrado el sistema Bitcoin se debe al sistema de datos que sostiene esta herramienta, una red de datos distribuida.

Un sistema de datos distribuido se basa en un conjunto de computadoras independientes conectadas por red y con soporte de software distribuido. Este sistema permite que las computadoras realicen de manera coordinada sus tareas y compartan unas con otras los recursos disponibles a nivel hardware, software y de datos, para garantizar el constante funcionamiento de la red. De esta manera, el usuario percibe un único núcleo de cómputo integrado aunque este pueda estar siendo ofrecido por varias máquinas en distintas ubicaciones [16].

En esencia, una red Blockchain es solo una sistema de almacenamiento de datos que permite realizar consultas de lectura y escritura en registros en una red de alamcenamiento de datos totalmente distribuida sin poder realizar modificaciones en ningún dato que se haya registrado anteriormente. Todos los registros que se guardan en ella están interrelacionados con una matemática muy avanzada haciendo imposible añadir o modificar algún registro cuyos datos no sean coherentes con el resto de datos registrados [4].

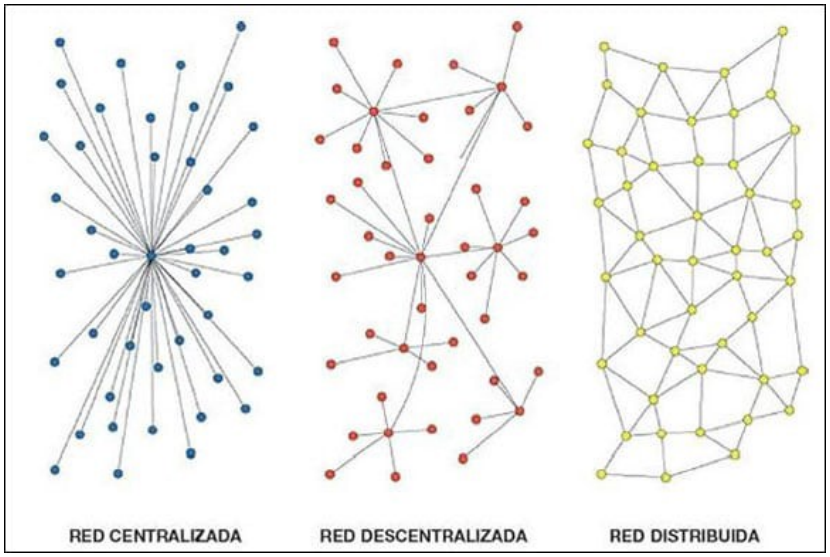


Figura 1.3: Comparación de bases de datos según distribución

1.2. Motivaciones para llevar acabo el proyecto

De cara a realizar el proyecto de fin de grado, tenía una gran cantidad de motivaciones y argumentos a favor de realizar este desarrollo. En primer lugar, el análisis y estudio del sistema Blockchain es una rama de investigación totalmente innovadora. Día a día, las empresas invierten grandes cantidades de capital en llevar acabo proyectos innovadores que hacen uso de esta tecnología con el fin de obtener resultados provechosos. Por tanto creo que la motivación principal de este proyecto ha sido el gran

valor real que puede llegar a tener el entender y analizar a fondo como funciona este sistema.

Por otro lado, encontré muy interesante este proyecto debido a que puede llegar a ser una herramienta muy potente para **dar a conocer el mundo de las criptomonedas a personas que no tienen conocimientos sobre tecnología y sobre fenómenos financieros**, por lo que considero que este proyecto cuenta también con un gran **valor social y divulgativo**. En este caso, se hace uso de las matemáticas y la teoría de grafos para conceptualizar el mapa de transacciones de Bitcoin, y entender de una manera más visual la estructura interna de esta red.

Por último creo que es importante añadir que este proyecto también tiene un importante **valor a nivel personal**. Desde hace meses, me he sentido tentado por sumergirme en la investigación de este campo, aunque por falta de recursos y tiempos no me ha sido posible. Es por este motivo que pensé que este proyecto sería la manera idónea de adquirir conocimientos sobre este área y comprender a fondo cómo funciona el sistema Bitcoin.

2. Estudio Previo

2.1. Introducción

La planificación del proyecto es la ordenación sistemática de las tareas para lograr un objetivo. En ella se expone lo que se necesita hacer y cómo debe llevarse a cabo [18].

La importancia de este análisis previo y planificación reside en que, gracias al desglose en horas y el establecimiento de un orden de procedimiento, es mucho más sencillo estudiar cuáles serán las desviaciones en la implementación y, posteriormente, realizar un análisis de rendimiento en cada tarea por medio del cálculo entre horas estimadas y el coste real de la tarea.

En este capitulo detallaremos el proceso de análisis y acciones tomadas de cara a afrontar el desarrollo de este proyecto. Al ser un proyecto de carácter muy abstracto en un principio, tanto la planificación a seguir como los objetivos del desarrollo **resultaron ser muy ambiguos y quedaban muy en el aire**, pero, tras un proceso de profundo análisis y estudio, los objetivos y las herramientas para elaborar el desarrollo quedaron claras.

2.2. Objetivos

El objetivo principal del proyecto es el procesamiento de archivos DAT, para su posterior conversión en archivos columnares (estructurados en columnas y filas) procesables por las librerías que nos ofrece Apache Spark [8], librería diseñada para el procesamiento de grandes volúmenes de datos y conversión en grafo por medio de GraphX [2]. Por medio de este motor de procesamiento, será posible obtener los conjuntos de datos de las carteras y las transacciones, y a partir de éstos, generar el grafo asociado a las transacciones.

Los archivos en su formato de entrada no son procesables por estas librerías, por lo que será necesaria la conversión de estos ficheros en archivos columnares por medio de un conversor programado en Scala (lenguaje de programación explicado en el punto 5.2.1). Una vez se tienen los archivos en el formato correcto se procesan mediante Spark y se construye un grafo con las transacciones asociadas al bloque analizado.

2.3. Metodología

Al ser éste un proyecto técnico de fin de grado, la metodología a seguir de cara a la realización del proyecto será de carácter analítico y de medición. Esta metodología se

basa en realizar un desarrollo para posteriormente analizar si cumple con los objetivos planteados y medir de manera cuantitativa el rendimiento del código.

Debido a que en el desarrollo se utilizan partes de código ya realizadas con anterioridad por otros desarrolladores, la parte de análisis del proyecto cobrará una mayor importancia que la parte del estudio de rendimiento.

Por tanto, la metodología a seguir se basará en el diseño de un modelo de código para su posterior análisis y estudio del rendimiento computacional de éste.

2.4. Planificación

Las actividades a realizar de cara a la elaboración de este proyecto se detallan en la siguiente lista:

1. Investigación sobre la temática del proyecto.
2. Investigación sobre el funcionamiento de las criptomonedas.
3. Investigación de las librerías de Apache Spark y vías disponibles para su implementación en Python.
4. Estudio del formato de entrada de los archivos que contienen la información sobre las transacciones.
5. Elaboración de un modelo en pseudocódigo sobre el desarrollo del proyecto.
6. Integración de un conversor de archivos para depurar los datos y poder trabajar con ellos en un formato procesable.
7. Integración y desarrollo en Python de un procesador de datos por medio de las librerías de Spark.
8. Elaboración del grafo asociado a las transacciones recogidas en los datasets devueltos por los métodos de Spark.
9. Desarrollo de ciertas funciones de interés que nos permitan estudiar el grafo por medio de sus propiedades matemáticas.
10. Elaboración de un análisis de los resultados obtenidos.
11. Redacción de memoria de proyecto.
12. Elaboración de la presentación del proyecto.
13. Reuniones con las tutoras y comunicación con las mismas para el seguimiento del proyecto.

Actividad	Actividades predecesoras	Estimación en horas
1	-	30
2	-	25
3	-	25
4	2,3	30
5	3,4	30
6	5,4	35
7	3,5,6	30
8	7	20
9	8	10
10	9,8	35
11	9,8,7,6	10
12	11	10
13	-	10

Cuadro 2.1: Tabla de planificación. Total = 300 horas aproximadamente

2.5. Presupuesto

De cara a afrontar el desarrollo de este proyecto, los recursos disponibles eran bastante limitados aunque suficientes para realizar la implementación del sistema. Desde el departamento de Matemática Aplicada I se me puso a disposición un servidor con gran volumen de almacenamiento y procesamiento para realizar pruebas sobre el modelo con un conjunto de datos de gran tamaño. Pero tras realizar un análisis previo de los datos llegué a la conclusión de que era más eficiente realizar el proyecto sobre un bloque de datos de pequeño tamaño procesable por mi computadora personal que cuenta con un buen procesador.

Por tanto, consideraré el único coste de este proyecto el sueldo en euros por hora que cobraría un programador o Data Analyst de cara a realizar el desarrollo del mismo. El sueldo lo he estipulado en base al sueldo reflejado en el intervalo de mercado de LinkedIn [14], que ronda entre los 27000 € - 30000 €. Lo que nos dejará un sueldo estimado de 11,37 € por hora de trabajo. Por tanto:

$$300\,\text{horas} \times 11,37\,€ = 4011\,€ \tag{2.1}$$

Esto nos dejará un coste total de 4011 € de presupuesto para el proyecto.

3. Análisis del problema

3.1. Introducción

La clave de realizar un desarrollo exitoso reside en saber todas las necesidades y establecer objetivos claros y medibles, pero también que sean alcanzables es imprescindible. Hay que huir de la ausencia de realismo en los objetivos propuestos. También es necesario hacer seguimiento periódico para solventar las dificultades en la gestión y la mejora del avance del proyecto[12]

En este capítulo haremos recopilación de requisitos técnicos necesarios para la realización del proyecto. Repasaremos desde los datos que necesitamos para su posterior análisis, hasta los requisitos de calidad que he decidido considerar para que el desarrollo sea limpio, visual, intuitivo y eficiente.

3.2. Requisitos de información

En este apartado se detallará la lista de requisitos de información necesarios para la realización del desarrollo:

Cuadro 3.1: RI-01. Información sobre las transacciones.

ID	Descripción del requisito
RI-01	El sistema deberá tener la información de las transacciones Bitcoin en un conjunto de datos de las carteras de los usuarios implicados en las transacciones, y otro conjunto de datos sobre las transacciones realizadas entre carteras.

Cuadro 3.2: RI-02. Grafo de las transacciones.

ID	Descripción del requisito
RI-02	El sistema deberá contener el grafo de transacciones, siendo los nodos las carteras de usuario, y las aristas las transacciones entre estas.

3.3. Requisitos funcionales

En este apartado se detallará la lista de requisitos funcionales necesarios para la realización del desarrollo, que nos facilitará las herramientas necesarias para efectuar un análisis sobre el grafo de transacciones:

Cuadro 3.3: RF-01. Representación parcial y total del grafo

ID	Descripción del requisito
RF-01	El sistema deberá poder representar gráficamente tanto una parte del grafo como el conjunto total de nodos y aristas del grafo.

Cuadro 3.4: RF-02. Rastreo de patrones en las transacciones.

ID	Descripción del requisito
RF-02	El sistema deberá poder filtrar las transacciones con parámetros escogidos por el usuario.

Cuadro 3.5: RF-03. Ranking de paginación.

ID	Descripción del requisito
RF-03	El sistema deberá poder realizar un ranking de la relevancia de los nodos en base al algoritmo del ranking de paginación.

Cuadro 3.6: RF-04. Grado de los nodos del grafo

ID	Descripción del requisito
RF-04	El sistema deberá poder realizar un calculo del grado de cada nodo.

Cuadro 3.7: RF-05. Estadísticas del grafo a partir del grado de sus nodos

ID	Descripción del requisito
RF-05	El sistema deberá poder realizar un calculo de estadísticas del grafo a partir del grado de los nodos.

Cuadro 3.8: RF-06. Correlación entre transacciones entrantes y salientes.

ID	Descripción del requisito
RF-06	El sistema deberá poder realizar un calculo de la correlación entre transacciones entrantes y salientes de cada nodo.

Cuadro 3.9: RF-07. Conteo de triángulos del grafo

ID	Descripción del requisito
RF-07	El sistema deberá poder obtener un conjunto de datos con los triángulos entre nodos que conforman el grafo.

3.4. Requisitos no funcionales

En este apartado se detallarán las funcionalidades que se van a implementar para el procesamiento y representación gráfica de la red de transacciones.

Cuadro 3.10: RNF-01. Calidad del código.

ID	Descripción del requisito
RNF-01	El desarrollo de código debe ser eficiente y tener buen rendimiento en coste computacional.

Cuadro 3.11: RNF-02. Legibilidad del código.

ID	Descripción del requisito
RNF-02	El desarrollo de código debe ser legible, visual e intuitivo.

4. Modelo del sistema

4.1. Introducción

El modelado de sistemas software es una fase de los proyectos software que se realiza para analizar y tratar con la complejidad inherente a estos sistemas. El diseño de un modelo ayuda al ingeniero de software y a los desarrolladores a tener una concepción visual del sistema a implementar. Además, los modelos de un nivel de abstracción mayor pueden utilizarse para la comunicación con el cliente o persona que hará uso de este proyecto (en mi caso, la comunidad de investigación) [3].

En este capítulo, hablaremos de como se ha realizado esta fase de modelado del sistema. Comentaremos las decisiones arquitectónicas que se han llevado a cabo con el fin de garantizar una solución óptima, así como el modelo del flujo de los datos a través del sistema para llegar desde un archivo Blockchain hasta su conversión en un grafo de transacciones.

4.2. Diseño

4.2.1. Arquitectura del modelo

De cara a realizar un modelo arquitectónico del sistema, se llegó a la conclusión de que la solución más óptima era separar el sistema en dos grandes bloques.

En primer lugar tendremos el módulo donde se encontrará el conversor de archivos .DAT a ficheros .parquet procesables por el motor de procesamiento del bloque de análisis. Este módulo está escrito en Scala, y lo analizaremos en profundidad en el capítulo 5, en el que se detallará la implementación del sistema y las herramientas utilizadas.

Por otro lado, se implementará un módulo de análisis escrito en Python. En este módulo se trabajará con los archivos procesados por el modulo de conversión anteriormente mencionado. En primer lugar se utilizará Spark para el procesamiento de los ficheros y la creación de conjuntos de datos, y luego se desarrollarán todas las funcionalidades de análisis en bloques de instrucciones aislados de manera que se distingan todas las funcionalidades a implementar.

De cara a no sobrecargar el disco del equipo donde se lanzará el proyecto, no se trabajará en un sistema de persistencia de los datos. Es decir, los datos no se almacenarán en ninguna base de datos, sino que se crearán y procesarán cada vez que lancemos el archivo del módulo de análisis del proyecto. Los únicos datos que se almacenarán serán los archivos .parquet procesados por el módulo de conversión. Se ha tomado esta decisión debido a la extrema complejidad de los versionados de Scala y con intención de facilitar la corrección de este proyecto.

4.2.2. Elaboración del modelo de flujo de datos

De cara a hablar del flujo de datos dentro del sistema, es importante diferenciar los dos estados en los que se encontraran los datos a lo largo del flujo. Por un lado tendremos los datos almacenados en archivos, este será el formato en el que entrarán en el sistema y sufrirán la primera transformación en un archivo en formato parquet.

Una vez tengamos los datos en ese estado, podremos proceder a su procesamiento, convirtiéndolos así en un conjunto de datos que se crea y almacena en memoria cada vez que se ejecuta el notebook con los bloques de ejecución escritos en Python. Una vez que dispongamos de estos datos, éstos dejaran de estar almacenados en archivos, y formaran parte de la ejecución de las instrucciones.

Por último, una vez obtenidos estos datos y almacenarlos en conjuntos de datos separados en carteras y transacciones, gracias a las librerías escogidas, podremos convertirlos en un sistema de datos orientado a un grafo, al cual, se le realizarán múltiples cálculos matemáticos para estudiar propiedades de interés de este sistema de transacciones Bitcoin.

En este gráfico, se muestra el flujo y las transformaciones que sufren los datos a lo largo del proceso con la intención final de crear y analizar el grafo:

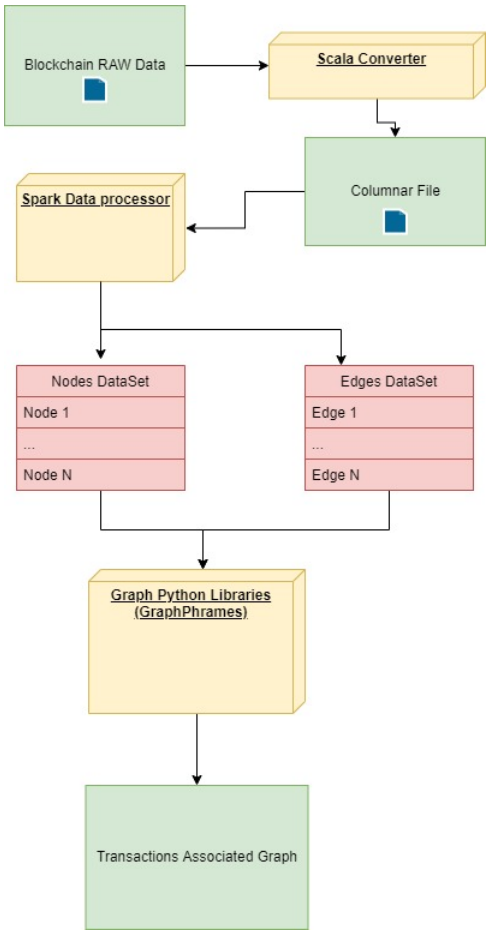


Figura 4.1: Representación gráfica del flujo de los datos a través del programa empleado

5. Implementación

5.1. Introducción

La implementación de un sistema informático es el proceso principal dentro de un proyecto de desarrollo software. Consiste en la producción del código necesario para cumplimentar las necesidades extraídas del análisis del problema.

En este capítulo explicaremos cómo, a partir de un modelo y un diseño previo, se realiza una implementación de código para realizar las funcionalidades detalladas en el apartado de análisis del proyecto. así como las herramientas utilizadas y las ventajas que éstas nos ofrecen.

5.2. Herramientas

5.2.1. Scala

Scala es un lenguaje de programación multi-paradigma implementado para expresar patrones comunes de programación de forma clara, sintáctica, visual y concisa. Este lenguaje de programación es ejecutado en la maquina virtual de Java por lo que se producirá una unión muy beneficiosa para los lenguajes de programación Java y Scala. Esta relación simbiótica entre lenguajes permitirá la unión de ambos en el desarrollo de una aplicación, pudiendo, entre otras muchas cosas, heredar clases e implementar interfaces [6, 11].

5.2.2. Librerías de Python de Apache Spark: Pyspark

Apache Spark es un framework de programación utilizado para el procesamiento de datos de manera distribuida. Está diseñado para trabajar de manera rápida y eficiente y ser de propósito general. Además este framework esta diseñado dentro del marco de proyecto Apache, lo que garantiza licencia OpenSource en todas sus funcionalidades [5].

Spark cuenta con librerías en una gran variedad de idiomas. Lo que hace que podamos disfrutar de sus funcionalidades en amplio espectro de proyectos de distinto ámbito y calibre.

En este apartado hablaremos de las ventajas que tiene utilizar este framework y qué papel ha cobrado a lo largo del desarrollo de este proyecto.

Popularidad en la comunidad

Una amplia variedad de proveedores y multinacionales tecnológicas se sumaron rápidamente a respaldar el proyecto Spark, reconociendo la oportunidad de ampliar el mercado de sus productos de Big Data existentes a áreas con un gran potencial como el aprendizaje automático, donde Spark ofrece un valor real [19].

Además de esto, Spark se considera, a día de hoy, uno de los proyectos más activos gestionados por la Fundación Apache Software, y la comunidad que ha crecido en torno al proyecto incluye tanto contribuidores particulares como patrocinadores corporativos con un gran respaldo económico como Databricks, IBM y Huawei [19].

Calidad de la documentación y compatibilidad con múltiples lenguajes

Cuando hablamos de Spark, hablamos de una de las ramas más populares y famosas del proyecto Apache. Desde el inicio del proyecto, se invirtieron grandes cantidades de capital por parte de la organización y múltiples inversores, para crear un software accesible, comprensible y mantenible. Y una de las mayores ventajas de este framework es la calidad de su documentación.

La documentación del framework cuenta con manuales a disposición de usuario durante todas las fases de uso del framework. Dispone de un manual de instalación muy detallado para todos los sistemas operativos. Así como un repositorio donde se pueden encontrar todas las versiones estables del framework para encontrar la que se adecue más a las necesidades del proyecto a realizar. En mi caso, he estado utilizando la última versión estable a disposición, la versión 3.2.1.

Además, también se encuentra a disposición del usuario una documentación para el uso de Spark por medio de distintas APIs escritas en los lenguajes en los que el framework se encuentra disponible. Actualmente Spark cuenta con APIs para Java, Scala, Python, R (para el estudio estadístico de grandes volúmenes de datos) y SQL. En mi caso he utilizado la versión adaptada a Python.

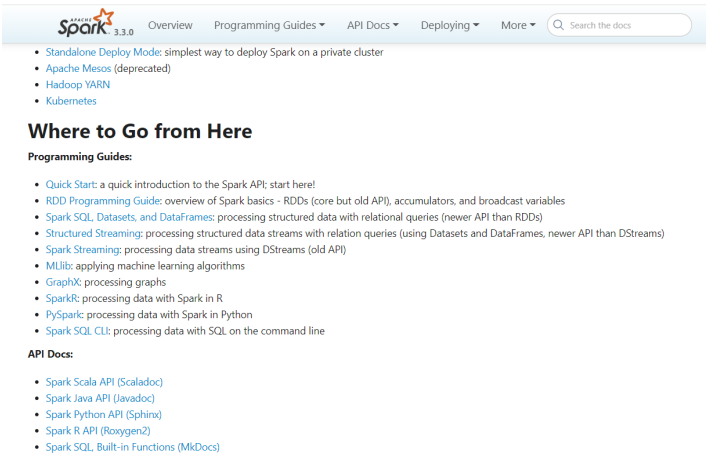


Figura 5.1: Documentación de Apache Spark

Y por último, la parte más relevante de esta documentación es que cuenta con manuales de programación de todo el espectro de funcionalidades anexas a Spark. En este proyecto en concreto, esta parte de la documentación resultó ser de vital importancia debido a que además de procesar los datos mediante Spark, el grafo de transacciones se construye mediante una rama de este framework llamada GraphX, para la cual se dedicará un apartado en este proyecto.

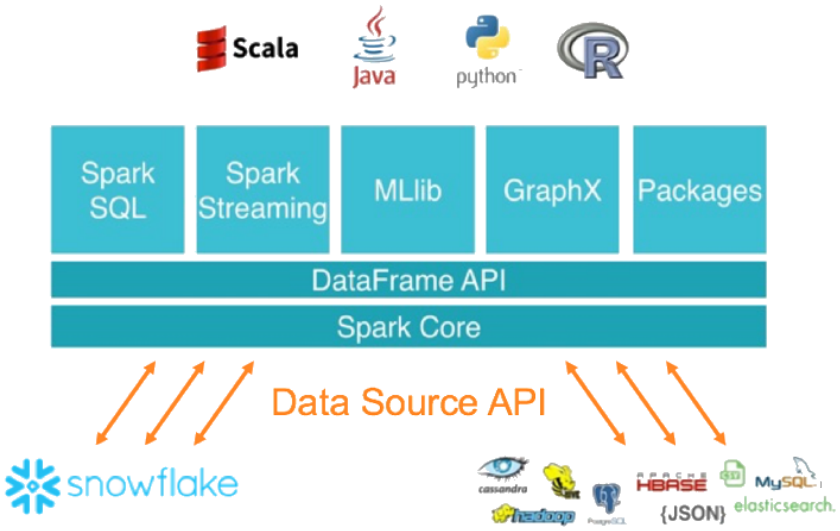


Figura 5.2: Esquema de módulos y compatibilidades de Spark

Rendimiento: simplicidad, velocidad y soporte

Hay muchas razones para elegir Spark, pero tres son clave: **simplicidad, velocidad y soporte.**

Cuando hablamos de Spark, hablamos de un sistema montado sobre un clúster de computación que permite trabajar con Big Data mediante la paralelización de procesadores en distintas máquinas. Está diseñado para ser rápido, fácilmente accesible y cuenta con una amplia gama APIs nativas de Java, Scala, R y Python y bibliotecas de desarrolladores Spark[17].

Apache Spark puede funcionar junto a Hadoop, con Apache Mesos o por sí mismo sobre los sistemas operativos Linux, Windows y OS X.

Desde comienzos del desarrollo del framework, Spark fue diseñado para ejecutarse de manera optimizada en memoria. Esta optimización, nos hace más fácil la tarea de procesar datos rápidamente, ganando en eficiencia a su alternativa al esquema de procesamiento de datos mediante MapReduce de Hadoop. Esta herramienta nos ofrece la capacidad de procesar datos hasta 100 veces más rápido cuando se utiliza in-memory (corriendo en la memoria), y 10 veces más rápido cuando procesan datos sobre disco [19].

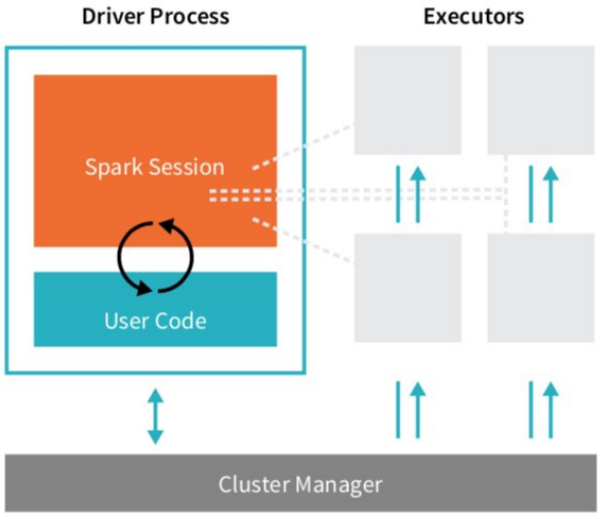


Figura 5.3: Esquema de computación del sistema Spark

5.2.3. Librerías de Python de GraphX: GraphFrames y NetworkX

GraphFrames es un paquete de métodos escrito en Scala programado sobre las librerías de Apache Spark que ofrece herramientas de creación de grafos basados en conjuntos de datos. He considerado esta herramienta como idónea para la realización del proyecto debido a que es compatible con Spark y nos brinda prácticamente todas las funcionalidades de GraphX.

Además, GraphFrames cuenta con APIs de alto nivel desarrolladas para los principales lenguajes de programación (Python y Scala), lo que ha resultado de utilidad debido a que ha sido posible integrar el módulo y hacer uso de sus funcionalidades en este proyecto escrito en Python. La calidad de la documentación de las APIs es muy buena y contiene una gran cantidad de referencias y ejemplos que han sido de gran ayuda para comprender las funcionalidades que nos ofrece el módulo. De cara a desarrollar este proyecto, la versión escogida de GraphFrames será la versión 0.6.0.

Por otro lado, para el dibujo de los grafos se ha hecho uso de la librería de Python NetworkX. Esta librería ofrece una amplia gama de métodos de análisis y dibujo de redes y grafos. Además, se complementa a la perfección con GraphFrames, lo que ha hecho que la combinación de ambas librerías haya sido un éxito y se hayan podido procesar y dibujar los grafos. La versión utilizada de esta librería será la última versión estable, la 2.8.4.

5.3. Desarrollo

5.3.1. Conversor de archivos .DAT: Scala Converter

De cara a poder procesar mediante Spark el bloque de datos ofrecido por la herramienta Bitcoin Core, era necesario realizar una transformación del formato del archivo, debido a que Spark trabaja de manera más eficiente con archivos columnares. Por tanto, se integró un módulo de conversión de archivos .DAT en el proyecto, programado en Scala, que garantizaba que el formato de entrada que tuviera Spark fuera analizable.

Este módulo fue desarrollado por el Analista de Datos **Jörn Franke**, y el módulo se encuentra en el repositorio "Using Apache Hive to analyze Bitcoin Blockchain data". También, se ha aprovechado la documentación y ciertos módulos del repositorio "Bitcoin Insights" elaborado por el analista de datos **Jirka Kremser**, que integraba este conversor en un proyecto escrito sobre lenguaje Python [13, 10].

El módulo esta estructurado en distintos paquetes que nos ofrecen las funcionalidades que necesitamos para realizar la conversión[9].

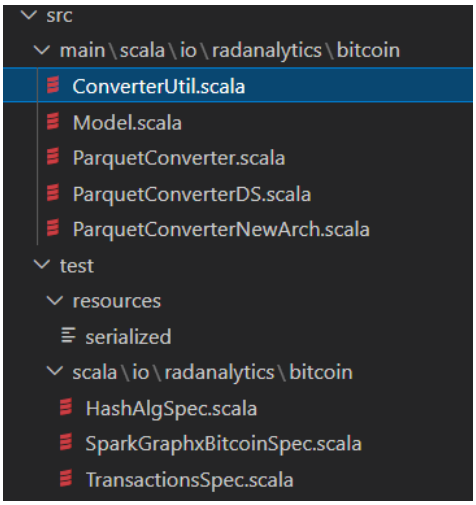


Figura 5.4: Distribución de paquetes del conversor Scala

Después de ejecutar el comando de construcción del proyecto **sbt clean assembly** se creará un archivo .jar con el ejecutable necesario para realizar las conversiones. Para convertir será necesario ejecutar el siguiente comando en la consola de administración del sistema.

De cara a simplificar el proyecto, se han añadido directamente los archivos .parquet convertidos gracias al conversor. La compatibilidad de los versionados es tremendamente compleja y el tamaño de los archivos .DAT es demasiado grande (unos 200 MB). Por eso, he decido que lo más óptimo sería realizar la conversión aparte y trabajar directamente con los archivos procesables por Spark. Aunque, si quisiéramos obtener directamente los datos del blockchain, podríamos hacerlo a través de la herramienta para desarrolladores de Bitcoin: **Bitcoin Core**. En este programa se podrán consultar todos los registros Blockchain completamente actualizados. En el caso de este proyecto

se utilizarán los bloques de ejemplo del repositorio "Bitcoin Insights"que contienen alrededor de 2 millones de transacciones, que he considerado una cifra bastante correcta para generar un grafo de gran tamaño para realizar un estudio preciso.

blk02058	05/04/2022 18:38	Archivo DAT	129.937 KB
blk02059	05/04/2022 18:56	Archivo DAT	130.118 KB
blk02060	05/04/2022 19:07	Archivo DAT	130.623 KB
blk02061	05/04/2022 19:34	Archivo DAT	129.899 KB
blk02062	05/04/2022 19:42	Archivo DAT	129.983 KB
blk02063	05/04/2022 19:50	Archivo DAT	130.071 KB
blk02064	06/04/2022 14:34	Archivo DAT	130.022 KB
blk02065	06/04/2022 14:52	Archivo DAT	130.657 KB
blk02066	06/04/2022 14:53	Archivo DAT	131.069 KB
blk02067	06/04/2022 14:53	Archivo DAT	130.854 KB
blk02068	06/04/2022 14:53	Archivo DAT	130.140 KB
blk02069	06/04/2022 14:54	Archivo DAT	131.061 KB
blk02070	06/04/2022 14:54	Archivo DAT	130.496 KB
blk02071	06/04/2022 14:54	Archivo DAT	130.195 KB
blk02072	06/04/2022 14:54	Archivo DAT	130.630 KB
blk02073	06/04/2022 14:54	Archivo DAT	131.055 KB
blk02074	30/04/2022 16:08	Archivo DAT	130.690 KB
blk02075	30/04/2022 16:27	Archivo DAT	130.242 KB

Figura 5.5: Directorio local de bloques tras la descarga de Bitcoin Core

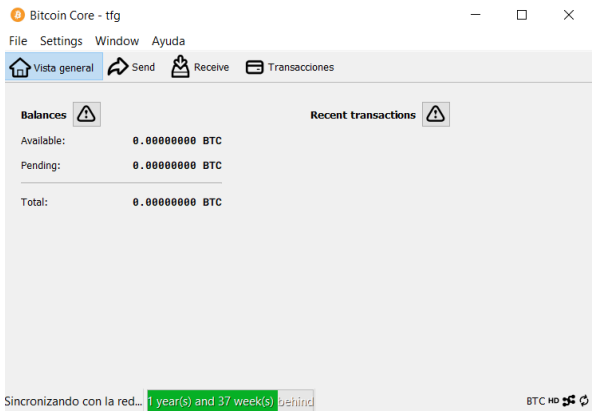


Figura 5.6: Interfaz de Bitcoin Core

```
ls ~/bitcoin/input/
# datosBlockchain.dat

spark-submit \
  --driver-memory 2G \
  --executor-memory 2G \
  --class io.radanalytics.bitcoin.ParquetConverter \
  --master local[8] \
  ./target/scala-2.11/bitcoin-insights.jar ~/bitcoin/input ~/
     bitcoin/output

# ... <Salida de la conversion> ...

ls ~/bitcoin/output
```

Extracto de código 5.1: Ejecución del conversor

5.3.2. Procesamiento de los datos mediante Spark

En primer lugar importaremos la librería de Spark para Python 'pyspark'. Por defecto‘, utilizaremos una SparkSession para ejecutar y procesar los datos de manera local en una sola máquina. Tratamos con datos "relativamente pequeños", por tanto, no tiene sentido ejecutar este sistema sobre un cluster, pero el parámetro local[1]‘' se podría modificar para insertar la IP del cluster.

```
import pyspark
from pyspark.context import SparkContext
from pyspark.sql import SparkSession, SQLContext
from pyspark.sql.functions import *

spark = SparkSession.builder \
                    .master("local[4]") \
                    .config("spark.driver.memory", "4g") \
                    .getOrCreate()

sc = spark.sparkContext
```

Extracto de código 5.2: Comando Load de Spark

Una vez ejecutada la conversión, podremos acceder a estos datos mediante el motor de procesamiento de Spark. En primer lugar, realizaremos la carga de los datos en el programa mediante el comando Load:

```
raw_nodes = spark.read.load("../parquet-converter/output/
   nodes") \
                      .withColumnRenamed("_1", "id") \
                      .withColumnRenamed("_2", "Address")
raw_nodes.show(5)
```

Extracto de código 5.3: Comando Load de Spark para los nodos

```
raw_edges = spark.read.load("../parquet-converter/output/
   edges") \
                      .withColumnRenamed("srcId", "src") \
                      .withColumnRenamed("dstId", "dst") \
                      .drop("attr") \
                      .cache()
raw_edges.show(5)
raw_edges.count()
```

Extracto de código 5.4: Comando Load de Spark para las aristas

Estos comandos recogerán los archivos .parquet generados a partir de la conversión de los archivos .DAT. Una vez cargados recibiremos dos datasets:

1. Un dataset con los nodos. Cuyos atributos serán un Id auto-generado, y la dirección pública de la cartera Bitcoin.
2. Un dataset con los aristas. Cada arista tendrá un atributo Id auto-generado, y dos atributos src y dst que representarán el origen y el destino de una transacción.

De esta manera, podremos construir el grafo a partir de los dos datasets mediante GraphX.

5.3.3. Creación del grafo y estudio de propiedades de éste

Creación del grafo

Una vez obtenidos los datos a través de Spark, tenemos a nuestra disposición la posibilidad de crear el grafo asociado a las transacciones Bitcoin de dicho bloque y realizar algunas operaciones de análisis sobre éste. En esta sección, se detallan los extractos de código necesarios para realizar las funcionalidades necesarias detalladas en el catalogo de requisitos del capítulo 3.

En primer lugar crearemos el grafo a partir de los dos datasets y el constructor que nos pone a disposición la librería:

```
from graphframes import *

g = GraphFrame(nodes, edges).cache()
```

Extracto de código 5.5: Construcción del grafo

Cálculo del grado de los nodos

A partir de este grafo podremos empezar a realizar operaciones de análisis. Estas operaciones comprenden desde cálculos básicos hasta la extracción de propiedades complejas como el ranking de paginación, el grado de triangulación del grafo o una medida de las componentes conexas del grafo para aproximar clústeres.

En primer lugar calcularemos el grado de los nodos utilizando el método **inDegrees()** y el método **outDegrees()**. Estos métodos calcularán el número de transacciones entrantes y salientes del nodo. De cara a calcular el grado, se realiza una operación **join()** con los dos cálculos. Aunque si quisiéramos, podríamos realizar estos cálculos por separado.

```
vertexDegreesAndIds = g.inDegrees.join(g.outDegrees, "id")
vertexDegrees = vertexDegreesAndIds.drop("id")
vertexDegrees.show(5, False)
```

Extracto de código 5.6: Cálculo del grado de los nodos

A partir de estos cálculos, se ha implementado una funcionalidad para disponer de estadísticas básicas de los nodos como su media, su conteo total, desviación típica, máximos y mínimos.

```
vertexDegrees.describe() \
             .show()
```

Extracto de código 5.7: Cración de DataSet con estadísticas sobre los nodos

Además, se ha implementado la funcionalidad de poder calcular la correlación entre aristas entrantes y salientes, y la creación de un diagrama de dispersión. Esto nos ayudará a estudiar si existe una relación de linealidad entre los aristas entrantes y salientes de cada nodo.

```
sns.jointplot(x="inDegree", y="outDegree", data=vertexDegrees
   .sample(False, 0.01, 42).toPandas());
```

Extracto de código 5.8: Creación de diagrama de dispersión de aristas entrantes y salientes

```
vertexDegrees.select(corr("inDegree", "outDegree")) \
             .show()
```

Extracto de código 5.9: Cálculo de la correlación entre aristas entrantes y salientes

Rastreo de patrones en los grafos mediante filtros

También se ha implementado una funcionalidad para encontrar nodos con unas determinadas condiciones por medio de los métodos **.find()** y **.filter()**.

Podremos usar esta funcionalidad para crear grafos con los nodos que cumplan las condiciones aplicadas. De esta manera, si queremos aplicar un estudio únicamente sobre ciertos nodos con unas propiedades concretas, podremos hacerlo a través de estas consultas.

El código esta diseñado para escoger una dirección cualquiera del dataSet, para luego iterar el grafo extrayendo cuales son sus transacciones salientes por medio del método select().

Por último se hará uso de la librería de tratamiento de grafos NetworkX para la representación del fragmento de grafo seleccionado.

```
graph_with_degrees = GraphFrame(g.vertices.join(
   vertexDegreesAndIds, "id"), edges)
graph_with_degrees.vertices.show()


motifs = graph_with_degrees.find("(a)-[]->(b)") \
                           .filter("a.outDegree > 1000") \
                           .filter("a.inDegree = 1") \
                           .filter("b.outDegree = 1") \
                           .filter("b.inDegree > 1000")
motifs.show()
motifs.count()
```

Extracto de código 5.10: Cálculo del grado de los nodos

Calculo del número de triángulos del grafo

La función del conteo de triángulos dentro de un grafo se utiliza para estudiar el numero de triángulos que pasan por cada nodo del grafo. Este algoritmo itera el grafo nodo por nodo, calculando el número de triángulos en los que el nodo participa siendo uno de los 3 vértices.

Este sistema es una muy buena medida de análisis para estudiar el grado de interconexión de un grafo. Cuanto mayor es el grado de triangulación de un grafo, mayor fuerza de interconexión existirá entre sus componentes y más relaciones entre nodos.

Para ejecutar este algoritmo, lo único que tendremos que hacer será llamar a la función **triangleCount()** de la librería GraphFrames con la que construimos el grafo.

```
ptriang = g.triangleCount()
ptriang.count()
```

Extracto de código 5.11: Cálculo del grado de triangulación del grafo

Esta función devolverá un conjunto de datos con los triángulos asociados a cada nodo. Por cuestiones de déficit de recursos (solo se contaba con un procesador para ejecutar estos cálculos) ha sido imposible estudiar este conjunto de datos. Pero sí se ha podido extraer el número total de registros de triángulos del grafo, gracias a la función **count()**, y se hablará sobre ello en el análisis de resultados.

Ranking de paginación de los nodos del grafo

El algoritmo del ranking de paginación mide la relevancia de cada nodo con respecto al grafo por medio de su número de relaciones y la relevancia de sus nodos vecinos. Esta medida nos servirá para obtener las carteras más importantes basadas en su volumen de transacciones.

Para ejecutar este algoritmo, necesitaremos la función **pagerank()**, que nos devolverá un dataset con los nodos y su ranking de paginación asociado. Como queremos obtener únicamente los más importantes, haremos uso de la función **orderBy()** para ordenar en base al ranking, y por último mostraremos los 5 primeros.

```
results = g.pageRank(resetProbability=0.15, tol=0.01)
results.vertices.select("id", "pagerank").orderBy("pagerank",
    ascending=False).show(5)
```

Extracto de código 5.12: Cálculo de los nodos más relevantes

Representación parcial y total del grafo

Debido a que el dataset cuenta con muchos nodos, de cara a estudiar ciertos sectores del grafo, se ha implementado la funcionalidad de representar parcialmente el grafo. En este caso, la funcionalidad que se pondrá de ejemplo es la representación de un nodo y todas sus transacciones salientes.

```
from pyspark.sql.functions import col
import random

vertexOutDegrees = g.outDegrees
senders = vertexOutDegrees.join(nodes, vertexOutDegrees.id ==
    nodes.id) \
                          .drop("id") \
                          .orderBy("outDegree", ascending=
                             False)

#Podriamos escoger cualquier direccion del conjunto de datos
   para representar

address = senders.take(1000)[999].Address

sub_graph = g.find("(src)-[e]->(dst)") \
             .filter(col('src.Address') == address)

def node_to_dict(r):
    return {
        'id': r[0],
        'label': r[1],
        'x': random.uniform(0,1),
        'y': random.uniform(0,1),
        'size': random.uniform(0.2,1)
    }

sub_nodes = sub_graph.select("dst.id", "dst.Address").
   distinct()
sub_edges = sub_graph.select("e.src", "e.dst")

target_nodes_dict = map(node_to_dict, sub_nodes.collect())

def edge_to_dict(i, r):
    return {
        'id': i,
        'source': r[0],
        'target': r[1]
    }

sub_edges_dict = [edge_to_dict(i, r) for i, r in enumerate(
   sub_edges.collect())]

target_nodes_dict.append({
    'id': sub_edges.first()['src'],
    'label': address,
```

```
    'color': '#999',
    'x': -1,
    'y': 0.5,
    'size': 2
})
```

Extracto de código 5.13: Representación parcial del grafo mediante filtro por transacciones salientes

```
import networkx as nx
G = nx.Graph()
G.add_edges_from(sub_edges.collect())
```

Extracto de código 5.14: Creación mediante NetworkX

```
import matplotlib.pyplot as plt
options = {
    'node_color': 'g',
    'node_size': 70,
    'width': 0.2,
    'node_shape': 'd',
    'with_labels': True,
    'font_size': 5,
}
nx.draw(G, **options)
```

Extracto de código 5.15: Representación del grafo mediante NetworkX y Matplotlib

Por otro lado, vamos a representar gráficamente el conjunto completo de todos los nodos y aristas del grafo.

Este proceso se realizará mediante las librerías GraphFrames para el procesado y la librería NetworkX para su representación gráfica, respectivamente.

```
sample_data = edges.sample(False, 0.0004).collect()
G2 = nx.Graph()
G2.add_edges_from(sample_data)

options = {
    'node_color': 'g',
    'node_size': 1,
    'width': 0.05,
    'node_shape': 'o',
    'vmin': 100.1,
    'vmax': 10.1,
    'with_labels': False,
}
nx.draw_random(G2, **options)
```

Extracto de código 5.16: Representación total del grafo

6. Exposición y análisis de resultados

6.1. Introducción

En este capítulo explicaremos la interpretación de los resultados obtenidos a partir del procesamiento de datos del Blockchain, así como las conclusiones a las que se han llegado a raíz de los resultados y las posibles mejoras del sistema que podrían realizarse.

6.2. Análisis de las propiedades del grafo

6.2.1. Grado de los nodos

En esta sección se detallarán las estadísticas recogidas a partir del estudio del grado de cada nodo del grafo. En primer lugar, generamos un conjunto de datos para estudiar el número de transacciones salientes y entrantes.

En primer lugar, se genera un conjunto de datos con las aristas salientes y entrantes de cada nodo del grafo.

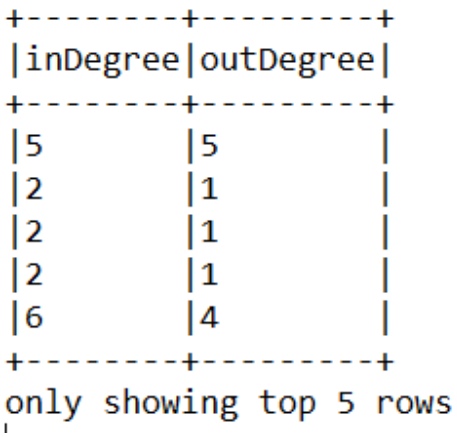


Figura 6.1: Conjunto de datos con el conteo de aristas salientes y entrantes

Y luego, gracias a la función **describe()** podemos calcular estadísticas básicas de este conjunto de datos. En esta tabla se detallan datos sobre el conteo total, la media, la desviación típica, el mínimo, y el máximo de aristas salientes y entrantes del grafo. Estas propiedades resultan de gran utilidad para realizar estudios estadísticos sobre grafos de gran volumen y establecer conclusiones a partir de estos análisis.

En esta tabla podemos observar que las transacciones salientes totales coinciden con las entrantes totales, lo cual concuerda debido a que cada arista tiene una cartera

summary	inDegree	outDegree
count	324431	324431
mean	6.188986255937318	4.691345771519984
stddev	275.2198543574708	19.964350474456104
min	1	1
max	121885	1994

Figura 6.2: Estadísticas recogidas del conteo de aristas salientes y entrantes

entrante y una saliente. Tenemos una media de 6.18 transacciones salientes y 4.69 transacciones entrantes, y por último es bastante interesante el dato del registro máximo de **121885** transacciones entrantes (posiblemente asociada a un banco Bitcoin), y **1994** transacciones salientes.

Por último, con la intención de determinar si existe una relación linealidad entre transacciones salientes y entrantes de cada cartera se ha realizado un diagrama de dispersión con una semilla aleatoria de nodos que nos muestra el siguiente gráfico:

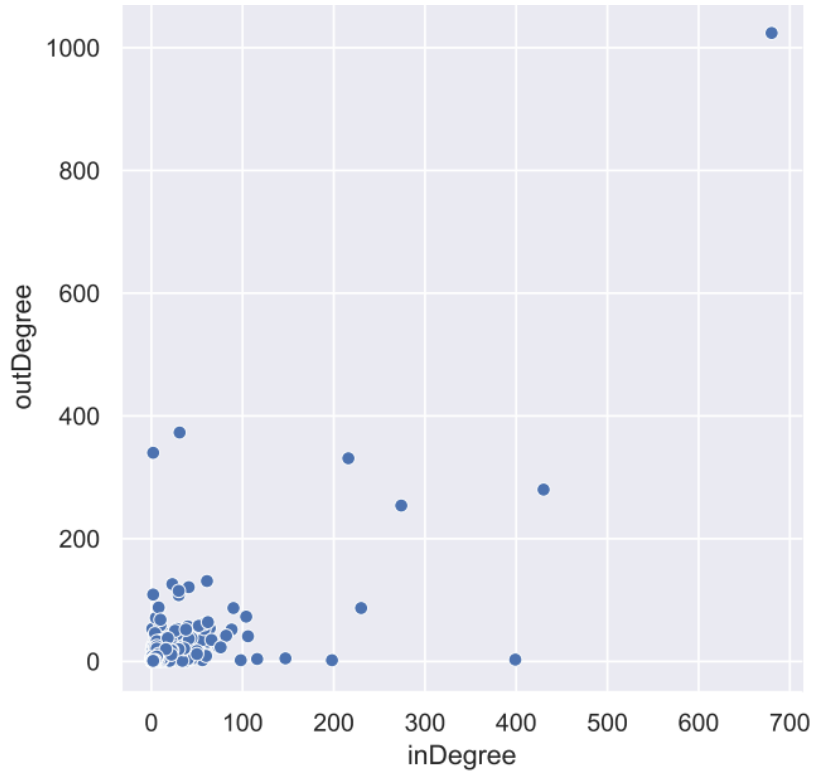


Figura 6.3: Diagrama de dispersión de aristas salientes y entrantes

Y a partir de estos datos también se calculó la correlación entre aristas entrantes y salientes.

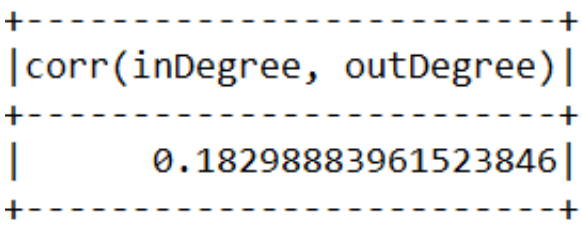


Figura 6.4: Coeficiente de correlación entre transacciones entrantes y salientes

A partir de este calculo, podemos determinar que estos dos atributos de los nodos mantienen una relación extremadamente débil. Esto se debe a que el coeficiente de correlación, que en este caso es 0.182 es muy próximo a 0. Como conclusión, añadiría que no ser las transacciones salientes y entrantes dos variables proporcionales, existe una gran comunidad de nodos que no se dedican a mover su capital. Una razón posible sería que el mercado Bitcoin en ese momento se encontraba en un momento de **fase alcista o periodo de acumulación**. En este momento los usuarios se dedican a conservar su inversión en lugar de distribuir y vender debido a que se espera una subida extremadamente grande, como la que se produjo en 2017, lo cual concuerda con los datos debido a que este historial de transacciones está registrado hasta 2016, donde la gente empezó a creer en el fenómeno Bitcoin y decidió invertir para esperar esta subida.

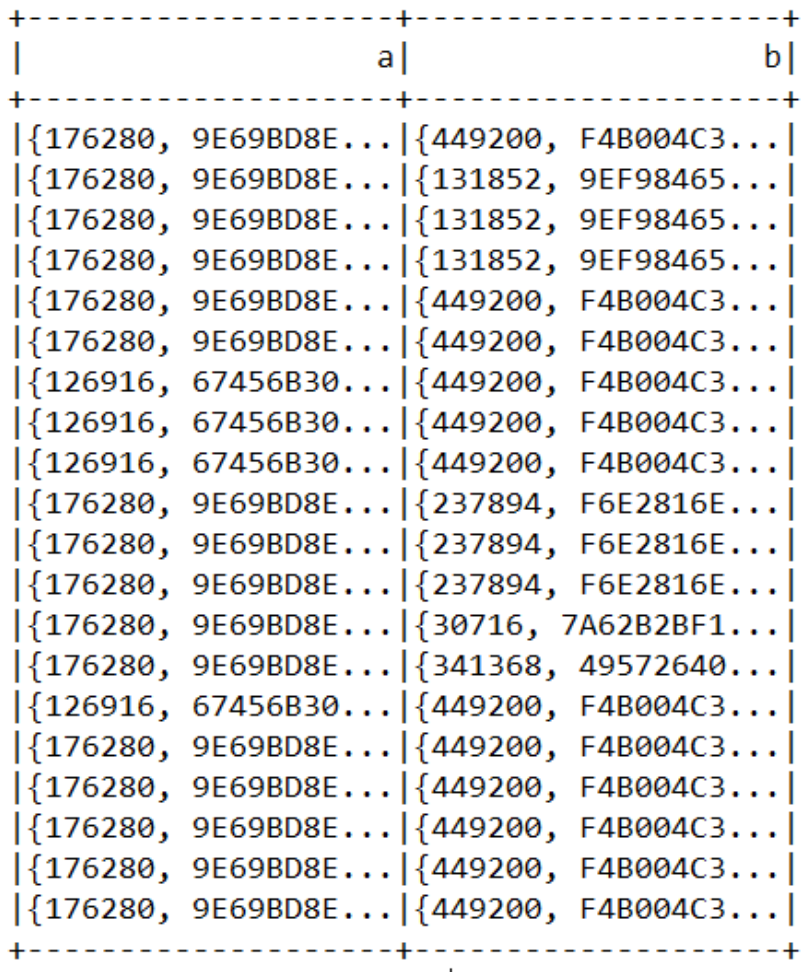
6.2.2. Búsqueda de patrones por medio de filtros

Gracias al sistema de filtrado que nos ofrece GraphX, podemos encontrar ciertos patrones dentro del conjunto de transacciones, y obtener subconjuntos más pequeños que reúnan condiciones interesantes para realizar un análisis. En este caso se ha realizado una consulta basada en que el nodo de entrada debe tener más de 1000 transacciones entrantes, y el nodo de salida debe tener más de 1000 transacciones salientes.

Gracias a estas herramientas de filtrado, se podrían implementar herramientas como sistemas de detección de transacciones legalmente irregulares, dado que podremos encontrar patrones en grandes volúmenes de transaciones dentro de la red.

6.2.3. Triangle counting

Como ya detallamos en la sección 5.3.3, que explicaba el conteo de triangulos, la exploración de este método en profundidad ha sido imposible debido al inmenso coste computacional de pasar el algoritmo por un grafo de gran tamaño. Al final, el resultado que ha sido posible obtener ha sido el número total de triangulos del grafo, que nos da una cifra final de **546651 resgistros**, esto nos indica que todos los nodos del grafo son perteneciantes como mínimo a un triángulo, dado que el número de registros coincide con el numero de nodos del grafo. Este dato se ha calculado en un tiempo de ejecución de **17 minutos**. En conclusión, teniendo en cuenta que el coste de ejecución medio de un bloque de instrucciones de este proyecto es de 1 minuto, podríamos anotar que esta



a	b
{176280, 9E69BD8E...	{449200, F4B004C3...
{176280, 9E69BD8E...	{131852, 9EF98465...
{176280, 9E69BD8E...	{131852, 9EF98465...
{176280, 9E69BD8E...	{131852, 9EF98465...
{176280, 9E69BD8E...	{449200, F4B004C3...
{176280, 9E69BD8E...	{449200, F4B004C3...
{126916, 67456B30...	{449200, F4B004C3...
{126916, 67456B30...	{449200, F4B004C3...
{126916, 67456B30...	{449200, F4B004C3...
{176280, 9E69BD8E...	{237894, F6E2816E...
{176280, 9E69BD8E...	{237894, F6E2816E...
{176280, 9E69BD8E...	{237894, F6E2816E...
{176280, 9E69BD8E...	{30716, 7A62B2BF1...
{176280, 9E69BD8E...	{341368, 49572640...
{126916, 67456B30...	{449200, F4B004C3...
{176280, 9E69BD8E...	{449200, F4B004C3...
{176280, 9E69BD8E...	{449200, F4B004C3...
{176280, 9E69BD8E...	{449200, F4B004C3...
{176280, 9E69BD8E...	{449200, F4B004C3...
{176280, 9E69BD8E...	{449200, F4B004C3...

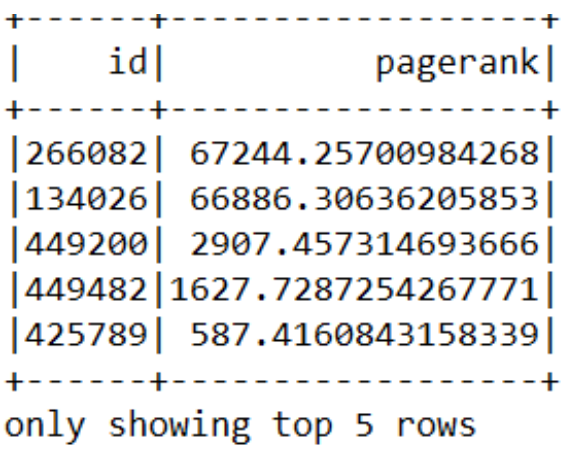
Figura 6.5: Búsqueda de patrones en las transacciones del grafo por medio de filtrado

función sobrepasa de manera excesiva el coste de procesamiento y por tanto no sería rentable en este contexto del proyecto.

Además, esta medida resulta de gran utilidad para estudiar agrupaciones de nodos dentro de la red y calcular coeficientes de clustering, pero por razones de insuficiencia de recursos de computación no han podido ejecutarse sobre este grafo.

6.2.4. Ranking de paginación

En este apartado se mostrará el listado de los 5 nodos más relevantes dentro del grafo de transacciones, así como su ranking de paginación asociado:



```
+------+------------------+
|    id|          pagerank|
+------+------------------+
|266082| 67244.25700984268|
|134026| 66886.30636205853|
|449200| 2907.457314693666|
|449482|1627.7287254267771|
|425789| 587.4160843158339|
+------+------------------+
only showing top 5 rows
```

Figura 6.6: Listado de nodos con mayor relevancia

Gracias a esta función, podemos apreciar qué nodos tienen más peso dentro del grafo total, y podría ser utilizada para identificar empresas y entidades bancarias dentro de la red Bitcoin, dado que estos nodos con mayor importancia serán los que más y mejor relacionados se encontrarán dentro del grafo.

6.2.5. Representación parcial y total del grafo

En este apartado nos centraremos en el dibujado del grafo de transacciones. Con el fin de explorar a fondo las funcionalidades de GraphX, se ha dividido esta sección en dos bloques: la representación de las transacciones asociadas a un nodo y la representación total en distintas formas.

Transacciones asociadas a un nodo

En primer lugar, se representarán gráficamente los subgrafos asociados a las transacciones salientes de un único nodo. En las siguientes figuras se presentan 3 ejemplos ordenados por volumen de transacciones de menor a mayor tamaño. La figuras 6.2.4 y 6.2.5 tendrán pocas transacciones, mientras que la figura 6.2.5 presentará muchos más nodos debido a que tiene muchas más transacciones salientes.

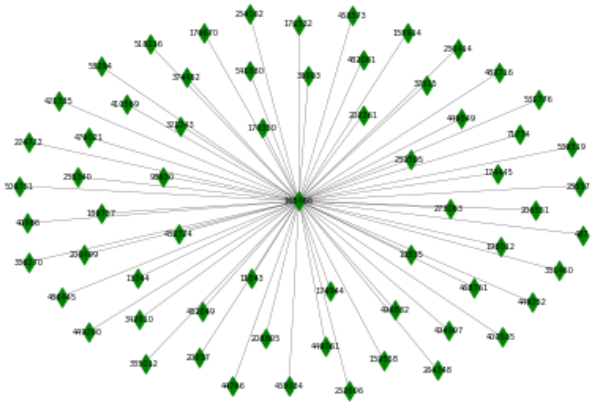


Figura 6.7: Representación de transacciones salientes del nodo 12

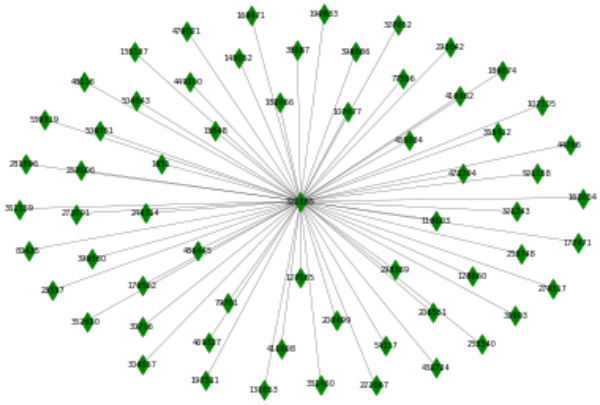


Figura 6.8: Representación de transacciones salientes del nodo 48

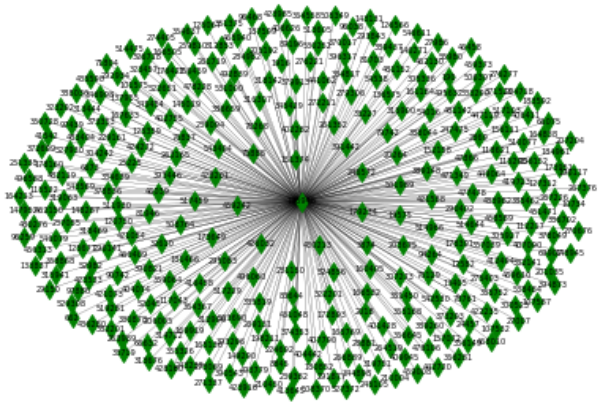


Figura 6.9: Representación de transacciones salientes del nodo 556

Representación total en distintas formas

Tras representar los subgrafos asociados a las transacciones de un único nodo, se han implementado dos maneras de dibujar el grafo completo. Gracias a la librería NetworkX, es posible realizar representaciones visuales de grafos de extrema complejidad.

Dado que dibujar la totalidad de los nodos es prácticamente imposible debido a las limitaciones técnicas, se ha hecho uso de la función de Spark **sample()** que coge un conjunto aleatorio de aristas de menor tamaño que el conjunto total y se realiza la representación gráfica a partir de este conjunto menor.

En el caso concreto de este grafo, se han representado el 0.003 % de los aristas del conjunto total. Cuando fuera posible, a nivel computacional, representar la totalidad del grafo contando con más recursos, lo único que tendríamos que hacer es no utilizar la función **sample()** y coger la totalidad del conjunto para realizar la representación.

Para la representación total se han utilizado dos gráficos. En primer lugar tenemos una representación en forma elíptica, donde los nodos conforman el perímetro de la elipse, mientas que las relaciones entre ellos se representan dentro de la figura. En segundo lugar, tenemos una representación rectangular. Aquí, los nodos se distribuyen de manera aleatoria dentro de la figura.

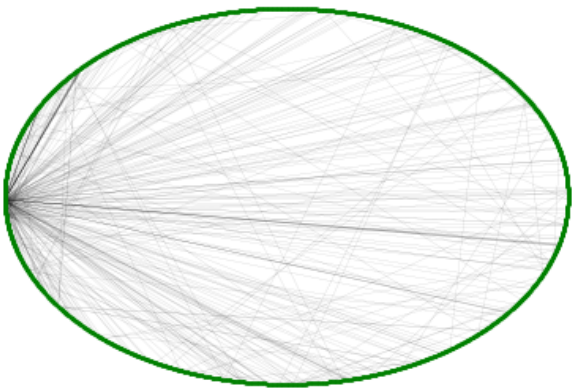


Figura 6.10: Representación total del grafo en forma elíptica

Mediante estos gráficos, podemos observar de manera visual cómo se presenta el grafo de transacciones. En las representaciones totales del grafo, pueden observarse con facilidad ciertos nodos que cuentan con un volumen muy alto de transacciones (muchas aristas salientes y entrantes). Estos denominados coloquialmente "Supernodos", corresponden en el contexto de la red Bitcoin, a grandes entidades bancarias con un frente de inversión en Bitcoin, o bien, a empresas que acumulan una parte importante de su capital en Bitcoin. Por tanto, gracias a esta representación visual, podemos entender con más claridad como se presenta la red de transacciones de esta criptodivisa.

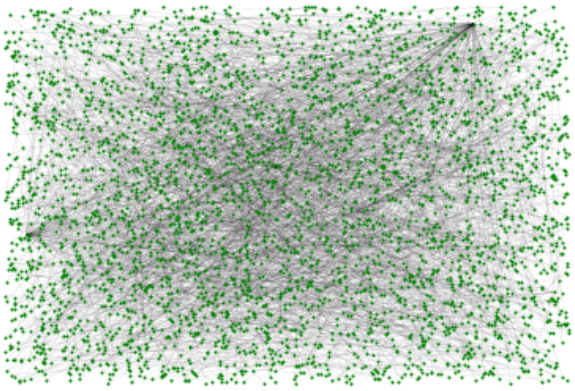


Figura 6.11: Representación rectangular del grafo

6.3. Posibles utilidades reales de los resultados

A la vista de los resultados obtenidos a lo largo de esta investigación, se han recopilado una serie de aplicaciones prácticas reales que podrían tener las funcionalidades desarrolladas en este proyecto.

En primer lugar, el hecho de que sea posible la representación gráfica de las transacciones por medio de su concepción como grafo, nos permite realizar análisis visuales de la distribución del gran volumen de transacciones que se encuentra en la red. Gracias a esta visualización, se pueden percibir de manera intuitiva, como ya se indicó en la sección 6.2.5, aquellos nodos que acumulan una gran cantidad de volumen de transacciones.

Esta herramienta, unida al desarrollo de una funcionalidad de fechado de transacciones, podría resultar de gran utilidad para detectar situaciones de **acaparamiento de capital**. Este fenómeno se produce cuando un único contribuyente de la red realiza grandes adquisiciones de capital y acumula un porcentaje relevante de una criptodivisa. Estos acaparamientos son llevados a cabo por bancos y grandes entidades internacionales, que actualmente poseen en 78 % del total de Bitcoin, dejando solo un 22 % para los traders. Las consecuencias que éste produce son la inestabilidad del valor de la moneda y el aumento del miedo entre los inversores, tal y como se indica en la figura 6.12. Por tanto, de cara a analizar este tipo de fenómenos, las funciones de representación gráfica pueden resultar de gran utilidad.

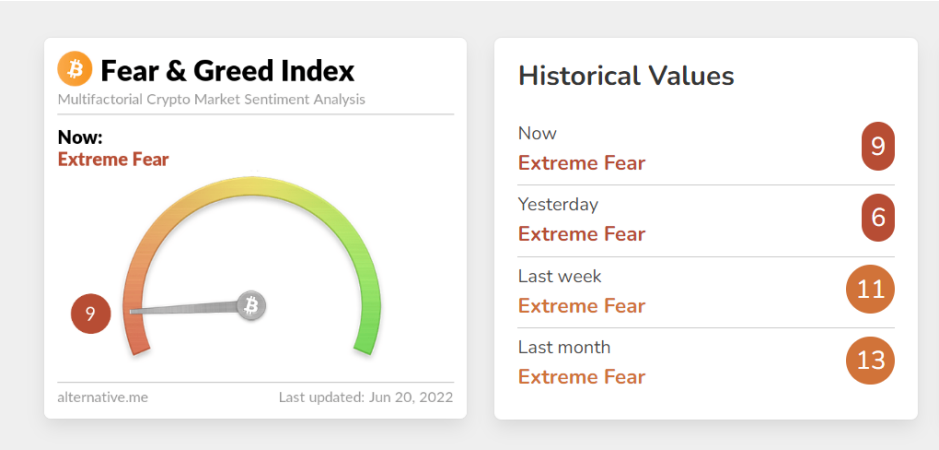


Figura 6.12: Índice de miedo y avaricia de Bitcoin del dia 20 de junio de 2022

6.4. Vectores de mejora del proyecto para futuras investigaciones

En este apartado se detallarán los posibles vectores de mejora de este proyecto, para que alumnos que quieran continuar esta investigación en un futuro cuenten con unas pautas para orientar el desarrollo de nuevas funcionalidades.

Como mejora principal, anotaría que una funcionalidad muy interesante podría ser el calculo del valor monetario de la transacción para la creación de un grafo direccionado con pesos. Esta funcionalidad nos permitiría saber qué cantidad de moneda se mueve en cada transacción. Y unido con el estudio de la relevancia de los nodos gracias al ranking de paginación que se ha implementado, permitiría **refinar el concepto de nodo relevante**, sabiendo no sólo cuántas transacciones maneja, sino también el valor de éstas. De esta manera, sería más fácil identificar qué carteras pertenecen a grandes instituciones y bancos.

Figura 6.13: Titular de acaparamiento de Bitcoin por Instituciones. 31 de Diciembre de 2020

Por otro lado, sería de gran ayuda para continuar con este estudio, la obtención de más propiedades dentro de las transacciones, tales como la fecha, el minero que confirma la transacción, y el número de contribuyentes y receptores. De cara a facilitar este desarrollo, existen una amplia gama de APIs que proporcionan de manera gratuita y pública toda la información necesaria.

Figura 6.14: Información detallada de una transacción del Blockchain

Por último, creo que otra mejora interesante sería la realización de un análisis profundo de rendimiento del sistema con mayores recursos de almacenamiento y computación. La mayor ventaja del motor Spark es la posibilidad de paralelizar el procesamiento de los datos a través de agrupaciones de máquinas. Por tanto, sería de gran ayuda un estudio que midiera el rendimiento del código con una mayor cantidad de recursos disponibles.

7. Conclusiones

7.1. Beneficios del Big Data y su análisis para Bitcoin y el resto de criptodivisas

La primera conclusión a la que he podido llegar durante el transcurso de este proyecto ha sido descubrir el enorme beneficio que tiene el análisis de grandes volúmenes de datos para la criptomonedas, en especial, para Bitcoin. Por supuesto, se ha recopilado información a lo largo de este proyecto que respalda esta conclusión. El sector financiero y bancario esta sufriendo un proceso de evolución y transición digital en estos últimos años. Esta evolución comprende desde el desarrollo de nuevas divisas digitales, hasta entidades bancarias totalmente digitalizadas y la aparición de nuevos proyectos de seguridad avanzada. Las organizaciones modernas tienen diversas soluciones para elegir.

Además, las nuevas teorías de Fintech (tecnologías de finanzas) [7, 15] van de la mano de otros avances tecnológicos relacionados como la Inteligencia Artificial, el Machine Learning y el analisis de grafos. El objetivo de la mayoría de las compañías de todos los sectores es reducir los errores costosos, la mala interpretación de las tendencias y las amenazas a la ciberseguridad.

Por tanto, gracias a todos estos avances, las divisas digitales cuentan con los recursos necesarios para garantizar factores de calidad fundamentales en los procesos financieros como son:

1. Reducción del error humano.
2. Aumento de la seguridad y reducción del fraude.
3. Herramientas para comprender las tendencias de los criptomercados.
4. Automatización del proceso.

7.2. Potencial de las herramientas de teoría de grafos para el análisis de datos

A lo largo de este estudio, se han investigado ciertas propiedades sobre la teoría de grafos que han aportado gran valor a los conocimientos sobre Bitcoin con los que ya contaba. Por tanto, la primera conclusión a la que he llegado desarrollando este proyecto es que el uso de las herramientas que nos brinda la teoría de grafos aplicado a los volúmenes de datos con relaciones entre ellos puede aportar nuevos caminos de análisis que no estarían disponibles sin contar con el factor relacional que nos prporciona un conjunto de datos estructurado en forma de grafo [7].

En el caso de Bitcoin, estas herramientas cobran aún más valor debido a que el concepto de relación entre carteras representa una transacción monetaria entre estas. El plantear este conjunto de datos como un grafo nos proporciona los recursos necesarios para estudiar fenómenos bursátiles como el acaparamiento bancario, el fraude fiscal, el blanqueo por movimiento continuado, etc [7].

7.3. Conclusión personal sobre el campo de investigación

En este capítulo se abordarán las conclusiones personales a las que he llegado a raíz de realizar este proyecto y estudiar en profundidad las técnicas y herramientas utilizadas para la implementación del código.

7.3.1. La complejidad del estudio

De cara a afrontar este proyecto, el concepto que tenía sobre la complejidad del mismo ha resultado ser muy diferente de la complejidad final. El área de conocimiento del Blockchain abarca una gran cantidad de campos de investigación debido a su popularidad y el grado de complejidad del sistema en sí.

Durante los primeros días del proyecto, dediqué una gran cantidad de horas a estudiar en profundidad este campo para escoger las herramientas óptimas para el desarrollo, y esta tarea de estudio acabó convirtiéndose en una tarea "agujero negro", que absorbía más horas de las que me podía permitir. Los motivos fueron: la complejidad de la estructura de los datos provenientes del Blockchain y la gran variedad de herramientas disponibles para su tratamiento.

Por otro lado, también empleé una parte importante del cómputo total de horas en estudiar fenómenos del mundo financiero con el propósito de buscar aplicaciones reales para las herramientas implementadas en el proyecto. Y este campo de estudio resultó ser inmensamente complejo a la vez que apasionante.

En conclusión, anotaría que haberme formado en este área ha supuesto un gran esfuerzo, pero al mismo tiempo ha sido una experiencia muy estimulante. Por lo que no consideraría la complejidad de este proyecto un punto negativo de éste.

7.3.2. Desvíos del plan de desarrollo inicial

Como ya mencioné con anterioridad en la sección 2.2, durante la fase inicial de este proyecto, los objetivos eran muy ambiguos. Esto provocó desvíos considerables en la planificación y la distribución de carga de trabajo del proyecto. Tras un análisis profundo de las causas de este desvío, y el impacto que tuvo en el desarrollo, llegué a la conclusión de que las causa principales fueron **que los objetivos eran bastante ambiciosos y fueron subestimados** y la **extrema complejidad de este campo de estudio**.

Una desviación en la línea de alcance o planificación de un proyecto puede tener consecuencias catastróficas en la puesta en marcha del desarrollo. Esta fue la mayor de mis preocupaciones cuando me sumergí en el desarrollo del código. Y un factor determinante para el éxito del proyecto fue la posibilidad de poder presentar este proyecto para otra asignatura (Complementos de Bases de Datos) junto con mi compañero **George Laurentiu Bogdan**, al cual me gustaría hacer una mención especial por su ayuda desinteresada en la fase de implementación de código. Sus conocimientos resultaron ser vitales para la finalización de este proyecto y la corrección de esos errores de planificación que cometí en las fases iniciales de mi investigación.

En la siguiente tabla se mostrará la desviación en horas que ha acabado sufriendo cara tarea:

Actividad	Coste en horas estimado	Coste en horas real	Porcentaje de desviación
1	30	40	33 %
2	25	30	20 %
3	25	25	0 %
4	30	35	17 %
5	30	25	-17 %
6	35	45	29 %
7	30	20	-50 %
8	20	10	-50 %
9	10	10	0 %
10	35	35	0 %
11	10	20	50 %
12	10	10	0 %
13	10	10	0 %

Cuadro 7.1: Tabla de desviaciones temporales de cada tarea

Como podemos observar, gran parte de las tareas han sufrido desviaciones temporales con respecto a mi estimación inicial. Pero afortunadamente, el computo de horas no se ha visto gravemente afectado debido a que ciertas tareas de desarrollo han resultado menos costosas de lo esperado. Como conclusión anotaría que la verdadera dificultad del proyecto consistía en aprender los conocimientos necesarios sobre la tecnología Blockchain, pero una vez adquiridos estos conocimientos, la implementación no resultaba excesivamente costosa.

7.4. Cierre

En esta sección me gustaría realizar una retrospectiva más personal del proyecto realizado. Así como un mensaje para alumnos que quieran continuar con esta rama de investigación en un futuro.

En primer lugar, como resultado de la realización de este proyecto, creo que el conocimiento que he adquirido documentando y realizando este proyecto es tremendamente valioso y me será de gran utilidad en el futuro. Me hacía una gran ilusión poder

cerrar mi paso por la universidad combinando dos campos que encuentro fascinantes como lo son la tecnología, y las matemáticas como herramienta de análisis de datos. Ni la complejidad, ni la extensión del campo de estudio, me han desmotivado en ningún momento. Por eso, animo a futuros compañeros a continuar realizando proyectos en este campo porque a pesar de su dificultad, lo he encontrado apasionante.

Y por último, pero no menos importante. Me gustaría subrayar la importancia y el valor de los conocimientos sobre gestión de proyectos adquiridos durante estos cuatro años de formación universitaria. Herramientas como el análisis técnico de requisitos, estrategias de descomposición del desarrollo, gestión de versiones; tienen un impacto significativo en la calidad del proyecto. Por esta razón creo que la formación en ingeniería informática te brinda los recursos necesarios tanto como para coordinar un proyecto, como para llevarlo acabo.

Es por estas dos grandes razones, por las que creo que este proyecto se ha ejecutado de manera exitosa. Y gracias al mismo, soy realmente consciente de lo importante y lo valiosa que es la enseñanza que he recibido.

"Las matemáticas son el idioma que usó Dios para escribir el mundo"

"Galileo Galilei"

8. Bibliografía

[1] Guía de los fundamentos para la dirección de proyectos, 2022. URL https://es.wikipedia.org/wiki/Gu%C3%ADa_de_los_fundamentos_para_la_direcci%C3%B3n_de_proyectos.

[2] Graphx programming guide, 2022. URL https://spark.apache.org/docs/latest/graphx-programming-guide.html.

[3] Modelado de software. wikipedia: La enciclopedia libre, 2022. URL https://es.wikipedia.org/wiki/Modelado_del_software#:~:text=El%20modelado%20de%20sistemas%20software,la%20comunicaci%C3%B3n%20con%20el%20cliente.

[4] Bit2Me Academy. ¿cómo funciona el blockchain- cadena de bloques?, 2022. URL https://academy.bit2me.com/como-funciona-blockchain-cadena-de-bloques/#:~:text=En%20esencia%20una%20red%20blockchain,con%20una%20matem%C3%A1tica%20muy%20avanzada.

[5] ESIC Business and Marketing School. Apache spark: Introducción, qué es y cómo funciona, 2018. URL https://www.esic.edu/rethink/tecnologia/apache-spark-introduccion-que-es-y-como-funciona.

[6] Comunidad. Scala - wikipedia, 2021. URL https://es.wikipedia.org/wiki/Scala.

[7] CEUPE: Centro Europeo de Postgrado. ¿cómo se benefician las criptomonedas del big data?, 2021. URL https://masterbigdataceupe.com/como-se-benefician-las-criptomonedas-del-big-data/#Las_criptomonedas_pueden_sacar_provecho_del_Big_Data.

[8] The Apache Foundation. Apache spark documentation, 2022 (Latest update). URL https://spark.apache.org/docs/latest/.

[9] Jorn Franke. Using hive to analyze bitcoin blockchain data, 2017. URL https://github.com/ZuInnoTe/hadoopcryptoledger/wiki/Using-Hive-to-analyze-Bitcoin-Blockchain-data.

[10] Jörn Franke. Using hive to analyze bitcoin blockchain data, 2017. URL https://github.com/ZuInnoTe/hadoopcryptoledger/wiki/Using-Hive-to-analyze-Bitcoin-Blockchain-data.

[11] Redacción KeepCoding. 10 motivos por los que debes aprender scala, 2022. URL https://keepcoding.io/blog/10-motivos-por-los-que-debes-aprender-scala/.

[12] Grupo Korporate. Fracaso de los proyectos informáticos, 2021. URL https://grupokorporate.com/por-que-fracasa-un-proyecto-ti/.

[13] Jirka Kremser. Bitcoin insights, 2018. URL https://github.com/jkremser/bitcoin-insights.

[14] LinkedIn. Sueldo promedio de un analista de datos en españa, 2022. URL https://www.linkedin.com/salary/explorer?countryCode=es&geoId=105646813&titleId=340.

[15] Raúl Jaime Maestre. Qué es fintech y por qué es el futuro de las finanzas, 2022. URL https://www.iebschool.com/blog/que-es-fintech-finanzas/.

[16] Patricia Mabel Pesado. Sistemas de software distribuidos y bases de datos distribuidas, junio 2006. URL http://sedici.unlp.edu.ar/handle/10915/20809.

[17] Hari Santanam. Apache spark para el procesamiento paralelo de big data, 2018. URL https://medium.com/datos-y-ciencia/apache-spark-para-el-procesamiento-paralelo-de-big-data-rapido-facil-c132243cee0.

[18] UNIR. ¿qué es la planificación de un proyecto?, 2018. URL https://www.unir.net/empresa/revista/planificacion-proyecto/.

[19] María Elena Verjaga Felgueras. Análisis de datos y extracción de conocimiento utilizando bigdata, 2018. URL https://hdl.handle.net/10953.1/8380.

[20] Wikipedia. Bitcoin, 2022. URL https://es.wikipedia.org/wiki/Bitcoin.