Anàlisi de la blockchain de Bitcoin

Ferran Plaza Rodríguez

Resum– Bitcoin, el conegut sistema de criptomoneda digital distribuït, s'ha estès considerablement en els dos últims anys. El gran volum de moviments monetaris que ha sofert fa que s'acumulin ja gairebé 80Gb de dades de transaccions registrades en l'històric públic, conegut com *blockchain*. Existeix una petita varietat de projectes open source dissenyats per llegir aquestes dades i representar-les en altres bases de dades per realitzar estudis sobre elles. La majoria d'aquests projectes obliguen a començar la lectura des de la primera transacció que existeix. Afegint que, o bé tenen un cost de rendiment i un temps d'execució elevats, o per altra banda fan una lectura massa superficial de les dades. Amb l'objectiu d'analitzar les dades contingudes a la blockchain i extreure estadístiques d'ús, en aquest article es proposa la implementació d'un sistema capaç de llegir les dades de les transaccions de Bitcoin sense fer servir el principi de la blockchain com a punt d'inici.

Paraules clau– Bitcoin, blockchain, mineria de dades

Abstract– Bitcoin, the well-known distributed digital cryptocurrency system, has expanded significantly in the last two years. The high volume of monetary movements that has experienced makes it accumulate almost 80Gb of registered transactions in his public history, known as *blockchain*. Exists a small set of open source projects designed for reading this data and represent them in different databases in order to realize the studies about transactions. The vast majority of them make mandatory to start from the first transaction to start reading. Adding the fact that on the one hand, they have a huge execution time and high performance costs, or either, on the other hand, they perform a superficial reading of the data. With the main goal to analyze the data contained in the blockchain and extract use statistics, in this paper, we introduce an implementation of a system capable of reading the blockchain data without the need to start from the first block of the chain.

Keywords– Bitcoin, blockchain, data mining

1 Introducció

Bitcoin és un sistema de criptomoneda descentralitzat, basat en una xarxa peer-to-peer. A diferència dels ecosistemes centralitzats als quals estem acostumats avui en dia, Bitcoin depèn únicament dels seus nodes participants per validar i certificar les transaccions. La forma d'establir un protocol per realitzar transaccions és repartir el poder de certificació entre un col·lectiu d'usuaris que s'encarreguen de filtrar els moviments monetaris. La funció d'aquest col·lectiu és captar les transaccions emeses per usuaris de la xarxa, validar-les i publicar-les al registre públic de transaccions.

Bitcoin és un sistema complex, ja que involucra diverses tecnologies focalitzades en les Tecnologies de la Informació (e.g. xarxes P2P, criptografia de corba el·líptica, etc.). Per aquest motiu, Bitcoin també atrau l'atenció en el camp de la recerca. Mitjançant els conceptes que implementa, molts altres sistemes que existeixen avui en dia són possibles (e.g. el sistema de DNS distribuït anomenat Namecoin, el concepte d'Internet distribuït que fa servir ZeroNet, etc.). Un d'aquests grans conceptes és el registre públic de transaccions que rep el nom de *blockchain*. Aquest emmagatzema gairebé 80Gb de dades referents a transaccions de la criptomoneda. Aquestes dades no contenen explícitament identificats als pagadors ni als receptors. Tot i així poden ser dades de caràcter sensible, ja que es poden establir relacions i heurístiques per realitzar apropaments a les persones involucrades en els pagaments.

La blockchain es troba replicada en tots els nodes de la xarxa que fan servir un client (i.e. un wallet) complet [1]. Depèn del wallet que es faci servir, l'estructura de carpetes pot prendre diferents formes. Tot i així, la blockchain presenta la mateixa forma en tots ells. La blockchain s'emmagatzema mitjançant operacions de baix nivell que compacten les dades en fitxers amb format .dat.

L'interès d'aquest projecte és buscar noves formes de fer

• E-mail de contacte: ferran.plaza@e-campus.uab.cat
• Menció realitzada: Tecnologies de la Informació
• Treball tutoritzat per: Jordi Herrera Joancomartí (dEIC)

servir les dades de la blockchain. La meta principal és establir un model de dades que permeti resoldre consultes per realitzar estadístiques. Aquestes estadístiques serviran tant per veure com es fa servir Bitcoin i discutir si és la forma correcta de fer-ho, com per proposar heurístiques que ens apropin a trencar la despersonalització dels pagaments.

En sistemes ja existents, o bé la lectura de la blockchain és superficial, o per altra banda el procés és molt lent i és obligatori començar pel primer bloc de la cadena per realitzar validacions. En aquest article es proposa desenvolupar un sistema que permeti llegir fitxers de la blockchain per separat, sense necessitat de començar pel primer. El sistema està format per un conjunt de mòduls que llegeixen el contingut de l'arxiu, analitzen i tracten les dades, per posteriorment presentar-les en una base de dades relacional sobre la qual realitzar consultes.

La resta de l'article està organitzat de la següent forma. La secció 1.1 és un estudi de viabilitat que exposa els possibles problemes que poden sorgir. La secció 2 explica que és Bitcoin i quines dades es poden fer servir. La secció 3 exposa la metodologia emprada per assolir els objectius del treball, mentre que la secció 4 mostra l'estat de l'art dels sistemes de bolcat d'informació de la blockchain. La secció 5 descriu la solució proposada i com s'ha implementat. A la secció 6 es presenta el funcionament junt amb el rendiment aconseguit. Finalment, a la secció 7 s'exposen els resultats obtinguts i a la secció 8 s'expliquen les conclusions.

1.1 Estudi de viabilitat

Per una banda, els objectius funcionals que es volen assolir són clars. Aquests són la correcta lectura de la base de dades (BBDD) blockchain i el tractament de les dades adient, que es faran servir com a base per realitzar estadístiques. Per altra banda, per obtenir un objectiu ideal, seria idoni processar la blockchain sencera en un temps raonablement petit. Seria encara millor, establir un model de dades adient que no penalitzés de forma exagerada realitzar les consultes desitjades. Aquests objectius no funcionals comporten possibles problemes que poden sorgir durant el desenvolupament. Intentar reduir els temps de còmput sense recórrer a la paral·lelització en threads o en nodes es limita a la potència del hardware de la màquina i a usar les eines d'implementació adequades, fent-les servir de forma adient. L'estudi de viabilitat conclou que el projecte es pot dur a terme tot i que poden sorgir imprevistos en forma de demandes de rendiment alt i temps de còmput elevats.

2 Criptomoneda Bitcoin

Bitcoin [1] és un sistema de pagament de criptomoneda virtual, descentralitzat, que no depèn de cap entitat financera. Bitcoin fa ús d'una xarxa P2P, on els usuaris són nodes participants que ajuden a sostenir el sistema. Els usuaris de la xarxa bitcoin es divideixen en dos grups que no són excloents. Per una banda hi ha els usuaris que fan ús de la moneda per pagar i rebre pagaments. Els pagaments que efectuen són transaccions realitzades mitjançant adreces semblants a comptes bancàries. Els usuaris poden posseir tantes adreces com vulguin fer servir. El principal motiu és perquè estan pensades per tenir un sol ús, per fer més difícil seguir les transaccions per la mateixa persona. Les adreces es creen mitjançant criptografia de clau asimètrica, on la clau pública demostra que un usuari posseeix una clau privada que no ha de compartir amb ningú. La moneda que es fa servir rep el nom de bitcoin. Les transaccions de bitcoins s'efectuen publicant les transaccions en un registre públic anomenat blockchain. Aquest procés no és trivial i en aquesta acció entra en joc el segon tipus d'usuaris de Bitcoin, els miners. Els miners s'encarreguen de recollir les transaccions emeses pels altres usuaris i les publiquen a la blockchain. Per fer-ho, necessiten calcular un hash menor que un llindar establert que permetrà a aquest bloc quedar registrat a la blockchain. El procés de càlcul del hash es coneix com "proof-of-work" [2].

2.1 Explicació de la blockchain

Una blockchain és una BBDD distribuïda que conté una llista de registres de dades en continu creixement. Aquests registres són una encadenament de blocs, on cada nou bloc va lligat a l'anterior mitjançant un hash. Això vol dir que els nous blocs obtenen el seu hash incorporant el hash del bloc anterior, evitant que un bloc anterior sigui modificat sense haver de canviar tots els blocs següents. Aquest concepte és conegut com a servidor de timestamps, amb el qual es pot demostrar que un fet ha esdevingut abans que un altre.

2.1.1 Dades incloses en el bloc

Cada estructura de bloc, en el cas de Bitcoin, té associades un conjunt de transaccions, que tanmateix contenen diverses adreces d'input i diverses d'output. A l'estructura bloc (Entitat 1) s'observen tots els camps que es troben a l'inici de la lectura d'un fitxer de la blockchain.

Entitat 1: Block

Magic number Nombre que es fa servir per conèixer l'inici d'un bloc.
Block size Mida del block en bytes.
Version Nombre de versió del bloc llegit.
Previous hash Hash del bloc anterior, si es modifica el bloc anterior aquest hash deixa de ser vàlid.
Merkle hash Hash calculat a partir de les transaccions contingudes. Evita que el client comprovi els hash de totes les transaccions.
Time El timestamp de creació del bloc.
Difficulty bits El hash d'un nou bloc ha de ser menor que l'objectiu de dificultat.
Nonce Camp que es pot modificar a voluntat per poder calcular hash diferents.
Transaction count Nombre de transaccions que conté el bloc.

El bloc conté el número de transaccions que indica al camp transaction count. Les transaccions (mostrades en l'Entitat 2) exerceixen de lligam entre inputs i outputs.

Entitat 2: Transaction

Version number Nombre de versió de la transacció.
Input count Nombre d'inputs que inclou la transacció.
Output count Nombre d'outputs que inclou la transacció.
Lock time Temps que triga la transacció a fer-se efectiva després de la seva inclusió en un nou bloc.

Els inputs de la transacció (Entitat 3) contenen un punter a l'output que estan gastant, això vol dir que si tenen més d'un output a gastar s'ha d'escriure dos inputs diferents amb la mateixa adreça.

Entitat 3: Transaction Input	
Previous hash	Hash de la transacció en la qual es van transferir bitcoins a aquesta adreça.
Output id	Output concret dins de la transacció indicada pel camp previous hash.
Script length	Mida que ocupa l'script que conté aquest input.
Script	Dades serialitzades que contenen informació del pagador que permet validar que aquest vol gastar una quantitat de bitcoins.

Finalment, els outputs (Entitat 4) contenen el script que conté l'adreça que ha de rebre el pagament més la quantitat de bitcoins que rebrà.

Entitat 4: Transaction Output	
Value	Quantitat de satoshis (unitat de mesura de bitcoins) que rep el receptor del pagament.
Script length	Mida que ocupa l'script que conté l'output.
Script	Dades serialitzades que contenen informació del receptor del pagament i que permetran en una transacció futura validar que realment pot gastar la quantitat rebuda.

2.1.2 Dades no incloses en el bloc

Indirectament, la blockchain conté totes les dades que es poden conèixer sobre els blocs. La part negativa és que no totes són fàcilment accessibles. Conèixer, per exemple, les adreces que fan i reben els pagaments requereix un procés de tractat de l'script. A més a més, durant els anys, mitjançant els Bitcoin Improvement Proposals (BIP) s'ha anat afegint més formes de fer servir els scripts per assolir nous objectius. Una altra dada interessant que no està inclosa directament a la blockchain són les adreces que ja han sigut gastades. Calcular això de forma convencional, significa recórrer les dades llegides de la blockchain en ordre invers, marcant com a gastades aquelles adreces que tenen un input que valida un output anterior. Per recòrrer la blockchain realitzant les verificacions s'ha de seguir els camps transaction hash i id, aquests senyalen la transacció d'output on van rebre el pagament anterior.

2.1.3 Estructura de les dades en un wallet

Teòricament, la forma que pren una blockchain és com s'ha dit fins ara. Tot i així, els wallets de Bitcoin necessiten fer ús de les dades que conté la blockchain de forma mínimament eficient. Un wallet necessita validar que una transacció es pot efectuar des d'una adreça, recorrent la blockchain a la inversa buscant si l'adreça té suficients bitcoins per permetre's el pagament. Per fer això, alguns wallets com Bitcoin Core [3] guarden metadades de la blockchain. Seguidament es pot observar l'estructura de carpetes que ens interessa de Bitcoin Core.

blocks/blk*.dat fitxers .dat que formen la blockchain.

blocks/index/* conté una base de dades en format LevelDB que indexa dades sobre la localització dels blocs dins dels fitxers .dat.

chainstate/* una altra base de dades en format LevelDB que emmagatzema metadades usades pel wallet Bitcoin Core. Aquestes indexen les adreces que encara no estan gastades per facilitar la feina de busca.

3 METODOLOGIA

La metodologia emprada per assolir aquest projecte és iterativa. Es comença fent una cerca per provar diferents sistemes existents i poder avaluar quin és més adient per al propòsit d'aquest treball. Seguidament es desenvolupen, una per una, noves funcionalitats que ajudin a dur a terme els objectius proposats. Per cada nova funcionalitat implementada, fent ús d'un explorador de la blockchain oen línia tal com blockchain.info [5], es contrasten dades en comú per identificar errors de lectura, tractament o anàlisi. Amb aquest punt de referència es pensa una possible solució i s'aplica. Aquest procés es repeteix reiteradament fins que les dades són congruents.

4 ESTAT DE L'ART

El punt de partida que es pren per cercar informació sobre com parsejar la blockchain de Bitcoin és Bitcoin Abe [4]. Aquest parser està escrit en python i és capaç de fer ús de diferents motors de bases de dades en SQL. Abe és molt complet, tot i així no permet carregar un bloc de la blockchain si no has carregat el seu anterior, per tant s'ha de començar des del primer bloc. Això és deu a què Abe possiblement realitza les validacions de les transaccions mentre carrega els blocs en la base de dades. Abe tampoc inclou les adreces dels inputs i els outputs.

L'apropament final que es pren és fer servir el codi presentat per Alex Gorale [6], el qual fa servir per ensenyar quin format té i com es llegeix la blockchain. És un programa curt, senzill i fàcil de modificar, cosa que el fa perfecte per afegir-li les modificacions adients per adquirir els resultats buscats.

5 DESCRIPCIÓ I IMPLEMENTACIÓ

Al llarg d'aquest apartat es mencionarà l'evolució que va prenent el codi adquirit [6] per permetre noves funcionalitats. L'evolució s'estructura en tres subseccions que segueixen l'ordre en què s'ha dut a terme el treball.

5.1 Procés d'extracció

En primer lloc, extraure les dades dels fitxers .dat és la part més senzilla, ja que el parser ja duu quasi tota la funcionalitat implementada. El parser fa servir el mòdul struct de python per desempaquetar els bytes del fitxer i fer-los llegibles. També té en compte que els bytes estan en format de xarxa, i.e. Big Endian. Com el sistema realitza la lectura sense fer el procés de validació, no és capaç de reconèixer els blocs

orfes, per tant els tracta com els altres blocs. Només executar el codi es pot apreciar com les dades contingudes a la blockchain surten impreses per la sortida estàndard de python. L'estructura que es mostra duu decoracions per indicar quan comencen seccions com els blocs o les transaccions. En veure l'output es pot fer servir alguns hash per executar cerques a blockchain.info. Lamentablement diverses cerques resulten fallides. El motiu sembla que és que alguns hash són més curts que els altres. Això és degut al fet que aquests no són correctes perquè els hi falten números, aparentment zeros. La falta de zeros és deguda a la funció que desempaqueta variables de tipus string no llegeix bé. Corregint la funció fa que tots els hash puguin ser investigables en els blockchain explorer. L'esmentat error és reportat al creador del parser perquè conegui l'esdeveniment.

Posseint un parser ja funcional es treballa en les millores a afegir. Una d'elles és obtenir les adreces per dividirles lògicament en dos tipus, multisignatura i convencionals. Cal dir que la subdivisió proposada no s'apropa gens a aquests termes en qüestió de com estan classificades en el protocol Bitcoin. A Bitcoin, els tipus d'adreça es diferencien per la forma que pren el camp anomenat script. Els scripts són cadenes de caràcters que contenen dades concatenades, a més de ser de mida variable. Per conèixer a priori la llargària del script i ser capaç de llegir-lo des del fitxer físic, existeix un camp de dades situat just abans del mateix script i que s'anomena script length. El nom del camp és auto significatiu, indica la llargària del script. Per entendre el context, la validació del script és una operació de dos o més passos. Existeix el concepte de locking script i el de unlocking script. Respectivament, el locking script tanca la possibilitat de gastar els bitcoins continguts a l'adreça de l'output. Només li serà permès a qui sigui capaç de resoldre el locking script fent ús d'un unlocking script.

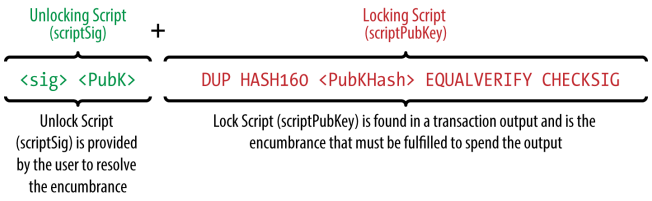


Fig. 1: Exemple de scripting

Els tipus de scripts que es poden apreciar més freqüentment en una anàlisi de la blockchain són a) P2PK, b) P2PKH, c) P2SH, d) Non spendable i e) Generation.

Un script d'input de generació és ignorat per a les validacions. Quan es generen nous bitcoins apareixen sense venir de cap adreça existent, el sistema "imprimeix nova moneda".

El més antic dels scripts que podem trobar, a part d'un input de generació, és el P2PK, on per fer el pagament s'ha de realitzar a la clau pública d'una adreça. Per altra banda, quan es vol gastar aquest output, fa falta que l'usuari de l'adreça signi amb la clau privada la nova transacció. Així es pot verificar que només l'usuari que posseïa la clau privada d'aquella adreça podia gastar aquell output. Es comprova mitjançant la clau pública.

P2PKH i P2SH són els més usats actualment i respectivament es corresponen a l'actual model d'adreça convencional i al model de multisignatura o al model de dipòsit.

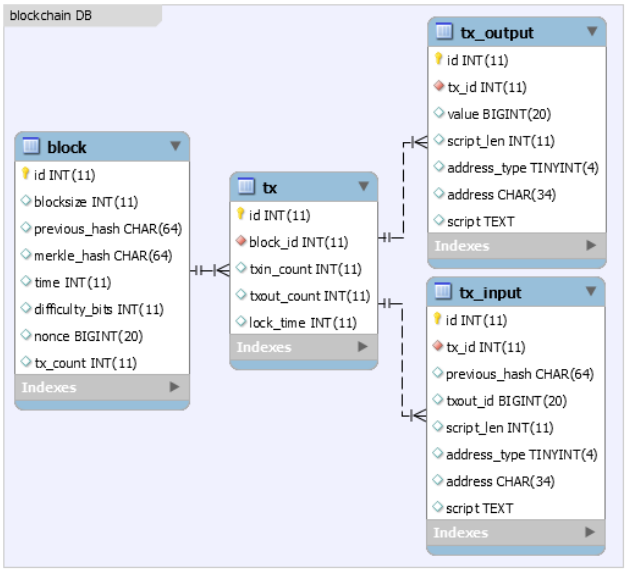


Fig. 2: Estructura de la base de dades

P2PKH és una versió modificada de l'anterior P2PK, en el que en comptes de posar una clau pública sencera es deixa en una forma més curta prèvia a transformar-se en adreça. El P2SH per altra banda, es fa servir per fer verificacions amb més d'una adreça en el mateix pagament. Aquest tipus de script apareix amb la proposta BIP 16 i una de les seves utilitats és permetre la multisignatura per verificar un pagament. Un altre dels seus usos coneguts són els anomenats smart contracts o escrow.

Finalment, el tipus non spendable és un output que ha sigut marcat amb una operació OP_RETURN, i.e. que el deixa inservible. Es fa servir habitualment per escriure timestamps a la blockchain o simples missatges que es quedaran per sempre.

Per fer servir la informació continguda en els scripts de les entrades i les sortides s'ha fet servir la llibreria de python pybitcointools [7]. Aquesta duu escrites funcions no acoblades que inclouen moltes operacions matemàtiques i processos que Bitcoin fa servir per transformar i tractar les seves dades.

5.2 Procés d'emmagatzematge

El software fa servir un sistema gestor de BBDD relacional. El motiu és que les dades que duu la blockchain emmagatzemades tenen una estructura definida i que és coneguda. El sistema gestor escollit és MySQL i l'esquema de la base de dades és el que es veu a la Figura 2. Les taules escollides són una imitació de l'estructura que contenen els fitxers de la blockchain. Alguns pocs camps han estat obviats (e.g. Magic number) pel fet que no són relevants per a realitzar cap de les estadístiques. Altres camps són nous, aquests són el tipus de script i l'adreça en cas d'haver-la. Aquests dos camps sorgeixen de la secció 5.1. L'esquema ha estat dissenyat fent servir l'eina MySQL Workbench. Aquesta eina és molt útil per fer esquemes entitat-relació visualment, així despreocupant a l'usuari que la fa servir de la nomenclatura específica de MySQL. Un cop fet l'esquema, l'eina duu una comanda per exportar l'esquema a fitxer SQL. Per la correcta manutenció i facilitat de canviar l'esquema, la creació de

les taules s'ha gestionat fent que el parser executi el fitxer SQL que es genera. Les insercions, per altra banda, estan escrites dins del codi. Fer servir la llibreria de python per executar consultes a MySQL permet fer les insercions a les taules, tot i així, aquestes operacions es poden fer transaccionalment (i.e. executar tots els *INSERT* alhora fent ús de la funció $commit$). Això evita que algun error d'inserció provoqui un problema d'integritat a la resta de taules de la BBDD en cas que alguna clau forana quedi a $null$. El commit de les insercions es fa cada cop que totes les operacions d'inserció d'un bloc han estat carregades al buffer.

5.3 Procés d'anàlisi

Per al procés d'anàlisi, amb la blockchain ja carregada en BBDD, es decideix executar consultes sobre el mateix MySQL Workbench. Gran part de les estadístiques que resulten interessants relacionen els inputs i els outputs unint-los en transaccions. Treballar amb tantes files de dades en una base de dades relacional és un problema. Executar sentències que contenen subqueries o joins resulta ser lent. En MySQL existeixen els conceptes de temporary tables i vistes. Les temporary tables són taules que existeixen en una sessió de MySQL i.e. una pestanya de MySQL Workbench. En canvi, una view és una taula precompilada que no conté dades sinó que les recupera de taules ja existents. En resum, el que es pot fer amb aquests dos tipus de consultes és pre-generar una consulta que es fa servir com a subconsulta. Això ajuda al temps de resposta d'una sentència complexa. Fent una prova ràpida de temps de resposta s'arriba a la conclusió que la que provoca una millora significativa en el rendiment són les vistes. Ja que interessa tenir a mà consultes que relacionin inputs i outputs, es creen dues vistes, una que junta inputs i outputs fent servir el id de la transacció que els relaciona i un altre que calcula el nombre d'inputs i el nombre d'outputs que conté cada transacció de la BBDD.

6 Funcionament del sistema

En aquesta secció es presenten els resultats obtinguts en cadascuna de les etapes explicades a l'apartat 5. Per al procés d'extracció es mostren temps de rendiment en l'obtenció de les dades des d'un fitxer .dat extret de la blockchain sense fer ús de la base de dades per emmagatzemar. El procés de guardat fa ús de l'extracció més el temps de penalització per inserir totes les dades a la base de dades en MySQL mitjançant els inserts natius de SQL. Al procés d'anàlisi és on finalment es presenten algunes de les estadístiques que es volien extreure com a objectiu.

6.1 Procés d'extracció

En l'extracció es té un bon punt de partida fent servir una eina ja desenvolupada amb anterioritat. L'eina conté poc codi, és ràpida i efectiva. Les modificacions afegides fan que hagi de fer servir poder de càlcul per obtenir les adreces a partir dels scripts. Tot i això, l'extracció no és un punt penalitzant del programa. Com es pot veure en la Taula 1 els temps que es triga a processar un arxiu de la blockchain de l'any 2016, de mida 130.529 KB, és aproximadament un minut. Per contrastar, es fa servir també el primer arxiu de la blockchain, que tot i tenir una mida semblant, té una densitat de blocs més alta, ja que gairebé tots els primers blocs generats a Bitcoin només tenen la transacció de generació. El temps de càrrega del fitxer és d'un minut aproximadament també. La mateixa prova es realitza amb fitxers dels anys 2012 i 2014. La prova dóna com a resultat un minut i mig de mitja aproximadament. Tenint en compte que hi ha més de 500 arxius a la blockchain, en menys d'un dia es faria possible processar tots els fitxers prenent en consideració el temps mig esmentat.

TAULA 1: Temps procés d'extracció

Fitxer blockchain	Any	Temps
blk00000.dat	2009	1 min 27 seg
blk00020.dat	2012	1 min 32 seg
blk00200.dat	2014	1 min 37 seg
blk00522.dat	2016	1 min 31 seg

6.2 Procés d'emmagatzematge

Aquesta secció presenta el resultat d'afegir a l'extracció el component per emmagatzemar les dades a les taules de MySQL. Fent la mateixa classe de proves sobre els mateixos fitxers de dades, a la Taula 1, es pot observar un increment considerable en el temps d'execució. Començant pel fitxer més recent, el de 2016, el temps d'execució ha incrementat cinc minuts i mig. Això és per al millor dels casos, ja que un fitxer amb poca densitat de transaccions ha de guardar més estructures del tipus bloc. Anys enrere, quan Bitcoin no es feia servir tant com ara, els blocs també es generaven cada deu minuts aproximadament. Els blocs contenien menys transaccions, per tant, ocupaven menys espai i cabien més blocs per arxiu. Guardar més estructures de bloc proboca que es realitzin moltes més operacions de commit a la base de dades. Els commits buiden el buffer d'instruccions i executen les comandes de forma atòmica. La Taula 1 reflecteix l'increment significatiu de temps de còmput que significa tenir més estructures de bloc per arxiu.

La Figura 3 mostra la diferència entre els temps d'extracció i d'emmagatzematge. En total, per al fitxer més recent, un 79% del temps d'execució és d'emmagatzematge, mentre que un 21% és el que triga a llegir el fitxer i calcular les adreces de cada input i cada output. En el cas del fitxer més antic, aquesta diferència encara es fa més evident, ja que el 89% del temps d'execució es fa servir en el procés d'emmagatzematge. Amb les noves mesures de temps preses es pot estimar que llegir la blockchain sensera trigaria al voltant de 3 dies prenent 9 minuts com temps mig aproximat.

6.3 Procés d'anàlisi

En el procés d'anàlisi es comprova l'efectivitat de realitzar consultes sobre les dades extretes. S'observa tant la rapidesa en la resposta com l'escalabilitat que pot tenir incrementant el nombre de dades emmagatzemades alhora.

El primer pas realitzat, abans d'operar amb MySQL Workbench, és augmentar el temps de timeout al màxim,

TAULA 2: TEMPS D'EXTRACCIÓ I EMMAGATZEMATGE

Fitxer blockchain	Any	Temps
blk00000.dat	2009	12 min 10 seg
blk00020.dat	2012	8 min 21 seg
blk00200.dat	2014	8 min 17 seg
blk00522.dat	2016	7 min 04 seg

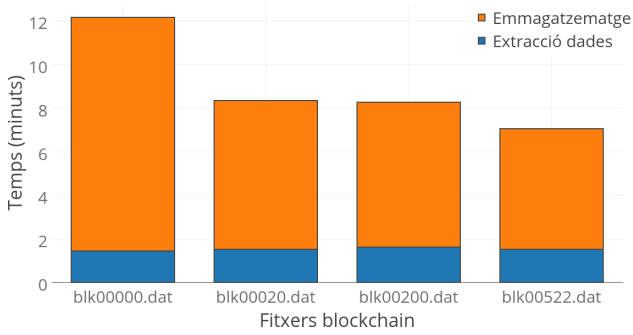


Fig. 3: Temps d'extracció i emmagatzematge per arxiu de blocs, mesurat en minuts i segons

canviant-lo de deu minuts a un dia. Independentment del pas anterior, les consultes realitzades sobre acumulacions de milions de files afecten greument el rendiment del servei MySQL, arribant fins i tot a col·lapsar-lo en alguns casos. La solució proposada per disminuir l'impacte d'aquest contratemps és fer ús de taules temporals o de vistes. Ambdues són taules que aparentment existeixen i es poden fer servir com les altres, tot i que no contenen dades, en lloc d'això permet que els seus valors siguin el resultat d'una consulta que romandrà precompilada. La diferència més significativa entre la vista i la taula temporal és que la taula només existeix en el context de la sessió que s'està executant en MySQL. En el cas de MySQL Workbench, cada pestanya representa una sessió. Tot i que la vista es crea amb relativa facilitat i rapidesa, el cas de la taula temporal es completament contrari. Ambdues s'intenten crear amb la mateixa consulta de *SELECT*. Tot i així, la taula temporal triga molt més a executar-se i atura la consulta forçadament llançant una excepció.

L'única opció viable, la vista, dóna bons resultats d'execució, per sobre de les operacions de *JOIN* directes. A la Sentència SQL 1 es mostra la vista que es fa servir per realitzar una unió entre les taules d'entrades i sortides de cada transacció mitjançant la clau forana *tx_id*. Aquesta vista es fa servir en la majoria de consultes, ja que les estadístiques que es volen extreure relacionen freqüentment les adreces d'entrada i les de sortida entre elles o bé amb la transacció que les engloba.

La Sentència SQL 2 crea una vista que realitza una conta de les adreces d'entrada i les de sortida per separat. Aquesta vista consumeix més recursos, ja que conté una unió junt amb una conta d'elements diferents. En SQL, tant el *JOIN* com el *DISTINCT* són costosos de calcular.

Aquestes dues vistes són les que es fan servir per alleugerar el temps de càlcul amb les sentències proposades a la secció 7.

Sentència SQL 1: Vista que uneix les taules tx_input i tx_output mitjançant un *JOIN* sobre la clau forana tx_id. Vista anomenada a partir d'ara $joinsinsouts$.

```
1:  CREATE OR REPLACE VIEW joininsouts AS (
2:      SELECT tx_input.tx_id,
3:          tx_input.id AS in_id,
4:          tx_output.id AS out_id,
5:          tx_input.address_type AS in_type,
6:          tx_input.address AS in_addr,
7:          tx_output.address_type AS out_type,
8:          tx_output.address AS out_addr,
9:          tx_input.script AS in_script,
10:         tx_output.script AS out_script,
11:         tx_output.value
12:     FROM tx_input
13:         INNER JOIN tx_output
14:         ON tx_input.tx_id = tx_output.tx_id
15: );
```

7 RESULTATS

En aquesta secció es presenten els resultats de fer consultes amb diferents quantitats de dades de la blockchain per extraure estadístiques d'ús del sistema. La forma en què es representaran és plantejant preguntes que siguin possibles de resoldre executant consultes a la base de dades. També els límits que l'aplicació pot abastar a causa de contratemps de rendiment sorgeixen i que es plantejaven a l'estudi de viabilitat (Secció 1.1).

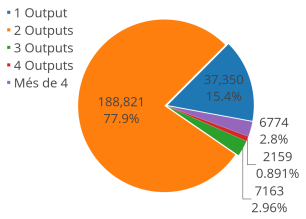
La primera qüestió tracta de saber quantes transaccions contenen exactament un nombre d'outputs concret. Per resoldre aquesta qüestió la consulta realitzada és la que es presenta a la Sentència SQL 3.

Com es menciona a la secció 6.3, la vista presentada a la Sentència 2 consumeix recursos elevats. En el cas exposat a la Sentència 3 es fa difícil d'executar a partir de les 250.000 transaccions. En els resultats que s'exposen, en conseqüència, el nombre de transaccions és inferior.

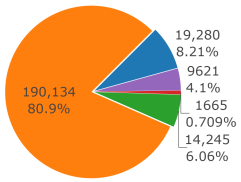
A la Figura 4 s'exposen tres conjunts de dades de diferents èpoques; la Figura 4a mostra el nombre d'outputs habitual en blocs generats en l'any de realització d'aquest tre-

Sentència SQL 2: Vista que uneix les taules $inputs$ i $outputs$ i conta quantes adreces conté cada transacció, tant les d'entrada com les de sortida. Rep el nom de $joinnuminsouts$.

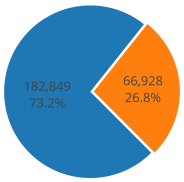
```
1:  CREATE OR REPLACE VIEW joinnuminsouts AS
    (
2:      SELECT tx_input.tx_id,
3:          COUNT(DISTINCT tx_input.id)
4:              AS in_count,
5:          COUNT(DISTINCT tx_output.id)
6:              AS out_count,
7:      FROM tx_input
8:          INNER JOIN tx_output
9:          ON tx_input.tx_id = tx_output.tx_id
10: );
```



a) blk00522.dat de principis de 2016. Carregats 164 blocs amb un total de 242.267 transaccions, equivalents a 27 hores de moviments de bitcoins aprox.



b) fitxer blk00200.dat de finals de 2014. Carregats 419 blocs amb un total de 234.945 transaccions, equivalents a 3-4 dies de moviments de bitcoins aprox.



c) part del fitxer blk00000.dat de l'any 2009. Carregats 105300 blocs amb un total de 249.780 transaccions, equivalents a 2 anys de moviments de bitcoins aprox.

Fig. 4: Representació del percentatge de número de sortides per transacció observat en diferents anys

Sentència SQL 3: Trobar el número de transaccions amb un nombre concret d'outputs, on n és el nombre d'outputs a escollir.

```
1: SELECT COUNT(1)
2: FROM joinnuminsouts
3: WHERE out_count = n
```

ball. El número d'outputs més corrent és dos o un, mentre que la resta són minoritaris. El mateix succeeix amb la Figura 4b, tot i que les transaccions amb més de dos outputs tenen una mica més de protagonisme que en el cas anterior. Finalment, la Figura 4c mostra aproximadament 2 anys de transaccions amb només un i dos outputs.

La següent qüestió tracta de relacionar les adreces involucrades en una transacció a un possible mateix usuari. Per plantejar la situació s'haurà d'establir una heurística que ens permeti aproximar una forma de vincular diverses adreces a la mateixa persona. Estimar de forma correcta si una adreça pertany a un mateix individu és complicat. La forma de procedir que es proposa és la següent: primer, es planteja efectuar una cerca de les transaccions que contenen exactament un input i dos outputs. La qüestió a partir d'ara es resoldrà sobre aquest col·lectiu reduït. A partir d'aquest subconjunt de transaccions, la relació més fàcil de trobar, per identificar si una persona es retorna el canvi a si mateixa, és que una de les adreces dels outputs sigui la mateixa que realitza el pagament. La Sentència SQL 4 és la proposada per trobar aquest conjunt de transaccions. En els resultats es diferència entre les transaccions que l'input és multisignatura i les que és no multisignatura.

Sentència SQL 4: Trobar el número de transaccions amb un input i dos outputs i que una de les adreces d'output sigui igual que la de l'input, on t és el tipus d'adreces que fer servir (multisignatura o no).

```
1: SELECT COUNT(DISTINCT joininsouts.tx_id)
2: FROM joininsouts
3:     INNER JOIN joinnuminsouts
4:     ON joininsouts.tx_id = joinnuminsouts.tx_id
5: WHERE in_count = 1 AND out_count = 2 AND
   in_type = t AND in_addr = out_addr
```

La resta del grup d'estudi es proposa subdividir-lo en dos conjunts més per cada tipus d'adreça. Tenint en compte que cap de les adreces d'output és igual a la d'input (resolt per la Sentència SQL 4), quins inputs, tant multisignatura o no, paguen a una adreça del mateix tipus i a una del tipus oposat. Per decidir un resultat a aquest grup, es proposa com a heurística donar per suposat que un usuari no té intenció de canviar el seu tipus d'adreça. Per tant es proposa contar les adreces que pertoquen a aquest grup com que pertanyen a una mateixa persona i que s'està retornant el canvi a sí mateix fent servir una nova adreça.

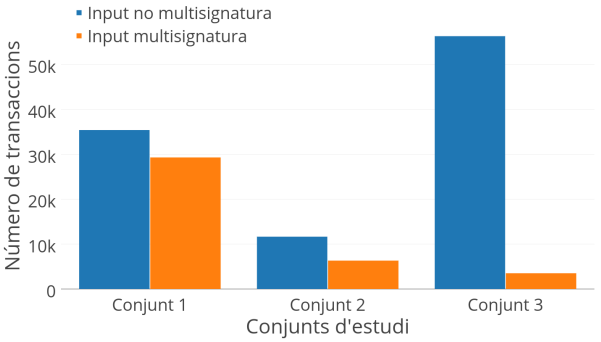
Fent servir la mateixa consulta es pot trobar el conjunt de transaccions que falta, ja que el detall que els diferencia és tenir un output del mateix tipus que l'input o tots dos. En el cas que totes tres adreces involucrades en la transacció siguin del mateix tipus i cap d'elles sigui igual definim la següent heurística: només s'afirmarà que una dues d'aquestes adreces pertanyen al mateix pagador en cas que un dels dos outputs rebi un import inferior o igual a un 1% de la quantitat que rep l'altre. Per classificar finalment aquest últim grup només es necessita modificar lleugerament la consulta anterior per afegir un *HAVING* SQL en què $MIN(value) \leq MAX(value)$ / 100.

Les proves es realitzen amb el conjunt de blocs de l'arxiu blk00522.dat. Es fan servir 142.510 transaccions de punt de partida, ja que són les que tenen un sol input i dos outputs. La Figura 5 conté tota la informació extreta d'aquestes transaccions. En el conjunt de dades, durant tot el procés, es tracta per separat per una banda la part en que l'input és multisignatura i la part en què no ho és. En primer lloc, a la Figura 5a es classifiquen les adreces separant-les en els tres col·lectius proposats anteriorment. El Conjunt 1 és aquell on s'inclouen adreces que fan servir incorrectament les adreces de Bitcoin. Això, sense necessitat de cap heurística, permet afirmar que aquesta persona està retornant-se el canvi de la transacció a si mateix i la resta l'està pagant a l'altre output involucrat. Tant en adreces multisignatura com en normals s'aprecia un mal ús força extès, d'entre 30.000 i 40.000 transaccions mal efectuades.

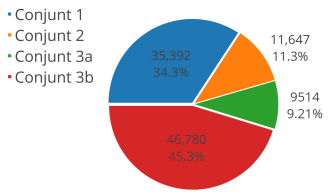
Per al Conjunt 2 no es veu masses casos en cap dels dos tipus d'input. L'heurística que s'ha definit permet marcar totes aquestes transaccions indicant que l'input i l'output que coincideixen en tipus pertanyen a la mateixa persona, i aquesta s'està tornant el canvi. Finalment, per al Conjunt 3 hi ha molts menys casos d'adreces multisignatura que paguen a una altra multisignatura i es retornen el canvi a una nova multisignatura que el mateix cas plantejat per una adreça no multisignatura. De fet, és força freqüent que una adreça convencional tingui dos outputs del mateix ti-

pus. Per classificar aquest conjunt, com s'ha comentat, s'ha proposat una heurística que determina en aquelles transaccions en que l'output rep menys d'un 1% del valor pagat a l'atre output, es decideix que l'output amb l'import menor pertany a la mateixa persona l'adreça amb la que es paga. Es pot veure a la Figura 5b que el Conjunt 3 s'ha separat en dos parts, on 3a és el cas en que l'import és menor a l'1%. Passa el mateix amb la Figura 5c.

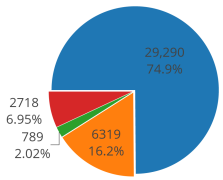
Finalment, la conclusió a la qüestió s'exposa a continuació: de les 142.510 transaccions amb un input i dos outputs, per a inputs no multisignatura, el 34,3% pertany al Conjunt 1; el 11,3% al Conjunt 2 i un 9,21% al Conjunt 3. Això significa que a un 55% dels inputs els trobem relació amb un dels outputs de la transacció, estimant que possiblement s'estiguin retornant el canvi de la transacció. Això és gairebé un 40% sobre les 142.510 transaccions inicials. Per altra banda, un 45% de les transaccions que pertanyen al Conjunt 3b no queden coberts per cap de les heurístiques proposades, el que vol dir que no queda clar si hi ha una relació entre un input i un output de la transacció. En el cas de que l'input sigui multisignatura, les heurístiques plantejades classifiquen més transaccions. A la Figura 5c s'observa com un 75% de les transaccions pertany al Conjunt 1, que ja sabem que es retornen el canvi. Afegint un 16% del Conjunt 2 i un 2% més del Conjunt 3a, s'ha trobat relació entre l'input i un dels outputs al 93% de les transaccions amb input multisignatura, que és equivalent a un 25% sobre les 142.510 transaccions inicials. Igual que en el cas anterior, el Conjunt 3b queda sense classificar. Del total de 142.510 s'han classificat un 65% com a positives i un 35% no s'ha pogut classificar.



a) Representació, en nombre de transaccions, de cada col·lectiu exposat. El Conjunt 1 són les transaccions que directament opten per pagar-se el canvi a la mateixa adreça. En el Conjunt 2 són adreces que paguen a dos outputs de diferents tipus entre ells i el Conjunt 3 són adreces que paguen a dos outputs del mateix tipus que l'input



b) Divisió de les adreces amb un input no multisignatura i dos outputs separats en els percentatges que representen a cada col·lectiu, contant com a 3a la part del 3 que la heurística aprova i 3b les que no.



c) Divisió igual que 5b però amb un input multisignatura

Fig. 5: Representació dels col·lectius proposats en aquesta secció per aproximar relacions entre diferents adreces que pertanyen a la mateixa persona fent servir els blocs de l'arxiu blk00522.dat.

8 Conclusions

S'ha presentat un programa que permet llegir els fitxers de dades de la blockchain i extreure'n els camps que conté. El programa no necessita començar pel primer bloc de transaccions per llegir les dades. Està escrit en python i fa servir MySQL per emmagatzemar dades massivament. Es fa servir MySQL i el seu software MySQL Workbench per realitzar mineria de dades massivament. Mitjançant el sistema implementat s'han pogut extraure una part de les estadístiques desitjades.

Agraïments

M'agradaria agraïr en primer lloc a Jordi Herrera el temps que m'ha dedicat setmanalment i l'ajut que m'ha proporcionat per dur aquest treball endavant. També voldria agraïr el suport de la meva parella i els amics que m'han animat fins al final.

Referències

[1] Andreas M. Antonopoulos. Mastering Bitcoin. Editorial O'Reilly (Primera edició), 2014.

[2] Satoshi Nakamoto. Bitcoin: A Peer-to-Peer Electronic Cash System. https://bitcoin.org/bitcoin.pdf. Ultim accés 26-06-2016.

[3] Bitcoin Core. https://bitcoin.org/en/bitcoin-core/. Últim accés 05-06-2016.

[4] Bitcoin Abe. https://github.com/bitcoin-abe/bitcoin-abe. Últim accés 08-06-2016.

[5] Explorador de la blockchain de Bitcoin online blockchain.info. https://blockchain.info/. Últim accés 11-06-2016.

[6] Alex Gorale. How to program block chain explorers with python. http://alexgorale.com/how-to-program-block-chain-explorers-with-python-part-1. Últim accés 26/06/2016.

[7] Vitalik Buterin. Paquet python PyBitcoinTools. https://github.com/vbuterin/pybitcointools. Últim accés 26/06/2016.