ĐẠI HỌC QUỐC GIA TP HỒ CHÍ MINH
TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN
KHOA CÔNG NGHỆ THÔNG TIN

BÁO CÁO MÔN
CÔNG NGHỆ CHUỖI KHỐI

CAO HỌC KHÓA 32 - 2022

Lab 1: Simplified Blockchain Implementation and Verification

Giáo viên hướng dẫn: **PGS. TS. Nguyễn Đình Thúc**

Học viên thực hiện: **22C11026 – Lâm Phạm Bá Tùng**

22C11022 – Nguyễn Trương Tấn Sâm

22C11049 – Trương Công Triều

TPHCM - 12/2023

MỤC LỤC

1 Giới thiệu

Trong thế giới số hóa ngày nay, Blockchain đã nổi lên như một công nghệ cách mạng, mang lại khả năng bảo mật và minh bạch cho các giao dịch số. Blockchain là một công nghệ cơ sở dữ liệu phân tán, nổi bật với khả năng bảo mật và minh bạch thông qua việc sử dụng các hàm băm mật mã. Mỗi khối trong chuỗi chứa dữ liệu được mã hóa, với cơ chế liên kết chắc chắn, đảm bảo tính toàn vẹn và không thể thay đổi sau khi đã được ghi.

Đồ án này tập trung vào việc khám phá và triển khai một hệ thống Blockchain đơn giản, không bao gồm tính năng peer-to-peer (p2p) giữa các nút. Mục tiêu chính bao gồm:

- **Tìm hiểu về Merkle Tree**: Tìm hiểu cấu trúc và ứng dụng của Merkle Tree, đặc biệt là vai trò của nó trong Blockchain.
- **Tìm hiểu về Cơ chế Đồng Thuận (Consensus Mechanism)**: Khám phá cách thức hoạt động của cơ chế đồng thuận trong Blockchain.
- **Cài đặt Blockchain Đơn Giản**: Cài đặt một hệ thống Blockchain đơn giản, kết hợp sử dụng Merkle Tree và cơ chế đồng thuận và phát triển giao diện dòng lệnh (Command Line Interface - CLI) để tương tác với Blockchain.

Thư mục đồ án có cấu trúc như sau:

```
|   report.pdf
|
\---code
    |   Blockchain_cli.py
    |   README.md
    |
    +---Blockchain_without_merkle_tree
    |       Blockchain.py
    |       Blockchain_server.py
    |
    \---Blockchain_with_merkle_tree
            Blockchain.py
            Blockchain_server.py
```

Trong đó:

- **report.pdf**: Tài liệu báo cáo chi tiết về đồ án.
- **code**: Thư mục chính chứa mã nguồn đồ án.
    - **Blockchain_cli.py**: File Python chứa giao diện dòng lệnh (CLI) để tương tác với hệ thống Blockchain.
    - **README.md**: File hướng dẫn sử dụng CLI.

- **Blockchain_without_merkle_tree**: Thư mục con chứa phiên bản Blockchain không sử dụng Merkle Tree.
    - **Blockchain.py**: File Python chứa cấu trúc cơ bản của Blockchain.
    - **Blockchain_server.py**: File Python để chạy server Blockchain.
- **Blockchain_with_merkle_tree**: Thư mục con chứa phiên bản Blockchain tích hợp Merkle Tree.
    - **Blockchain.py**: File Python chứa cấu trúc Blockchain tích hợp Merkle Tree.
    - **Blockchain_server.py**: File Python để chạy server Blockchain.

2 Tổng quan về lý thuyết

Chương này sẽ trình bày một cách tổng quan về Merkle Tree và cơ chế đồng thuận cũng như ứng dụng của chúng.

2.1 Merkle Tree

2.1.1 Giới thiệu Merkle Tree

Merkle Tree, được đặt tên theo Ralph Merkle [1], là một cấu trúc dữ liệu cốt lõi trong công nghệ Blockchain. Là cấu trúc cây nhị phân mà mỗi nút lá chứa dữ liệu được băm và mỗi nút không phải lá là hàm băm của các nút con của nó. Cấu trúc này cung cấp một phương pháp hiệu quả để xác minh tính toàn vẹn của dữ liệu trong một tập hợp lớn, với ứng dụng rộng rãi từ tiền điện tử đến hệ thống kiểm tra phiên bản.

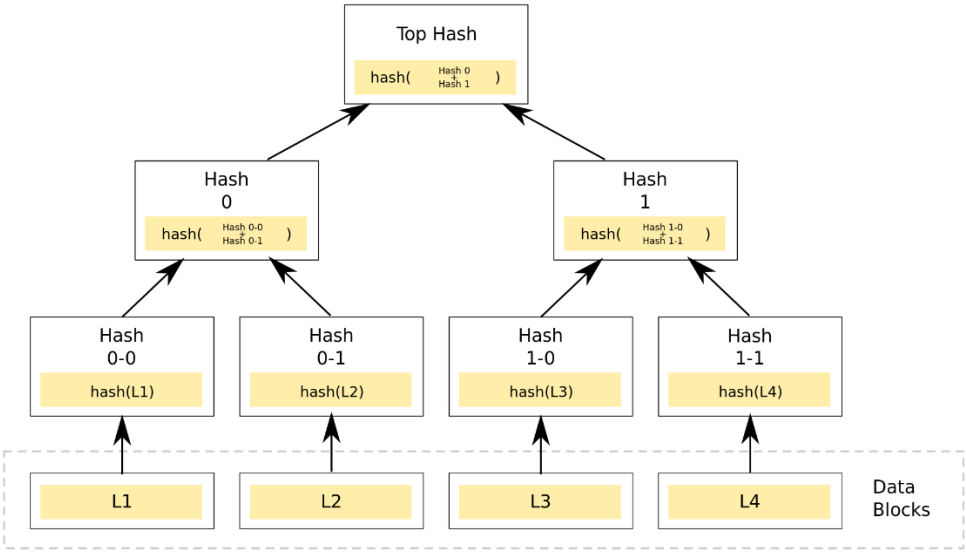
Merkle Tree đóng một vai trò quan trọng trong công nghệ Blockchain, được ứng dụng rộng rãi trong các nền tảng như Ethereum và Bitcoin. Cho phép Blockchain có thể xác minh dữ liệu một cách đáng tin cậy. Cách mà Merkle Tree tổ chức dữ liệu giúp giảm thiểu lượng tài nguyên cần thiết cho việc xử lý, chia sẻ và xác minh dữ liệu. Hơn nữa, Merkle Tree cũng góp phần tăng cường an toàn cho các giao dịch thông qua việc sử dụng các hàm băm và mã hóa.

2.1.2 Cấu trúc Merkle Tree

Merkle Tree được cấu trúc là một cây nhị phân, bắt đầu từ các nút lá, mỗi nút chứa dữ liệu đã băm. Nó được xây dựng từ dưới lên bằng cách liên tục băm các cặp nút, thực hiện lặp đi lặp lại các cặp nút băm cho đến khi chỉ còn lại một giá trị băm (giá trị băm cuối cùng này được gọi là root hash hoặc merkle root).

Mỗi nút lá (leaf node) là một hàm băm của dữ liệu và mỗi nút không lá (non leaf node – parent node) là một hàm băm của hai giá trị băm trước đó gộp lại. Merkle Tree có bản chất nhị

phân vì là nhị phân nên yêu cầu số nút chẵn. Nếu số lượng giao dịch là số lẻ, nút lá cuối cùng sẽ được nhân đôi một lần để tạo số nút chẵn.



Hình 2.1 Ví dụ Merkle Tree [1]

Để kiểm tra tính toàn vẹn của dữ liệu trong Merkle Tree ta có thể sử dụng Merkle Path. Merkle Path bao gồm những mã hash dùng để ghép cặp trong quá trình tính toán Merkle Root. Quá trình này giúp xác thực tính chính xác của dữ liệu thông qua một chuỗi các bước kiểm tra. Đầu tiên, cần phải xác định liệu dữ liệu cần xác thực có phải là một trong những nút lá của cây không. Nếu có, nút lá và nút anh em của nó sẽ được thêm vào Merkle Path. Quá trình này tiếp tục đệ quy với nút cha cho đến khi đạt được Merkle Root.

Merkle Path được coi như bằng chứng chứng minh tính xác thực và đáng tin cậy.Việc làm giả Merkle Path, yêu cầu tính toán lại các mã hash trong đường dẫn để trùng khớp với root hash, là một quá trình tính toán cực kỳ lớn và hiện nay được coi là không khả thi. Qua đó, Merkle Tree không chỉ cung cấp cách thức tổ chức dữ liệu một cách hiệu quả mà còn đảm bảo an toàn thông tin trong các giao dịch điện tử.

2.1.3 Ứng dụng của Merkle Tree

Với cấu trúc cây nhị phân đặc trưng của mình, được sử dụng rộng rãi trong nhiều lĩnh vực, chủ yếu là để xác thực dữ liệu, đảm bảo tính nhất quán, và đồng bộ hóa dữ liệu. Dưới đây là ba lĩnh vực chính mà Merkle Trees được ứng dụng:

2.1.3.1 Tiền Kỹ Thuật Số và Bitcoin

Trong thế giới của tiền điện tử, Merkle Trees đóng một vai trò quan trọng, đặc biệt là trong hệ thống của Bitcoin [2].

- **Cấu trúc Bitcoin Blockchain**: Trong Bitcoin, mỗi khối chứa một Merkle Root Hash, là kết quả của việc ghép nối và băm tất cả các giao dịch trong khối đó. Cấu trúc này không chỉ giúp xác minh tính toàn vẹn của từng giao dịch mà còn đảm bảo liên kết giữa các khối trong chuỗi.
- **Simple Payment Verification (SPV)**: SPV là một phương pháp cho phép các nút hạng nhẹ (lightweight nodes) xác minh các giao dịch mà không cần tải xuống toàn bộ Blockchain. Bằng cách sử dụng Merkle Trees, các nút này chỉ cần tải xuống tiêu đề của khối và sử dụng Merkle Path để kiểm tra tính hợp lệ của giao dịch cụ thể.

2.1.3.2 Hệ Thống Quản Lý Phiên Bản như Git và Mercurial

Trong hệ thống quản lý phiên bản như Git và Mercurial [2]:

- **So Sánh Dữ Liệu**: Merkle Trees được sử dụng để tạo ra mã hash cho từng file và thư mục. Khi có sự thay đổi giữa hai commit, hệ thống so sánh mã hash để xác định những thay đổi cụ thể, giúp quản lý phiên bản hiệu quả và chính xác.
- **Tối Ưu Hóa Hiệu Năng**: Khi làm việc với lượng lớn dữ liệu và thay đổi liên tục, việc sử dụng Merkle Trees giúp giảm bớt thời gian cần thiết để xác định các thay đổi, tối ưu hóa hiệu năng và tài nguyên hệ thống.

2.1.3.3 Cơ Sở Dữ Liệu Phân Tán

Ứng dụng của Merkle Trees trong các cơ sở dữ liệu phân tán như Apache Cassandra hoặc Amazon DynamoDB [2]:

- **Đảm Bảo Tính Nhất Quán**: Trong môi trường phân tán, việc duy trì tính nhất quán giữa các bản sao dữ liệu là cực kỳ quan trọng. Merkle Trees giúp nhanh chóng xác định những bất nhất quán và kích hoạt quá trình đồng bộ hóa dữ liệu.
- **Hiệu Quả Trong Đồng Bộ Dữ Liệu**: Thay vì đồng bộ toàn bộ dữ liệu, việc sử dụng Merkle Trees cho phép xác định và chỉ cập nhật những phần dữ liệu cụ thể có sự khác biệt, làm giảm đáng kể lưu lượng mạng và thời gian xử lý cần thiết.

2.2 Cơ chế Đồng thuận

2.2.1 Giới thiệu về Cơ chế đồng thuận

2.2.1.1 Định nghĩa

Cơ chế đồng thuận trong Blockchain bắt nguồn từ các hệ thống phân tán trong lĩnh vực khoa học máy tính. Vấn đề chính mà các cơ chế đồng thuận phải giải quyết là tạo ra sự chắc chắn và nhất quán trong dữ liệu trên toàn bộ mạng. Trước khi Blockchain được giới thiệu, các cơ chế đồng thuận có thể được xem như là các cơ chế đồng thuận phân tán cổ điển và cùng với các công nghệ liên quan, chúng đã đặt nền móng lý thuyết vững chắc cho các cơ chế đồng thuận trong Blockchain. Các lý thuyết này được tổng hợp từ các lĩnh vực như hệ thống phân tán từ khoa học máy tính, mật mã từ toán học và lý thuyết trò chơi từ kinh tế học. [3]

2.2.1.2 Mục đích và Tầm quan trọng của Cơ chế Đồng thuận

Cơ chế đồng thuận được tích hợp vào Blockchain như một cơ chế kháng lỗi (fault-tolerant) cho việc xác minh giao dịch trong quá trình thêm khối vào chuỗi. Vì Blockchain là một cấu trúc dữ liệu phân tán, nên cơ chế đồng thuận cần được sử dụng để duy trì sự nhất quán giữa các nút trong mạng. Đồng thời, cơ chế đồng thuận cũng phải đảm bảo tính bảo mật để xác minh và xác thực một cách chính xác cho các giao dịch để đạt được sự đồng thuận của những người tham gia trong mạng. Khi mạng mở rộng và số lượng nút tăng lên, việc đảm bảo sự đồng thuận dần trở thành một thách thức.

2.2.2 Giới thiệu về Proof of Work

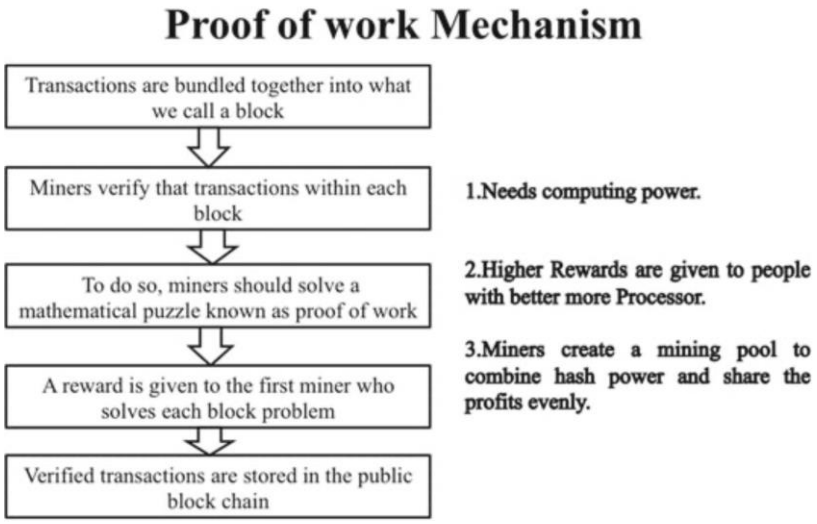
Proof of Work (PoW) lần đầu tiên được đề xuất trong bài báo "Bitcoin: A Peer-to-Peer Electronic Cash System". PoW được thiết kế để giải quyết vấn đề tin cậy giữa các nút trên cơ sở phân quyền, giúp Blockchain đạt được sự cân bằng giữa nhiều nút và giải quyết vấn đề truyền tải thông tin và chuyển giao giá trị trên các kênh không tin cậy. [4]

2.2.2.1 Các thức hoạt động của PoW

Cách Thức Hoạt Động: Trong quá trình tạo khối mới ứng với các giao dịch mới, PoW sẽ yêu cầu các nút (miners) thực hiện tính toán để tạo ra một giá trị băm (Block Hash) khớp với các yêu cầu nhất định (ví dụ, số lượng số không đứng đầu nhất định). Quá trình này yêu cầu nhiều thử nghiệm tính toán và thời gian tính toán phụ thuộc vào tốc độ băm của máy.

Quá Trình Đào (Mining): Trong PoW, quá trình đào bắt đầu bằng việc broadcast các giao dịch mới đến toàn bộ các miners trong mạng. Mỗi miner sẽ thu thập giao dịch và xây dựng một Merkle Tree mới. Tiếp theo, miner sử dụng tài nguyên máy tính của mình để tìm ra một giá trị

random (gọi là “nonce”) dựa trên một bài toán có độ khó tỷ lệ thuận với số lượng miners trong mạng hiện tại. Khi một giá trị nonce khả thi được tìm thấy, miner sẽ broadcast kết quả hash của khối, nonce và các thông tin khác đến toàn mạng để các miners khác xác minh. Nếu kết quả hợp lệ và khối trong kết quả thuộc chuỗi khối dài nhất, Blockchain sẽ tiến hành thêm khối này vào chuỗi tức là giao dịch được ghi nhận thành công vào mạng. Miner đã tính toán được nonce để hoàn thành việc tạo khối sẽ nhận được phần thưởng. [5]



Hình 2.2. Cơ chế PoW [5]

2.2.2.2 Ưu và Nhược Điểm

- **Ưu Điểm**:
    - **Độ phân quyền cao**: giải thuật đơn giản và dễ triển khai, các nút có thể tự do tham gia, và độ phân quyền cao.
    - **Bảo Mật Cao**: PoW yêu cầu giải quyết các bài toán mật mã khó, làm tăng độ an toàn và khó khăn cho các cuộc tấn công.
    - **Không Thể Thay Đổi**: Một khi khối được thêm vào chuỗi, việc thay đổi nó là rất khó, bảo vệ tính toàn vẹn của Blockchain.
- **Nhược Điểm**:
    - **Thời Gian Xác Nhận Lâu**: Việc xác nhận giao dịch trong PoW có thể mất thời gian dài, làm giảm hiệu quả giao dịch.
    - **Khả năng mở rộng kém**: Khả năng xử lý giao dịch hạn chế, không phù hợp với số lượng giao dịch lớn và mở rộng mạng lưới.

- **Lãng phí tài nguyên**: Quá trình mining PoW đòi hỏi một lượng lớn, tính toán khó khăn và nâng cấp phần cứng dẫn đến lãng phí tài nguyên và phần cứng.

3 Cài đặt

Chương này trình bày về cách cài đặt của Blockchain trong đồ án, bao gồm cấu trúc dữ liệu của Blockchain, cấu trúc dữ liệu của Merkle Tree, các hàm của Blockchain và cách sử dụng CLI. Tất cả các chương trình đều được cài đặt bằng Python.

3.1 Cài đặt Blockchain đơn giản không sử dụng Merkle Tree

Trước tiên, để hiểu về cách cài đặt Blockchain, nhóm thực hiện cài đặt một Blockchain đơn giản. Cài đặt của nhóm được đựa trên nguồn [6]. Blockchain có cấu trúc dữ liệu như sau.

Cấu trúc dữ liệu của một Block:

- `index`: Số thứ tự của khối trong chuỗi.
- `transactions`: Danh sách giao dịch. Ở đây, các giao dịch chỉ đơn tuần là các văn bản chứa thông tin dữ liệu và không có cơ chế xác thực giao dịch hợp lệ.
- `timestamp`: Dấu thời gian cho biết thời điểm khối được tạo.
- `previous_hash`: Giá trị hash của khối trước đó trong chuỗi. Điều này tạo ra một liên kết giữa các khối, giúp đảm bảo tính toàn vẹn của chuỗi.
- `nonce`: Một giá trị số được sử dụng trong quá trình mining. `Nonce` được sử dụng để tạo ra một giá trị hash phù hợp với một điều kiện nhất định (ví dụ: một số lượng zero định trước ở đầu giá trị hash).
- `hash`: Giá trị hash của khối. Nó chuyển đổi thông tin của khối thành một chuỗi (bao gồm `index`, `transactions`, `timestamp`, `previous_hash`, và `nonce`), và sau đó sử dụng thuật toán băm SHA-256 để tạo ra một giá trị hash duy nhất cho khối đó.

Cấu trúc dữ liệu của Blockchain:

- `unconfirmed_transactions`: Một danh sách chứa các giao dịch chưa được xác nhận. Các giao dịch này chưa được đưa vào bất kỳ khối nào trong chuỗi và đang đợi để được khai thác.
- `chain`: Một danh sách chứa toàn bộ các khối của chuỗi Blockchain. Mỗi phần tử trong danh sách này là một đối tượng `Block`.

Khi Blockchain được khởi tạo:

- Một giá trị `difficulty` được chọn. Giá trị này xác định độ khó cho quá trình mining, tức là số lượng zero đầu tiên mà giá trị hash của một khối phải có để được coi là hợp lệ trong quá trình mining. Giá trị này càng lớn thì độ khó để mining càng cao.
- Một khối genesis được khởi tạo. Nó là khối đầu tiên trong bất kỳ Blockchain nào.

Blockchain gồm các phương thức sau:

proof_of_work:

- Đây là phương thức thực hiện thuật toán "proof of work".
- Nó nhận vào một khối và cố gắng tìm một giá trị `nonce` sao cho giá trị hash của khối bắt đầu bằng một số lượng zero nhất định (được xác định bởi giá trị `difficulty`).
- Phương thức này tăng giá trị `nonce` một cách tuần tự cho đến khi tìm được hash phù hợp, sau đó trả về giá trị hash đó.

add_block:

- Thêm một khối mới vào chuỗi Blockchain sau khi đã xác minh.
- Kiểm tra xem hash của khối trước đó có trùng khớp với `previous_hash` của khối mới không.
- Kiểm tra xem giá trị hash được cung cấp có phải là một proof of work hợp lệ cho khối mới không.
- Nếu cả hai điều kiện trên đều đúng, khối mới sẽ được thêm vào chuỗi.

is_valid_proof:

- Kiểm tra xem một giá trị hash có phải là proof of work hợp lệ cho một khối cụ thể không.
- Điều kiện để là proof of work hợp lệ là hash bắt đầu bằng một số lượng zero nhất định và hash phải trùng với giá trị hash được tính toán từ khối.

mine:

- Phương thức khai thác một khối mới từ các giao dịch chưa được xác nhận.
- Tạo một khối mới, thực hiện proof of work để tìm ra hash phù hợp, sau đó thêm khối vào chuỗi.
- Làm trống danh sách các giao dịch chưa được xác nhận sau khi khối được thêm vào.

- Trả về chỉ số của khối mới.

is_chain_valid:

- Kiểm tra tính hợp lệ của toàn bộ chuỗi Blockchain.
- Kiểm tra từng cặp khối liên tiếp trong chuỗi để đảm bảo rằng hash của mỗi khối trùng khớp với giá trị hash được tính toán và `previous_hash` của mỗi khối trùng với hash của khối trước đó.
- Nếu bất kỳ điều kiện nào không được thỏa mãn, chuỗi sẽ được coi là không hợp lệ.

verify_transaction:

- Kiểm tra xem một giao dịch cụ thể có tồn tại trong chuỗi Blockchain hay không.
- Lần lượt kiểm tra từng khối trong chuỗi để tìm giao dịch.
- Trả về `True` nếu tìm thấy giao dịch, ngược lại trả về `False`.

Để tương tác với Blockchain, nhóm thực hiện cài đặt một ứng dụng Flask đơn giản, cung cấp các API. Dưới đây là mô tả chi tiết cho từng API:

GET /chain:

- API này trả về toàn bộ chuỗi Blockchain hiện tại.
- Dữ liệu trả về là một danh sách các khối, mỗi khối chứa thông tin như chỉ số (`index`), các giao dịch (`transactions`), dấu thời gian (`timestamp`), hash của khối trước đó (`previous_hash`), giá trị `nonce`, và hash của khối hiện tại.

POST /mine:

- API này kích hoạt quá trình khai thác một khối mới từ các giao dịch chưa được xác nhận.
- Nếu có giao dịch để khai thác, trả về thông báo khối đã được khai thác cùng với chỉ số của khối đó.
- Nếu không có giao dịch nào để khai thác, trả về thông báo tương ứng.

POST /add_transaction:

- API này cho phép thêm một giao dịch mới vào danh sách các giao dịch chưa được xác nhận.
- Dữ liệu giao dịch được lấy từ phần thân của yêu cầu (request body) dưới dạng JSON.
- Trả về thông báo xác nhận rằng giao dịch đã được thêm vào.

GET /validate:

- API này kiểm tra xem chuỗi Blockchain hiện tại có hợp lệ hay không.
- Trả về thông báo xác nhận tính hợp lệ của chuỗi.

POST /verify_transaction:

- API này được sử dụng để kiểm tra xem một giao dịch cụ thể có tồn tại và hợp lệ trong Blockchain hay không.
- Giao dịch cần kiểm tra được truyền qua phần thân của yêu cầu dưới dạng JSON.
- Trả về thông báo cho biết giao dịch có được xác minh là hợp lệ hay không.

3.2 Cài đặt Blockchain tích hợp Merkle Tree

Sau khi đã cài đặt được một Blockchain đơn giản, nhóm tiến hành tích hợp Merkle Tree vào Blockchain. Trước tiên nhóm thực hiện cài đặt Merkle Tree. Cài đặt của nhóm dựa trên cài đặt từ nguồn [7] và thêm một số phương thức để xác thực giao dịch trong Blockchain. Để cài đặt Merkle Tree, cần cài đặt 2 lớp mới.

Lớp Node:

- Đây là lớp đại diện cho mỗi nút trong Merkle Tree.
- Mỗi nút (`Node`) có các thuộc tính như `left` và `right` (đại diện cho nút con trái và phải), `value` (giá trị hash của nút), `content` (dữ liệu gốc hoặc nội dung được băm), `is_copied` (đánh dấu nếu nút là bản sao của nút khác), và `parent` (nút cha của nút hiện tại).
- Có phương `hash` để băm chuỗi đầu vào, và phương thức `copy` để tạo bản sao của nút.

Lớp MerkleTree: Tạo và quản lý cấu trúc cây Merkle từ danh sách các giá trị. Lớp Merkle Tree có các phương thức sau.

`__buildTree`:

- Phương thức này xây dựng cấu trúc cơ bản của cây Merkle từ danh sách các giá trị đầu vào.
- Đầu tiên, nó tạo ra các nút lá (`leaves`) bằng cách băm mỗi giá trị đầu vào và tạo một nút (`Node`) cho mỗi hash đó.
- Nếu số lượng nút lá là lẻ, phương thức sẽ nhân đôi nút lá cuối cùng để đảm bảo số lượng chẵn.
- Cuối cùng, nó gọi phương thức `__buildTreeRec` để tạo ra nút gốc (`root`) của cây.

`__buildTreeRec`:

- Đây là phương thức đệ quy được sử dụng trong quá trình xây dựng cây.
- Nếu danh sách nút có số lượng lẻ, phương thức cũng sẽ nhân đôi nút cuối cùng.
- Phương thức chia danh sách nút thành hai nửa và gọi đệ quy cho mỗi nửa để xây dựng các nhánh của cây.
- Nút cha được tạo ra bằng cách kết hợp giá trị hash của hai nút con.
- Quá trình này tiếp tục cho đến khi chỉ còn lại một nút (nút gốc).

`getRootHash`:

- Trả về giá trị hash của nút gốc. Điều này hữu ích cho việc xác minh toàn bộ cây hoặc đối chiếu với dữ liệu lưu trữ trong một khối của blockchain.

`verify_transaction`:

- Kiểm tra xem một giao dịch (hoặc dữ liệu) cụ thể có tồn tại trong cây hay không.
- Bắt đầu bằng việc tìm kiếm nút lá chứa giao dịch.
- Từ nút lá này, phương thức tái tạo lại đường dẫn lên nút gốc, so sánh hash tại mỗi bước để đảm bảo tính nhất quán của dữ liệu.
- Nếu bất kỳ hash nào không khớp trong quá trình này, phương thức sẽ trả về `False`, ngược lại trả về `True`.

`_get_leaves` và `_get_leaves_rec`:

- `_get_leaves` là phương thức để thu thập tất cả các nút lá của cây, nơi chứa dữ liệu gốc.
- `_get_leaves_rec` là phương thức đệ quy hỗ trợ cho `_get_leaves`, đi qua từng nút của cây và thu thập nút lá.

Vì tích hợp Merkle Tree vào Blockchain nên lớp Block có 2 thay đổi sau:

- Thêm `merkle_root` là chuỗi đại diện cho root hash của Merkle Tree, được tạo từ danh sách các giao dịch (`transactions`) của khối.
- Trong phiên bản Blockchain không tích hợp Merkle Tree, hash của khối được tính dựa trên chuỗi chứa `index`, `transactions`, `timestamp`, `previous_hash`, và `nonce`. Trong phiên bản mới, chuỗi được sử dụng để tính hash bao gồm `merkle_root` thay vì toàn bộ danh sách `transactions`.

Đồng thời, lớp Blockchain cũng có những thay đổi như sau:

Thêm Phương Thức `get_merkle_root`

- Phương thức này tạo một Merkle Tree từ danh sách các giao dịch và trả về hash của nút gốc (Merkle Root).

Thay Đổi trong Phương Thức `mine`

- Trong phương thức `mine`, Merkle Root của các giao dịch chưa được xác nhận được tính toán và gán cho `merkle_root` của khối mới trước khi thực hiện proof of work.

Thay Đổi trong Phương Thức `is_chain_valid`

- Ngoài việc kiểm tra tính liên tục của chuỗi và tính toàn vẹn của hash từng khối, phiên bản mới còn kiểm tra tính chính xác của Merkle Root bằng phương thức `validate_merkle_root`.

Thêm Phương Thức `validate_merkle_root`

- Phương thức này kiểm tra xem Merkle Root lưu trữ trong khối có phù hợp với Merkle Root được tính toán từ danh sách giao dịch trong khối hay không.

Thay Đổi trong Phương Thức `verify_transaction`

- Trong phiên bản mới, việc xác minh một giao dịch được thực hiện không chỉ bằng cách kiểm tra sự tồn tại của giao dịch trong danh sách giao dịch của khối, mà còn thông qua việc sử dụng Merkle Tree để xác định xem giao dịch có thuộc về khối đó hay không.

API của server Blockchain tích hợp Merkle Tree không có thay đổi ngoại trừ việc trả thêm thông tin về Merkle Root của khối khi xem thông tin hoặc đào.

3.3 CLI để tương tác với Blockchain

Để tương tác với Blockchain, trước tiên cần chạy server Flask của Blockchain bằng lệnh sau:

```
$ python blockchain_server.py -p 8000
```

Sau khi chạy server Flask, mở một terminal mới và sử dụng các dòng lệnh sau với các chức năng tương ứng

Xem thông tin Blockchain:

```
$ python blockchain_cli.py --server http://localhost --port 8000 --
view
```

Thêm giao dịch vào Blockchain:

```
$ python blockchain_cli.py --server http://localhost --port 8000 --add
'{"sender": "Alice", "receiver": "Bob", "amount": 50}'
```

Đào các giao dịch chưa được đào trong Blockchain:

```
$ python blockchain_cli.py --server http://localhost --port 8000 --
mine
```

Xác minh giao dịch:

```
$ python blockchain_cli.py --server http://localhost --port 8000 --
verify '{"sender": "Alice", "receiver": "Bob", "amount": 50}'
```

Xác thực Blockchain:

```
$ python blockchain_cli.py --server http://localhost --port 8000 --
validate
```

4 Kết luận

Việc triển khai một hệ thống Blockchain đơn giản được mô tả trong báo cáo này là bước đệm quan trọng để hiểu sâu hơn về công nghệ Blockchain. Đồ án đã cung cấp cái nhìn cận cảnh về cách thức hoạt động của Blockchain, từ cấu trúc dữ liệu cơ bản đến các cơ chế phức tạp như Merkle Tree và cơ chế đồng thuận.

Quá trình triển khai đã gặp không ít thách thức:

- Việc hiểu được cách hoạt động của Merkle Tree và Merkle Path và cách áp dụng chúng vào Blockchain. Nhóm đã gặp khá nhiều bug khi cài đặt hàm để xác minh giao dịch trong lớp Merkle Tree. Nhờ vào khả năng mô tả lỗi khá dễ hiểu của Python nên nhóm có thể giải quyết các bug phát sinh.
- Nhóm đã thử cài đặt Blockchain có khả năng hoạt động giữa nhiều node bao gồm việc broadcast các giao dịch cũng như đồng bộ Blockchain giữa các node nhưng do không có kinh nghiệm trong việc cài đặt p2p network, cũng như sự phức tạp của các cơ chế này và thời gian có hạn nên nhóm đã không thể thực hiện được.

Bài Lab này đã đem tới nhiều kiến thức mới. Một trong những hiểu biết quan trọng nhất là sự hiểu biết sâu sắc về cách thức mà các giao dịch được xác minh và đảm bảo trong một hệ thống phân tán. Việc triển khai Merkle Tree và các cơ chế đồng thuận cung cấp cơ hội để phát triển kỹ năng lập trình và giải quyết vấn đề.

Tài liệu tham khảo

[1] Merkle tree. (2023, November 6). In *Wikipedia*. https://en.wikipedia.org/wiki/Merkle_tree

[2] (2020, January 22). *Merkle Trees: What They Are and the Problems They Solve*. Www.Codementor.io. Retrieved December 23, 2020, from https://www.codementor.io/blog/merkle-trees-5h9arzd3n8

[3] Zhou, S.; Li, K.; Xiao, L.; Cai, J.; Liang, W.; Castiglione, A. (2023) - A Systematic Review of Consensus Mechanisms in Blockchain

[4] Ziad Hussein1, May A. Salama1 and Sahar A. El-Rahman1 (2023) - Evolution of blockchain consensus algorithms: a review on the latest milestones of blockchain consensus algorithms

[5] B. Sriman, S. Ganesh Kumar, and P. Shamili (2020) - Blockchain Technology: Consensus Protocol Proof of Work and Proof of Stake

[6] (2020, April 21). *Develop a blockchain application from scratch in Python*. Ruochi.AI. Retrieved December 23, 2023, from https://zhangruochi.com/Develop-a-blockchain-application-from-scratch-in-Python/2020/04/21/

[7] (2022, September 9). *Introduction to Merkle Tree*. GeeksforGeeks. Retrieved December 23, 2023, from https://www.geeksforgeeks.org/introduction-to-merkle-tree/