ĐẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH
TRƯỜNG ĐẠI HỌC BÁCH KHOA
KHOA KHOA HỌC - KỸ THUẬT MÁY TÍNH

CÔNG NGHỆ PHẦN MỀM (MỞ RỘNG)

Báo cáo

Ứng dụng Blockchain xây dựng từ điển bất động sản

GVHD: Quản Thành Thơ
SV: Nguyễn Khoa Gia Cát 1912749
Huỳnh Tấn Luân 1914054
Trịnh Nguyên Bảo Tuấn 1912371

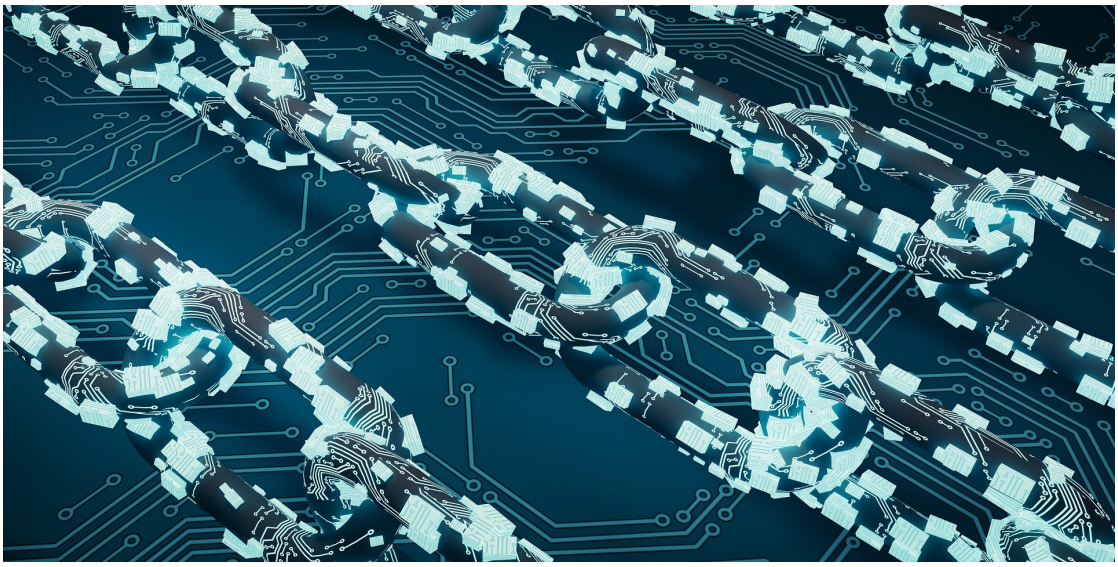
TP. HỒ CHÍ MINH, THÁNG 12/2021

Mục lục

1 Tìm hiểu về blockchain

1.1 Giới thiệu blockchain

Blockchain là một công nghệ dùng để lưu trữ thông tin đang ngày càng được phổ biến hiện nay do nhiều ưu điểm của nó. Blockchain nghĩa đen là chuỗi các khối, các khối ở đây lưu trữ dữ liệu và các khối này sẽ liên kết với nhau thành chuỗi. Trên thực tế, bởi vì sự liên kết đó nên nếu người ta thay đổi nội dung của một khối nào đó thì sẽ xung đột với những khối khác, do đó dữ liệu đã được chấp nhận thì sẽ coi như không chỉnh sửa được. Vì vậy, một trong những ưu điểm của công nghệ blockchain là chống việc gian lận dữ liệu, khiến blockchain phù hợp để ứng dụng vào việc ghi chép giao dịch, hồ sơ, công chứng...
Công nghệ blockchain kết hợp nhiều loại công nghệ khác nhau như mã hóa, mạng ngang hàng, các luật đồng thuận... Các loại mã hóa khiến cho dữ liệu được toàn vẹn, minh bạch. Mạng ngang hàng phân tán sẽ lưu trữ các bản sao dữ liệu, nếu có một máy tính trong mạng blockchain bị sập thì dữ liệu không bị mất đi.



Hình 1: Nguồn ảnh: pixabay.com

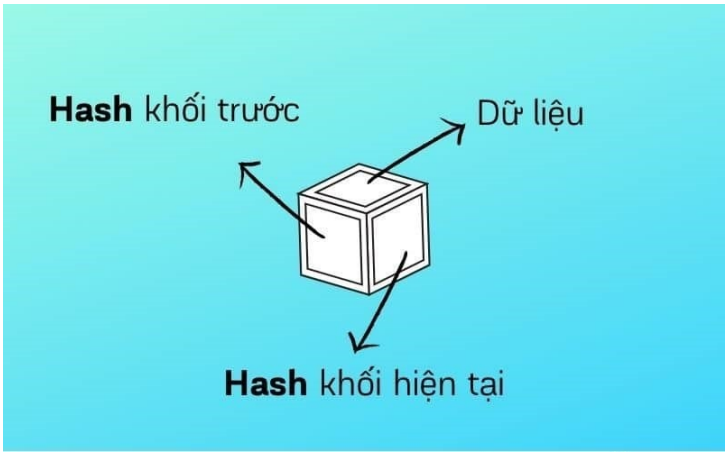
Hiện nay, ứng dụng được biết đến phổ biến của blockchain là hệ thống các đồng tiền điện tử (crypto-currency), cũng như để quản lý các giao dịch tài chính. Tuy nhiên, có nhiều ngành nghề khác cũng phù hợp để ứng dụng công nghệ này, chẳng hạn như:

- Dược phẩm: Công nghệ blockchain có thể ứng dụng để xây dựng hệ cơ sở dữ liệu của các dược phẩm đang lưu thông trên thị trường, chống việc giả mạo dược phẩm hay nguồn gốc xuất xứ của chúng.
- Bảo hiểm: Giúp bảo mật việc xác minh hay nhận tiền, đảm bảo sự minh bạch của hợp đồng.
- Bất động sản: Dữ liệu được công khai, minh bạch sẽ giúp ích cho người quan tâm đến bất động sản.

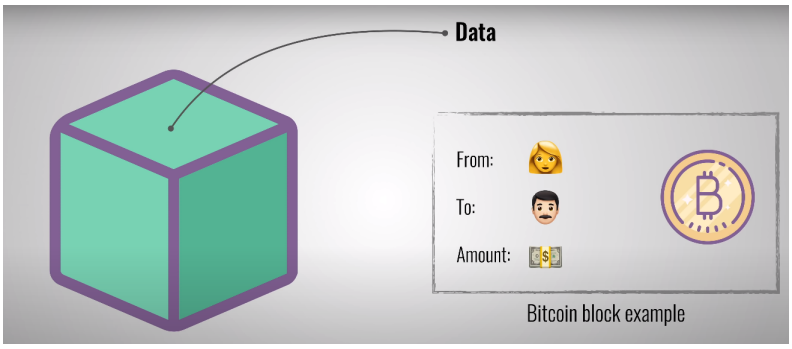
Từ đó, có thể thấy công nghệ blockchain có rất nhiều tiềm năng ứng dụng trong thực tiễn. Hiện nay và trong tương lai, yêu cầu bảo mật và toàn vẹn về dữ liệu là xu hướng tất yếu, cộng với sự phát triển của công nghệ, blockchain sẽ càng được ứng dụng rộng rãi hơn trong nhiều lĩnh vực.

1.2 Cấu trúc của của blockchain

- BlockChain là một chuỗi các Block. Mỗi Block sẽ được lưu trữ gồm 3 phần:
  - Dữ liệu
  - Hash của khối hiện tại
  - Hash khối trước

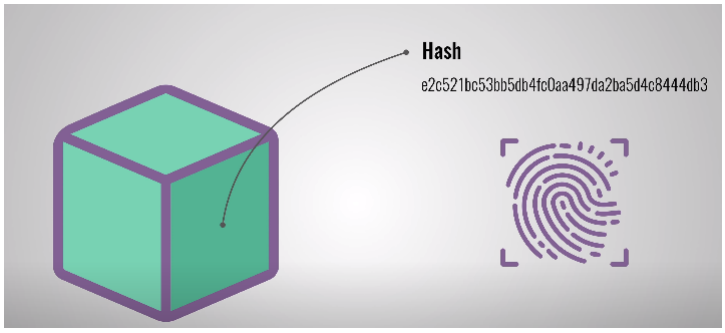


- Dữ liệu của một block phụ thuộc vào các loại blockchain

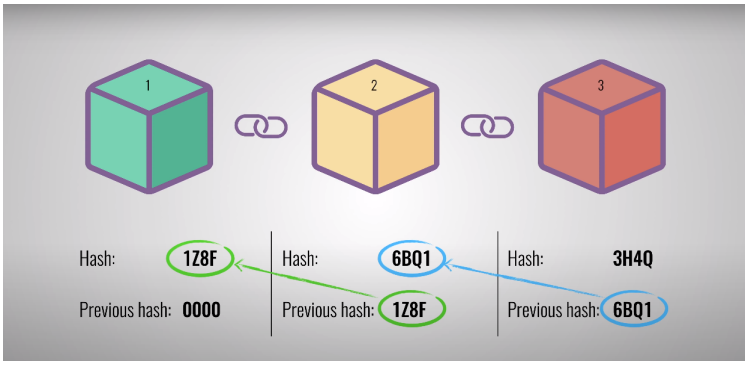


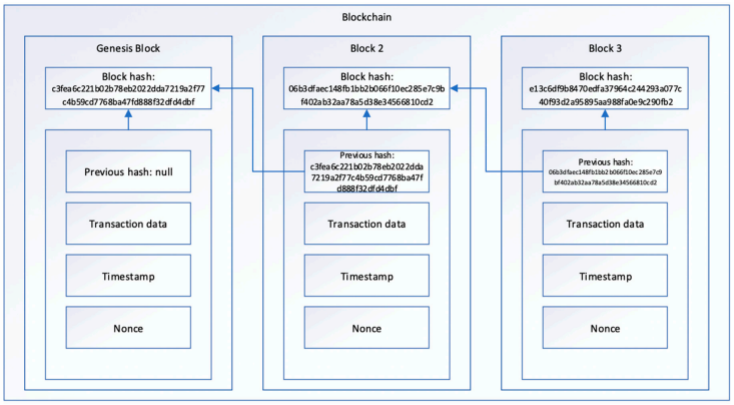
- Ví dụ Blockchain của Bitcoin sẽ chứa các dữ liệu về thông tin giao dịch như người gửi, người nhận, số tiền giao dịch,...

- Hash của một Block được tính dựa trên các thành phần có trong Block và xác định nó là duy nhất trong chuỗi. Bất cứ thay đổi dữ liệu nào trên Block điều làm Hash thay đổi



- Hash của Block trước được chứa trong Block hiện tại giúp hình thành một BlockChain. Hask của khối đầu luôn là 0 (Khối nguyên thủy)





1.3 Các đặc điểm của blockchain

Công nghệ blockchain có một số đặc điểm chính sau:

- Mạng blockchain được phân tán và đồng bộ, tùy theo loại blockchain (public, private hay permissioned) mà mạng blockchain sẽ có mức độ phân tán khác nhau. Càng phân tán thì các thực thể trong mạng càng ít quyền kiểm soát đối với mạng.
- Không phải mọi thao tác đều có thể được thực hiện đối với dữ liệu mà chỉ có một số thao tác được cho phép, những thao tác đó được lưu trữ ở blockchain dưới dạng smart contract, giúp kiểm soát việc quản lý dữ liệu.
- Block được tạo ra cần phải có sự đồng thuận của các đối tượng khác trong mạng để tránh tình trạng giả mạo.
- Dữ liệu khi đã được đồng thuận và ghi lại thì coi như không thể thay đổi được (nếu cố tình thay đổi sẽ để lại dấu vết).
- Block (hay giao dịch) có thể được tạo ra mà không để lộ danh tính thực sự. Điều này đảm bảo tính riêng tư của hệ thống.
- Dữ liệu được lưu trữ có tính minh bạch (mọi người đều có thể truy cập vào dữ liệu), tùy theo loại blockchain mà mức độ minh bạch sẽ khác nhau.

1.4 Các loại blockchain

1.4.1 Public Blockchain

Dữ liệu trên Public Blockchain được công khai để mỗi người có thể theo dõi được dữ liệu nếu họ muốn. Public blockchain không có giới hạn truy cập và bất kỳ ai cũng có thể gửi và xác thực các transaction. Public blockchain thường sử dụng cho mục đích kinh tế, nhất là khi xác thực các transaction và sử dụng thuật toán đồng thuận, chẳng hạn như PoW.
Trong public blockchain, dữ liệu thường minh bạch (transparent) và không thể giả mạo nhưng độ phức tạp cao, thời gian tính toán chậm và có chi phí tính toán cao.

1.4.2 Private Blockchain

Trong private blockchain, quyền truy xuất dữ liệu lưu trong block của các bên tham gia (peer) thường bị hạn chế. Chỉ có tổ chức điều hành và các yếu tố xác định được tham gia vào các nghiệp vụ trong hệ thống.
Hệ thống private blockchain phân phối dữ liệu mang tính tập trung hơn, bù lại thường có tốc độ truy xuất cao, chi phí tính toán ít.

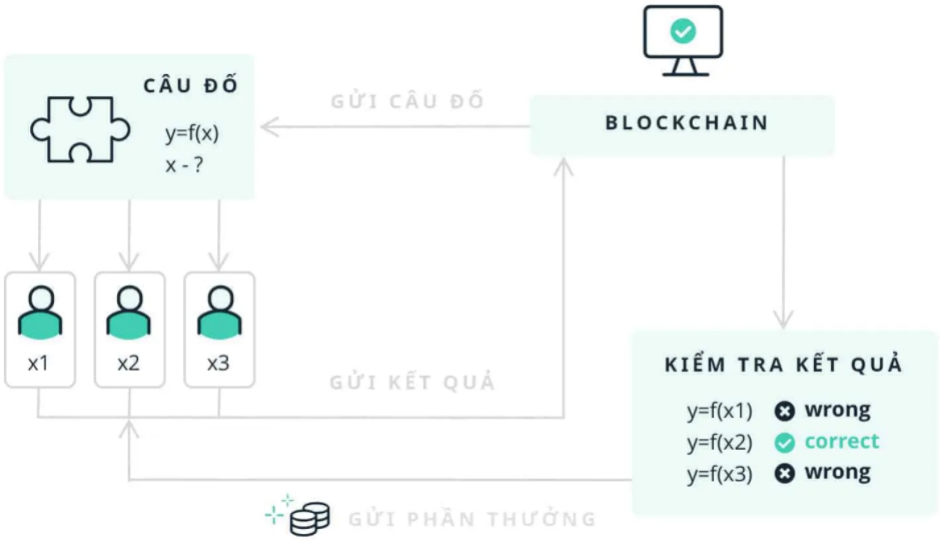
1.4.3 Permissioned/ Hybrid Blockchain

Hybrid blockchain được xem như một sự hợp nhất của hai loại public và private, mục đích để kết hợp các ưu điểm của cả hai loại đồng thời cố gắng hạn chế các khuyết điểm.
Trong hybrid blockchain, public blockchain có thể được sử dụng để công chúng có thể truy cập các lịch sử giao dịch, đồng thời sử dụng private blockchain để kiểm soát ai có thể tạo giao dịch. Do đó, cách tiếp cận này kết hợp các lợi ích về quyền riêng tư của loại private blockchain và sự bảo mật, minh bạch của public blockchain. Điều này giúp các công ty, tổ chức linh hoạt hơn. Họ có thể chọn những dữ liệu nào muốn công khai hoặc thông tin nào muốn giữ nội bộ.

1.5 Các cơ chế đồng thuận trong blockchain

1.5.1 Proof of Work

- Proof of Work (POW) là thuật toán đồng thuận đầu tiên được tạo ra trong mạng Blockchain. Được sử dụng để xác nhận giao dịch và sản xuất các block mới trong chuỗi.
- Khi một giao dịch được thực hiện trên Blockchain, nó sẽ được gom vào một Block cùng một số giao dịch khác. Các miner sẽ sử dụng hệ thống gồm nhiều máy tính mạnh để xác minh giao dịch.
- Một câu đố toán học phức tạp sẽ được hệ thống đưa ra. Nhiệm vụ của miner là sử dụng sức mạnh của hệ thống mining để tìm ra câu trả lời, sau khi tìm được sẽ thông báo cho các miner còn lại.
- Khi phần lớn thành viên xác nhận đó là câu trả lời đúng, Block mới sẽ được tạo ra, giao dịch được xác nhận.
- Khi hoàn thành, miner sẽ nhận được phần thưởng là phí giao dịch và phần thưởng khối. Tuy nhiên, đây là quá trình sử dụng rất nhiều tài nguyên, điện, thời gian.



- Nếu câu đố quá khó, sẽ mất rất nhiều thời gian để tìm ra câu trả lời, khiến Block mới không được tạo ra, hệ thống sẽ bị tắc nghẽn, giao dịch không thể thực hiện.
- Nhưng nếu câu đố quá dễ, hệ thống sẽ dễ bị tấn công, các giao dịch có khả năng bị làm giả
- PoW giải quyết vấn đề này bằng một thuật toán điều chỉnh độ khó phù hợp với tốc độ khai thác của các miner, sao cho Block mới sẽ sinh ra trong một khoảng thời gian cố định.

1.5.2 Proof of Stake

- Proof of Stake (POS) được tạo ra thay thế cho Proof of Work (POW). Được sử dụng để xác nhận giao dịch và thêm các block mới vào chuỗi.
- Proof-of-Stake đạt được sự đồng thuận bằng cách yêu cầu người dùng đóng góp một lượng token của họ để có cơ hội được chọn để xác thực các block giao dịch và được thưởng vì đã làm như vậy.

- Trong PoS, các block được "rèn" thay vì được khai thác. Yếu tố đầu tiên được xem xét trong quá trình lựa chọn này là cổ phần (stake) của người dùng.
- Mỗi người muốn tham gia vào quá trình phải sở hữu một cổ phần trong mạng. Staking liên quan đến việc khóa một số tiền nhất định vào mạng làm cổ phần của họ. Sử dụng nó làm tài sản thế chấp để chứng minh cho block.
- Càng nhiều người dùng đặt cược, cơ hội được lựa chọn của họ càng cao. Số lượng cổ phần (stake) quyết định cơ hội mà node được chọn làm người xác thực để rèn block kế tiếp. Cổ phần càng lớn, thì cơ hội càng lớn so với người đặt cược (staking) ít hơn.
- Trong PoS, khuyến khích tham gia xác thực các khối phần thưởng là một khoản thanh toán dưới dạng phí giao dịch. Trái ngược với tiền tệ mới được tạo ra trong các hệ thống PoW.
- Để tránh việc nghĩ đây là cơ hội cho những node giàu có trong mạng. Ngày càng nhiều phương thức độc nhất được thêm vào quá trình lựa chọn. Chìa khóa ở đây là bao gồm một mức độ cơ hội cho quá trình lựa chọn để tránh trường hợp người dùng giàu nhất luôn được chọn để xác thực các giao dịch, luôn gặt hái những phần thưởng và ngày càng giàu hơn.
- Hai phương thức được sử dụng phổ biến nhất là Lựa chọn block ngẫu nhiên và Lựa chọn tuổi Coin:
  - Lựa chọn block ngẫu nhiên: Thuật toán Proof-of-Stake sẽ lựa chọn validator kiểm định block tiếp theo một cách ngẫu nhiên. Bằng cách sử dụng công thức tìm kiếm Hashrate thấp nhất, kết hợp với khoản đặt cược cao nhất (stake).
  - Lựa chọn tuổi Coin: Các node được chọn dựa trên thời gian mà các token của họ đã được lưu giữ làm cổ phần. Tuổi đồng coin được tính bằng cách nhân số ngày các coin được giữ làm cổ phần với số lượng các coin đó.

1.6 Một số nền tảng dựa trên Blockchain hiện nay

1.6.1 Ethereum

- Sau sự thành công của Bitcoin, một loại tiền điện tử khác cũng gây tiếng vang trong thị trường số hiện nay là Ethereum. Ethereum cho phép mọi người xây dựng và sử dụng các ứng dụng phi tập trung dựa trên công nghệ Blockchain. Nó là dự án mã nguồn mở, có thể chuyển đổi và linh hoạt hơn Bitcoin.
- Ethereum có các đặc điểm sau:
  - Là mạng mở
  - Sử dụng mô hình đồng thuận bằng chứng công việc
  - Có lượng người theo dõi trên Github cao
  - Hỗ trợ các ngôn ngữ như C++, Go và Python

1.6.2 Hyperledger Fabric

- Đây là một trong những nền tảng Blockchain phát triển gần đây nhất và được biết đến như là cuốn siêu sổ cái vào năm 2016, do Linux Foundation tạo ra. Mục tiêu của nó là đẩy nhanh sử dụng công nghệ Blockchain trong các ngành công nghiệp khác nhau như tài chính ngân hàng, IoT, chuỗi cung ứng...
- Hyperledger Fabric có các đặc điểm sau:
  - Có thể sử dụng cho mục đích mở hoặc đóng
  - Tích cực cập nhật trên Github
  - Sử dụng mô hình đồng thuận Pluggable
  - Hỗ trợ ngôn ngữ Python

1.6.3 IBM Blockchain

- Là công ty tiên phong liên doanh Blockchain vì vậy mà nó có thể tạo một nền tảng điều hành kinh doanh minh bạch. IBM tự hào về một cơ chế đồng thuận hiệu quả hơn, tạo sự chú ý cho nhiều người.
- IBM Blockchain có các đặc điểm sau:
  - Nó thuộc về mạng Blockchain đóng, do đó có sự bảo mật cao
  - Phổ biến ở mức trung bình nhưng tích cực cập nhật trên Github
  - Phiên bản miễn phí hạn chế, có thể nâng cấp lên gói Doanh nghiệp
  - Hỗ trợ các ngôn ngữ như Go và Javascript

1.6.4 Multichain

- Multichain là nền tảng Blockchain mã nguồn mở, được dùng trong mạng Blockchain đóng. Nó được sử dụng trong các doanh nghiệp khác nhau. Bằng cách cung cấp quyền riêng tư và sự kiểm soát mạng ngang hàng, nó như là sự cải thiện của Bitcoin cho các giao dịch tài chính riêng tư.
- Multichain có các đặc điểm sau:
  - Là mạng mang tính chất đóng
  - Phổ biến ở mức trung bình nhưng tích cực cập nhật trên Github
  - Miễn phí và mã nguồn mở
  - Hỗ trợ các ngôn ngữ như Python, C#, JavaScript , PHP, Ruby

1.6.5 Hydrachain

- Hydrachain là một sáng kiến hợp tác giữa Ethereum và công nghệ brainbot. Nó được dùng để tạo một sổ cái riêng tư hữu ích cho doanh nghiệp mặc dù nó không được phổ biến.
- Hydrachain có các đặc điểm sau:
  - Sử dụng giao thức Ethereum
  - Là mạng đóng
  - Ít phổ biến hơn nhưng tích cực cập nhật trên Github
  - Hỗ trợ ngôn ngữ Python.

1.6.6 Openchain

- Openhain là một nền tảng mã nguồn mở, cực kỳ hữu ích cho các công ty đang tìm kiếm giải pháp quản lý tài sản kỹ thuật số. Nó còn cho phép tùy biến quyền theo các mức độ khác nhau.
- OpenChain có các đặc điểm sau:
  - Dùng cho mạng đóng
  - Phổ biến ở mức trung bình nhưng tích cực cập nhật trên Github
  - Hỗ trợ ngôn ngữ JavaScript
  - Sử dụng mô hình đồng thuận phân vùng

1.6.7 BigchainDB

- BigchainDB là một nền tảng mã nguồn mở. Là một cơ sở dữ liệu nhưng mang các tính chất của Blockchain.
- BigchainDB có các đặc điểm sau:
  - Tùy biến tài sản
  - Không tích hợp sẵn tiền ảo
  - Có thể dùng cho cả mạng đóng và mở
  - Hỗ trợ các ngôn ngữ như Java, Python, Javascript và các ngôn ngữ khác do cộng đồng hỗ trợ

2 BigChainDB

2.1 BigChainDB là gì

BigchainDB là 1 trong nhiều nền tảng Blockchain (Blockchain platform), được thiết kế với ý tưởng kế thừa và kết hợp blockchain với cơ sở dữ liệu phân tán (cụ thể là MongoDB). Có thể xem BigchainDB như là 1 cơ sở dữ liệu blockchain giúp xây dựng một ứng dụng phi tập trung mà không cần tạo một blockchain riêng biệt từ đầu.

BigchainDB giúp bổ sung các đặc điểm của blockchain, bao gồm tính bất biến, kiểm soát phi tập trung và chuyển giao tài sản kỹ thuật số. Do đó BigchainDB có để được sử dụng để triển khai các ứng dụng quy mô lớn vào nhiều ngành khác nhau, từ lĩnh vực sở hữu trí tuệ cho đến quản lý chuỗi cung ứng và Internet-of-Things.

2.2 Ưu điểm của BigchainDB

Nhờ vào sự kế thừa và kết hợp giữa Blockchain và MongoDB, BigChain có các thế mạnh sau:

- Kiểm soát phi tập trung. BigchainDB không tồn tại điểm kiểm soát duy nhất (single point of control). Do sự kiểm soát phi tập trung nhờ vào liên kết của các nút ngang hàng, nó tạo thành một mạng P2P.
- Cho phép truy vấn dữ liệu. BigChainDB Được hỗ trợ bởi MongoDB nên chúng ta có thể viết và chạy bất kỳ truy vấn MongoDB nào, tìm kiếm các phần tử liên quan đến các giao dịch, siêu dữ liệu, nội dung trong các block.
- Tính bất biến: Dữ liệu không thể bị thay đổi.
- Khả năng thích nghi của Byzantine (BFT). Nếu không may mắn, một số nút trong mạng đang gặp lỗi, phần còn lại của mạng vẫn đi đến quy trình đồng thuận trên khối tiếp theo.
- Cung cấp độ trễ thấp. Mất khoảng một giây để đi đến quá trình đồng thuận trên một block mới, làm cho việc hoàn tất giao dịch diễn ra cực kỳ nhanh chóng.

- Linh hoạt. BigchainDB cho phép phát triển mạng riêng của người dùng với các thông tin tùy chỉnh, đảm bảo sự linh hoạt trong cấp quyền (permissions), giao dịch và tính minh bạch.
- Mã nguồn mở
- Triển khai các mạng Private hoặc Public. BigchainDB cho phép chúng ta triển khai các hệ thống mạng Private hoặc Public, tùy cơ ứng biến cho các trường hợp sử dụng cụ thể.

2.3 Một vài Ứng dụng của BigchainDB

Nhiều trường hợp sử dụng BigchainDB giống như mô hình blockchain truyền thống; khi đó bỏ qua các lợi thế của BigchainDB như thông lượng cao hơn, nhiều dung lượng hơn, độ trễ thấp hơn, truy vấn tốt hơn hoặc cấp phép phong phú hơn.

Bên cạnh đó, cũng có một số trường hợp dùng BigchainDB như 1 cơ sở dữ liệu phân tán truyền thống, ngoại trừ việc tập trung vào nơi các đặc điểm của blockchain có thể được tận dụng. Ví dụ: cải thiện độ tin cậy của DataBase bằng cách không có một điểm lỗi nào hoặc lưu trữ tài liệu mật một cách an toàn.

Cụ thể các trường hợp sử dụng BigchainDB:

- Các hợp đồng ràng buộc về mặt pháp lý có thể được lưu trữ trực tiếp trên BigchainDB bên cạnh giao dịch, ở định dạng mà con người lẫn máy tính có thể đọc được.
- Tạo và di chuyển theo thời gian thực của các dữ liệu có khối lượng lớn. Chỉ chủ sở hữu của nội dung mới có thể di chuyển dữ liệu. Khả năng này giúp giảm chi phí, giảm thiểu độ trễ giao dịch và hỗ trợ các ứng dụng khác.
- Theo dõi các tài sản vật chất có khối lượng lớn dựa theo toàn bộ chuỗi cung ứng. BigchainDB có thể giúp giảm gian lận, tiết kiệm chi phí lớn.
- Theo dõi các tài sản sở hữu trí tuệ. BigchainDB có thể giảm chi phí, thời gian cấp phép trong các kênh, hệ thống kết nối các tác giả với khán giả và cung cấp nguồn gốc rõ ràng cho các tài sản kỹ thuật số. Ví dụ: Giả sử cần một dịch vụ âm nhạc có hàng triệu bài hát - BigchainDB có thể lưu trữ thông tin này trong tích tắc, cùng với thông tin bản quyền về từng bài hát và thông tin về giấy phép sử dụng của người đăng ký.
- Giảm thời gian đóng dấu, chứng nhận, công chứng. BigchainDB làm giảm xung đột pháp lý bằng cách cung cấp bằng chứng hợp pháp, chặt chẽ về các hành vi trên không gian mạng.
- Cải thiện độ tin cậy của cơ sở dữ liệu truyền thống bằng cách tạo ra khả năng chống lại các điểm lỗi đơn lẻ (single points of failure). Việc tăng cường bảo mật này sẽ giúp đề phòng các nguy cơ mà một lần hack dẫn đến mất mát dữ liệu lớn như vụ việc Sony 2014.

3 Cài đặt BigChainDB

3.1 Tải BigChainDB Python driver

- BigChainDB Python driver hoạt động trên các hệ điều hành Linux và macOS
- Để sử dụng driver này, đầu tiên cần tải các dependencies sau
  - **Python 3.5+**: Tải về bằng lệnh

```
sudo apt-get install python3.7
```

    hoặc kiểm tra phiên bản Python đang sử dụng bằng lệnh

```
$ python --version
OR
$ python3 --version
```

  - **Pip3**: Tải về bằng lệnh

```
sudo apt-get install python3-pip
```

    hoặc kiểm tra phiên bản đang sử dụng bằng lệnh

```
$ pip --version
OR
$ pip3 --version
```

  - **Setuptools**: Khi có pip hoặc pip3, có thể nâng cấp setuptools bằng lệnh

```
$ pip install --upgrade setuptools
OR
$ pip3 install --upgrade setuptools
```

  - **Cryptography và Cryptoconditions**: BigChainDB (Server và Client) cũng phụ thuộc vào Cryptography và Cryptoconditions
    - Cryptography phụ thuộc vào libssl, libcrypto. Hai thư viện này phụ thuộc vào (Python development library and header files)
    - Cryptoconditions phụ thuộc vào PyNaCl (Networking and Cryptography library) và thư viện này phụ thuộc vào ffi.h
    - Có thể tải bằng lệnh sau:

```
$ sudo apt-get update

$ sudo apt-get install python3-dev libssl-dev libffi-dev
```

- Sau đó, tải BigChainDB Python driver để sử dụng

```
$ pip install bigchaindb_driver
OR
$ pip3 install bigchaindb_driver
```

3.2 Cài đặt môi trường để chạy BigchainDB Node trên máy

3.2.1 Cấu hình tối thiểu

Các BigchainDB Node có thể chạy trên máy ảo, máy thật hoặc trên cloud server (Azure, AWS, ...). Trên mỗi máy, cần thiết lập các nội dung sau:

- **Địa chỉ IP:** Mỗi máy chạy node cần phải có địa chỉ IP công cộng (public IP)
- **Hệ điều hành:** Ubuntu 18.04 hoặc Ubuntu Server 18.04
- **Thiết lập an ninh mạng** (Nếu muốn cho phép các Users kết nối thông qua giao thức HTTPs):
  - TCP: port 22 (SSH)
  - TCP: port 80 (HTTP)
  - TCP: port 443 (HTTPS)
  - Các giao thức còn lại: port 26656 (Tendermint P2P)

  Nếu không muốn Users kết nối thông qua HTTP, không sử dụng port 443, thay thế port 80 bằng port 9984 (port BigchainDB HTTP mặc định).
- **Update hệ điều hành:**

```
sudo apt update
sudo apt full-upgrade
```

- Thiết lập DNS và cấu hình hệ thống để tăng tính bảo mật của các Node (nếu cần thiết)

3.2.2 Thiết lập NGINX (nếu có sử dụng HTTPs)

Cài đặt NGINX:

```
sudo apt update
sudo apt install nginx
```

Cấu hình NGINX (lấy SSL certificate cho Node's subdomain)

- Copy SSL private key vào */etc/nginx/ssl/cert.key*
- Tạo "PEM file" (text file) bằng cách nối SSL certificate với các certificate trung gian.
- Copy PEM file vào */etc/nginx/ssl/cert.pem*
- Trong repo *bigchaindb/bigchaindb* trên GitHub, tìm file *nginx/nginx.conf* và ghi đè nội dung vào file */etc/nginx/nginx.conf* trong máy local
- Sửa file */etc/nginx/nginx.conf* bằng cách thay thế string *example.testnet2.com* bởi subdomain
- Reload NGINX:

```
sudo service nginx reload
```

3.3 Cài đặt một BigchainDB node

Mỗi một node trong BigchainDB network, cần phải cài đặt và chạy các phần mềm: BigchainDB server, MongoDB và Tendermint.

3.4 Cài đặt BigchainDB Server

BigchainDB server yêu cầu python 3.6 trở lên. Để cài đặt BigchainDB cần sử dụng các lệnh:

```
sudo apt install -y python3-pip libssl-dev
sudo pip3 install bigchaindb==2.0.0
```

Ở trên cài đặt BigchainDB phiên bản 2.0.0. Các phiên bản khác tham khảo ở https://pypi.org/project/BigchainDB/#history.
Sau khi cài đặt, ta cần cấu hình BigchainDB bằng cách chạy lệnh:

```
bigchaindb configure
```

Sau đó trả lời một số câu hỏi để tiến hành cấu hình BigchainDB server. Câu hỏi đầu tiên là `API Server bind? (default ‘localhost:9984‘)`, tùy theo mục đích mà trả lời như sau:

- Nếu sử dụng NGINX (tức là dùng HTTPS) thì đồng ý với giá trị mặc định.
  Trong trường hợp này, cần chỉnh file thiết lập BigchainDB (`$HOME/.bigchaindb` và thay đổi trường `"wsserver"`:

  ```
  "advertised_scheme": "wss",
  "advertised_host": "sub.example.com",
  "advertised_port": 443
  ```

  Trong đó `sub.example.com` là subdomain của node đang cài đặt.
- Nếu không sử dụng NGINX, cần nhập giá trị `0.0.0.0:9984`

3.4.1 Cài đặt MongoDB

MongoDB là cơ sở dữ liệu mà BigchainDB dùng để quản lý dữ liệu. Các phiên bản hiện tại của BigchainDB yêu cầu MongoDB 3.4 trở lên. Để cài đặt MongoDB cần sử dụng lệnh:

```
sudo apt install mongodb
```

Package `mongodb` cài đặt như trên không phải là bản chính thức của MongoDB. Để cài đặt bản chính thức, cần tham khảo trang https://docs.mongodb.com/manual/installation/

3.5 Cài đặt Tendermint

Tendermint là phần mềm giúp nhân bản một ứng dụng trên nhiều thiết bị khác nhau một cách an toàn và ổn định, phù hợp sử dụng cho các công nghệ phi tập trung. Để xây dựng BigchainDB network thì mỗi BigchainDB node cần phải cài đặt Tendermint.
BigchainDB Server hầu như chỉ hoạt động tốt với Tendermint 0.31.5 (không phải phiên bản cao hơn). Để cài đặt cần thực hiện các lệnh:

```
sudo apt install -y unzip
wget https://github.com/tendermint/tendermint/releases/download/v0.31.5
    /tendermint_v0.31.5_linux_amd64.zip
unzip tendermint_v0.31.5_linux_amd64.zip
rm tendermint_v0.31.5_linux_amd64.zip
sudo mv tendermint /usr/local/bin
```

Do Tendermint liên quan đến nhiều node trong network nên chỉ có thể cấu hình Tendermint khi có được thông tin về những nodes khác trong network.

4 Hướng giải quyết cho bài toán xây dựng từ điển bất động sản bằng blockchain

4.1 Xây dựng transaction và giải pháp thay đổi dữ liệu

4.1.1 Xây dựng dữ liệu thành một transaction và lưu trữ

Dữ liệu của bài toán có thể tổ chức thành nhiều object (mỗi object là thông tin của một property). Để thêm một object mới vào hệ thống, cần thực hiện một transaction là create asset, với asset là object cần thêm.
Sử dụng ngôn ngữ Python3, sau khi đã thiết lập môi trường, cần import các thư viện cần thiết; sau đó kết nối với server:

```
from bigchaindb_driver import BigchainDB
from bigchaindb_driver.common.utils import serialize
from bigchaindb_driver.crypto import generate_keypair
import json

bdb_root_url = 'https://test.ipdb.io'

bdb = BigchainDB(bdb_root_url)
```

Khởi tạo dữ liệu cần thêm vào hệ thống blockchain. Sau đó generate public key và private key cho user.

```
test_data = {
    'data': {
        'Area' : 50,
        'AreaHome' : 200,
        'Bedrooms' : 6,
        'Bathrooms' : 6,
        'id' : 'H1234',
        'Owner' : 111000222,
        'Price' : 10000000000
    }
}

user = generate_keypair()
```

Tạo transaciton với data và user vừa khai báo

```
# Prepare new transaction
prepare_creation = bdb.transactions.prepare(
    operation='CREATE',
```

```
    signers=user.public_key,
    asset=test_data,
)

print('PREPARE:')
print(json.dumps(prepare_creation, indent=4))
print()
```

Kí transaction bằng private key của user để chuẩn bị đưa lên BigChainDB

```
# Sign the transaction
fulfilled_creation  = bdb.transactions. fulfill (
    prepare_creation, user.private_key
)

print('FULFILL:')
print(json.dumps(fulfilled_creation, indent=4))
print()
```

Đưa transaction lên server BigChainDB

```
# Send the transaction to BigchainDB
sent_creation = bdb.transactions.send_commit(fulfilled_creation)
print('sent_creation == fulfilled_creation: ', sent_creation == fulfilled_creation)
print()
```

4.1.2 Giải pháp thay đổi dữ liệu

Dữ liệu đã được lưu trữ trên blockchain thì không thể thay đổi được. Tuy nhiên trong phạm vi bài toán, thông tin về property có thể thay đổi theo thời gian:

- Thay đổi chủ sở hữu của property.
- Một property sau khi quy hoạch có thể tách thành nhiều property hoặc gộp lại với những property khác thành property mới.
- Thay đổi một số thông tin của property.

Mặt khác, những thay đổi này cần phải được ghi lại để đảm bảo tính minh bạch. Do đó nhóm đề xuất ghi lại những thay đổi này bằng cách thực hiện một transaction là transfer asset đã tạo ra (thông tin ban đầu của property), những nội dung thay đổi sẽ lưu trong thuộc tính metadata. Để chuẩn bị cho transaction loại này, người ta cần có `id` của transaction đã tạo ra asset (object). Sau đó thực hiện các lệnh sau để tạo transaction:

```
transfer_asset = {
    'id' : asset_id
}

data_changed = {
    'data' : {
        'Owner' : 111222333
    }
}

output_index = 0

output = fulfilled_creation['outputs'][output_index]
```

```
transfer_input = {
    ' fulfillment ': output['condition' ][' details ' ],
    ' fulfills ': {
    'output_index': output_index,
    'transaction_id':  fulfilled_creation ['id' ],
    },
    'owners_before': output['public_keys'],
}

prepare_creation = bdb.transactions.prepare(
    operation='TRANSFER',
    asset = transfer_asset,
    inputs = transfer_input,
    metadata= data_changed,
    recipients =user.public_key
)
```

Việc ký và gửi transaction lên server thực hiện giống như transaction để tạo ra asset.
Sau đó, khi cần lấy dữ liệu của một property, chỉ cần thực hiện các bước sau:

- Tìm trong các asset đã tạo trong hệ thống để lấy `id` của transaction tạo property cần lấy dữ liệu.
- Tìm tất cả các transaction trong hệ thống liên quan tới `id` trên.

Từ các transaction đã tìm ra có thể thấy được lịch sử thay đổi của thông tin property, cũng như suy ra được thông tin mới nhất của property.

4.2 Xây dựng BigChainDB Network

- Các node tham gia vào mạng gọi là Members, chúng sẽ chia sẻ thông tin với các node còn lại. Từ đó, tạo thành một mạng blockchain
- Có một Member đặc biệt giúp các Member khác kết nối với nhau gọi là Coordinator

4.2.1 Member

- Đầu tiên, Member cần chia sẻ định danh của nó với tất cả các member khác trong mạng
- Mỗi node BigChainDB sẽ được định danh bởi
  - Hostname : Đây chính là DNS subdomain của node (Vd: bnode.example.com) hoặc là địa chỉ IP (46.145.17.32)
  - Tendermint pub_key.value
  - Tendermint node_id
- Tendermint pub.key_value sẽ được lưu trữ ở file $HOME/.tendermint/config/priv_validator.json. Có dạng như sau:

```
{
  "address": ...,
  "pub_key": {
    "type": "tendermint/PubKeyEd25519",
    "value": "P+aweH73Hii8RyCmNWbwPsa9o4inq3I+0fSfprVkZa0="
  },
  "last_height": ...,
  "last_round": ...,
  "last_step": ...,
  "priv_key": {
    "type": ...,
    "value": ...
  }
}
```

- Để lấy Tendermint node_id, sử dụng lệnh sau:

```
tendermint show_node_id
```

4.2.2 Coordinator

- Sau đó, Coordinator sẽ nhận được tất cả dữ liệu với nhau và nối chúng thành một file $HOME/.tendermint/config/genesis.json

```
{
  "genesis_time":"0001-01-01T00:00:00Z",
  "chain_id":"test-chain-la6HSr",
  ...
  "validators":[
    {
      "pub_key":{
        "type":"tendermint/PubKeyEd25519",
        "value":"<Member 1 public key>"
      },
      "power":10,
      "name":"<Member 1 name>"
    },
    ...
    {
      "pub_key":{
        "type":"tendermint/PubKeyEd25519",
        "value":"<Member N public key>"
      },
      "power":10,
      "name":"<Member N name>"
    }
  ],
  "app_hash":""
}
```

- File genesis.json được tạo sẽ chứa dữ liệu diễn tả mạng
- Sau đó, Coordinator phải chia sẻ file genesis.json này với các Member trong mạng

4.2.3 Kết nối giữa các Member

- Sau khi coordinator chia sẻ file genesis.json thì tất cả các Member sẽ nhận được
- Khi đó, các Member cần phải copy file genesis.json vào thư mục local $HOME/.tendermint/config
- Từ đó, các Member trong mạng sẽ chia sẻ cùng chain_id, genesis_time (dùng để nhận dạng mạng BlockChain) và cùng một danh sách các validators
- Sau đó, các Member cần chỉnh sửa file $HOME/.tendermint/config/config.toml theo form như sau:

```
moniker = "Name of our node"
create_empty_blocks = false
log_level = "main:info,state:info,*:error"

persistent_peers = "<Member 1 node id>@<Member 1 hostname>:26656,\
<Member 2 node id>@<Member 2 hostname>:26656,\
<Member N node id>@<Member N hostname>:26656,"

send_rate = 102400000
recv_rate = 102400000

recheck = false
```

4.3 Hiện thực các hàm

Trong 2 tuần qua, nhóm đã viết các hàm để tương tác với server blockchain dựa trên data được cung cấp (`rv_listing_dataset.json`. Trong đó:

- Hàm add() thêm 1 asset là object trong file json.
- Hàm get_asset () lấy dữ liệu của một object dựa vào id.
- Hàm update() cập nhật một object đã thêm.

Để thuận tiện cho việc hiện thực thì nhóm đã chuyển đổi file `.jsonl` thành sang định dạng `.json`.

4.3.1 add(dict_values)

Hàm này thêm một asset mới vào hệ thống.
Input: dict_values là 1 object dạng json miêu tả asset cần thêm vào.
Output: return id của transaction tương ứng.
Hiện thực:

```
## dict_values:    a line in file json
def add(dict_values):
    addAsset = {}
    addAsset['data']=dict_values

    prepare_creation = bdb.transactions.prepare(
        operation='CREATE',
        signers=user.public_key,
        asset=addAsset,
    )

    fulfilled_creation  = bdb.transactions. fulfill (
        prepare_creation, user.private_key
    )

    # Send the transaction to BigchainDB
    sent_creation = bdb.transactions.send_commit(fulfilled_creation)
    return  fulfilled_creation ["id"]
```

4.3.2 set_asset_fields(asset, fields)

Hàm này dùng để phụ trợ cho hàm get_asset(asset_id).
Input: asset là một object dạng json, fields cũng là một object dạng json có các key là con của asset.
Sau khi thực hiện thì các trường trong asset sẽ thay đổi tương ứng theo fields.
Hiện thực:

```
def set_asset_fields(asset,  fields ):
  if  fields  == {}:
    return
  for  field  in  fields .keys():
    value  =  fields [ field ]
    if  type(value)  is  dict:
      set_asset_fields(asset [ field ],  value)
    else:
      asset [ field ]  = value
```

4.3.3 get_asset(asset_id)

Hàm này dùng để lấy thông tin của một asset ở thời điểm hiện tại.
Input: asset_id là id của asset cần lấy.
Output: return thông tin asset tương ứng
Hiện thực:

```
def get_asset(asset_id):
  try:
    transactions = bdb.transactions.get(asset_id=asset_id)
  except:
    print("Unable to get transactions!")
    return
  if transactions == [] or transactions is None:
    return {}
  asset = {}
  for transaction in transactions:
    if asset == {}:
      asset = transaction["asset"]["data"]
    else:
      set_asset_fields(asset, transaction["metadata"]["data"])
  return asset
```

4.3.4 update(asset_id, input_data_changed)

Hàm này thay đổi thông tin của asset cần thêm, bằng cách ghi lại những thay đổi trong trường metadata của transaction mới.
Input: asset_id là id của asset cần update, input_data_changed miêu tả sự thay đổi thông tin của asset (có dạng json chứa các trường cần thay đổi tương ứng với asset ban đầu).
Output: return id của transaction tương ứng.
Hiện thực:

```
def update(asset_id, input_data_changed):
    transfer_asset = {
        'id' : asset_id
    }
    data_changed = {}
    data_changed['data']=input_data_changed

    output_index = 0

    transactions = bdb.transactions.get(asset_id=asset_id)
    last_transaction = transactions[-1]

    output = last_transaction['outputs'][output_index]

    transfer_input = {
        'fulfillment': output['condition']['details'],
        'fulfills': {
        'output_index': output_index,
        'transaction_id': last_transaction['id'],
        },
        'owners_before': output['public_keys'],
    }

    prepare_creation = bdb.transactions.prepare(
        operation='TRANSFER',
        asset = transfer_asset,
```

```
        inputs = transfer_input,
        metadata= data_changed,
        recipients =user.public_key
    )

    last_transaction = bdb.transactions. fulfill (
        prepare_creation,
        private_keys=user.private_key
    )

    sent_creation = bdb.transactions.send_commit(last_transaction)
    return last_transaction["id"]
```

4.4 Sử dụng trong thực tế

Việc áp dụng hợp lý các hàm đã hiện thực ở trên sẽ xây dựng được ứng dụng từ điển bất động sản. Dưới đây là một ví dụ về việc áp dụng các hàm trên:

- Trước tên, để thay đổi một đối tượng (ở đây là thông tin của căn hộ), cần có một UI (User Interface) cho phép người dùng nhập vào các thông tin cần tìm kiếm của căn hộ. Sử dụng các hàm cần thiết để lấy được thông tin của các căn hộ và id tương ứng của asset. Sau đó dựa vào thông tin của các căn hộ để hiển thị cho người dùng.

```
data_searched = bdb.assets.get(search="D'Lusso")

if data_searched == []:
  print("not found")
else:
  for item in data_searched:
    print(json.dumps({"data" : get_asset(item["id"]), "id" : item["id"]}, ensure_ascii=False,
        indent=4))
```

- Sau đó, sẽ có một UI cho phép người dùng chọn căn hộ mong muốn. Dựa vào đối tượng mà người dùng chọn, suy ra được asset id tương ứng. Khi đã có asset id thì có thể gọi các hàm đã hiện thực ở trên tùy theo nhu cầu sử dụng.

```
asset_chosen_id = data_searched[-1]["id"]

update(asset_chosen_id, {"furniture_status" : "Full"})
```

4.5 Nhận xét

- Việc quản lý dữ liệu đảm bảo tính minh bạch.
- Việc lấy thông tin của asset ở thời điểm hiện tại tốn nhiều thời gian nếu có nhiều sự thay đổi trên asset đó.
- Phần hiện thực ở trên chưa bao gồm phân quyền truy cập cho các user khác nhau.

5 Những vấn đề khi áp dụng vào thực tế và giải pháp

5.1 Những vấn đề khi áp dụng vào thực tế

Cũng giống như các nền tảng blockchain khác, khi từ điển bất động sản ứng dụng công nghệ blockchain được sử dụng trong thực tế, khả năng cao sẽ xuất hiện các vấn đề như:

- Cộng đồng sử dụng lớn
- Cần phải lưu trữ một lượng lớn dữ liệu bất động sản
- Mật độ giao dịch dày đặt (gia tăng số transactions mỗi giây)

Vì blockchain luôn lưu lại mọi trạng thái và lịch sử giao dịch, những vấn đề trên sẽ làm gia tăng đáng kể kích thước dữ liệu cần lưu trữ. Việc phình to dữ liệu cũng ảnh hưởng đáng kể đến chất lượng và tốc độ xử lý thông tin nếu không cải thiện kiến trúc của hệ thống. Nhất là khi được sử dụng rộng rãi, số transaction mỗi giây tăng cao sẽ kéo theo sự chậm chạp trong việc đọc ghi dữ liệu, vì hệ thống sẽ phải duyệt qua từng transaction để đối chiếu và thao tác với dữ liệu.

5.2 Giải pháp giả thuyết

Các vấn đề nảy sinh khi áp dụng ứng dụng vào thực tế hầu như đều liên quan đến tốc độ truy xuất dữ liệu. Do cơ chế của blockchain, để đổi lấy tính minh bạch cũng như một số tính chất có lợi khác, người ta khó có thể cải thiện tốc độ truy xuất dữ liệu (thông tin về một bất động sản nào đó chẳng hạn) một cách rõ rệt. Vì vậy, nhóm đề xuất tiếp cận giải pháp cho các vấn đề này từ một góc nhìn khác, với yêu cầu hệ thống có thể đạt tốc độ truy xuất cao và vẫn đảm bảo các tính chất như blockchain, nhưng thỉnh thoảng có thể đánh đổi các tính chất đó để đảm bảo tốc độ.
Giải pháp này là xây dựng một cơ sở dữ liệu thông thường (không phải ứng dụng blockchain) để lưu trữ dữ liệu về từ điển bất động sản song song với cơ sở dữ liệu ứng dụng blockchain. Các thao tác cập nhật sẽ được thực hiện cùng lúc trên cả hai cơ sở dữ liệu, nhưng việc truy xuất chỉ được thực hiện trên cơ sở dữ liệu thông thường. Sau một chu kỳ thời gian nhất định (có thể là một số ngày đến một số tuần), dữ liệu ở cơ sở dữ liệu thông thường sẽ được kiểm tra xem có khớp với dữ liệu đang được lưu trữ ở cơ sở dữ liệu ứng dụng blockchain hay không. Nếu có, hệ thống hoạt động bình thường, còn nếu không, cần chỉnh sửa lại và/hoặc báo cáo cho người quản lý cơ sở dữ liệu.
Bằng cách trên, tuy trong một chu kỳ thời gian, nếu cơ sở dữ liệu thông thường bị tấn công, dữ liệu có thể không minh bạch nhưng khi hết chu kỳ đó thì có thể phát hiện ngay. Mặt khác, do truy xuất dữ liệu từ cơ sở dữ liệu thông thường nên tốc độ truy xuất sẽ nhanh hơn, và thường sẽ không phụ thuộc vào kích thước của dữ liệu theo thời gian (do cơ sở dữ liệu thông thường đã tối ưu việc này). Điều đó dẫn đến có thể vận dụng từ điển bất động sản này cho các ứng dụng thời gian thực (real-time application) như xác nhận thông tin listing mà người dùng đăng tải có đúng hay không.

6 Tổng kết

6.1 Những gì đã đạt được

- Hiểu được kiến thức cơ bản về blockchain: cấu trúc, đặc điểm và các cơ chế đồng thuận
- Tìm hiểu được một số nền tảng cơ bản dựa trên blockchain. Đặc biệt là BigChainDB
- Cài đặt được BigChainDB driver và hiện thực một số hàm cần thiết cho việc xây dựng từ điển bất động sản (add, set_asset, get_asset, update)
- Đề ra được giải pháp cho ứng dụng khi mở rộng cơ sở dữ liệu

6.2 Thuận lợi và khó khăn

- Thuận lợi:
  - Có sự hỗ trợ dữ liệu từ doanh nghiệp
  - Có sự phối hợp giữa các thành viên
- Khó khăn:
  - Kiến thức mới nên khó khăn trong việc tìm hiểu

6.3 Định hướng phát triển

- Phát triển một ứng dụng dựa trên BigChainDB để có thể xây dựng từ điển bất động sản

7 Mã nguồn

- Github Repository
- Hướng dẫn sử dụng