Hướng dẫn Triển khai: Tích hợp Polygon PoS & L2 CDK vào ChainFLIP

1. Giới thiệu

Tài liệu này cung cấp hướng dẫn chi tiết từng bước để tích hợp kiến trúc multi-chain sử dụng Polygon PoS (làm lớp điều phối trung tâm) và các Layer 2 tùy chỉnh (xây dựng bằng Polygon CDK) vào hệ thống ChainFLIP hiện có của bạn. Phương án này nhằm mục đích tận dụng hệ sinh thái Polygon để tăng cường khả năng mở rộng, tính linh hoạt và kiểm soát dữ liệu cho ChainFLIP, đồng thời giữ cho việc triển khai ban đầu dễ tiếp cận hơn so với việc xây dựng Relay Chain riêng.

Kiến trúc Mục tiêu:

- **Lớp Điều phối Chính:** Polygon PoS (Amoy cho thử nghiệm, Mainnet cho sản xuất). Lưu trữ các Hub Smart Contract trung tâm.
- **Lớp Thực thi của Người tham gia:** Các Layer 2 tùy chỉnh (ví dụ: ZK-Rollups) được xây dựng bằng Polygon CDK, do các bên tham gia chính (nhà sản xuất, nhà phân phối, v.v.) vận hành.
- **Tương tác Cross-Chain:** Sử dụng cầu nối gốc của Polygon (FxPortal) để chuyển tài sản (NFT) và tin nhắn giữa Polygon PoS và các L2 CDK.
- **Hub Smart Contracts (trên PoS):** Quản lý đăng ký người tham gia/L2, sổ đăng ký NFT cốt lõi, tổng hợp mô hình FL toàn cục, xử lý tin nhắn từ L2.
- **L2 Smart Contracts (trên L2 CDK):** Xử lý logic nghiệp vụ chi tiết, lưu trữ dữ liệu cục bộ (bao gồm CID IPFS), kích hoạt đào tạo FL cục bộ, tương tác với cầu nối.

Lưu ý: Hướng dẫn này giả định bạn sử dụng Polygon Amoy cho môi trường thử nghiệm. Các bước tương tự áp dụng cho Mainnet nhưng cần cấu hình và bảo mật cẩn thận hơn.

2. Giai đoạn 0: Chuẩn bị & Thiết lập Môi trường

Mục tiêu: Đảm bảo đội ngũ và môi trường sẵn sàng cho việc triển khai.

- **Yêu cầu Kỹ năng:**
  - Solidity & Phát triển Smart Contract (Hardhat/Foundry).
  - Hiểu biết về EVM, Polygon PoS.
  - Kiến thức cơ bản về Layer 2 (đặc biệt là ZK-Rollups nếu chọn).
  - Kinh nghiệm với Node.js, Docker.

    - Quản lý hạ tầng cơ bản.
- **Công cụ Cần thiết:**
    - Node.js (phiên bản LTS).
    - Docker & Docker Compose.
    - Polygon CDK CLI (cài đặt theo tài liệu Polygon).
    - Hardhat hoặc Foundry.
    - Git.
    - Ví trình duyệt (MetaMask).
- **Thiết lập Môi trường:**
    - Cấu hình MetaMask với mạng Polygon Amoy.
    - Nhận MATIC thử nghiệm từ Faucet Polygon Amoy.
    - Chuẩn bị thư mục dự án.

3. Giai đoạn 1: Triển khai Hub Smart Contracts (trên Polygon Amoy)

Mục tiêu: Thiết lập nền tảng điều phối trung tâm trên Polygon PoS.

1. **Phát triển Hub Smart Contracts (Solidity):**
    - `ParticipantRegistry.sol`:
        - `registerParticipant(address l2Address, string memory participantInfo)`: Đăng ký L2 và thông tin người tham gia.
        - `getParticipantInfo(address participantAddress)`: Lấy thông tin.
        - Quản lý quyền truy cập (ví dụ: chỉ chủ sở hữu hợp đồng có thể đăng ký).
    - `NFTCoreHub.sol` (ERC721):
        - `mint(address to, uint256 tokenId, string memory tokenURI)`: Tạo NFT mới.
        - `transferOwnership(address newOwner)`: Chuyển quyền sở hữu hợp đồng (nếu cần).
        - Quản lý `tokenURI` (có thể trỏ đến IPFS CID).
        - Tích hợp với logic cầu nối để xử lý khóa/mở khóa khi chuyển sang/từ L2.
    - `FLHubAggregator.sol`:
        - `submitUpdate(bytes memory modelUpdateData, address senderL2)`: Nhận cập nhật từ L2 (cần xác thực người gửi qua `BridgeListenerPoS`).
        - Logic tổng hợp (ví dụ: trung bình cộng đơn giản hoặc cơ chế an toàn hơn).
        - `getGlobalModel()`: Trả về mô hình toàn cục hiện tại.
        - Logic kích hoạt gửi mô hình mới về L2s (thông qua `BridgeListenerPoS`).
    - `BridgeListenerPoS.sol`:
        - Triển khai giao diện `IFxMessageProcessor` của FxPortal.

        - `processMessageFromChild(bytes memory messageData)`: Hàm được FxPortal gọi khi có tin nhắn từ L2. Phân tích `messageData` để xác định người gửi (L2) và loại tin nhắn (cập nhật FL, yêu cầu cập nhật trạng thái, v.v.), sau đó gọi hàm tương ứng trong các Hub contract khác (`FLHubAggregator.submitUpdate`, `NFTCoreHub.updateState`, v.v.).
2. **Kiểm thử (Unit & Integration):** Sử dụng Hardhat/Foundry để kiểm thử kỹ lưỡng các chức năng và tương tác giữa các Hub contract.
3. **Triển khai lên Amoy:** Dùng script Hardhat/Foundry. Lưu lại địa chỉ các hợp đồng.
4. **Xác thực Mã nguồn:** Trên trình khám phá khối Amoy.

4. Giai đoạn 2: Triển khai L2 CDK Chain Đầu tiên

Mục tiêu: Khởi tạo và vận hành một blockchain Layer 2 tùy chỉnh.

1. **Chọn Loại L2:** ZK-Rollup thường là lựa chọn tốt cho tính bảo mật và tương thích EVM.
2. **Cấu hình L2:**
    - Tham khảo tài liệu Polygon CDK mới nhất. Tạo các tệp cấu hình cần thiết (`config.json`, `.env`, v.v.).
    - Xác định Chain ID, tên mạng, RPC URL, tài khoản ban đầu (genesis accounts).
3. **Chạy Lệnh CDK:** Sử dụng `cdk-validium deploy` (hoặc lệnh tương ứng cho ZK-Rollup) để khởi tạo và chạy các node L2 bằng Docker.
4. **Xác minh:**
    - Kiểm tra logs Docker.
    - Lấy RPC URL của L2 mới.
    - Thêm mạng L2 vào MetaMask.
    - Kiểm tra kết nối và số dư tài khoản genesis.

5. Giai đoạn 3: Thiết lập Cầu nối (Polygon FxPortal)

Mục tiêu: Cho phép giao tiếp giữa Amoy (PoS) và L2 CDK.

1. **Hiểu FxPortal:** Đọc kỹ tài liệu về cách FxPortal hoạt động cho việc chuyển ERC721 và tin nhắn tùy chỉnh.
2. **Triển khai Hợp đồng Tunnel:**
    - **Trên Amoy (Root):** Triển khai `FxERC721RootTunnel`.
    - **Trên L2 CDK (Child):** Triển khai `FxERC721ChildTunnel`.
3. **Cấu hình & Ánh xạ:**
    - Gọi các hàm `setFxRootTunnel` và `setFxChildTunnel` trên các hợp đồng tunnel tương ứng để liên kết chúng.

    - Gọi `mapToken` trên `FxERC721RootTunnel` để ánh xạ hợp đồng `NFTCoreHub` (trên Amoy) với một hợp đồng ERC721 đại diện sẽ được triển khai trên L2 (trong Giai đoạn 4).
4. **Kiểm thử Tin nhắn:**
    - Sử dụng các hàm `sendMessageToChild` (trên Amoy) và `sendMessageToRoot` (trên L2) của FxPortal để gửi tin nhắn kiểm tra.
    - Xác nhận `BridgeListenerPoS` (trên Amoy) và một hợp đồng lắng nghe tương tự trên L2 (sẽ tạo ở Giai đoạn 4) nhận và xử lý được tin nhắn.

6. Giai đoạn 4: Triển khai L2 Smart Contracts & Tích hợp Logic ChainFLIP

Mục tiêu: Đưa logic nghiệp vụ ChainFLIP lên L2.

1. **Phát triển L2 Smart Contracts (Solidity):**
    - `L2NFTReceiver.sol` (ERC721): Hợp đồng đại diện cho NFT trên L2. Được ánh xạ với `NFTCoreHub` qua `mapToken`. Nó sẽ được `FxERC721ChildTunnel` gọi khi có NFT được bridge sang.
    - `L2Tracking.sol`:
        - `recordTrackingEvent(uint256 tokenId, string memory eventData, string memory ipfsCid)`: Ghi sự kiện, lưu CID IPFS.
        - `requestStateUpdate(uint256 tokenId, bytes memory updateData)`: Kích hoạt gửi tin nhắn cập nhật trạng thái về PoS qua cầu nối.
    - `L2FLTrainer.sol`:
        - `triggerLocalTraining()`: Logic kích hoạt (có thể gọi off-chain).
        - `submitModelUpdate(bytes memory modelUpdateData)`: Chuẩn bị và gửi cập nhật FL về PoS qua cầu nối.
    - `L2BridgeListener.sol`:
        - Triển khai giao diện `IFxMessageProcessor`.
        - `processMessageFromRoot(bytes memory messageData)`: Xử lý tin nhắn từ PoS (ví dụ: nhận mô hình FL toàn cục mới, cập nhật cấu hình).
2. **Triển khai Lên L2 CDK:** Dùng Hardhat/Foundry. Lưu địa chỉ.
3. **Cập nhật Backend/Frontend:**
    - Kết nối với RPC của L2 CDK.
    - Gọi các hàm trên L2 contracts (ví dụ: `recordTrackingEvent` khi quét QR).
    - Xử lý hiển thị dữ liệu từ cả PoS Hub và L2 contracts.

7. Giai đoạn 5: Hiện thực hóa Luồng Công việc Cross-Chain

Mục tiêu: Đảm bảo các quy trình chính hoạt động liền mạch.

1. **Mint & Chuyển NFT sang L2:**
    - Mint NFT trên `NFTCoreHub` (Amoy).
    - Gọi `deposit` trên `FxERC721RootTunnel` (Amoy).
    - Xác nhận NFT đại diện xuất hiện trên L2 (do `FxERC721ChildTunnel` mint vào `L2NFTReceiver`).
2. **Ghi nhận & Cập nhật Theo dõi:**
    - Ghi sự kiện trên `L2Tracking` (L2).
    - Kích hoạt `requestStateUpdate` trên `L2Tracking` (L2).
    - Xác nhận `BridgeListenerPoS` (Amoy) nhận và xử lý tin nhắn (ví dụ: cập nhật trạng thái liên quan đến NFT trong `NFTCoreHub`).
3. **Luồng Federated Learning:**
    - Kích hoạt `submitModelUpdate` trên `L2FLTrainer` (L2).
    - Xác nhận `BridgeListenerPoS` (Amoy) nhận và chuyển cho `FLHubAggregator`.
    - Xác nhận `FLHubAggregator` tổng hợp.
    - (Tùy chọn) Kích hoạt gửi mô hình mới từ `FLHubAggregator` về L2 và xác nhận `L2BridgeListener` nhận được.
4. **Chuyển NFT về PoS:**
    - Gọi hàm `withdraw` trên `L2NFTReceiver` hoặc `FxERC721ChildTunnel` (L2).
    - Xác nhận NFT đại diện bị burn trên L2 và NFT gốc được mở khóa trên `NFTCoreHub` (Amoy).

8. Giai đoạn 6: Kiểm thử Toàn diện, Triển khai & Giám sát

Mục tiêu: Đảm bảo hệ thống sẵn sàng và ổn định.

1. **Kiểm thử E2E:** Thực hiện các kịch bản đầy đủ trên môi trường thử nghiệm (Amoy + L2 CDK).
2. **Kiểm toán Bảo mật:** Audit tất cả smart contracts và logic cross-chain.
3. **Chuẩn bị Mainnet:**
    - Lập kế hoạch chi tiết.
    - Triển khai L2 CDK cho production.
    - Triển khai contracts lên Polygon PoS Mainnet và L2 production.
    - Cấu hình cầu nối cho Mainnet.

4. **Thiết lập Giám sát:**
    - Sử dụng các công cụ giám sát blockchain (ví dụ: Tenderly, Blockscout, hoặc custom scripts) để theo dõi:
        - Sức khỏe node L2.
        - Trạng thái giao dịch bridge (thành công/thất bại/pending).
        - Các sự kiện quan trọng từ smart contracts.
    - Thiết lập cảnh báo (ví dụ: qua PagerDuty, Slack).

9. Kết luận

Việc tích hợp này đòi hỏi sự cẩn thận trong thiết kế smart contract, cấu hình cầu nối và kiểm thử cross-chain. Bằng cách tuân theo các giai đoạn này và sử dụng các công cụ của Polygon, bạn có thể xây dựng một hệ thống ChainFLIP multi-chain mạnh mẽ và dễ quản lý hơn trong hệ sinh thái quen thuộc. Luôn tham khảo tài liệu mới nhất từ Polygon CDK và FxPortal trong quá trình triển khai.