Vite: Nền tảng Ứng dụng Phân quyền Không đồng bộ Hiệu suất cao

Vite là một nền tảng ứng dụng phân quyền phổ thông đáp ứng yêu cầu của các ứng dụng công nghiệp về lưu lượng lớn, thời gian xử lý yêu cầu nhanh, và khả năng nhân rộng, trong khi vẫn tập trung vào bảo mật tài khoản. Vite sử dụng cấu trúc sổ cái DAG, giao dịch trong sổ cái được nhóm lại theo tài khoản. Cấu trúc Snapshot Chain trong hệ thống Vite có thể bù đắp lại cho mức độ bảo mật thấp trong sổ cái DAG. Thuật toán đồng thuận HDPoS, mà qua đó việc ghi và xác nhận giao dịch được thực hiện không đồng bộ, cung cấp hiệu suất và khả năng nhân rộng cao. Máy ảo của Vite tương thích với EVM, ngôn ngữ hợp đồng thông minh mở rộng từ Solidity, cung cấp khả năng mô tả cao hơn. Thêm vào đó, một cải tiến quan trọng trong thiết kế của Vite là mô hình Kiến trúc Event Driven, mô hình truyền thông tin thông qua dữ liệu giữa các hợp đồng thông minh giúp cải thiện đáng kể lưu lượng và khả năng nhân rộng của hệ thống. Ngoài token bản địa được xây dựng sẵn, Vite cũng hỗ trợ người dùng phát hành tài sản mã hóa riêng của họ, đồng thời cung cấp giao dịch và chuyển giao giá trị xuyên chuỗi dựa trên Giao thức Loopring [1]. Vite thực hiện phân bổ nguồn lực thông qua hạn ngạch và người dùng không thường xuyên không phải trả phí giao dịch. Vite cũng hỗ trợ lập kế hoạch hợp đồng, dịch vụ cung cấp tên miền, cập nhật hợp đồng, kỹ thuật gọt mỏng blockchain và các tính năng khác.

1 Giới thiệu

1.1 Định nghĩa

Vite là một ứng dụng phân quyền phổ thông hỗ trợ một nhóm hợp đồng thông minh, trong đó mỗi hợp đồng là một máy trạng thái với trạng thái độc lập và các phép logic toán tử khác nhau mà có thể giao tiếp thông qua quá trình truyền tải thông điệp.

Nhìn chung đây là một máy trạng thái giao dịch. Trạng thái của hệ thống $s \in S$, cũng được biết đến như trạng thái thế giới, bao gồm trạng thái của mỗi tài khoản độc lập. Một sự kiện tạo thay đổi trong trạng thái tài khoản được gọi là giao dịch. Định nghĩa chính thức như sau:

Định nghĩa 1.1 (Máy trạng thái giao dịch) *một máy trạng thái giao dịch bao gồm 4 bộ dữ liệu (4 tuple): (T , S, g, δ), trong đó T là tập hợp các giao dịch, S là tập hợp trạng thái, g ∈ S là trạng thái ban đầu, cũng được gọi là khối nguyên thủy, δ : S x T →S là chức năng chuyển đổi trạng thái.*

Ngữ nghĩa học của máy trạng thái giao dịch này là một hệ thống chuyển tiếp gián đoạn, được định nghĩa như sau:

Định nghĩa 1.2 (Ngữ nghĩa học của máy trạng thái giao dịch) *ngữ nghĩa học của máy trạng thái giao dịch (T , S, s0, δ) là một hệ thống chuyển tiếp gián đoạn (S, s0, →).→∈ S × S là một quan hệ chuyển tiếp.*

Đồng thời, nền tảng ứng dụng phân quyền là một hệ thống phân quyền với mô hình nhất quán cuối cùng. Thông qua một số thuật toán đồng thuận, trạng thái cuối cùng có thể đạt được giữa các điểm nút. Trong thực tế, những gì được lưu trữ trong trạng thái hợp đồng thông minh là tập hợp dữ liệu hoàn chỉnh trong một ứng dụng phân quyền với khối lượng lớn và không thể được truyền đi giữa các điểm nút. Do đó, các điểm nút cần chuyển giao một nhóm giao dịch để đạt được tính nhất quán của trạng thái cuối cùng. Chúng tôi tổ chức nhóm giao dịch như thế này thành một cấu trúc dữ liệu cụ thể, thường được gọi là sổ cái.

Định nghĩa 1.3 (Sổ cái) *Sổ cái bao gồm một nhóm giao dịch với một kiểu dữ liệu trừu tượng được xây dựng theo công thức đệ quy. Nó được định nghĩa như sau:*

$$\begin{cases} l = \Gamma(T_t) \\ l = l_1 + l_2 \end{cases}$$

Trong đó,T_t ∈ 2T, biểu thị tập hợp giao dịch, Γ ∈2T → L biểu thị chức năng xây dựng một sổ ghi thông qua tập hợp giao dịch, L là tập hợp sổ cái, + : L x L → L biểu thị hoạt động hợp nhất hai sổ cái con thành một.

Nên chú ý rằng trong những hệ thống như thế này, sổ cái thường được sử dụng để biểu thị một nhóm giao dịch, thay vì một trạng thái. Ở Bitcoin [2] và Ethereum [3] sổ cái là một cấu trúc blockchain mà ở đó lệnh giao dịch được đặt trên trên toàn hệ thống. Để thay đổi một giao dịch trong sổ cái, chúng ta cần xây dựng lại một sổ cái con trong sổ ghi tài khoản, nhờ đó gia tăng chi phí của việc thực hiện giao dịch giả mạo.

Theo cùng nhóm giao dịch, những sổ ghi hợp lệ khác nhau có thể được xây dựng, nhưng chúng biểu thị một thứ tự giao dịch khác và có thể làm cho hệ thống chuyển sang một trạng thái khác. Khi điều này xảy ra, nó thường được gọi là ”fork”.

Định nghĩa 1.4 (Fork) *Giả sử T_t, $T_{t'}$ ∈2T ,$T_t \subseteq T_{t'}$. Nếu l = Γ1(T_t),l' = Γ2($T_{t'}$),và không đáp ứng l ⪯ l',chúng ta có thể gọi l và l' là sổ cái fork. ⪯ biểu thị quan hệ tiền tố.*

Theo ngữ nghĩa học của máy trạng thái giao dịch, chúng ta có thể dễ dàng chứng minh rằng từ trạng thái ban đầu, nếu sổ cái không được fork, cuối cùng mỗi điểm nút sẽ đi vào cùng một trạng thái. Do đó nếu một sổ cái fork được nhận, liệu nó có chuyển sang trạng thái khác không? Điều này phụ thuộc vào tính logic riêng của giao dịch trong sổ cái, và cách sổ cái sắp xếp thứ tự bộ phận giữa các giao dịch. Trên thực tế, thường có một số giao dịch thoản mãn luật giao hoán nhưng do vấn đề của thiết kế tài khoản, chúng thường gây ra các fork. Khi hệ thống khởi động từ một trạng thái ban đầu, tiếp nhận hai sổ cái được fork và kết thúc ở cùng trạng thái đó, chúng ta gọi hai sổ cái này là những sổ cái fork giả.

Định nghĩa 1.5 (Fork giả*) trạng thái ban đầu s0 ∈ S,sổ cái l_1, l_2 ∈ L, s_0 (l_1) → s_1 , s_0 (l_2) → s_2 và nếu s_1 = s_2 chúng ta gọi hai sổ cái l_1, l_2 là sổ cái fork giả.*

Một sổ cái được thiết kế tốt sẽ hạn chế khả năng fork giả.

Khi fork diễn ra, mỗi điểm nút cần chọn một trong số nhiều sổ cái được fork. Để đảm bảo tính nhất quán của trạng thái, các điểm nút cần sử dụng cùng một thuật toán để hoàn thành quá trình lựa chọn. Thuật toán này gọi là thuật toán đồng thuận.

Định nghĩa 1.6 (Thuật toán đồng thuận) *Thuật toán đồng thuận là một chức năng nhận một tập hợp sổ cái và trả lại một sổ cái duy nhất: Φ : 2L → L*

Thuật toán đồng thuận là một phần quan trọng của hệ thống. Một thuật toán đồng thuận nên có tốc độ hội tụ cao để giảm thiểu ảnh hưởng của đồng thuận trong các fork khác nhau, và nên có khả năng chống lại các cuộc tấn công.

1.2 Tiến độ hiện tại

The Ethereum[4] dẫn đầu trong phát triển những hệ thống như vậy. Theo thiết kế của Ethereum, định nghĩa về trạng thái thế giới là S = ΣA, một ánh xạ từ tài khoản a ∈ A và trạng thái tài khoả σa ∈ Σ. Do đó, bất kỳ trạng thái nào trong máy trạng thái của the Ethereum đều mang tính tổng quát, có nghĩa là một điểm nút có thể đạt được trạng thái của bất kỳ tài khoản nào ở bất kỳ thời điểm nào.

Chức năng chuyển tiếp trạng thái δ của Ethereum được xác định bởi tập hợp mã code chương trình. Mỗi nhóm code được gọi là một hợp đồng thông minh. Ethereum xác định một máy ảo hoàn chỉnh Turing, gọi là EVM, với bộ chỉ dẫn gọi là EVM code. Người dùng có thể phát triển hợp đồng thông minh thông qua

ngôn ngữ lập trình Solidity tương tự như JavaScript, và tập hợp chúng thành EVM code, sau đó sử dụng chúng trên Ethereum [5]. Khi hợp đồng thông minh được tạo lập thành công, nó tương đương với việc xác định tài khoản hợp đồng a nhận chức năng chuyển tiếp trạng thái δa. EVM được sử dụng rộng rãi nhưng có một số vấn đề như thiếu hỗ trợ chức năng và vấn đề bảo mật.

Cấu trúc sổ cái của Ethereum là một block chain [2], block chain này được cấu thành từ các khối, mỗi khối chứa một danh sách các giao dịch, và khối sau tham chiếu tới hàm băm của khối trước để tạo nên một cấu trúc chuỗi.

$\Gamma(\{t1, t2, ...|t1, t2, ... \in T\}) = (..., (t1, t2, ...))$

Ưu điểm lớn nhất của cấu trúc này là ngăn chặn giao dịch không bị thay đổi một cách hiệu quả, nhưng vì nó duy trì thứ tự hoàn chỉnh của mọi giao dịch, việc chuyển đổi thứ tự của hai giao dịch sẽ tạo ra một sổ cái mới, có khả năng cao tạo ra fork. Trên thực tế, theo định nghĩa này, không gian trạng thái của máy trạng thái giao dịch được coi là một cây: trạng thái ban đầu là điểm nút rễ, thứ tự giao dịch khác nhau biểu thị các hướng đi khác nhau, và điểm nút lá là trạng thái cuối cùng. Trên thực tế, trạng thái của một số lượng lớn điểm nút lá đều giống nhau, tạo ra một lượng lớn fork giả.

Thuật toán đồng thuận Φ được gọi là PoW, được đề xuất lần đầu tiên trong giao thức Bitcoin [2].Thuật toán PoW phụ thuộc vào một bài toán có thể dễ dàng xác minh nhưng khó để giải. Ví dụ, dựa trên chức năng hàm băm h : N □ N ,tìm ra kết quả x, thỏa mã yêu cầu $h(T + x) \geq d$, d là một số đã cho, gọi là độ khó, T là biểu diễn nhị phân của danh sách giao dịch trong khối. Mỗi khối trong block chain chứa một lời giải của bài toán này. Tăng độ khó cho tât cả các khối, là tổng độ khó của một sổ cái blockchain:

$$D(l) = D(\sum_i li) = \sum_i D(li) \qquad (2)$$

Do đó để chọn tài khoản đúng từ fork, hãy chọn fork có độ khó cao nhất:

$$\Phi(l_1, l_2, ..., l_n) = l_m \text{ where } m = \arg\max_{i \in 1..n}(D(l_i)) \qquad (3)$$

Thuật toán đồng thuận PoW có bảo mật cáo hơn và chạy tốt hơn trên Bitcoin và Ethereum. Tuy nhiên có hai vấn đề chính trong thuật toán này. Đầu tiên là để giải một bài toán yêu cầu một lượng lớn nguồn lực máy tính, gây ra tình trạng hao phí năng lượng. Thứ hai là tốc độ hội tụ của thuật toán chậm, ảnh hưởng đến lưu lượng chung của hệ thống. Hiện tại tốc độ giao dịch/giây của Ethereum chỉ khoảng 15 giao dịch, không thể đáp ứng yêu cầu của các ứng dụng phân quyền.

1.3 Hướng cải thiện

Sau khi Ethereum ra đời, cộng đồng Ethereum và các dự án tương tự khác bắt đầu cải thiện hệ thống theo các hướng khác nhau. Từ mô hình trừu tượng của hệ thống, có thể cải thiện theo những hướng sau:

Cải thiện trạng thái hệ thống S
Cải thiện chức năng chuyển tiếp trạng thái δ
Cải thiện cấu trúc sổ cái Γ
Cải thiện thuật toán đồng thuận Φ
Cải thiện trạng thái hệ thống

Ý tưởng chính của việc cải thiện trạng thái hệ thống là bản địa hóa trạng thái toàn phần, mỗi điểm nút không còn liên quan đến chức năng chuyển giao tất cả mọi trạng thái và giao dịch nữa, mà chỉ duy trì một tập hợp con của máy trạng thái. Bằng cách này, thế năng của tập S và T giảm đáng kể, nhờ đó giúp cải thiện khả năng nhân rộng của hệ thống. Những hệ thống như vậy bao gồm: Cosmos [6], Aelf[7], Pchain,....

Về cơ bản, phương thức dựa trên chuỗi bên này hi sinh tính toàn vẹn của trạng thái hệ thống để đổi lấy khả năng nhân rộng. Điều này làm cho sự phân tán của các ứng dụng phân quyền chạy trên hệ thống yếu đi – lịch sử giao dịch của một hợp đồng thông minh không còn được lưu lại bởi mọi điểm nút trong mạng nữa, mà chỉ được lưu bởi một phần điểm nút. Thêm vào đó, tương tác giữa các hợp đồng sẽ trở thành điểm bế tắc của hệ thống đó. Ví dụ trong Cosmos tương tác trong những khu vực (Zone) khác nhau cần một Trung tâm (Hub) chuỗi để hoàn thành [6].

1.3.2 Cải thiện chức năng chuyển tiếp trạng thái

Dựa trên việc cải thiện EVM, một vài dự án cung cấp nhiều ngôn ngữ lập trình cho hợp đồng thông minh hơn. Ví dụ, một ngôn ngữ hợp đồng thông minh Rholang được xác định trong Rchain dựa trên phép tính π; Hợp đồng thông minh của NEO được gọi là NeoContract, có thể được phát triển bằng ngôn ngữ lập trình phổ thông như Java,C# etc; EOS được lập trình với C/C++.

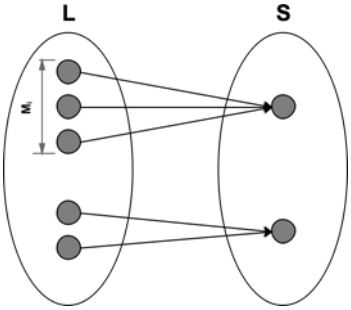
1.3.3 Cải thiện cấu trúc sổ cái

Hướng cải thiện cấu trúc sổ cái là xây dựng các lớp tương đương. Sổ cái tuyến tính với thứ tự toàn phần của nhiều giao dịch được cải thiện thành một sổ cái phi tuyến tính mà chỉ ghi lại những quan hệ thứ tự bộ phận. Cấu trúc sổ cái phi tuyến tính này là một DAG (đồ thị có hướng không chu trình). Hiện tại, Byteball [8], IOTA[9], Nano[10] và một số dự án khác đã thực hiện chức năng mã hóa tiền tệ dựa trên cấu trúc tài khoản của DAG. Một số dự án đang cố gắng sử dụng DAG để cải thiện hợp đồng thông minh, nhưng cho đến bây giờ những cải thiện theo hướng này vẫn đang được tìm hiểu

1.3.4 Cải thiện thuật toán đồng thuận

Việc cải thiện thuật toán đồng thuận chủ yếu là để cải thiện lưu lượng của hệ thống, và hướng đi chính là ngăn chặn

fork giả được tạo ra. Tiếp theo chúng tôi sẽ nói về những yếu tố liên quan đến fork giả.



Hình 1: Fork giả

Như trong hình, L là tập hợp các tài khoản có thể được fork cho một tập hợp các giao dịch, S là tập hợp các trạng thái có thể đạt được theo thứ tự khác nhau. Theo định nghĩa 1.4, ánh xạ f : L → S là ánh xạ toàn ánh; và theo định nghĩa 1.5, ánh xạ này là ánh xạ đơn ánh. Ở đây chúng tôi tính toán khả năng của fork giả như sau:

Giả sử C người dùng có quyền tạo ra sổ cái, M = |L|, N = |S|, Mi = |Li|, trong đó Li = {l|f (l) = si, si ∈ S}. Khả năng của fork giả sẽ là:

$$Pff = \sum_{i=1}^{n} \left(\frac{Mi}{M}\right)^{c} - \frac{1}{M^{c-1}} \qquad (4)$$

Từ công thức này chúng ta có thể thấy rằng, để giảm khả năng fork giả, có hai cách sau:

Tạo lập các quan hệ tương đương trên L của tập hợp sổ cái, chia lớp tương đương thành chúng, rồi xây dựng ít sổ cái hơn.

Hạn chế người dùng có quyền tạo sổ cái, do đó hạn chế C

Cách đầu tiên là hướng đi quan trọng trong thiết kế của Vite. Hướng đi này sẽ được thảo luận chi tiết ở phần sau. Cách thứ hai được sử dụng bởi nhiều thuật toán. Trong thuật toán PoW, bất kỳ người dùng nào cũng có quyền tạo ra một khối; còn thuật toán PoS thì hạn chế quyền tạo khối của những người có quyền trong hệ thống; thuật toán DPoS [11] hạn chế người dùng có quyền tạo khối trong một nhóm điểm nút tác nhân.

Hiện tại, thông qua thuật toán đồng thuận được cải thiện, một vài dự án có tầm ảnh hưởng lớn đã xuất hiện. Ví dụ, Carnado sử dụng thuật toán PoS gọi là Ouroboros, và lý thuyết [12] đưa ra một bằng chứng chặt chẽ về những đặc điểm liên quan của thuật toán; thuật toán BFT-DPOS sử dụng bởi EOS[13] là một biến thể của thuật toán DpoS và cải thiện lưu lượng hệ thống bằng việc sản xuất các khối một cách nhanh chóng; Thuật toán đồng thuận của Qtum [14] cũng là một thuật toán PoS; thuật toán Casper sử dụng bởi Rchain [15] cũng là một trong những thuật toán PoS.

Có nhiều dự án khác cũng đưa ra đề xuất riêng cho việc cải thiện thuật toán đồng thuận. NEO[16] sử dụng thuật toán BFT gọi là dBFT, Cosmos[6] dùng thuật toán Tendermint [17].

2 Sổ cái

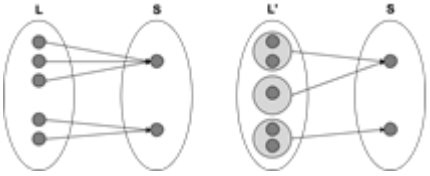
2.1 Tổng quan

Vai trò của sổ cái là quyết định thứ tự của giao dịch và thứ tự của giao dịch sẽ ảnh hưởng đến hai khía cạnh sau:

Tính nhất quán của trạng thái: Do trạng thái của hệ thống không phải một CRDT (Loại dữ liệu được sao chép không xung đột) [18], không phải tất cả giao dịch đều có thể trao đổi được, và trình tự thực hiện giao dịch khác nhau có thể đưa hệ thống vào những trạng thái khác nhau.

Độ hiệu quả của hàm băm: Trong sổ cái, giao dịch sẽ được nhóm thành các khối có chứa hàm băm được tham

Thiết kế của sổ cái có hai mục đích chính:

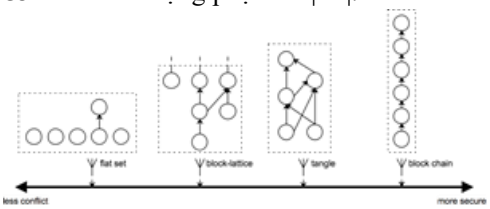


Hình hai 2: Hợp nhất sổ cái

Giảm tỉ lệ fork giả: như đã nói ở phần trên, việc giảm thiểu tỉ lệ fork giả có thể được thực hiện bằng cách tạo lập một lớp tương đương và kết hợp một nhóm tài khoản mà đưa hệ thống vào cùng một trạng thái của một tài khoản duy nhất. Như đã chỉ ra ở trên, theo công thức tỉ lệ fork giả, tỉ lệ fork giả của sổ cái bên trái là: $Pff = \left(\frac{3}{5}\right)^{c} + \left(\frac{2}{5}\right)^{c} - \frac{1}{5^{c-1}}$

Sau khi hợp nhất không gian sổ cái, tỉ lệ fork giả của đồ thị bên phải là $Pff' = \left(\frac{2}{3}\right)^{c} + \left(\frac{1}{3}\right)^{c} - \frac{1}{3^{c-1}}$ Nó sẽ được tìm ra khi C > 1, Pff' < Pff. Như vậy chúng ta nên giảm thiểu quan hệ thứ tự bộ phận giữa các giao dịch và cho phép nhiều giao dịch được trao đổi tuần tự.

Chống giả mạo: khi một giao dịch t được thay đổi trong sổ cái l, trong hai sổ cái con của sổ ghi l = l1 + l2, sổ cái con l1 không bị ảnh hưởng, và tham chiếu băm trong sổ cái con l2 cần xây dựng lại để tạo nên một sổ cái mới hợp lệ l′ = l1 + l2′. Sổ cái con bị ảnh hưởng l2 = Γ(T2), T2 = {x | x ∈ T, x > t} do đó để tăng chi phí giao dịch giả mạo, cần duy trì quan hệ thứ tự bộ phận giữa các giao dịch mở mức cao nhất có thể để mở rộng phạm vi |T2|.



Hình 3: So sánh cấu trúc sổ cái

Rõ ràng hai mục tiêu trên mâu thuẫn nhau và cần phải thực hiện sự đánh đổi khi thiết kế cấu trúc tài khoản. Do việc duy trì tài khoản là một thứ tự bộ phận giữa các giao dịch, nó là tập hợp thứ tự bộ phận (poset) [19], nếu được biểu thị

bởi biểu đồ Hasse (Hasse diagram)[20], nó là một DAG trên cấu trúc liên kết.

Hình trên so sánh một số cấu trúc sổ cái thông thường; những sổ cái gần phía bên trái được duy trì với thứ tự mang tính bộ phận ít hơn. Biểu đồ Hasse rộng hơn và tỉ lệ fork giả thấp; Các sổ cái gần phía bên phải duy trì nhiều quan hệ thứ tự bộ phận hơn, và biểu đồ Hasse hẹp hơn và chống giả mạo tốt hơn

Trong hình, ngoài cùng phía bên trái là một cấu trúc tập hợp thông thường trong một hệ thống tập trung không có tính năng chống giả mạo; ngoài cùng bên phải là một sổ cái blockchain với tính năng chống giả mạo tốt nhất; hai mô hình ở giữa, có hai sổ cái DAG, tài khoản block-lattice [10] sử dụng bởi Nano ở bên trái; và ở bên phải là sổ ghi tangle [9] sử dụng bởi IOTA . Về mặt đặc tính, blocklattice duy trì ít quan hệ thứ tự bộ phận hơn và thích hợp hơn cho cấu trúc tài khoản của nền tảng ứng dụng phân quyền hiệu suất cao. Do đặc điểm chống giả mạo kém, nó chứa nhiều rủi ro bảo mật, gần đây không dự án nào sử dụng cấu trúc sổ cái này ngoài Nano.

Để tăng cường hiệu suất, Vite sử dụng cấu trúc sổ cái DAG. Đồng thời, bằng việc đưa ra một cấu trúc chuỗi bổ sung Snapshot Chain và cải thiện thuật toán đồng thuận, những khuyết điểm của bảo mật block-lattice được bù đắp lại thành công và cả hai sự cải thiện trên sẽ được thảo luận chi tiết ở mục sau.

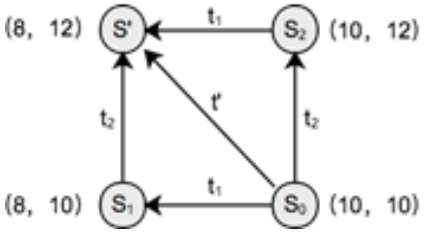
2.2 Ràng buộc

Đầu tiên, hãy xem những điều kiện tiên quyết để sử dụng cấu trúc sổ cái này cho mô hình máy trạng thái. Cấu trúc này cơ bản là sự kết hợp của toàn bộ máy trạng thái như một tập hợp nhiều máy trạng thái, mỗi tài khoản tương ứng với một máy trạng thái độc lập, mỗi giao dịch chỉ ảnh hưởng đến trạng thái của một tài khoản. Trong sổ cái, tất cả giao dịch được nhóm thành các tài khoản và tổ chức thành một chuỗi giao dịch trong cùng một tài khoản. Do đó chúng ta có những hạn chế sau đây trên trạng thái S và giao dịch T trong hệ thống Vite:

Định nghĩa 2.1 (Ràng buộc bậc tự do đơn lẻ) *trạng thái hệ thống $s \in S$ là véc-tơ $s = (s1, s2, ..., sn)$tạo thành từ trạng thái si của mỗi tài khoản. Đối với $\forall ti \in T$, sau khi thực hiện giao dịch ti, trạng thái hệ thống chuyển đổi như sau s: $(s1', ..., si', ..., sn') = \sigma(ti, (s1, ..., si, ..., sn))$,cần thỏa mãn:$sj' = sj, j \neq i$. Ràng buộc này gọi là ràng buộc bậc tự do đơn lẻ đối với một giao dịch*.

Một giao dịch bậc tự do đơn lẻ sẽ chỉ thay đổi trạng thái của một tài khoản mà không gây ảnh hưởng đến trạng thái của các tài khoản khác trong hệ thống. Trong không gian đa chiều nơi mà véc-tơ không gian trạng thái được xác định, một giao dịch được thực hiện, và trạng thái hệ thống chỉ chuyển động theo hướng song song với một trục tọa độ. Vui lòng nhớ rằng định nghĩa này cụ thể hơn so với định nghĩa giao dịch trong Bitcoin, Ethereum và các mô hình khác. Một giao dịch trong Bitcoin sẽ thay đổi trạng thái tài khoản của cả người gửi và người nhận; Ethereum có thể thay đổi trạng thái của hơn hai tài khoản thông qua một lựa chọn mua.

Theo ràng buộc này, quan hệ giữa các giao dịch có thể được đơn giản hóa. Bất kỳ một giao dịch sẽ là giao dịch trực giao hoặc song song. Điều này cung cấp điều kiện cho giao dịch nhóm theo các tài khoản. Dưới đây là một ví dụ minh họa:



Hình 4: Giao dịch bậc tự do đơn lẻ và and trạng thái trung gian

Như hình trên, giả sử Alice và Bob mỗi người có 10 USD. Trạng thái hệ thống ban đầu là s0 = (10, 10). Khi Alice muốn chuyển 2 USD cho Bob, trong mô hình Bitcoin và Ethereum, một giao dịch t′,có thể khiến hệ thống chuyển trực tiếp sang trạng thái cuối cùng: $s0 \rightarrow^{t'} s'$.

Theo định nghĩa của Vite, giao dịch t' thay đổi trạng thái tài khoản của Alice và Bob, điều không tuân theo nguyên lý bậc tự do đơn lẻ. Do đó, giao dịch không thể phân chia làm hai:

1) Giao dịch t1 biểu thị việc chuyể 2 USD của Alice

2) Giao dịch t2 biểu thị việc nhận 2 USD của Bob

Theo cách này từ trạng thái ban đầu đến trạng thái cuối s', có hai hướng đi khác nhau: $s0 \rightarrow^{t1} s1 \rightarrow^{t2} s'$ và $s0 \rightarrow^{t2} s2 \rightarrow^{t1} s'$. Hai hướng đi này lần lượt được qua trạng thái trung gian s1 và s2. Hai trạng thái trung gian này là ánh xạ của trạng thái cuối cùng trong kích thước hai tài khoản. Nói cách khác, nếu bạn quan tâm về trạng thái của một tài khoản, bạn chỉ cần thực hiện tất cả giao dịch tương ứng với tài khoản đó mà không cần phải thực hiện giao dịch của tài khoản khác.

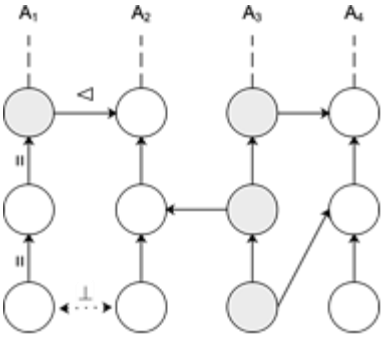
Tiếp theo chúng tôi sẽ xác định cách chia giao dịch trong Ethereum thành giao dịch bậc tự do đơn lẻ mà Vite yêu cầu:

Định nghĩa 2.2 (Phân tách giao dịch) *Phân chia một giao dịch với bậc tự do lớn hơn 1 thành một nhóm giao dịch tự do bậc đơn, được gọi là Phân tách giao dịch. Một giao dịch chuyển đổi có thể được chia thành một giao dịch gửi và một giao dịch nhận; một giao dịch hợp đồng chọn mua có thể được chia thành một giao dịch yêu cầu hợp đồng và một giao dịch đáp ứng hợp đồng; một quyền chọn mua trong mỗi hợp đồng có thể chia thành một*

giao dịch yêu cầu hợp đồng và một giao dịch đáp ứng hợp đồng.
Do đó sẽ có hai loại giao dịch khác nhau trong sổ cái. Hai giao dịch này được gọi là "cặp giao dịch":
Định nghĩa 2.3 (Cặp giao dịch) *một giao dịch gửi hoặc giao dịch yêu cầu hợp đồng, được gọi chung là "giao dịch yêu cầu"; một giao dịch nhận hoặc một giao dịch đáp ứng hợp đồng, được gọi chung là "giao dịch đáp ứng". Một giao dịch yêu cầu và một giao dịch đáp ứng được gọi là cặp giao dịch. Tài khoản để bắt đầu giao dịch yêu cầu t được ghi lại là A(t); giao dịch đáp ứng tương ứng được ghi là t ˜, tài khoản tương ứng với giao dịch được ghi lại là A(t ˜).*
Dựa vào định nghĩa trên chúng ta có thể kết luận quan hệ giữa hai giao dịch trong Vite như sau:
Định nghĩa 2.4 (Quan hệ giao dịch) *Có thể tồn tại các quan hệ sau giữa hai giao dịch t1 và t2:*
Trực giao: Nếu A(t1) ≠ A(t2),Hai giao dịch vuông Góc với nhau , được ghi là t1 ⊥ t2;
Song song: Nếu A(t1) = A(t2),hai giao dịch song song, được ghi là t1 ∥ t2;
Nhân quả: Nếu t2 = t˜1,hai giao dịch hai giao dịch quan hệ nhân quả được ghi là t1 ▷ t2, hoặc t2 ◁ t1.

2.3 Định nghĩa Sổ cái

Để định nghĩa một sổ cái, phải định nghĩa một poset (tập hợp thứ tự bộ phận). Trước hết hãy cùng định nghĩa quan hệ thứ tự bộ phận giữa cac giao dịch trong Vite:
Định nghĩa 2.5 (Thứ tự bộ phận của giao dịch) *chúng tôi sử dụng quan hệ nhị nguyên < để thể hiện cho mối quan hệ thứ tự bộ phận giữa hai giao dịch*
Một giao dịch đáp ứng phải theo sau một giao dịch yêu cầu tương ứng :t1 < t2 ⇔ t1 ▷t2;
Mọi giao dịch trong một tài khoản phải theo đúng thứ tự nhất quán chung: ∀t1 ∥ t2, phải :t1 < t2, hoặc t2 < t1.
Do quan hệ thứ tự bộ phận được tạo lập trên nhóm giao dịch T phải đáp ứng các điểm sau:
Không phản xạ: ∀t ∈ T ,không có t < t;
Bắc cầu: ∀ t1, t2, t3 ∈ T , nếu t1 < t2, t2 < t3, thì t1 < t3;
Phản đối xứng:∀t1, t2 ∈ T , if t1 < t2, và không tồn tại t2 < t1
Theo cách này, chúng tôi định nghĩa tài khoản Vite theo một tập hợp thứ tự bộ phận chặt chẽ:
Định nghĩa 2.6 (Sổ cái Vite) *Sổ cái Vite là một poset chặt chẽ được tạo thành bởi nhóm T của giao dịch đã cung cấp, và poset <*



Hình 5: Quan hệ giữa sổ cái và giao dịch trong Vite
Một poset chặt chẽ có thể tương ứng với một cấu trúc DAG. Như trong hình trên, các vòng tròn biểu thị các giao dịch, mũi tên biểu thị sự phụ thuộc giữa giao dịch. a → b thể hiện a phục thuộc vào b.
Sổ cái Vite được định nghĩa ở trên có cấu trúc tương tự với block-lattice. Các giao dịch được chia ra thành giao dịch yêu cầu và giao dịch đáp ứng, mỗi giao dịch ứng với một khối riêng, mỗi tài khoản Ai ứng với một chuỗi, mỗi cặp giao dịch và mỗi giao dịch đáp ứng, tham chiếu tới hàm băm của giao dịch yêu cầu tương ứng.

3 Chuỗi Snapshot

3.1 Xác nhận giao dịch

Khi một tài sản được fork, kết quả đồng thuận có thể thay đổi giữa hai sổ cái được fork. Ví dụ, dựa trên cấu trúc blockchain, nếu một điểm nút nhận chuỗi fork dài hơn, fork mới sẽ được chọn như kết quả đồng thuận, fork ban đầu sẽ bị bỏ lại, và giao dịch trên fork ban đầu sẽ buộc phải lùi trở lại trạng thái cũ. Trong một hệ thống như vậy, việc lùi giao dịch này là một sự kiện nghiêm trọng, có thể gây ra tình trạng double-spend. Hãy tưởng tượng rằng một doanh nghiệp nhận một thanh toán, cung cấp dịch vụ và hàng hóa, sau đó thanh toán đó bị rút về, như vậy doanh nghiệp đó sẽ bị lỗ vốn. Do đó khi một người dùng nhận giao dịch thanh toán, họ phải đợi cho hệ thống xác nhận giao dịch để đảm bảo hạn chế khả năng lùi dữ liệu ở mức thấp nhất
Định nghĩa 3.1 (Xác nhận giao dịch) *khi khả năng giao dịch bị lùi lại trạng thái ban đầu thấp hơn mức cho trước ϵ, giao dịch được xác nhận. Pr(t) < ϵ ⇔ t được xác nhận.*
Xác nhận giao dịch là một khái niệm khá phức tạp, bởi vì việc giao dịch có được xác nhận hay không phụ thuộc vào độ tin cậy ngầm từ 1 - ϵ. Một thương nhân bán kim cương và một người bán cà phê sẽ chịu thiệt hại khác nhau khi họ bị tấn công double spend. Do đó thương nhân bán kim cương cần đặt mức ϵ nhỏ hơn trong giao dịch. Đây cũng chính là bản chất của số xác nhận trong Bitcoin. Trong Bitcoin, số xác nhận cho thấy chiều sâu của một giao dịch trong blockchain. Số xác nhận càng lớn, khả năng lùi trạng thái giao dịch [2] càng thấp. Do đó, người thương nhân có thể gián tiếp điều chỉnh mức độ tin cậy ngầm của xác nhận thông qua số xác nhận.

Khả năng đẩy lùi trạng thái giao dịch giảm theo thời gian do quan hệ tham chiếu hàm băm trong cấu trúc tài khoản. Như đã đề cập ở trên, khi thiết kế sổ cái có tính năng chống giả mạo tốt, việc đẩy lùi trạng thái giao dịch cần xây dựng lại tất cả khối giao dịch theo sau. Khi các giao dịch mới liên tục được thêm vào sổ cái, sẽ có ngày càng nhiều điểm nút nối tiếp trong một giao dịch, do đó khả năng giả mạo sẽ giảm đi.

Trong cấu trúc block-lattice, khi giao dịch được nhóm theo tài khoản, một giao dịch sẽ chỉ được gắn vào cuối chuỗi tài khoản giao dịch của tài khoản đó, và giao dịch tạo ra bởi hầu hết tài khoản khác sẽ không được tự động trở thành một điểm nút tiếp theo của giao dịch. Do đó, cần phải thiết kế một thuật toán đồng thuận để tránh những nguy cơ tiềm ẩn của double-spend.

Nano sử dụng một thuật toán đồng thuận dựa trên hình thức bỏ phiếu [10], giao dịch được ký bởi một tập hợp điểm nút đại diện được lựa chọn bởi một nhóm người dùng. Mỗi điểm nút đại diện có một trọng lượng. Khi chữ ký của một giao dịch có đủ trọng lượng, giao dịch được cho là đã được xác nhận. Trong thuật toán này, có một số vấn đề như sau:

Đầu tiên, nếu độ xác nhận cao hơn được yêu cầu, ngưỡng trọng lượng bỏ phiếu cần được nâng lên. Nếu không có đủ điểm nút đại diện online, tốc độ giao nhau không thể được đảm bảo, and và có thể một người dùng sẽ không bao giờ thu được số phiếu cần thiết để xác nhận một giao dịch;

Thứ hai, khả năng giao dịch bị đẩy lùi trạng thái không giảm theo thời gian. Bởi vì ở bất kỳ thời điểm nào, chi phí chiếm đoạt quyền bỏ phiếu lịch sử đều như nhau.

Cuối cùng, kết quả bỏ phiếu không được ghi lại trong sổ cái, mà chỉ được lưu trong các điểm nút cục bộ. Khi một điểm nút nhận tài khoản từ điểm nút khác, không có cách nào xác định chính xác khả năng giao dịch lịch sử bị đẩy lùi.

Về cơ bản, cơ chế bỏ phiếu là một giải pháp tập trung cục bộ. Chúng ta có thể coi kết quả bỏ phiếu như một ảnh chụp (snapshot) của trạng thái sổ cái. Snapshot này sẽ được phân bố trong lưu trữ cục bộ của mỗi điểm nút trong mạng lưới. Để có khả năng chống giả mạo như của blockchain, chúng ta có thể sắp xếp các snapshot này thành cấu trúc chuỗi – một trong những ý tưởng cốt lõi của thiết kế Vite – chuỗi snapshot [21].

3.2 Định nghĩa chuỗi snapshot

Chuỗi Snapshot là cấu trúc lưu trữ quan trọng nhất trong Vite. Chức năng chính của nó là duy trì đồng thuận của sổ cái Vite. Trước hết chúng tôi sẽ đưa ra định nghĩa về chuỗi snapshot:

Định nghĩa 3.2 (Khối snapshot và chuỗi snapshot) *một khối snapshot lưu trữ một snapshot trạng thái của sổ cái Vite, bao gồm số dư tài khoản, rễ Merkle của trạng thái hợp đồng và hàm băm của khối cuối cùng trong mỗi chuỗi tài khoản. Chuỗi snapshot là một cấu trúc chuỗi bao gồm các khối snapshot, chuỗi snapshot tiếp theo tham chiếu tới hàm băm của khối snapshot trước.*

Trạng thái tài khoản người dùng bao gồm số dư và hàm băm của khối trước trong chuỗi tài khoản; ngoài hai yếu tố trên, trạng thái tài khoản hợp đồng bao gồm hàm băm rễ Merkle. Cấu trúc trạng thái của một tài khoản như sau:

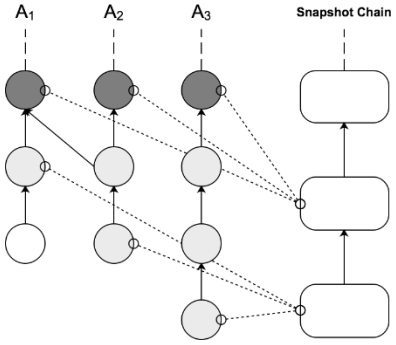
```
struct AccountState {
// account balance
map<uint32, uint256> balances;
//Merklerootofthecontractstate optional uint256storageRoot;
// hash of the last transaction
//oftheaccountchain uint256  lastTransaction;
}
```

Cấu trúc của khối snapshot được định nghĩa như sau:

```
struct SnapshotBlock {
//hashofthepreviousblock uint256 prevHash;
// snapshot information
map<address, AccountState>snapshot;
// signature uint256signature;
}
```

Để hỗ trợ nhiều token cùng một lúc, cấu trúc ghi lại thông tin số dư trong trạng thái tài khoản của Vite không phải là uint256, mà là một mapping từ ID của token đến số dư.

Khối snapshot đầu tiên trong chuỗi snapshot được gọi là "genesis snapshot", genesis snapshot có nhiệm vụ lưu lại các snapshot của khối genesis trong tài khoản.



Hình 6: Chuỗi snapshot

Do mỗi khối snapshot trong chuỗi đều có mối liên hệ với fork duy nhất của sổ cái Vite, ta có thể xác định được kết quả đồng thuận của sổ cái Vite bằng khối snapshot khi khối này không fork trong khối snapshot.

3.3 Xác nhận giao dịch và chuỗi snapshot

Sau khi giới thiệu chuỗi snapshot, các lỗ hổng bảo mật tự nhiên của cấu trúc mạng khối được khắc phục. Nếu kẻ tấn công muốn tạo giao dịch chi tiêu gấp đôi, ngoài việc xây dựng lại hash reference trong sổ cái Vite, nó còn cần phải được xây dựng lại trong chuỗi snapshot cho tất cả các khối sau khối snapshot đầu tiên của giao dịch và cần phải tạo một chuỗi snapshot dài hơn. Bằng cách này, chi phí tấn công sẽ tăng lên rất nhiều.

Trong Vite, cơ chế xác nhận giao dịch tương tự như Bitcoin,

được xác định như sau:

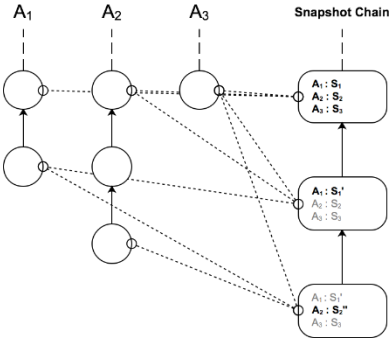
Định nghĩa 3.3 (Xác nhận giao dịch trong Vite) trong Vite, nếu một giao dịch được snapshot bởi chuỗi snapshot, giao dịch được xác nhận., Độ sâu của khối snapshot trong snapshot đầu tiên, được gọi là số xác nhận của giao dịch.

Theo định nghĩa này, số lượng các giao dịch được xác nhận sẽ tăng lên 1 khi chuỗi snapshot tăng lên và xác suất của một cuộc tấn công với giao dịch chi tiêu tăng gấp đôi sẽ giảm, cùng với đó là sự gia tăng của chuỗi snapshot. Bằng cách này, người dùng có thể tùy chỉnh số xác nhận yêu cầu bằng cách chờ các số xác nhận khác nhau theo tình huống cụ thể.

Bản thân chuỗi snapshot dựa trên thuật toán đồng thuận. Nếu chuỗi snapshot được fork, thì fork dài nhất sẽ được lựa chọn là fork hợp lệ. Khi chuỗi snapshot được chuyển sang một fork mới, thông tin của snapshot ban đầu sẽ được khôi phục, điều này có nghĩa là sự đồng thuận ban đầu trên sổ cái bị lật đổ và thay thế bởi sự đồng thuận mới. Do đó, chuỗi snapshot là nền tảng của toàn bộ hệ thống bảo mật và cần phải được xử lý nghiêm túc.

3.4 Bộ nhớ nén

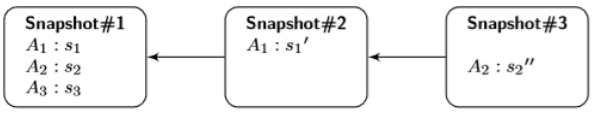
Tất cả các trạng thái tài khoản đều cần phải được lưu trong mỗi khối snapshot trong chuỗi snapshot, vì vậy, dung lượng lưu trữ rất lớn, việc nén các chuỗi snapshot là cần thiết.



Hình 7: Snapshot trước khi nén

Cách tiếp cận cơ bản của việc nén không gian lưu trữ chuỗi snapshot là sử dụng dung lượng gia tăng: một khối snapshot chỉ lưu trữ dữ liệu được thay đổi so với khối snapshot trước đó. Nếu không có giao dịch cho một tài khoản giữa hai snapshot, khối snapshot sau sẽ không lưu dữ liệu của tài khoản.

Để khôi phục thông tin snapshot, bạn có thể duyệt qua khối snapshot từ đầu đến cuối và bao gồm dữ liệu của mỗi khối snapshot theo dữ liệu hiện tại.



Hình 8: Snapshot sau khi nén

Chỉ trạng thái cuối cùng của mỗi snapshot của tài khoản được lưu khi snapshot, trạng thái trung gian sẽ không được tính đến, do đó, chỉ có một bản sao dữ liệu trong snapshot được lưu, bất kể số lượng giao dịch được tạo bởi một tài khoản giữa hai snapshot. Do đó, một khối snapshot chiếm tối đa S A byte. Trong số đó, S = sizeof (si), là số byte của mỗi trạng thái tài khoản, và A là tổng số tài khoản hệ thống. Nếu tỷ lệ trung bình của các tài khoản đang hoạt động đối với tổng số tài khoản là, tỷ lệ nén là 1 - a.

4 Đồng thuận

4.1 Mục tiêu thiết kế

Khi thiết kế một giao thức đồng thuận, chúng ta cần phải xem xét đầy đủ các yếu tố sau:

• Hiệu suất. Mục tiêu chính của Vite là nhanh. Để đảm bảo hệ thống có thông lượng cao và it chậm trễ, chúng ta cần phải áp dụng thuật toán đồng thuận với tốc độ hội tụ cao hơn.

• Khả năng mở rộng. Vite là một nền tảng công khai cho tất cả các ứng dụng phi tập trung, do đó khả năng mở rộng cũng là một yếu tố quan trọng.

• An toàn. Nguyên tắc thiết kế của Vite không theo đuổi sự an toàn tuyệt đối, tuy nhiên, nó vẫn cần phải đảm bảo đủ đường cơ sở an toàn và bảo vệ chống lại tất cả các loại tấn công có hiệu quả.

So với một số thuật toán đồng thuận hiện có, bảo mật của PoW tốt hơn, và có thể đạt được sự đồng thuận nếu sức mạnh tính toán của các node độc hại dưới 50%. Tuy nhiên, tốc độ giao nhau của PoW chậm và không đáp ứng được yêu cầu về hiệu suất; PoS và các thuật toán biến thể của nó loại bỏ các bước để giải quyết các vấn đề toán học, cải thiện tốc độ giao nhau và chi phí tấn công đơn, và giảm tiêu thụ năng lượng. Tuy nhiên, khả năng mở rộng của PoS vẫn còn kém, và vấn đề "Nothing at stake" "Stake trống" [22] khó giải quyết; thuật toán BFT có hiệu suất tốt hơn trong việc bảo mật và hiệu suất, nhưng Khả năng mở rộng của nó là một vấn đề, thường phù hợp hơn cho chuỗi tư nhân hoặc chuỗi liên kết; thuật toán DPoS [11] series có hiệu quả trong việc giảm khả năng phát sinh fork giả bằng cách hạn chế các quyền tạo khối. Hiệu suất và khả năng mở rộng tốt. Kết quả là, DPoS phải hi sinh một chút về vấn đề an ninh, và số lượng các node độc hại không được lớn hơn 1/3 [23].

Nói chung, thuật toán DPoS có lợi thế rõ ràng về hiệu suất và khả năng mở rộng. Vì vậy, chúng tôi chọn DPoS làm cơ sở cho giao thức đồng thuận Vite và mở rộng nó một cách chính xác trên chính cơ sở này. Thông qua giao thức đồng thuận được phân cấp theo thứ bậc và mô hình không đồng bộ, hiệu suất tổng thể của nền tảng có thể được cải thiện hơn nữa.

4.2 Sự đồng thuận phân cấp

Giao thức đồng thuận của Vite là HDPoS (Bằng chứng phân cấp theo cấp bậc của cổ phần). Ý tưởng cơ bản là phân tích hàm đồng thuận Φ (phân tích theo hàm):

$$\Phi(l_1, l_2, \ldots, l_n) = \Psi(\Lambda_1(l_1, l_2, \ldots, l_n), \Lambda_2(l_1, l_2, \ldots, l_n), \ldots \Lambda_m(l_1, l_2, \ldots, l_n)) \quad (5)$$

Λi: 2^L L, được gọi là hàm đồng thuận địa phương, kết quả trả về được gọi là sự đồng thuận địa phương; Ψ : 2L L, được gọi là hàm đồng thuận toàn cầu, nó lựa chọn một kết quả duy nhất từ một nhóm ứng cử viên trong sự đồng thuận địa phương làm kết quả đồng thuận cuối cùng.

Sau sự tách biệt này, sự đồng thuận của toàn bộ hệ thống đã trở thành hai quá trình độc lập:

Sự đồng thuận của địa phương tạo ra các khối tương ứng với các giao dịch truy vấn lại và giao dịch phản hồi trong tài khoản người dùng hoặc tài khoản hợp đồng và ghi vào sổ cái.

Sự đồng thuận toàn cầu snapshot dữ liệu trong sổ cái và các khối snapshot. Nếu sổ cái được fork, hãy chọn một trong số chúng.

4.3 Quyền của nhóm phát triển và hỗ trợ khối

Vậy, ai sẽ có quyền tạo khối giao dịch trong sổ cái và khối snapshot trong chuỗi snapshot? Thuật toán đồng thuận nào được chấp nhận để đạt được sự đồng thuận? Vì cấu trúc sổ cái của Vite được sắp xếp thành nhiều chuỗi tài khoản theo các tài khoản khác nhau, chúng tôi có thể xác định một cách thuận lợi cả quyền sản xuất của các khối trong sổ cái theo kích thước của tài khoản và quyền sản xuất của khối snapshot cho một nhóm người dùng. Bằng cách này, chúng ta có thể đặt một số lượng chuỗi tài khoản hoặc chuỗi snapshot vào nhóm đồng thuận, và trong nhóm đồng thuận, chúng ta có thể sử dụng một cách thống nhất để tạo ra khối và đạt được sự đồng thuận.

Khái niệm 4.1 (Nhóm đồng thuận) Nhóm đồng thuận là một bộ (L, U, Φ, P), mô tả cơ chế đồng thuận của một phần của tài khoản hoặc chuỗi snapshot., L A As, đại diện cho một hoặc một số chuỗi tài khoản, hoặc chuỗi snapshot của nhóm đồng thuận trong sổ cái; U đại diện cho người dùng với quyền sản xuất khối ngay trên chuỗi được chỉ định bởi L; Φ xác định thuật toán đồng thuận của nhóm đồng thuận; và P chỉ định các tham số của thuật toán đồng thuận.

Theo định nghĩa này, người dùng có thể thiết lập các nhóm đồng thuận một cách linh hoạt và chọn các thông số đồng thuận khác nhau tùy theo nhu cầu của họ. Tiếp theo, chúng tôi sẽ giải thích các nhóm đồng thuận khác nhau.

4.3.1 Nhóm đồng thuận của Snapshot

Nhóm đồng thuận các chuỗi snapshot được gọi là nhóm đồng thuận snapshot, là nhóm đồng thuận quan trọng nhất trong Vite. Thuật toán đồng thuận Φ của nhóm đồng thuận snapshot áp dụng thuật toán DPoS và tương ứng với Ψ trong mô hình phân cấp. Số lượng tác nhân và khoảng thời gian tạo khối được xác định bởi tham số P.

Ví dụ, chúng ta có thể chỉ định các nhóm đồng thuận snapshot với 25 proxy node để tạo ra các khối snapshot trong khoảng thời gian 1 giây. Điều này đảm bảo rằng giao dịch được xác nhận là đủ nhanh. Để đạt được xác nhận giao dịch 10 lần, cần phải chờ tối đa 10 giây.

4.3.2 Nhóm đồng thuận tư nhân

Nhóm đồng thuận tư nhân chỉ áp dụng cho việc xây dựng các khối giao dịch trong sổ cái, chúng thuộc về chuỗi tài khoản của nhóm đồng thuận tư nhân. Các khối chỉ có thể được tạo bởi chủ sở hữu khóa riêng của tài khoản. Theo mặc định, tất cả tài khoản người dùng thuộc nhóm đồng thuận riêng tư.

Ưu điểm lớn nhất của nhóm đồng thuận tư nhân là giảm xác suất fork. Do chỉ có một người dùng có quyền sản xuất các khối, khả năng duy nhất của fork là người dùng tự bắt đầu một cuộc tấn công với chi tiêu tăng gấp đôi hoặc điều này xảy ra do lỗi chương trình.

Nhược điểm của nhóm đồng thuận tư nhân là các node người dùng phải trực tuyến trước khi đóng gói giao dịch. Điều này không phù hợp với tài khoản hợp đồng. Khi node của chủ sở hữu không thành công, không có node nào khác có thể thay thế giao dịch phản hồi mà nó tạo hợp đồng, tương đương với việc giảm khả năng cung cấp dịch vụ của dApp.

4.3.3 Nhóm đồng thuận ủy thác

Trong nhóm đồng thuận ủy thác, thay vì tài khoản người dùng, một tập hợp các proxy node được sử dụng để đóng gói giao dịch qua thuật toán DPoS. Cả tài khoản người dùng và tài khoản hợp đồng có thể được thêm vào nhóm đồng thuận. Người dùng có thể thiết lập một tập hợp các node tác nhân riêng biệt và thiết lập một nhóm đồng thuận mới. Ngoài ra, còn có một nhóm đồng thuận mặc định ở Vite để giúp đóng gói các giao dịch cho tất cả các tài khoản chưa thiết lập nhóm đồng thuận ủy thác riêng của mình, hay còn được gọi là nhóm đồng thuận công cộng.

Nhóm đồng thuận ủy thác phù hợp với hầu hết các tài khoản hợp đồng, vì hầu hết các giao dịch trong tài khoản hợp đồng là các giao dịch phản hồi hợp đồng, trong đó độ sẵn có cao hơn và it chậm trễ hơn các giao dịch phải thu trong tài khoản người dùng.

4.4 Mức độ ưu tiên của sự đồng thuận

Trong giao thức Vite, ưu tiên đồng thuận toàn cầu có vị trí cao hơn đồng thuận của địa phương. Khi đồng thuận địa phương bị fork, kết quả của sự lựa chọn đồng thuận toàn cầu sẽ chiếm ưu thế. Nói cách khác, một khi đồng thuận toàn cầu chọn một fork của đồng thuận địa phương làm kết quả cuối cùng, thậm chí một fork dài hơn của một chuỗi tài khoản nhất định trong các tài khoản tương lai xảy ra, nó sẽ không khiến các kết quả toàn cầu quay trở lại. Vấn đề này cần được chú ý nhiều hơn khi thực hiện giao thức chuỗi chéo. Bởi một chuỗi mục tiêu có thể quay trở lại,

chuỗi tài khoản tương ứng của mapping hợp đồng chuyển tiếp chuỗi cũng cần phải quay lại. Tại thời điểm này, nếu sự đồng thuận địa phương của chuỗi chuyển tiếp đã được thông qua bởi sự đồng thuận toàn cầu, nó sẽ không thể khôi phục lại, điều này có thể gây ra sự không nhất quán giữa dữ liệu giữa hợp đồng chuyển tiếp và chuỗi mục tiêu.

Cách để tránh vấn đề này là thiết lập tham số *độ trễ* trong tham số nhóm đồng thuận P, trong đó xác định nhóm đồng thuận snapshot để có một snapshot riêng cho sự đồng thuận địa phương được hoàn thành sau các khối *trễ*. Điều này sẽ làm giảm đáng kể khả năng không nhất quán của các hợp đồng chuyển tiếp, nhưng ta không thể tránh được nó một cách hoàn toàn. Trong logic mã của các hợp đồng chuyển tiếp, việc đối phó với phục hồi của chuỗi đích một cách riêng biệt là việc cần phải làm.

4.5 Mô hình không đồng bộ

Để cải thiện thông lượng hệ thống hơn nữa, chúng ta cần hỗ trợ một mô hình không đồng bộ hoàn hảo hơn trên cơ chế đồng thuận.

Vòng đời của giao dịch bao gồm khởi tạo giao dịch, thực hiện giao dịch và xác nhận giao dịch. Để cải thiện hiệu suất của hệ thống, chúng ta cần thiết kế ba bước này theo chế độ không đồng bộ. Điều này là do vào các thời điểm khác nhau, số lượng giao dịch do người dùng khởi tạo khác nhau, tốc độ ghi giao dịch và xác nhận giao dịch được xử lý bởi hệ thống được cố định có tính tương đối. Chế độ không đồng bộ giúp làm phẳng các điểm đỉnh và đáy, từ đó cải thiện thông lượng tổng thể của hệ thống.

Mô hình không đồng bộ của Bitcoin và Ethereum rất đơn giản: giao dịch được khởi tạo bởi tất cả người dùng được đặt trong một nhóm chưa được xác nhận. Khi người đào coin gói chúng vào một khối, giao dịch được viết và xác nhận cùng một lúc. Khi chuỗi khối tiếp tục phát triển, giao dịch cuối cùng đạt đến mức độ tin cậy xác nhận đặt trước.

Có hai vấn đề trong mô hình không đồng bộ này:

• Các giao dịch không được tiếp tục đối với sổ cái trong trạng thái chưa được xác nhận. Giao dịch chưa được công nhận có tính không ổn định và không tồn tại sự đồng thuận có liên quan, không thể ngăn chặn việc gửi giao dịch nhiều lần.

• Không có cơ chế không đồng bộ để viết và xác nhận giao dịch. Các giao dịch chỉ được viết khi được xác nhận và tốc độ viết bị giới hạn bởi tốc độ xác nhận.

Giao thức Vite thiết lập mô hình đồng bộ cải tiến hơn: đầu tiên, giao dịch được chia thành một cặp chuyển đổi dựa trên mô hình "yêu cầu - phản hồi", cho dù đó là chuyển giao hay cuộc gọi hợp đồng, và giao dịch được khởi chạy thành công khi một giao dịch yêu cầu được ghi vào sổ cái. Ngoài ra, việc viết và xác nhận giao dịch cũng không đồng bộ. Các giao dịch có thể được ghi vào tài khoản DAG của Vite trước và sẽ không bị chặn bởi quá trình xác nhận. Xác nhận giao dịch được thực hiện thông qua chuỗi snapshot và hành động snapshot cũng không đồng bộ.

Đây là một mô hình tiêu dùng - sản xuất tiêu biểu. Trong vòng đời của giao dịch, bất kể tốc độ sản xuất thay đổi như thế nào ở thượng nguồn, hạ lưu có thể xử lý giao dịch với tốc độ không đổi, để tận dụng tối đa nền tảng tài nguyên và cải thiện thông lượng của hệ thống.

5 Máy ảo

5.1 Khả năng tương thích EVM

Hiện nay, có rất nhiều nhà phát triển trong lĩnh vực Ethereum, và nhiều hợp đồng thông minh được áp dụng dựa trên Solidity và EVM. Do đó, chúng tôi quyết định cung cấp khả năng tương thích EVM trên máy ảo Vite, và ngữ nghĩa gốc trong hầu hết các tập lệnh EVM được lưu giữ trong Vite. Do cấu trúc tài khoản và định nghĩa giao dịch của Vite khác với Ethereum, ngữ nghĩa của một số lệnh EVM cần phải được định nghĩa lại, ví dụ, một tập hợp các lệnh để lấy thông tin khối. Sự khác biệt ngữ nghĩa chi tiết có thể được tìm thấy trong phụ lục A.

Trong số đó, sự khác biệt lớn nhất là ngữ nghĩa của các cuộc gọi tin nhắn. Tiếp theo, chúng ta sẽ thảo luận chi tiết vấn đề này.

5.2 Theo dòng sự kiện

Trong giao thức Ethereum, một giao dịch hoặc tin nhắn có thể ảnh hưởng đến trạng thái của nhiều tài khoản. Ví dụ, một giao dịch yêu cầu hợp đồng có thể làm cho trạng thái của các tài khoản hợp đồng thay đổi cùng một lúc thông qua các cuộc gọi tin nhắn. Những thay đổi này xảy ra cùng một lúc hoặc không xảy ra. Do đó, giao dịch trong Ethereum thực chất là một loại giao dịch cứng nhắc đáp ứng các đặc điểm của ACID (Atomicity, Consistency, Isolation, Durability) [24], cũng là một lý do quan trọng cho việc thiếu tính mở rộng trong Ethereum.

Dựa trên những cân nhắc về khả năng mở rộng và hiệu suất, Vite đã thông qua một chương trình nhất quán cuối cùng thỏa mãn các ngữ nghĩa BASE (Basically Available, Soft state, Eventual consistency) [25]. Cụ thể, chúng tôi thiết kế Vite như một Kiến trúc theo hướng sự kiện (EDA) [26]. Mỗi hợp đồng thông minh được coi là một dịch vụ độc lập và có thể trao đổi tin nhắn giữa các hợp đồng, nhưng không có trạng thái nào được chia sẻ.

Do đó, trong EVM của Vite, chúng ta cần hủy bỏ ngữ nghĩa của các hàm cuộc gọi đồng bộ giữa các hợp đồng và chỉ cho phép liên lạc qua tin nhắn. Hướng dẫn EVM bị ảnh hưởng chủ yếu là CALL và STATICCALL. Trong Vite EVM, hai hướng dẫn này không thể được thực thi ngay lập tức, cũng như không thể trả lại kết quả của cuộc gọi. Chúng chỉ tạo ra một giao dịch yêu cầu để ghi vào sổ cái. Vì vậy, trong Vite, ngữ nghĩa của các hàm cuộc gọi sẽ không được bao gồm trong hướng dẫn này, mà sẽ gửi các tin nhắn đến một tài khoản.

5.3 Ngôn ngữ hợp đồng thông minh

Ethereum cung cấp một chương trình hoàn thiện về ngôn ngữ lập

trình Turing để phát triển các hợp đồng thông minh. Để hỗ trợ ngữ nghĩa không đồng bộ, chúng tôi mở rộng Solidity và xác định một bộ cú pháp cho liên lạc bằng tin nhắn. Solidity mở rộng được gọi là Solidity ++.

Hầu hết cú pháp của Solidity đều được hỗ trợ bởi So- lidity ++, nhưng không bao gồm các hàm cuộc gọi bên ngoài hợp đồng. Nhà phát triển có thể xác định tin nhắn thông qua từ khóa *message* và xác định bộ xử lý tin nhắn (MessageHandler) thông qua từ khóa *on* để thực hiện hàm giao tiếp chéo.

Ví dụ, hợp đồng A cần gọi phương thức add () trong hợp đồng B để cập nhật trạng thái của nó dựa trên giá trị trả về. Trong Solidity, điều này có thể được thực hiện bằng hàm cuộc gọi. Mã như sau:

```
pragma solidity^0.4.0; contract B {
function add(uint a, uint b) returns
(uint ret) {
return a + b;
}
}

contract A {
uint total;

functioninvoker(addressaddr,uinta, uint b) {
// message call to A.add() uintsum=B(addr).add(a,b);
//usethereturnvalue if (sum > 10){
total += sum;
}
}
}
```

Trong Solidity ++, hàm cuộc gọi codeuint sum = B (addr) .add (a, b); không còn giá trị; thay vào đó, hợp đồng A và hợp đồng B giao tiếp không đồng bộ bằng cách gửi tin nhắn cho nhau. Mã như sau:

```
pragmasolidity++^0.1.0; contract B{
message Add(uint a, uintb);
message Sum(uint sum);

Add.on {
// read message
uinta=msg.data.a; uintb=msg.data.b;
address sender =msg.sender;
// do things
uint sum = a +b;
//sendmessagetoreturnresult send(sender, Sum(sum));
}

}
```

```
contract A {
uint total;

functioninvoker(addressaddr,uinta, uint b) {
// message call to B send(addr, Add(a,b))
// you can do anything after sending
// a message other than using the
// return value
}
Sum.on {
//getreturndatafrommessage uint sum =msg.data.sum;
//usethereturndata if (sum > 10){
total += sum;
}
}
}
```

Trong dòng đầu tiên, mã pragma solidity ++ 0.1.0; cho thấy mã nguồn được viết bằng Solidity ++ nhưng sẽ không được biên dịch trực tiếp với trình biên dịch Solidity để tránh mã EVM đã biên dịch không phù hợp với ngữ nghĩa dự kiến. Vite sẽ cung cấp một trình biên dịch chuyên biệt để biên dịch Solidity ++. Trình biên dịch này tương thích một phần trước: nếu không có mã Solidity xung đột với ngữ nghĩa của Vite, nó có thể được biên dịch trực tiếp, nếu không thì lỗi sẽ được báo cáo. Ví dụ, cú pháp của các cuộc gọi hàm cục bộ vẫn tương thích khi chuyển sang các tài khoản khác; có được giá trị trả về của cuộc gọi hàm hợp đồng chéo, đơn vị tiền tệ ether cũng sẽ không được biên dịch.

Trong hợp đồng A, khi hàm invoker được gọi, tin nhắn Add sẽ được gửi đến hợp đồng B, điều này không đồng bộ và kết quả sẽ không được trả về ngay lập tức. Do đó, cần xác định bộ xử lý tin nhắn trong hợp đồng A bằng cách sử dụng từ khóa on để nhận kết quả trả về và cập nhật trạng thái.

Trong hợp đồng B, tin nhắn Add được giám sát. Sau khi xử lý, một tin nhắn Sum được gửi đến người gửi tin nhắn Add để trả lại kết quả.

Các tin nhắn trong Solidity ++ sẽ được biên dịch thành các lệnh CALL và một giao dịch yêu cầu sẽ được thêm vào sổ cái. Trong Vite, sổ cái dùng làm tin nhắn trung gian cho việc giao tiếp không đồng bộ giữa các hợp đồng. Sổ cái đảm bảo lưu trữ tin nhắn đáng tin cậy và ngăn chặn sự trùng lặp. Nhiều tin nhắn được gửi đến một hợp đồng bởi cùng một hợp đồng có thể đảm bảo FIFO (First In First Out); tin nhắn được gửi bởi các hợp đồng khác nhau cho cùng một hợp đồng không đảm bảo FIFO.

Cần lưu ý rằng các sự kiện trong Solidity (Event) và các tin nhắn trong Solidity ++ không cùng một concept. Các sự kiện được gửi gián tiếp tới phía trước thông qua nhật ký EVM.

5.4 Thư viện chuẩn

Các nhà phát triển các hợp đồng thông minh trên Ethereum thường bị ngăn cản bởi sự thiếu thốn các thư viện chuẩn trong Solidity. Ví dụ, xác minh vòng lặp trong giao thức Loopring phải được thực hiện bên ngoài chuỗi, một trong những lý do quan trọng là Solidity không cung cấp hàm dấu phẩy động, đặc biệt là n bình phương [1] [1] cho các số động.

Trong EVM, một hợp đồng được triển khai trước có thể được gọi bằng lệnh DELEGATECALL để nhận ra chức năng của hàm thư viện. Ethereum cũng cung cấp một số hợp đồng soạn sẵn, chủ yếu là một vài hoạt động Hash. Tuy nhiên, các hàm này quá đơn giản để đáp ứng nhu cầu ứng dụng phức tạp.

Do đó, chúng tôi sẽ cung cấp một loạt các thư viện chuẩn trong Solidity ++, chẳng hạn như xử lý chuỗi, các phép toán chứa dấu phẩy động, các phép toán cơ bản, các thùng chứa, sắp xếp, v.v.

Dựa trên những cân nhắc về hiệu suất, các thư viện chuẩn này sẽ được thực hiện theo cách mở rộng địa phương (Native Extension), và hầu hết các hoạt động được xây dựng trong mã Vite địa phương, và hàm chỉ được gọi thông qua lệnh DELEGATECALL trong mã EVM.

Thư viện chuẩn có thể được mở rộng khi cần thiết, nhưng vì mô hình máy trạng thái của toàn bộ hệ thống đã xác định, nó không thể cung cấp các hàm như số ngẫu nhiên. Tương tự như Ethereum, chúng ta có thể mô phỏng các số ngẫu nhiên giả thông qua hash của chuỗi snapshot.

5.5 Gas

Có hai chức năng chính của Gas trong Ethereum, chức năng đầu tiên là định lượng tài nguyên máy tính và tài nguyên lưu trữ được tiêu thụ bởi việc thực thi mã EVM, và thứ hai là đảm bảo rằng mã EVM được tạm dừng. Theo lý thuyết tính toán, vấn đề dừng trên máy Turing là một vấn đề không thể tính toán được [27]. Điều đó có nghĩa là, không thể xác định liệu một hợp đồng thông minh có thể được dừng lại sau khi việc thực hiện được hạn chế bằng cách phân tích mã EVM hay không.

Do đó, việc tính toán gas trong EVM cũng được giữ lại trong Vite. Tuy nhiên, không có khái niệm Giá Gas Trong Vite. Người dùng không mua gas để trao đổi bằng cách trả phí, mà thông qua một mô hình dựa trên hạn ngạch để có được tài nguyên máy tính. Việc tính toán hạn ngạch sẽ được thảo luận chi tiết trong chương "mô hình kinh tế".

6 Mô hình kinh tế

6.1 Token gốc

Để định lượng điện toán nền tảng, nguồn lưu trữ và khuyến khích các node chạy, Vite đã xây dựng một token gọi là ViteToken. Đơn vị cơ bản của token là *vite*, đơn vị nhỏ nhất là *attov*,,1 vite= 10^{18} attov.

Chuỗi snapshot là chìa khóa của việc bảo mật và hiệu suất của nền tảng Vite. Để kích động node tham gia vào quá trình xác minh giao dịch, giao thức Vite thiết lập phần thưởng giả mạo cho việc sản xuất khối snapshot. Các ưu đãi giả mạo sẽ làm tăng tính thanh khoản của ViteToken và làm giảm đi những lợi ích của các chủ sở hữu vite. Do đó, chúng tôi sẽ hạn chế lạm phát đến 3% mỗi năm.

Ngược lại, khi người dùng phát hành token mới, triển khai hợp đồng, đăng ký tên miền VNS 1 và nhận hạn ngạch tài nguyên, họ cần tiêu thụ hoặc thế chấp ViteToken để giảm thanh khoản.

Theo hành động kết hợp của hai yếu tố này, thanh khoản của vite có thể được duy trì ở mức độ lành mạnh, và đồng thời, điều này lợi cho việc tối ưu hóa việc phân bổ tài nguyên hệ thống.

6.2 Phân bổ nguồn lực

Vì Vite là một nền tảng dApp phổ biến, khả năng của các hợp đồng thông minh được triển khai trên chúng khác nhau, và mỗi hợp đồng thông minh khác nhau có những yêu cầu khác nhau về thông lượng và độ trễ. Ngay cả đối với cùng một hợp đồng thông minh, yêu cầu hiệu suất ở các giai đoạn khác nhau cũng khác nhau.

Trong thiết kế của Ethereum, mỗi giao dịch cần phải được ấn định giá gas khi tung ra để cạnh tranh với các giao dịch khác để ghi tài khoản. Đây là một mô hình đấu thầu điển hình, về nguyên tắc có thể kiểm soát hiệu quả sự cân bằng giữa cung và cầu. Tuy nhiên, người dùng rất khó định lượng được tình hình cung và cầu hiện tại, và không thể dự đoán giá của các nhà cung cấp khác, do đó, thị trường dễ dàng thất bại. Hơn nữa, các tài nguyên cạnh tranh cho mỗi giá thầu được hướng ngược với một giao dịch và không có thỏa thuận về việc phân bổ tài nguyên hợp lý theo độ rộng của tài khoản.

6.2.1 Tính toán hạn ngạch

Chúng tôi đã áp dụng giao thức phân bổ tài nguyên dựa trên hạn ngạch trong Vite, cho phép người dùng nhận được hạn ngạch tài nguyên cao hơn theo ba cách:

- Một PoW được tính toán khi khởi tạo giao dịch;
- Thế chấp một số tiền nhất định trong tài khoản vite;
- Để tiêu diệt một số lượng nhỏ của vite trong thời gian ngắn.

Các hạn ngạch cụ thể có thể được tính toán thông qua công thức sau:

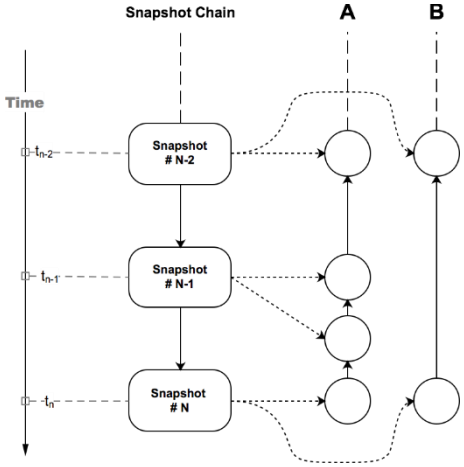
$$Q = Q_m \cdot \left(\frac{2}{1 + exp\left(-\rho \times \xi^{\top}\right)} - 1 \right) \qquad (6)$$

Trong số đó, Q_m là hằng số, đại diện cho giới hạn trên của một hạn ngạch tài khoản, có liên quan đến tổng thông lượng của hệ thống và tổng số tài khoản. $\xi = (\xi_d, \xi_s, \xi_f)$ là một vectơ đại diện cho chi phí của một người dùng để có được một nguồn tài nguyên: ξ_d là độ khó PoW mà người dùng tính toán khi tạo một giao dịch, ξ_s là số dư tài chính thế chấp vite trong tài khoản, và ξ_f là chi phí một lần mà người dùng sẵn sàng trả cho việc tăng hạn ngạch. Cần lưu ý rằng, ξ_f khác với phí xử lý. Những vite này sẽ bị phá hủy trực tiếp thay vì trả cho người đào coin.

Trong công thức, véc tơ $\rho = (\rho_d, \rho_s, \rho_f)$ đại diện cho trọng số của ba cách để đạt được hạn ngạch, nghĩa là, hạn ngạch thu được bằng việc hủy 1 vite tương đương với số tiền thế chấp ρ_s/ρ_f vite.
Có thể thấy từ công thức này rằng nếu người dùng không thế chấp vite hoặc không trả chi phí một lần, cần tính toán PoW, nếu không sẽ không có hạn ngạch để bắt đầu một giao dịch, việc này có hiệu quả ngăn chặn các cuộc tấn công bụi và bảo vệ tài nguyên hệ thống khỏi bị lạm dụng. Đồng thời, công thức này là một hàm Logistic. Người dùng có thể dễ dàng nhận được hạn ngạch thấp hơn, do đó giảm ngưỡng của người dùng tần suất thấp; và người dùng tần suất cao cần phải đầu tư nhiều tài nguyên để có được hạn ngạch cao hơn. Các chi phí bổ sung mà họ trả sẽ làm tăng lợi ích của tất cả người dùng.

6.2.2 Định lượng tài nguyên

Do chuỗi snapshot tương đương với đồng hồ toàn cầu, chúng tôi có thể sử dụng nó để định lượng tài nguyên sử dụng tài khoản một cách chính xác. Trong mỗi giao dịch, Hash của một khối snapshot được đưa ra, chiều cao của khối snapshot được lấy làm dấu thời gian của giao dịch. Do đó, theo sự khác biệt giữa hai dấu thời gian giao dịch, chúng tôi có thể đánh giá liệu khoảng thời gian giữa hai giao dịch có đủ dài hay không.



Hình 9: chuỗi snapshot tương tự như một chiếc đồng hồ toàn cầu

Như đã trình bày ở trên, tài khoản A tạo ra 4 giao dịch trong 2 khoảng thời gian, trong khi tài khoản B chỉ tạo ra 2 giao dịch. Do đó, TPS trung bình của A trong giai đoạn này gấp 2 lần so với B. Nếu đó chỉ là một giao dịch chuyển khoản, TPS trung bình của tài khoản được định lượng là đủ. Đối với các hợp đồng thông minh, mỗi trao đổi có mức tiêu thụ tài nguyên khác nhau, vì vậy cần phải tích lũy gas cho mỗi giao dịch để tính toán mức tiêu thụ tài nguyên trung bình trong một khoảng thời gian. Mức tiêu thụ tài nguyên trung bình của các giao dịch k gần đây trong chuỗi tài khoản có chiều cao "n" là:

$$Cost_k(T_n) = \frac{k \cdot \sum_{i=n-k+1}^{n} gas_i}{timestamp_n - timestamp_{n-k+1} + 1} \qquad (7)$$

Trong số đó, đối với một giao dịch T_n, $timestamp_n$ là dấu thời gian của giao dịch, tức là chiều cao của khối snapshot nó đề cập đến; gas_n là nhiên liệu tiêu thụ cho giao dịch.
Khi xác minh giao dịch, node sẽ xác định liệu hạn ngạch có thỏa mãn điều kiện: Chi phí (T) Q và nếu không thỏa mãn, giao dịch sẽ bị từ chối. Trong trường hợp này, người dùng cần đóng gói lại một giao dịch, tăng hạn ngạch bằng cách trả một khoản phí một lần hoặc chờ một khoảng thời gian để báo giá một snapshot cao hơn trong giao dịch.

6.2.3 Cho thuê hạn ngạch

Nếu người dùng nắm giữ tài sản vite phong phú, nhưng không cần sử dụng nhiều hạn ngạch tài nguyên, họ có thể cho người dùng khác thuê hạn ngạch của mình.
Hệ thống Vite hỗ trợ một loại giao dịch đặc biệt để chuyển quyền sử dụng hạn ngạch tài nguyên tài khoản. Trong giao dịch này, số lượng vite có thể được thế chấp, địa chỉ của người được chuyển nhượng và thời hạn thuê có thể được xác định. Khi giao dịch được xác nhận, hạn ngạch tài nguyên tương ứng với số lượng token sẽ được bao gồm trong tài khoản của người được chuyển nhượng. Khi thời gian thuê vượt quá, hạn ngạch sẽ được tính vào tài khoản của người chuyển khoản. Đơn vị thời gian cho thuê là giây. Hệ thống sẽ được chuyển đổi thành độ chênh lệch chiều cao của khối snapshot, do đó có thể có một chút chênh lệch.
Thu nhập cho thuê có thể thu được bởi người sử dụng. Hệ thống Vite chỉ cung cấp một giao dịch chuyển nhượng hạn ngạch, việc định giá và thanh toán tiền thuê có thể đạt được thông qua hợp đồng thông minh của bên thứ ba ..

6.3 Đảm bảo tài sản

Ngoài token gốc được gọi là ViteToken, Vite cũng hỗ trợ người dùng phát hành token của họ. Việc phát hành của token có thể được thực hiện thông qua một giao dịch đặc biệt gọi là giao dịch Mint. Địa chỉ đích của giao dịch Mint là 0. Trong *dữ liệu* của giao dịch, các tham số của token được chỉ định như sau:
Mint: {
name: "MyToken",
totalSupply: 999999999000000000000000000,
decimals: 18,
owner:"0xa3c1f4...fa", symbol: "MYT"
}
Sau khi yêu cầu được mạng chấp nhận, vite nằm trong giao dịch mint sẽ được khấu trừ từ tài khoản người khởi tạo như phí giao dịch. Hệ thống ghi nhận lại thông tin của loại token mới và gán cho nó một token_id . Tất cả số dư các token mới được tạo sẽ được thêm vào địa chỉ chủ sở hữu , có nghĩa là tài khoản chủ sở hữu là tài khoản gốc của token đó.

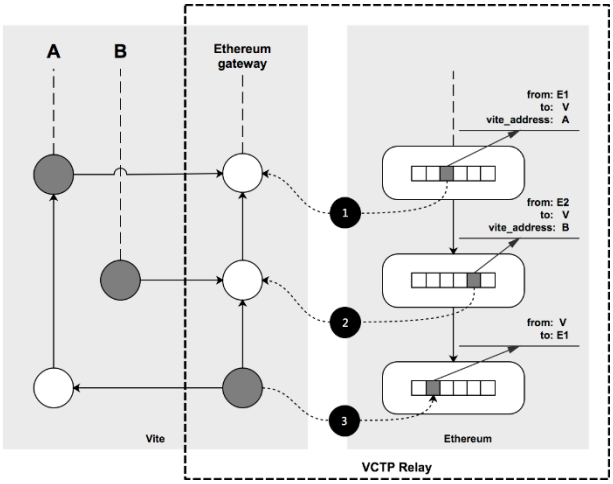
6.4 Giao thức liên chuỗi

Để hỗ trợ chuyển giao giá trị trên nhiều chuỗi tài sản kỹ thuật số và loại bỏ "đảo giá trị", Vite thiết kế một giao thức chuyển giao liên chuỗi (VCTP).

Đối với mỗi tài sản cần di chuyển qua nhiều chuỗi trên chuỗi mục tiêu, một token tương ứng với nó là cần thiết trong Vite được gọi là ToT (Token of Token). Ví dụ, nếu bạn muốn chuyển *ether* trong tài khoản Ethereum đến Vite, bạn có thể đặt ToT với xác minh *ETH* trong Vite, số lượng TOT ban đầu phải bằng nhau với tổng số lượng *ETH*.

Đối với mỗi chuỗi mục tiêu, có một Hợp đồng gateway trên Vite để duy trì mối quan hệ mapping giữa các giao dịch Vite và các giao dịch chuỗi đích. Trong nhóm đồng thuận nơi đặt hợp đồng, nút chịu trách nhiệm tạo khối được gọi là VCTP Relay. VCTP Relay cần phải cùng một lúc là nút Vite và nút đầy đủ của chuỗi đích, và lắng nghe các giao dịch trên cả hai mặt. Trên chuỗi mục tiêu, chúng ta cũng cần triển khai một hợp đồng Vite Gateway.

Trước khi VCTP Relay bắt đầu hoạt động, ToT trong Vite tương ứng sẽ được chuyển sang hợp đồng gateway. Sau đó, việc cung cấp ToT chỉ có thể được kiểm soát bởi hợp đồng gateway, và không ai được thêm vào để đảm bảo 1: 1 tỷ lệ trao đổi giữa ToT và tài sản đích. Đồng thời, các tài sản trên chuỗi đích được kiểm soát bởi hợp đồng Vite gateway và không người dùng nào có thể sử dụng nó, để đảm bảo rằng ToT có toàn bộ dự trữ chấp nhận.



Bảng 10: Giao thức liên chuỗi

Hình trên là ví dụ về sự chuyển giao giá trị liên chuỗi giữa Vite và Ethereum. Khi người dùng Ethereum E1 muốn chuyển token từ Ethereum sang Vite, họ có thể gửi giao dịch đến địa chỉ hợp đồng cổng Vite V , khi địa chỉ của người dùng Một trên Vite được đặt trong tham số. Số dư của vụ chuyển khoản sẽ bị khóa trong tài khoản hợp đồng gateway và trở thành một phần của dự trữ ToT. Sau khi nghe giao dịch, nút chuyển tiếp VCTP tạo tài khoản tương ứng trên giao dịch Vite, gửi cùng một lượng ToT đến tài khoản của người dùng A trong Vite. Trong các Vite là chứng minh của Token mục tiêu lưu hành trong hình ảnh ① và ② lần lượt có nghĩa là E1 and E2 Vite được gọi là ToT (Token of Token). Ví dụ, nếu bạn muốn chuyển *ether* trong tài khoản Ethereum đến Vite, bạn có thể đặt ToT với xác minh chuyển sang tài khoản Vite A và B. Cần lưu ý rằng nếu người dùng không chỉ định địa chỉ Vite khi chuyển, hợp đồng sẽ từ chối giao dịch.

Luồng ngược lại được hiển thị trong ③, Khi người dùng A bắt đầu chuyển từ tài khoản Vite sang tài khoản Ethereum, giao dịch sẽ được gửi đến hợp đồng Vite gateway, chuyển đến một số lượng nhất định ToT và chỉ định địa chỉ nhận E 1 Ethereum trong giao dịch. Nút VCTP relay sẽ tạo ra khối phản hồi tương ứng trên hợp đồng Ethereum Gateway và gói một giao dịch của Ethereum với hợp đồng cổng Vite trên Ethereum. Trong Ethereum, hợp đồng Vite gateway sẽ xác minh xem giao dịch này có được bắt đầu bằng VCTP relay đáng tin cậy hay không

Tất cả các nút relay liên chuỗi sẽ giám sát mạng mục tiêu, và họ có thể xác minh xem mỗi hoạt động liên chuỗi có chính xác hay không và đạt được sự đồng thuận trong nhóm đồng thuận. Nhưng nhóm đồng thuận snapshot sẽ không giám sát giao dịch của chuỗi mục tiêu, cũng như nó sẽ xác minh liệu việc mapping giữa hai chuỗi có chính xác hay không. Nếu mạng mục tiêu khôi phục hoặc xảy ra hard fork, các giao dịch được xác nhận trong hệ thống Vite không thể được phục hồi; tương tự, nếu các giao dịch liên chuỗi trong Vite được khôi phục, giao dịch tương ứng của mạng mục tiêu không thể được khôi phục cùng một lúc. Vì vậy, khi thực hiện các giao dịch xuyên chuỗi, cần phải xử lý việc khôi phục giao dịch trong logic hợp đồng. Đồng thời, như được mô tả trong phần 4.4, chúng ta cần đặt thông số *trì hoãn* cho nhóm đồng thuận Relay chuỗi chéo.

6.5 Giao thức Loopring

Giao thức Loopring [1] là một giao thức mở để xây dựng mạng giao dịch tài sản phi tập trung. So với các giải pháp DEX khác, giao thức Loopring dựa trên kết hợp vòng lặp đa, cung cấp kỹ thuật ủy quyền kép để ngăn chặn các giao dịch ưu tiên và hoàn toàn mở.

Chúng tôi xây dựng giao thức Loopring trong Vite, điều này có lợi cho việc thúc đẩy lưu thông tài sản kỹ thuật số trong Vite, để toàn bộ hệ thống giá trị có thể được lưu hành. Trong hệ thống giá trị này, người dùng có thể phát hành các tài sản số của riêng họ, chuyển các tài sản bên ngoài chuỗi thông qua VCTP và sử dụng giao thức Loopring để hoàn thành được trao đổi tài sản. Toàn bộ quá trình có thể được hoàn thành trong hệ thống Vite và hoàn toàn phi tập trung.

Trong Vite, hợp đồng thông minh Loopring Protocol (LPSC) là một phần của hệ thống Vite. Ủy quyền chuyển giao tài sản và bảo vệ nguyên tử đa bên đều được hỗ trợ trong Vite. Loopring relay vẫn mở để tích hợp hoàn toàn với hệ sinh thái của chính nó.

Người dùng có thể sử dụng *vite* để thanh toán cho giao dịch trao đổi tài sản, do đó, token kiếm được bởi các miner của Looping thực hiện kết hợp vòng lặp trong nền tảng Vite vẫn là *vite*

7 Những thiết kế khác

7.1 Quy trình

Trong Ethereum, các hợp đồng thông minh bị điều khiển bởi các giao dịch và việc thực hiện các hợp đồng chỉ có thể được kích hoạt khi người dùng bắt đầu một giao dịch. Trong một số ứng dụng, cần có chức năng lập lịch thời gian để kích hoạt việc thực hiện hợp đồng thông qua đồng hồ.

Trong Ethereum, chức năng này đạt được thông qua hợp đồng của bên thứ ba. 1 , hiệu suất và bảo mật không được đảm bảo. Trong Vite, chúng tôi thêm hàm lập lịch thời gian vào hợp đồng được tạo sẵn. Người dùng có thể đăng ký lịch biểu của họ vào hợp đồng được hẹn giờ. Nhóm đồng thuận công khai sẽ sử dụng chuỗi snapshot làm đồng hồ và gửi giao dịch yêu cầu đến hợp đồng đích theo lịch biểu do người dùng xác định.

Có thông báo Hẹn giờ chuyên biệt trong Solidity ++. Người dùng có thể thiết lập lịch biểu của riêng mình trong mã hợp đồng thông qua Timer.on.

7.2 Dịch vụ đặt tên

Trong Ethereum, hợp đồng sẽ tạo ra một địa chỉ để xác định hợp đồng khi nó được triển khai. Có hai vấn đề trong việc xác định hợp đồng với địa chỉ:

Địa chỉ là số nhận dạng 20 byte không có ý nghĩa. Nó không thân thiện với người dùng và bất tiện khi sử dụng.

Hợp đồng và địa chỉ là một đối một. Chúng không thể cung cấp điều hướng lại hợp đồng.

Để giải quyết hai vấn đề này, nhà phát triển Ethereum đã cung cấp hợp đồng của bên thứ ba ENS 2 . Tuy nhiên, trong kịch bản thực tế, việc sử dụng các dịch vụ đặt tên sẽ rất thường xuyên, và việc sử dụng hợp đồng của bên thứ ba không thể đảm bảo tính duy nhất toàn cầu của việc đặt tên, vì vậy chúng tôi sẽ xây dựng một dịch vụ tên VNS (ViteName Service) tại Vite. Người dùng có thể đăng ký một bộ tên dễ nhớ và đưa chúng đến địa chỉ thực tế thông qua VNS. Tên được sắp xếp dưới dạng tên miền, chẳng hạn như *vite.myname.mycontract* . Tên miền cấp cao nhất sẽ được hệ thống giữ lại cho các mục đích cụ thể. Ví dụ: *vite.xx* đại diện cho địa chỉ Vite và *eth.xx* đại diện cho một địa chỉ Ethereum. Tên miền thứ cấp được mở cho tất cả người dùng. Khi người dùng sở hữu tên miền thứ cấp, tên miền phụ có thể được mở rộng tùy ý. Chủ sở hữu tên miền có thể sửa đổi địa chỉ được chỉ dẫn bởi tên miền bất kỳ lúc nào, vì vậy chức năng này có thể được sử dụng cho hợp đồng nâng cấp

Độ dài của tên miền không bị hạn chế. Trong VNS, băm của tên miền thực sự được lưu trữ. Địa chỉ đích có thể là một địa chỉ không phải là Vite nhỏ hơn 256 bit, có thể được sử dụng cho tương tác liên chuỗi.

Cần lưu ý rằng VNS khác với hợp đồng thông minh Gói đặc tả EIP190 3 trong Ethereum. VNS là một dịch vụ phân giải tên, tên được thiết lập trong thời gian chạy, và các quy tắc giải quyết có thể được điều chỉnh; và EIP190 là một đặc tả quản lý gói, không gian tên có đặc tính tĩnh, và nó được thiết lập vào thời gian tổng hợp.

7.3 Cập nhật hợp đồng

Hợp đồng thông minh của Ethereum là bất biến. Sau khi triển khai, nó không thể được sửa đổi. Ngay cả khi có lỗi trong hợp đồng, nó không thể được cập nhật. Điều này rất không thân thiện với các nhà phát triển và làm cho việc lặp lại liên tục DApp rất khó khăn. Do đó, Vite cần cung cấp một kế hoạch để hỗ trợ cập nhật hợp đồng thông minh.

Tại Vite, quá trình cập nhật hợp đồng bao gồm:

A. Triển khai phiên bản mới của hợp đồng để kế thừa trạng thái của hợp đồng gốc.

B. Ghi tên hợp đồng vào địa chỉ mới trong VNS.

Ba bước này cần được hoàn thành cùng một lúc và giao thức Vite đảm bảo số nguyên tử số của hoạt động. Nhà phát triển cần đảm bảo rằng dữ liệu hợp đồng cũ được xử lý chính xác trong hợp đồng phiên bản mới.

Cần lưu ý rằng hợp đồng mới sẽ không kế thừa địa chỉ của hợp đồng cũ. Nếu được trích dẫn bởi địa chỉ, giao dịch sẽ vẫn được gửi đến hợp đồng cũ. Điều này là do các phiên bản khác nhau của hợp đồng cơ bản là hai hợp đồng hoàn toàn khác nhau, cho dù chúng có thể được sửa đổi hay không, tùy thuộc vào ngữ nghĩa của hợp đồng. Trong các hệ thống Vite, các hợp đồng thông minh được chia thành hai loại, cái đầu tiên là nền tảng dApp, và logic nghiệp vụ của nó được mô tả; và thứ hai là một loại hợp đồng lập bản đồ thế giới thực. Phiên bản trước đó tương đương với dịch vụ nền của ứng dụng, dịch vụ này cần được lặp lại liên tục thông qua nâng cấp; cái sau tương đương với một hợp đồng, và một khi nó có hiệu lực, không thể sửa đổi được, nếu không đó được coi là một vi phạm hợp đồng. Đối với một hợp đồng không được phép sửa đổi, hợp đồng đó có thể được trang trí bằng từ khóa *tĩnh* trong
Solidity++, ví dụ:

```
pragma solidity++ ^0.1.0;

cam kết hợp đồng tĩnh {
// hợp đồng không bao giờ thay đổi
}
```

7.4 Cắt khối

Trong sổ kế toán, mọi giao dịch đều không thay đổi và người dùng chỉ có thể thêm giao dịch mới vào sổ kế toán mà không thay đổi hoặc xóa giao dịch trong lịch sử. Do đó, với hoạt động của hệ thống, sổ cái sẽ ngày càng trở nên lớn hơn. Nếu một nút mới tham gia mạng muốn khôi phục trạng thái mới nhất, bắt đầu từ khối gốc và làm lại tất cả các giao dịch lịch sử. Sau khi chạy hệ thống trong một khoảng thời gian, không gian bị chiếm dụng bởi sổ kế toán và thời gian được sử dụng để thực hiện lại các giao dịch sẽ trở thành không thể chấp nhận được. Đối với hệ thống thông lượng cao của Vite, tốc độ tăng trưởng sẽ cao hơn nhiều so với Bitcoin và Ethereum, do đó, cần phải cung thiết kế kỹ thuật để cắt các khối trong sổ cái.

Cắt khối liên quan đến việc xóa các giao dịch lịch sử không thể được sử dụng trong sổ cái, và không ảnh hưởng đến hoạt động của máy trạng thái dùng trong giao dịch. Vậy các giao dịch nào có thể được xóa an toàn? Điều này phụ thuộc vào bối cảnh giao dịch sẽ được sử dụng, bao gồm:

Khôi phục. Vai trò chính của giao dịch là khôi phục trạng thái. Bởi vì trong Vite, chuỗi snapshot lưu trữ thông tin về trạng thái tài khoản, các nút có thể khôi phục trạng thái từ một khối snapshot. Tất cả các giao dịch trước *giao dịch cuối cùng* trong khối snapshot có thể được điều chỉnh để khôi phục trạng thái.

Xác minh giao dịch . Để xác minh giao dịch mới, cần xác minh giao dịch trước đó của giao dịch trong chuỗi tài khoản và nếu đó là giao dịch phản hồi, nó cũng cần phải xác minh giao dịch yêu cầu tương ứng. Do đó, trong sổ kế toán được điều chỉnh phù hợp, ít nhất một giao dịch cuối cùng sẽ được giữ lại trong mỗi chuỗi tài khoản. Ngoài ra, tất cả các giao dịch yêu cầu mở không thể được điều chỉnh vì băm của chúng có thể được tham chiếu bởi các giao dịch phản hồi tiếp theo.

Tính toán hạn ngạch. Liệu giao dịch có đáp ứng hạn ngạch được tính bằng cách đánh giá mức trung bình trượt của 10 tài nguyên giao dịch cuối cùng, vì vậy ít nhất 9 giao dịch cuối cùng cần được lưu trên mỗi chuỗi tài khoản.

Hỏi về lịch sử . Nếu nút cần truy vấn lịch sử giao dịch, giao dịch liên quan đến truy vấn sẽ không được điều chỉnh.

Theo các kịch bản sử dụng khác nhau, mỗi nút có thể chọn một số kết hợp từ phân đoạn cắt phía trên. Điều quan trọng cần lưu ý là việc cắt liên quan đến việc chuyển đổi trong sổ cái, trong khi các chuỗi chụp nhanh cần được giữ nguyên vẹn. Ngoài ra, những gì được ghi trong chuỗi snapshot là băm của trạng thái hợp đồng. Khi tài khoản bị cắt bớt, trạng thái tương ứng của snapshot cần được giữ nguyên.

Để đảm bảo tính toàn vẹn của dữ liệu Vite, chúng tôi cần giữ lại một số "Nút đầy đủ" trong mạng để lưu tất cả dữ liệu giao dịch. Các nút snapshot nhóm đồng thuận là các nút đầy đủ và ngoài ra, những người dùng quan trọng như trao đổi cũng có thể trở thành các nút đầy đủ.

8 Quản lí

Đối với một nền tảng ứng dụng phi tập trung, một hệ thống quản lý hiệu quả là điều cần thiết để duy trì một hệ sinh thái lành mạnh. Tính hiệu quả và công bằng cần được xem xét khi thiết kế hệ thống quản trị.

Hệ thống quản trị của Vite được chia thành hai phần: chuỗi chính và chuỗi phụ. On-chain là một cơ chế bỏ phiếu dựa trên giao thức, và off-chain là sự lặp lại của chính giao thức đó.

Trên cơ chế bỏ phiếu, nó được chia thành hai loại: Bỏ phiếu toàn cầu và bỏ phiếu địa phương. Việc bỏ phiếu toàn cầu được dựa trên *vite* được người dùng nắm giữ để tính các quyền như trọng số biểu quyết. Bỏ phiếu toàn cầu chủ yếu được sử dụng cho việc bầu chọn nút ủy quyền nhóm snapshot. Bỏ phiếu địa phương là nhằm vào một hợp đồng. Khi hợp đồng được triển khai, một token được chỉ định làm cơ sở để bỏ phiếu. Nó có thể được sử dụng để chọn các nút đại lý của nhóm đồng thuận trong đó hợp đồng được đặt.

Bên cạnh việc xác minh giao dịch, nút tác nhân của nhóm đồng thuận snapshot có quyền chọn có nâng cấp tính không tương thích của hệ thống Vite hay không. Nút ủy quyền của nhóm đồng thuận được ủy quyền có quyền quyết định liệu có cho phép hợp đồng được nâng cấp để tránh rủi ro tiềm ẩn phát sinh từ việc leo thang hợp đồng hay không. Nút tác nhân được sử dụng để nâng cấp sức mạnh ra quyết định thay mặt cho người dùng nhằm nâng cao hiệu quả của việc ra quyết định và tránh thất bại trong việc ra quyết định do không đủ người bỏ phiếu. Các nút ủy quyền này cũng bị hạn chế bởi giao thức đồng thuận. Chỉ khi hầu hết các nút tác nhân 1 được chuyển, thì bản nâng cấp sẽ có hiệu lực. Nếu các tác nhân này không đáp ứng được quyền quyết định của họ theo mong đợi của người dùng, người dùng cũng có thể hủy bỏ ủy quyền của mình bằng cách bỏ phiếu.

Việc quản lý chuỗi ngoại tuyến được thực hiện bởi cộng đồng. Bất kỳ người tham gia cộng đồng Vite nào đều có thể đề xuất một kế hoạch cải tiến cho chính giao thức Vite hoặc các hệ thống liên quan, được gọi là VEP (Đề xuất Tăng cường Vite). VEP có thể được thảo luận rộng rãi trong cộng đồng và quyết định thực hiện giải pháp được thực hiện bởi những người tham gia trong hệ sinh thái Vite. Cho dù giao thức sẽ được nâng cấp để thực hiện VEP sẽ được quyết định cuối cùng bởi nút tác nhân hay không. Tất nhiên, khi có sự khác biệt lớn, bạn cũng có thể bắt đầu một vòng bỏ phiếu trên chuỗi để thu thập một loạt các ý kiến người dùng và nút ủy quyền sẽ quyết định có nên nâng cấp theo kết quả bỏ phiếu hay không.

Mặc dù một số người tham gia Hội nghị có thể không có đủ *token* vite để bỏ phiếu cho ý kiến của họ. Nhưng họ có thể tự do gửi VEP và thể hiện đầy đủ quan điểm của họ. Những người sử dụng có quyền bỏ phiếu phải có đầy đủ minh chứng về mức độ khỏe mạnh của toàn bộ hệ sinh thái vì quyền lợi riêng của họ, và do đó xem xét nghiêm túc tất cả những người tham gia.

9 Việc cần làm trong tương lai

Xác minh giao dịch trên các chuỗi snapchat là một thách thức về hiệu suất lớn của hệ thống. Bởi vì Vite sử dụng thiết kế không đồng bộ và cấu trúc tài khoản DAG, việc xác thực giao dịch có thể được thực thi song song. Tuy nhiên, do sự phụ thuộc giữa các giao dịch của các tài khoản khác nhau, mức độ song song sẽ bị hạn chế rất nhiều. Làm thế nào để cải thiện tính song song của xác minh giao dịch hoặc áp dụng một chiến lược xác minh phân phối sẽ là một hướng quan trọng để tối ưu hóa trong tương lai.

Một số thiếu sót tồn tại trong thuật toán điều khiển HDPoS hiện tại. Nó cũng là một hướng tối ưu hóa để cải thiện thuật toán đồng thuận, hoặc để tương thích với các thuật toán đồng thuận hơn trong nhóm đồng thuận được giao.

Ngoài ra, việc tối ưu hóa máy ảo cũng rất quan trọng để giảm sự chậm trễ của hệ thống và cải thiện thông lượng hệ thống. Do thiết kế đơn giản của EVM và đơn giản hóa tập lệnh, có thể cần phải thiết kế một máy ảo mạnh hơn trong tương lai và xác định một ngôn ngữ lập trình hợp đồng thông minh với nhiều khả năng mô tả hơn, và ít các lỗ hổng bảo mật hơn.

Cuối cùng, bên cạnh thỏa thuận cốt lõi của Vite, việc xây dựng các cơ sở phụ trợ hỗ trợ phát triển hệ sinh thái cũng là một chủ đề quan trọng. Ngoài hỗ trợ SDK cho các nhà phát triển dApp, có nhiều việc phải làm trong việc xây dựng hệ sinh thái tiền cảnh dApp. Ví dụ, bạn có thể xây dựng một công cụ dApplet dựa trên HTML5 trong ứng dụng ví di động của Vite, cho phép các nhà phát triển phát triển và xuất bản dApp với chi phí thấp.

10 Tóm tắt

So với các dự án tương tự khác, đặc điểm của Vite bao gồm:

Thông lượng cao .Vite sử dụng cấu trúc sổ kế toán DAG, giao dịch trực giao có thể được viết theo mệnh đề của cuốn sách; ngoài ra, nhiều nhóm đồng thuận không phụ thuộc vào nhau trong thuật toán đồng thuận HDPoS, và có thể làm việc song song; điều quan trọng nhất là giao tiếp hợp đồng giữa các bên của Vite dựa trên mô hình không đồng bộ của thông điệp. Tất cả những tính năng này đều hữu ích để cải thiện thông lượng của hệ thống.

Độ trễ thấp .Vite sử dụng thuật toán đồng thuận HDPoS để cộng tác để hoàn thành khối sản xuất luân phiên thông qua nút proxy, mà không cần tính toán PoW, khoảng thời gian có thể giảm xuống 1 giây, có lợi cho giảm sự chậm trễ của xác nhận giao dịch.

Khả năng mở rộng . Để đáp ứng các yêu cầu về khả năng mở rộng, Vite tạo ra một mức độ giới hạn tự do duy nhất trên giao dịch, nhóm các giao dịch trong ac- count theo kích thước tài khoản, cho phép sản xuất khối khác nhau các tài khoản sẽ được hoàn thành bởi các nút khác nhau và để loại bỏ ngữ nghĩa ACID của các cuộc gọi hợp đồng chéo đến ngữ nghĩa BASE dựa trên thông báo. Bằng cách này, các nút không còn cần lưu tất cả trạng thái của thế giới nữa và dữ liệu được lưu trong toàn bộ mạng được phân phối theo cơ chế sharding

Tính khả dụng .Các cải tiến về khả năng sử dụng của Vite bao gồm việc cung cấp hỗ trợ thư viện chuẩn trong Solid ++, dành riêng cho xử lý cú pháp thư, lập lịch thời gian hợp đồng, dịch vụ đặt tên VNS, hỗ trợ nâng cấp hợp đồng, v.v.

Lưu thông giá trị . Hỗ trợ tài sản kỹ thuật số, chuyển giao giá trị xuyên chuỗi, trao đổi token dựa trên giao thức Loopring, v.v., tạo thành một hệ thống giá trị hoàn chỉnh. Từ quan điểm của người dùng, Vite là một sàn giao dịch phân quyền đầy đủ chức năng.

Kinh tế . Bởi vì Vite sử dụng mô hình phân bổ nguồn lực dựa trên hạn ngạch, người dùng không thường xuyên giao dịch không phải trả phí cao hoặc phí gas. Người dùng có thể chọn nhiều cách để thay đổi phép tính. Hạn ngạch bổ sung cũng có thể được chuyển giao cho người dùng khác thông qua thỏa thuận cho thuê hạn ngạch để nâng cao hiệu quả sử dụng tài nguyên hệ thống.

---

1 theo giao thức DPoS, phần lớn hợp lệ là 2/3 tổng số nút tác nhân.

11 Lời cảm ơn

Chúng tôi xin chân thành cảm ơn các nhà tư vấn của chúng tôi đã hướng dẫn và hỗ trợ cho bài viết này.
Đặc biệt chúng tôi xin trân trọng sự đóng góp của nhóm Loopring và cộng đồng Loopring cho dự án này.

Tham khảo

[1] **Daniel Wang, Jay Zhou, Alex Wang, and Matthew Finestone**. Loopring: A decentralized token exchange protocol. *URL* *https://github.com/Loopring/whitepaper/blob/master/en_whitepaper.pdf*

[2] **Satoshi Nakamoto**. Bitcoin: A peer-to-peer electronic cash system. 2008.

[3] **Gavin Wood**. Ethereum: A secure decentralised generalised transaction ledger. *Ethereum Project Yellow Paper*, 151, 2014.

[4] Ethereum: a next generation smart contract and decentralized application platform (2013). *URL http://ethereum. org/ethereum.html*, 2017.

[5] **Chris Dannen**. *Introducing Ethereum and Solidity*. Springer, 2017.

[6] Cosmos a network of distributed ledgers. *URL https://cosmos.network/whitepaper*.

[7] **Anonymous**.aelf-a multi-chain parallel computing blockchain framework. *URL https://grid.hoopox.com/aelf_whitepaper_en.pdf*, 2018.

[8] **Anton Churyumov**. Byteball: A decentralized system for storage and transfer of value. *URL* *https://byteball.org/Byteball.pdf*.

[9] **Serguei Popov**. The tangle. URL https://iota.org/IOTA_Whitepaper.pdf.

[10] **Colin LeMahieu**. Raiblocks: A feeless distributed cryptocurrency network. URL https://raiblocks.net/media/RaiBlocks_Whitepaper

English.pdf.

[11] **Anonymous**. Delegated proof-of-stake consensus, a robust and flexible consensus protocol. URL https://bitshares.org/technology/delegated-proof-of-stake-consensus/.

[12] **Bernardo David, Peter Gazi, Aggelos Kiayias, and Alexander Russell**. Ouroboros praos: An adaptively-secure, semi- synchronous proof-of-stake blockchain. URL https://eprint.iacr.org/2017/573.pdf, 2017.

[13] **Anonymous**. Eos.io technical white paper v2. URL *https://github.com/EOSIO/Documentation/blob/master/TechnicalWhiteP*

[14] **Dai Patrick, Neil Mahi, Jordan Earls, and Alex Norta**. Smart-contract value-transfer protocols on a distributed mobile application platform. URL https://qtum.org/uploads/files/cf6d69348ca50dd985b60425ccf282f3.pdf, 2017.

[15] **Ed Eykholt, Lucius Meredith, and Joseph Denman.** Rchain platform architecture. URL http://rchain-architecture.readthedocs.io/en/latest/.

[16] **Anonymous**. Neo white paper a distributed network for the smart economy. URL

[17] **Anonymous.** Byzantine consensus algorithm. URL https://github.com/tendermint/tendermint/wiki/Byzantine- Consensus-Algorithm.

[18] **Shapiro Marc, Nuno Preguiça, Carlos Baquero, and Marek Zawirski**. Conflict-free replicated data types. URL https://hal.inria.fr/inria-00609399v1, 2011.

[19] **Deshpande and Jayant V.** On continuity of a partial order. Proc. Amer. Math. Soc. 19 (1968), 383-386, 1968. *http://docs.neo.org/en- us/index.html*.

[20] **Weisstein and Eric W. Hasse diagram**. URL http://mathworld.wolfram.com/HasseDiagram.html.

[21] **Chunming Liu**. Snapshot chain: An improvement on block-lattice. URL https://medium.com/@chunming.vite/snapshot-chain-an-improvement-on-block-lattice-561aaabd1a2b.

[22] **Anonymous**. Problems. URL https://github.com/ethereum/wiki/wiki/Problems.

[23] **Dantheman**. Dpos consensus algorithm - the missing white paper. URL https://steemit.com/dpos/@dantheman/dpos-consensus-algorithm-this-missing-white-paper.

[24] **Theo Haerder and Andreas Reuter.** Principles of transaction-oriented database recovery. ACM **Comput. Surv**., 15(4):287–317, December 1983.

[25] **Dan Pritchett**. Base: An acid alternative. Queue, 6(3):48–55, May 2008.

[26] **Jeff Hanson**. Event-driven services in soa. URL https://www.javaworld.com/article/2072262/soa/event-driven-services-in-soa.html.

[27] **Michael Sipser**. Introduction to the Theory of Computation. PWS Publishing, second edition, 2006.

Phụ lục

Phụ lục: Bộ hướng dẫn EVM

A.0.1 0s: Điểm dừng và bộ chỉ dẫn hoạt động đại số

STT	Từ	POP	PUSH	Ngữ nghĩa trong EVM	Ngữ nghĩa trong Vite
0x00	STOP	0	0	Ngưng thực hiện	Cùng nghĩa
0x01	ADD	2	1	Thêm 2 toán hạng	Cùng nghĩa
0x02	MUL	2	1	Nhân đôi toán hạng.	Cùng nghĩa
0x03	SUB	2	1	Chia đôi toán hạng.	Cùng nghĩa
0x04	DIV	2	1	Chia 2 toán hạng Nếu số chia = 0 Về 0	Cùng nghĩa
0x05	SDIV	2	1	Chia theo kí hiệu	Cùng nghĩa
0x06	MOD	2	1	Hoạt động mô đun.	Cùng nghĩa
0x07	SMOD	2	1	Mô đun có biểu tượng	Cùng nghĩa
0x08	ADDMOD	3	1	Thêm 2 toán hạng đầu tiên và thứ 3 với mô đun	Cùng nghĩa
0x09	MULMOD	3	1	Nhân 2 toán hạng đầu với cái số 3	Cùng nghĩa
0x0a	EXP	2	1	Trung bình 2 toán hạng.	Cùng nghĩa
0x0b	SIGNEXTEND	2	1	Mở rộng ký hiệu	Cùng nghĩa

A.0.2 10s: Bộ hướng dẫn vận hành và so sánh bit

STT	Từ	POP	PUSH	Ngữ nghĩa trong EVM	Ngữ nghĩa trong Vite
0x10	LT	2	1	Ít hơn	Cùng nghĩa
0x11	GT	2	1	Nhiều hơn	Cùng nghĩa
0x12	SLT	2	1	Ít hơn với kí hiệu	Cùng nghĩa
0x13	SGT	2	1	Nhiều hơn với kí hiệu	Cùng nghĩa
0x14	EQ	2	1	Bằng	Cùng nghĩa
0x15	ISZERO	1	1	Nếu = 0	Cùng nghĩa
0x16	AND	2	1	Và theo bit	Cùng nghĩa
0x17	OR	2	1	Hoặc theo bit	Cùng nghĩa
0x18	XOR	2	1	Xor theo bit	Cùng nghĩa
0x19	NOT	1	1	Không theo bit	Cùng nghĩa
0x1a	BYTE	2	1	Lấy 1 byte từ toán hạng 2	Cùng nghĩa

A.0.3 20s: Bộ hướng dẫn SHA3

STT	Từ	PoP	PUSH	Ngữ nghĩa trong EVM	Ngữ nghĩa trong Vite
0x20	SHA3	2	1	Tính mã băm Keccak-256	Cùng nghĩa

A.0.4 30s: Bộ chỉ dẫn thông tin môi trường

STT	Từ	POP	PUSH	Ngữ nghĩa trong EVM	Ngữ nghĩa trong Vite
0x30	ADDRESS	0	1	Lấy địa chỉ tài khoản hiện tại	Cùng nghĩa
0x31	BALANCE	1	1	Lấy số dư 1 tài khoản	Cùng nghĩa.
0x32	ORIGIN	0	1	Nhận địa chỉ người gửi của giao dịch ban đầu	Nghĩa khác Về 0 mãi mãi
0x33	CALLER	0	1	Nhận địa chỉ của người gọi trực tiếp.	Cùng nghĩa.
0x34	CALLVALUE	0	1	Nhận số tiền đã chuyển trong giao dịch được gọi.	Cùng nghĩa
0x35	CALLDATALOAD	1	1	Nhận thông số trong cuộc gọi này	Cùng nghĩa
0x36	CALLDATASIZE	0	1	Lấy kích thước của dữ liệu tham số trong cuộc gọi này.	Cùng nghĩa
0x37	CALLDATACOPY	3	0	Sao chép dữ liệu tham số được gọi vào bộ nhớ.	Cùng nghĩa
0x38	CODESIZE	0	1	Lấy kích cỡ của mã đang chạy trong môi trường hiện tại.	Cùng nghĩa
0x39	CODECOPY	3	0	Sao chép mã đang chạy trong môi trường hiện tại vào bộ nhớ.	Cùng nghĩa
0x3a	GASPRICE	0	1	Lấy giá gas trong môi trường hiện tại	Khác nghĩa
0x3b	EXTCODESIZE	1	1	Lấy kích thước mã của một tài khoản.	Cùng nghĩa
0x3c	EXTCODECOPY	4	0	Sao chép mã của tài khoản vào bộ nhớ	Cùng nghĩa
0x3d	RETURNDATASIZE	0	1	Nhận kích thước dữ liệu được trả về từ lần gọi trước đó.	Cùng nghĩa
0x3e	RETURNDATACOPY	3	0	Sao chép cuộc gọi dữ liệu được trả về	Cùng nghĩa

A.0.7 60s and 70s: Hướng dẫn hoạt động ngăn xếp

STT	Từ	POP	PUSH	Ngữ nghĩa trong EVM	Ngữ nghĩa trong Vite
0x60	PUSH1	0	1	Đẩy một byte vào đầu ngăn xếp	Cùng nghĩa
0x61	PUSH2	0	1	Đẩy hai byte đối tượng vào đầu stack.	Cùng nghĩa
0x7f	PUSH32	0	1	Đẩy đối tượng 32 byte (toàn bộ từ) vào đầu ngăn xếp	Cùng nghĩa

A.0.8 80s: Hướng dẫn hoạt động sao chép

STT	Từ	POP	PUSH	Ngữ nghĩa trong EVM	Ngữ nghĩa trong Vite
0x90	SWAP1	2	2	Hoán đổi đối tượng thứ nhất và thứ 2 trong ngăn xếp.	Cùng nghĩa
0x91	SWAP2	3	3	Hoán đổi đối tượng thứ nhất và thứ 3 trong ngăn xếp.	Cùng nghĩa
0x9f	SWAP16	17	17	Hoán đổi đối tượng thứ nhất và thứ 17 trong ngăn xếp.	Cùng nghĩa

A.0.11 f0s: Hướng dẫn vận hành hệ thống

STT	Từ	POP	PUSH	Ngữ nghĩa trong EVM	Ngữ nghĩa trong Vite
0xf0	CREATE	3	1	Tạo hợp đồng mới.	Cùng nghĩa
0xf1	CALL	7	1	Gọi một hợp đồng khác.	Khác nghĩa. gửi một tin nhắn đến một tài khoản Trả về giá trị là 0 mãi mãi.
0xf2	CALLCODE	7	1	Gọi mã của hợp đồng khác Thay đổi trạng thái tài khoản.	Cùng nghĩa

0xf3	RETURN	2	0	Dừng thực thi và trả về giá trị.	Cùng nghĩa
0xf4	DELEGATECALL	6	1	Gọi mã của hợp đồng khác, thay đổi hợp đồng, thay đổi trạng thái tài khoản hiện tại giữ thông tin giao dịch ban đầu.	Cùng nghĩa
0xfa	STATICCALL	6	1	Gọi một hợp đồng khác, không cho phép thay đổi trạng thái.	Khác nghĩa. đại diện cho việc gửi tin nhắn tới hợp đồng, không thay đổi trạng thái của hợp đồng mục tiêu. về kết quả 0 mãi mãi.needed Gửi tin nhắn khác thông qua hợp đồng đích và trả lại
0xfd	REVERT	2	0	Dừng thực thi và khôi phục trạng thái và giá trị trả về	Cùng nghĩa
0xfe	INVALID	Ø	Ø	hướng dẫn không hợp lệ.	Cùng nghĩa
0xff	SELFDESTRUCT	1	0	Dừng thực hiện, thiết lập hợp đồng như chờ đợi cho việc xóa trả lại tất cả số dư.	Cùng nghĩa