Vite: Yüksek Performanslı Eşzamansız Merkezi Olmayan Uygulama Platformu

Chunming Liu
charles@vite.org
Daniel Wang
daniel@loopring.org
Ming Wu
woo@vite.org

Öz

Vite, endüstriyel uygulamaların gereksinimlerini karşılayan genelleştirilmiş bir merkezi olmayan uygulama platformudur.
güvenliği hesaba katarken yüksek verim, düşük gecikme ve ölçeklenebilirlik. Vite, DAG defter yapısını kullanır ve
defterlerdeki işlemler hesaplara göre gruplandırılır. Vite'daki Snapshot Chain yapısı,
DAG defterinin güvenliği. İşlemlerin yazılması ve onaylanmasını sağlayan HDPoS mutabakat algoritması
asenkron, yüksek performans ve ölçeklenebilirlik sağlar. Vite VM, EVM ile uyumludur ve akıllı Sözleşme dili Solidity'den genişletildi ve daha güçlü açıklama yeteneği sağladı. Ek olarak, önemli bir Vite tasarımındaki iyileştirme, bilgileri ileten eşzamansız Olay Odaklı Mimari'nin benimsenmesidir. Sistem verimliliğini ve ölçeklenebilirliği büyük ölçüde artıran akıllı sözleşmeler arasındaki mesajlar aracılığıyla. Ek olarak
yerleşik yerel belirteçler, Vite ayrıca kullanıcıların kendi dijital varlıklarını yayınlamalarını destekler ve ayrıca zincirler arası değer sağlar
Döngü Protokolüne [1] dayalı aktarım ve değişim. Vite, kaynak tahsisini kotalarla gerçekleştirir ve hafif kullanıcılar yapar
işlem ücreti ödemek zorunda değilsiniz. Vite ayrıca sözleşme planlamasını, ad hizmetini, sözleşme güncellemesini, blok kesmeyi destekler
ve diğer özellikler.

1. Giriş

1.1 Tanım

Vite, bir seti destekleyebilen evrensel bir dApp platformudur
her biri bir devlet makinesi olan akıllı sözleşmelerin
bağımsız durum ve farklı operasyonel mantık,
mesaj teslimi ile iletişim kurun.
Genel olarak, sistem işlemsel bir durum makinesidir.
Sistemin durumu $s \in S$, aynı zamanda dünya olarak da bilinir
devlet, her bağımsız hesabın durumundan oluşur.
Hesap durumunda değişikliklere neden olan bir olay denir
işlemler. Daha resmileştirilmiş tanım aşağıdaki gibidir:

Tanım 1.1 (İşlem Durumu Makinesi) *a işlemsel durum makinesi bir 4-demettir:* $(T, S, g,,)$ *, burada* T *bir set işlem olup,* s *durumlarının bir dizi,* $g \in S$ *olduğu başlangıç durumu, aynı zamanda genesis bloğu olarak da bilinir,* $\delta: S \times T \rightarrow S$ *bir durum geçiş işlevidir.*

Bu işlemsel durum makinesinin anlam bilgisi bir
aşağıdaki gibi tanımlanan ayrık geçiş sistemi:

Tanım 1.2 (İşlemsel Durum Makinesinin Anlamları) *İşlemsel durum makinesinin anlambilimi* (T, S, s_0, δ) *ayrık bir geçiş sistemidir:* $(S, s_0, \rightarrow)$ *.* $\rightarrow \in S \times S$ *bir geçiş ilişkisi.*

Aynı zamanda, merkezi olmayan uygulama platformu
nihai tutarlılığa sahip dağıtılmış bir sistemdir. Bazıları aracılığıyla
fikir birliği algoritması, son duruma arasında ulaşılabilir

düğümler. Gerçekçi senaryolarda, durumda depolananlar
akıllı sözleşmeler, merkezi olmayan bir ortamda tamamlanmış bir veri kümesidir.
uygulama, büyük hacimli ve iletilemez
düğümler arasında. Bu nedenle, düğümlerin bir dizi
son durumun tutarlılığını sağlamak için işlemler. Biz
Böyle bir işlem grubunu belirli bir veriye göre düzenlemek
yapı, genellikle defterler olarak anılır.

Tanım 1.3 (Defter) *Defter, bir dizi*
soyut bir veri türü ile özyinelemeli olarak
yapılandırılmış. Aşağıdaki gibi tanımlanır:
{l = Γ (T t)
l = l 1 + l 2
Bunların arasında, bir dizi işlemi temsil eden T t ∈ 2 T ,
Γ ∈ 2 T → L *, bir kitap oluşturmanın bir işlevini temsil eder*
bir dizi işlem aracılığıyla, L *bir dizi defterdir,* +: L × L →
L *, iki tali defteri birleştirme işlemini temsil eder.*
bir.

Bu tür sistemlerde defterlerin
genellikle bir işlem grubunu temsil etmek için kullanılır, bunun yerine
bir eyaletten daha. Bitcoin [2] ve Ethereum [3] 'te, defter
işlemlerin küresel olarak gerçekleştirildiği bir blok zinciri yapısıdır
sipariş edildi. Defterdeki bir işlemi değiştirmek için şuna ihtiyacımız var:
hesap defterinde bir tali defteri yeniden oluşturmak, böylece
işlemde kurcalamanın maliyetini artırmak.
Aynı işlem grubuna göre farklı
geçerli kitaplar oluşturulabilir, ancak bunlar farklı bir
işlem sırası ve sistemin girmesine neden olabilir
farklı bir durum. Bu olduğunda, genellikle denir
"çatal".
1

Sayfa 2

Tanım 1.4 (Çatal) T t , T t *varsayalım*
′ ∈ 2 T , T t ⊆ T t
′ . *Eğer*
l = Γ 1 (T t) , l ′ = Γ 2 (T t
′) *Ve karşılamak yok* l ⪯ l ′ *biz adlandırabilir* l
ve ben ′ *çatal kolçusuyuz.* ⪯ *önek* ilişkisini *temsil eder.*
İşlem durumunun anlambilimine göre
makine, bunu başlangıç durumundan kolayca kanıtlayabiliriz, eğer
defter çatallanmamıştır, her bir düğüm sonunda girecektir
aynı durum. Yani çatallı bir defter alınırsa,
kesinlikle farklı bir duruma mı giriyor? Bağlıdır
defterdeki işlemin doğal mantığı ve nasıl
defterler işlemler arasında kısmi siparişleri düzenler. İçinde
gerçekte, çoğu zaman tatmin edici bazı işlemler vardır.
değişmeli yasalar, ancak hesap sorunu nedeniyle
tasarım, sıklıkla çatallara neden olurlar. Sistem başladığında
ilk durumdan, iki çatallı defteri alır ve uçlar
aynı durumda, bu iki hesap defterine sahte çatal diyoruz
defter.

Tanım 1.5 (Yanlış Çatal) *Başlangıç durumu* s 0 ∈ S *, genel muhasebe*
l 1 , l 2 ∈ L , s 0
l 1

→ s 1 , s 0
l 2
→ s 2 . *Eğer* L 1 / = l 2 *ve* s 1 = s 2 , *biz*
bu iki defteri l 1 , l 2 *sahte çatal defteri olarak adlandırın.*
İyi tasarlanmış bir defter, olasılığı en aza indirmelidir
yanlış çatal
Çatal oluştuğunda, her düğümün birini seçmesi gerekir
birden fazla çatallı defterden. Tutarlılığı sağlamak için-
durumun tensi, düğümlerin aynı algoritmayı kullanması gerekir
seçimi tamamlamak için. Bu algoritmaya
fikir birliği algoritması.
Tanım 1.6 (Konsensus Algoritması) *Konsensüs al-*
gorithm, bir dizi defter ve geri dönüş alan bir işlevdir
tek defter:
Φ: 2 L → L
Konsensüs algoritması sistemin önemli bir parçasıdır
tasarım. İyi bir fikir birliği algoritması yüksek
fikir birliğinin salınımını azaltmak için yakınsama hızı
farklı çatallar ve yüksek koruma
kötü niyetli saldırılar.
1.2 Mevcut İlerleme
Ethereum [4] böyle bir sistemin gerçekleştirilmesinde başı çekti.
Ethereum'un tasarımında dünyanın tanımı
durum S = Σ A , a ∈ A hesabından bir eşleme ve
bu hesabın durumu σ a ∈ Σ . Bu nedenle, herhangi bir durum
Ethereum'un durum makinesi küreseldir, yani
bir düğüm herhangi bir zamanda herhangi bir hesabın durumuna ulaşabilir.
Ethereum'un durum geçiş işlevi δ tanımlandı
bir dizi program kodu ile. Her kod grubuna bir
akıllı sözleşme. Ethereum bir Turing tamamlandı
Komut seti adı verilen EVM adı verilen sanal makine
EVM kodu. Kullanıcılar, akıllı sözleşmeler geliştirebilir.
JavaScript'e benzer programlama dili Solidity ve
bunları EVM kodunda derleyin ve Ethereum'da konuşlandırın
[5]. Akıllı sözleşme başarıyla uygulandığında,
sözleşme hesabının tanımlanmasına eşdeğerdir a
durum geçiş işlevi δ a . EVM, bu tür alanlarda yaygın olarak kullanılmaktadır.
platformlar, ancak bazı sorunlar da var. Örneğin,
kütüphane işlevi desteği ve güvenliği eksikliği var
sorunlar.
Ethereum'un defter yapısı bir blok zinciridir
[2] blok zinciri bloklardan oluşur, her blok
işlemlerin bir listesi ve ikinci blok hash ile ilgilidir.
bir zincir yapısı oluşturmak için önceki bloğun.
Γ ({t 1 , t 2 , ... | t 1 , t 2 , ... ∈ T}) = (..., (t 1 , t 2 , ...))
(1)
Bu yapının en büyük avantajı,
işlemlerin tahrif edilmesini önlemek, ancak
tüm işlemlerin tam sırasını korur,
iki işlem emri yeni bir defter oluşturacaktır.
daha yüksek çatallanma olasılığı. Aslında bu tanıma göre,
işlemsel durum makinesinin durum uzayı dikkate alınır
ağaç olarak: ilk durum kök düğümdür, farklı
işlem sırası farklı yolları temsil eder ve yaprak
düğüm son durumdur. Gerçekte, çok sayıda devlet

yaprak düğümlerinin sayısı aynıdır, bu da çok sayıda
yanlış çatallar.
Konsensüs algoritması Φ PoW olarak adlandırılır.
Bitcoin protokolünde önerilmiştir [2]. PoW algoritması dayanır
kolayca doğrulanabilen, ancak
çözmesi zor. Örneğin, bir hash işlevine göre
h: N → N, gereksinimi karşılamak için x'in sonucunu bulmak
h (T + x) ≥ d, dis zorluk olarak adlandırılan belirli bir sayı, T
içinde bulunan ticaret listesinin ikili bir temsilidir.
blok. Blok zincirindeki her blok, aşağıdakilere bir çözüm içerir:
bu tür sorunlar. Tüm blokların zorluğunu toplayın.
bir blok zinciri defterinin toplam zorluğu:
D (l) = D (∑ ben
l ben) = ∑ ben
D (l ben)
(2)
Bu nedenle, doğru hesabı seçerken
çatal, en yüksek zorluk derecesine sahip çatalı seçin:
Φ (l 1 , l 2 , ..., l n) = l m burada m = arg max
i∈1..n
(D (l ben))
(3)
PoW fikir birliği algoritması daha iyi güvenliğe sahiptir ve
Bitcoin ve Ethereum'da iyi çalışıyor. Ancak,
Bu algoritmada iki ana problem vardır. Birincisi
büyük miktar gerektiren matematiksel bir problemi çözmek
enerji israfıyla sonuçlanan bilgi işlem kaynakları.
ikincisi, algoritmanın yavaş yakınsama hızıdır.
sistemin genel verimini etkileyen. Şu anda
Ethereum'un TPS'si yalnızca yaklaşık 15'tir, bu tamamen
merkezi olmayan uygulamaların ihtiyaçlarını karşılayamamaktadır.

1.3 İyileştirme Yönü

Ethereum'un doğumundan sonra, Ethereum topluluğu
ve diğer benzer projeler sistemi geliştirmeye başladı.
farklı güzergahlar. Sistemin soyut modelinden,
aşağıdaki talimatlar geliştirilebilir:
2

• Sistem durumunu iyileştirme S
• Durum geçiş işlevinin iyileştirilmesi δ
• Defterin yapısını iyileştirmek Γ
• Fikir birliği algoritmasını geliştirmek Φ

1.3.1 Sistemin durumunu iyileştirin

Sistemin durumunu iyileştirmenin ana fikri,
dünyanın küresel durumunu yerelleştirin, her düğüm artık değil
tüm işlemler ve devlet transferleri ile ilgilidir ve yalnızca
tüm durum makinesinin bir alt kümesini korur. Böylece,
S kümesinin ve T kümesinin potansiyelleri büyük ölçüde azalır,
böylece sistemin ölçeklenebilirliğini artırır. Bu tür sistemler
şunları içerir: Cosmos [6], Aelf [7], PChain vb.
Özünde, bu yan zincir temelli şema,
ölçeklenebilirlik karşılığında sistem durumunun bütünlüğü.
Bu, üzerinde çalışan her bir dApp'ın merkezden uzaklaştırılmasını sağlar

zayıflamış - akıllı bir sözleşmenin işlem geçmişi
artık tüm ağdaki her düğüm tarafından kaydedilmiyor, yalnızca
düğümün bir parçası tarafından. Ek olarak, çapraz sözleşme etkileşimi
böyle bir sistemin darboğazı haline gelecektir. Örneğin,
Cosmos'ta, farklı Zone'daki etkileşimler ortak bir
Zincir Hub'ı tamamlamak için [6].

1.3.2 Durum geçiş işlevini iyileştirin

EVM'yi iyileştirmeye dayalı olarak, bazı projeler daha fazlasını sağlar
bol akıllı sözleşme programlama dilleri.
İçin
Örneğin, akıllı bir sözleşme dili Rholang tanımlanmıştır
RChain'de π analizine dayalı; akıllı sözleşme
NEO, NeoContract olarak adlandırılır ve
Java, C # vb. gibi popüler programlama dilleri; EOS
C / C ++ ile programlanmıştır.

1.3.3 Defter yapısını iyileştirin

Defter yapısının iyileştirme yönü,
eşdeğer sınıfın yapımı. Doğrusal defter
birden fazla işlemin genel sıralaması bir
yalnızca kısmi düzen ilişkilerini kaydeden doğrusal olmayan defter.
Bu doğrusal olmayan defter yapısı bir DAG'dir (Yönlendirilmiş Asiklik
Grafik). Şu anda Byteball [8], IOTA [9], Nano [10] ve diğerleri
projeler parayı şifreleme işlevini gerçekleştirdi
DAG'nin hesap yapısına göre. Bazı projeler deniyor
akıllı sözleşmeleri uygulamak için DAG'yi kullanmak, ancak şu ana kadar
bu yöndeki gelişmeler halen araştırılmaktadır.

1.3.4 Fikir birliği algoritmasını geliştirin

Konsensüs algoritmasının iyileştirilmesi çoğunlukla
sistemin verimini ve ana yönü kanıtlayın
yanlış çatal oluşumunu bastırmaktır. Sonra yapacağız
Yanlış çatallanmada hangi faktörlerin yer aldığını tartışın.
Şekil 1: Yanlış Çatal
Gösterildiği gibi, L, tüm olası çatallı hesapların bir koleksiyonudur
bir dizi işlem için ve S,
farklı siparişlerde ulaşılabilir. Tanıma göre
, 1.4, f haritalama: L → S bir örtendir; Ve göre
tanım 1.5, bu eşleştirme bir amaç değildir. İşte biz
yanlış çatallanma olasılığını hesaplayın:
C kullanıcılarının defter üretme hakkına sahip olduğunu varsayalım,
M = | L |, N = | S |, M ben = | L ben |, burada L ben = {l | f (l) = s ben , s ben ∈
S}. Yanlış çatallanma olasılığı aşağıdaki gibidir:

$$P_{ff} = \sum_{i=1}^{N} \left(\frac{M_i}{M}\right)^{C-1} \quad (4)$$

Bu formülden, bunu azaltmak için görebiliriz
yanlış çatallanma olasılığı, iki yol vardır:
• □ Eşitlik ilişkilerini

defter seti, denklik sınıflarını bunlara bölün ve
daha az çatallı defter oluşturun.
• □ Defter üretme hakkına sahip kullanıcıları kısıtlayın,
böylece C'yi düşürür
İlk yol, Vite tasarımındaki önemli yöndür.
Daha sonra detaylı olarak tartışılacaktır. İkinci yollar var
birçok algoritma tarafından benimsenmiştir. PoW algoritmasında,
herhangi bir kullanıcının bir blok üretme hakkı vardır; ve PoS
algoritması, üretim bloğunun gücünü bunlarla sınırlar
sistem haklarıyla; DPoS algoritması [11] kullanıcıyı sınırlar
daha fazla kısıtlanacak bloğu üretme hakkı ile
bir ajan düğümleri grubu içinde.
Şu anda, geliştirilmiş fikir birliği algoritması sayesinde, bazıları
etkili projeler ortaya çıktı. Örneğin, Cardano bir
Ouroboros olarak adlandırılan PoS algoritması ve literatür [12] verir
algoritmanın ilgili karakterlerinin kesin bir kanıtı;
EOS [13] tarafından kullanılan BFT-DPOS algoritması,
DPoS algoritması ve hızlı bir şekilde sistem verimini iyileştirir
blok üretmek; Qtum [14] 'ün fikir birliği algoritması da bir
PoS algoritması; RChain tarafından benimsenen Casper algoritması
[15] aynı zamanda PoS algoritmalarından biridir.
Kendi kendini ortaya koyan başka projeler de var.
fikir birliği algoritmasını geliştirmek için öneriler. NEO [16]
dBFT adı verilen bir BFT algoritması kullanır ve Cosmos [6],
Tendermint [17] adlı algoritma.

---

4. sayfa

2 Defter

2.1 Genel Bakış

Defterlerin rolü, işlemlerin sırasını belirlemektir,
ve işlemlerin sırası aşağıdaki ikisini etkileyecektir
yönler:
• **Durum tutarlılığı:** Sistemin durumundan beri
bir CRDT değildir (Çatışmasız çoğaltılmış veri türleri)
[18], tüm işlemler değiştirilemez ve
farklı işlem yürütme dizisi yol açabilir
sisteme farklı bir duruma giriyor.
• **Hash'in Etkinliği:** Defterde, işlem
eylem, aşağıdakileri içeren bloklar halinde paketlenecektir:
birbirine referans verilen karma. Sırası
işlemler, alıntılanan karmanın bağlantısını etkiler
defterlerde. Bu etkinin kapsamı ne kadar büyükse,
işlemlere müdahale etmenin maliyeti o kadar yüksek olur.
Bunun nedeni, bir işlemdeki herhangi bir değişikliğin
doğrudan veya dolaylı olarak atıfta bulunan karma ile yeniden oluşturulmuştur
işlemin bloğu ..
Defterin tasarımının da iki ana amacı vardır:
Şekil 2: Defter birleştirme
• **yanlış çatal oranının azaltılması:** tartışıldığı gibi
önceki bölüm, yanlış çatalın azaltılması
bir eşdeğer kurularak oran elde edilebilir
sınıflandırma ve bir grup hesabı birleştirerek
sistemi aynı duruma tek bir hesaba aktarır.
Yukarıda gösterildiği gibi, yanlış formülüne göre

çatal oranı, defterin üzerindeki yanlış çatal oranı
sol P ff = (3
5) C
+ (2
5) C
- 1
5 C − 1 ; birleştikten sonra
defter alanı, doğru grafiğin yanlış çatal oranı
P ff
′ = (2
3) C
+ (1
3) C
- 1
3 C − 1. C>
1, P ff
′ <P ff . Yani, en aza indirmeliyiz
işlemler arasındaki kısmi sipariş ilişkisi ve
daha fazla işlemin sırayla değiştirilmesine izin verir.
• **Kurcalama koruması:** bir işlem t değiştirildiğinde
defterde l, kitabın iki alt defterinde
l = l 1 + l 2 , yardımcı defter l 1 etkilenmez ve
l 2 tali defterdeki hash referanslarının yeniden oluşturulması gerekiyor
yeni bir geçerli muhasebe defteri oluşturmak için l ′ = l 1 + l 2
′ . Etkilenen alt
genel muhasebe l 2 = Γ (T 2), T 2 = {x | x ∈ T, x> t}.
işlemlerle kurcalamanın maliyetini artırmak,
kısmi sipariş ilişkisini sürdürmek için gerekli
mümkün olduğunca işlemler arasında
kurcalama kapsamını genişletme | T 2 |.
Şekil 3: Defter yapısı karşılaştırması
Açıkçası, yukarıdaki iki amaç çelişkili,
ve tasarım yapılırken gerekli ödünleşmeler yapılmalıdır.
hesap yapısı. Hesap bakımı bir
işlemler arasında kısmi sipariş, esasen kısmi
Sıralı küme (poset) [19], Hasse diyagramı ile gösteriliyorsa
(Hasse diyagramı) [20], topolojide bir DAG'dir.
Yukarıdaki resim birkaç genel defteri karşılaştırmaktadır.
yapılar ve sol taraftaki defterler,
daha az kısmi düzen. Hasse diyagramı düz görünür ve bir
daha düşük yanlış çatal oranı; sağ taraftaki defterler
daha kısmi düzen ilişkileri ve Hasse diyagramı daha fazladır
ince ve kurcalamaya daha dayanıklı.
Resimde, en sol taraf ortak bir kümedir.
herhangi bir kurcalama olmaksızın bir merkezileştirme sisteminde yapı
prova özellikleri; en sağ taraf tipik bir blok zinciridir
En iyi kurcalamaya dayanıklı özelliklere sahip defter; arasında
iki, iki DAG defteri var, blok kafes hesabı
[10] solda Nano tarafından kullanılıyor; ve sağ taraf, arapsaçı
kitap [9] IOTA tarafından kullanılıyor. Özellikler açısından,
Blocklattice daha az kısmi sipariş ilişkilerini korur ve daha fazladır
yüksek performanslı muhasebe yapısına uygun
merkezi olmayan uygulama platformları. Fakir olduğu için
kurcalama özellikleri, şimdiye kadar güvenlik risklerini ortaya çıkarabilir,
Nano dışında başka hiçbir proje bu defter yapısını benimsemez.

Yüksek performans elde etmek için Vite,
DAG defter yapısı. Aynı zamanda, bir
ek zincir yapısı Snapshot Chain ve iyileştirme
fikir birliği algoritması, blok kafesin eksiklikleri
güvenlik başarıyla oluşturuldu ve iki iyileştirme
daha sonra detaylı olarak tartışılacaktır.

2.2 Ön Kısıtlama

Öncelikle, bu defteri kullanmanın ön koşuluna bir göz atalım.
durum makinesi modeli için yapı. Bu yapı
esasen tüm durum makinesinin bir küme olarak bir kombinasyonu
bağımsız durum makineleri, her hesap karşılık gelen
bağımsız bir durum makinesine ve her işlem yalnızca
bir hesabın durumunu etkiler. Defterde tüm işlemler
hesaplar halinde gruplandırılır ve bir zincir halinde düzenlenir
aynı hesaptaki işlemler. Bu nedenle, biz var
4

S durumu ve T işlemindeki kısıtlamaların ardından
Vite:

Tanım 2.1 (Tek serbestlik derecesi kısıtlaması)

Sistem devlet s ∈ S , *vektördür* s = (s 1 , s 2 , ..., s , n) *oluşan*
devlet tarafından s i , *her hesap.*
İçin ∀t i
∈ T , *sonra*
t i *işleminin gerçekleştirilmesi, sistem durumu transferleri*
aşağıdaki gibi: (s 1
′ , ..., s ben
′ , ..., s n
′) = Σ (t ben , (s 1 , ..., s ben , ..., s n)) , *ihtiyaç*
buluşmak için: s j
′ = S j , j/= i . *Bu kısıtlamaya tek*
bir işlem için serbestlik derecesi kısıtlaması.

Sezgisel olarak, tek bir serbestlik dereceli işlem,
sadece durumu etkilemeden bir hesabın durumunu değiştirin
sistemdeki diğer hesapların toplamı. Çok boyutlu olarak
durum uzay vektörünün bulunduğu uzay, bir işlem
yürütülür ve sistemin durumu yalnızca
koordinat eksenine paralel yön. Lütfen bunu unutmayın
tanım, işlem tanımından daha katıdır
Bitcoin, Ethereum ve diğer modellerde. Bir işlem
Bitcoin iki hesabın durumunu değiştirecek
gönderen ve alıcı; Ethereum durumu değiştirebilir
bir mesaj aramasıyla ikiden fazla hesap.
Bu kısıtlama altında, trans-
eylemler basitleştirilebilir. Herhangi iki işlem ya
ortogonal veya paralel. Bu, gruplama için koşullar sağlar
hesaplara göre işlemler. İşte bir örnek
gözünde canlandırmak:
Şekil 4: Tek serbestlik dereceli ticaret ve orta seviye
durum
Yukarıdaki şekilde gösterildiği gibi, farz edin ki Alice ve Bob
sırasıyla 10 USD var. Sistemin başlangıç durumu
s 0 = (10, 10). Alice, Bob'a 2 USD transfer etmek istediğinde,

Bitcoin ve Ethereum modelinde, bir işlem ′ ,
sistemin doğrudan son duruma geçmesini sağlayın: s 0
t ′
→ s ′ .
Vite tanımında, t ′ işlemi ,
Alice ve Bob'un iki hesabının durumu,
tek serbestlik derecesi ilkesine uymamak.
Bu nedenle, işlem iki işleme bölünmelidir.
tions:
1) 2 USD transferini temsil eden bir t 1 işlemi
Alice tarafından
2) 2 USD alımını temsil eden bir t 2 işlemi
Bob tarafından
Bu şekilde, başlangıç durumundan son duruma
s ′ iki farklı yol olabilir s 0
t 1
→ s 1
t 2
→ s ′ ve
s 0
t 2
→ s 2
t 1
→ s ′ . Bu iki yol sırasıyla geçilir
ara durum s 1 ve s 2 aracılığıyla ve bu ikisi
ara devletler son durum s haritalama vardır ' de
iki hesap boyutu. Başka bir deyişle, yalnızca umursuyorsanız
hesaplardan birinin durumu hakkında, yapmanız gereken tek şey
hesaba karşılık gelen tüm işlemleri yürütmek,
ve diğerlerinin işlemlerini gerçekleştirmesine gerek yoktur.
hesaplar.
Ardından, işlemlerin nasıl bölüneceğini tanımlayacağız
Tek serbestlik dereceli işlemlere Ethereum
Vite tarafından gerekli:
Tanım 2.2 (İşlem Ayrıştırma) *Bölme*
1'den büyük serbestlik derecesine sahip bir işlem
adı verilen tek serbestlik dereceli işlemlere
İşlem Ayrıştırma. Bir transfer işlemi olabilir
bir gönderme işlemi ve bir alma işlemi olarak bölünür;
bir sözleşme arama işlemi, bir sözleşme talebine bölünebilir
işlem ve bir sözleşme yanıt işlemi; bir mesaj
her sözleşme içindeki çağrı, bir sözleşme talebine bölünebilir
işlem ve bir sözleşmeye dayalı yanıt işlemi.
Böylece, iki farklı işlem türü olacaktır.
defterlerde. Bunlar "ticaret çiftleri" olarak adlandırılır:
Tanım 2.3 (İşlem Çifti) *bir gönderme işlemi veya*
toplu olarak bir
"İşlem isteme"; bir alım işlemi veya bir sözleşme
toplu olarak "yanıt" olarak anılan yanıt işlemi
işlem ". Bir istek işlemi ve karşılık gelen
yanıt işlemine işlem çiftleri denir.
The
t *işlemi için talebi başlatan hesap kaydedilir*
olarak bir (t) *; karşılık gelen yanıt işlemi kaydedilir*
as: ˜t *, işleme karşılık gelen hesap*

A (˜t) *olarak kaydedilir* .
Yukarıdaki tanıma dayanarak, şu sonuca varabiliriz:
Vite'daki herhangi iki işlem arasındaki olası ilişki:
Tanım 2.4 (İşlem İlişkisi) *Olabilir*
iki işlem için aşağıdaki ilişkiler için mevcuttur t 1 *ve*
t 2 *:*
***Ortogonallik** : Eğer* bir (t 1)/= A (t 2) *, iki işlem*
ortogonaldir, t 1 ⊥ t 2 *olarak kaydedilir ;*
***Paralel** :* A (t 1) = A (t 2) *ise, iki işlem*
paralel, t 1 ∥ t 2 *olarak kaydedilir ;*
***Nedensellik** :* t 2 = ˜t 1 ise *, o zaman iki işlem*
nedensel, t 1 > t 2 *veya* t 2 <t 1 olarak kaydedilir .
2.3 Defterin Tanımı
Bir defter tanımlamak, bir poset tanımlamaktır. İlk olarak, tanımlayalım
Vite'deki işlemler arasında kısmi sipariş ilişkisi:
Tanım 2.5 (Kısmi işlem sırası) *kullandığımız*
dualistik ilişki < kısmi düzen ilişkisini temsil etmek için
iki işlem:
5

Bir yanıt işlemi, karşılık gelen bir
işlem isteği: t 1 <t 2 ⇔ t 1 > t 2 *;*
Bir hesaptaki tüm işlemler kesinlikle ve
küresel olarak sıralı: ∀t 1 ∥ t 2 *, olmalıdır:* t 1 <t 2 *veya* t 2 <t 1 .
Üzerinde kurulan kısmi sipariş ilişkisi nedeniyle
işlem kümesi T şu özellikleri karşılar:
• Yansımasız: ∀t ∈ T, t <t yoktur;
• Geçişli: ∀t 1 , t 2 , t 3 ∈ T, eğer t 1 <t 2 , t 2 <t 3 , o zaman
t 1 <t 3 ;
• Asimetrik: ∀t 1 , t 2 ∈ T, eğer t 1 <t 2 ise , o zaman mevcut değildir
t 2 <t 1
Bu şekilde, Vite hesabını katı şekilde tanımlayabiliriz
kısmi sipariş seti:
Tanım 2.6 (Vite Ledger) *Vite Ledger sınırlıdır*
verilen işlemin T *kümesinden oluşan poset ve*
kısmi poset <
Şekil 5: Genel muhasebe ile hesap defteri arasındaki ilişki
Vite'da işlem
Kesin bir poset, bir DAG yapısına karşılık gelebilir. Gibi
Yukarıdaki şekilde gösterilen daireler işlemleri temsil eder ve
oklar işlemler arasındaki bağımlılıkları gösterir. a → b
a'nın b'ye bağlı olduğunu gösterir.
Yukarıda tanımlanan Vite defteri yapısal olarak benzerdir
blok kafes için.
İşlemler talebe ayrılmıştır
ve yanıt işlemleri, her biri bir
ayrı bir blok, her bir AI hesabı bir zincire karşılık gelir, bir
işlem çifti ve
karşılık gelen istek işleminin hash değeri.
3 Anlık görüntü zinciri
3.1 İşlem Onayı
Hesap çatallandığında, fikir birliğinin sonucu olabilir
iki çatallı yatay eleman arasında sallanın. Örneğin, şuna göre

bir düğüm daha uzun çatal alırsa bir blok zinciri yapısı
zincir, yeni çatal fikir birliği sonucu olarak seçilecektir,
ve orijinal çatal terk edilecek ve işlem
orijinal çatalda geri alınacaktır. Böyle bir sistemde
işlemin geri alınması çok ciddi bir olaydır ve
iki kat harcama yapmak. Bir işletmenin bir
ödeme, mal veya hizmet sağlar ve bu ödemeden sonra
geri çekilirse, tüccar zarar görebilir. Bu nedenle,
bir kullanıcı bir ödeme işlemi aldığında beklemesi gerekir
sistemin işlemi "onaylaması" için
geri dönme olasılığı yeterince düşük.

Tanım 3.1 (İşlem Onay) *zaman*
bir işlemin geri alınma olasılığı şundan azdır:
belirli bir eşik ϵ *, işlem onaylandı olarak adlandırılır.*
P r (t) '$\varepsilon \Leftrightarrow$ T *olduğu* doğrulandı .

İşlemlerin onaylanması çok kafa karıştırıcı bir kavramdır,
çünkü bir işlemin tanınıp tanınmayacağı aslında
1 - ϵ dolaylı güven düzeyinde. Satıcı satan bir tüccar
elmaslar ve bir kahve satıcısı farklı kayıplar yaşadı.
çifte harcama ile saldırıya uğradılar. Sonuç olarak, eski
işlemde daha küçük ayarlanması gerekiyor ϵ. Bu ayrıca
Bitcoin'deki onay sayısının özü. İçinde
Bitcoin, onay numarası bir
blok zincirinde işlem. Sayısı arttıkça
onaylar, işlemin olasılığı o kadar düşüktür
geri alınıyor [2]. Bu nedenle, tüccarlar dolaylı olarak
doğrulamanın güven düzeyini ayarlamak için
onay numaralarının bekleyen sayısı.

İşlemin geri alınma olasılığı şununla azalır:
hesaptaki karma referans ilişkisinden kaynaklanan süre
yapı. Yukarıda bahsedildiği gibi,
defterin daha iyi kurcalama özellikleri vardır,
işlemin sonraki tüm bloklarını yeniden yapılandırması gerekir
blokta değişim. Yeni işlemler sürekli olduğu için
defterlere eklendikçe, giderek daha fazla
bir işlemdeki düğümler, yani tahrif edilme olasılığı
ile azalacak.

Blok kafes yapısında, işlem olduğu gibi
hesaba göre gruplandıysa, yalnızca bir işlem eklenecektir
kendi hesabının hesap zincirinin sonuna ve
diğer çoğu hesap tarafından oluşturulan işlem,
otomatik olarak işlemin halef düğümü haline gelir.
Bu nedenle, bir fikir birliği algoritması tasarlamak gereklidir.
çifte harcamanın gizli tehlikelerinden kaçınmak için makul.
Nano, oylamaya dayalı bir fikir birliği algoritması kullanır, [10],
işlem, seçilen bir dizi temsili düğüm tarafından imzalanır
bir grup kullanıcı tarafından. Her temsili düğümün bir ağırlığı vardır.
Bir işlemin imzası yeterli ağırlığa sahip olduğunda,
işlemin onaylandığına inanılıyor. Var
bu algoritmada aşağıdaki problemler:
İlk olarak, daha yüksek bir güvenilirlik derecesi ise
ihtiyaç duyulduğunda, oylama ağırlığı eşiğinin yükseltilmesi gerekiyor.
Çevrimiçi olarak yeterli temsili düğüm yoksa,
kesişen hız garanti edilemez ve mümkündür

bir kullanıcının asla gerekli sayıda bilet toplamayacağını
bir değişimi onaylamak için;
İkincisi, işlemlerin geri alınma olasılığı
zamanla azalmaz. Bunun nedeni herhangi bir zamanda
tarihsel bir oylamayı devirmenin maliyeti aynıdır.
Son olarak, tarihsel oylama sonuçları
defterdir ve yalnızca düğümlerin yerel deposunda saklanır.
Bir düğüm, hesabını diğer düğümlerden aldığında,
tarihsel bir olasılığın güvenilir bir şekilde ölçülmesinin bir yolu yok
işlem geri alınıyor.
Esas itibarıyla, oylama mekanizması kısmi bir merkezileştirmedir.
yon çözümü. Oylama sonuçlarını anlık görüntü olarak görebiliriz
defterlerin durumunun. Bu anlık görüntü dağıtılacak
ağdaki her düğümün yerel deposunda. Sırayla
blok zinciri ile aynı kurcalamaya karşı dayanıklılık yeteneğine sahip olmak,
bu anlık görüntüleri zincir yapıları halinde de düzenleyebiliriz,
Vite tasarımının çekirdeklerinden biri olan anlık görüntü
zincir [21].

3.2 Anlık görüntü zincirinin tanımı

Snapshot zinciri, en önemli depolama yapısıdır.
Vite. Ana işlevi, Vite'ın fikir birliğini sürdürmektir.
defterler. İlk olarak, anlık görüntü zincirinin tanımını veriyoruz:

Tanım 3.2 (Anlık görüntü bloğu ve anlık görüntü zinciri)

bir Vite defterinin durum anlık görüntüsünü depolayan bir anlık görüntü bloğu,
hesabın bakiyesi de dahil olmak üzere, hesabın Merkle kökü
sözleşme durumu ve her hesaptaki son bloğun karması
Zincir. Anlık görüntü zinciri, aşağıdakilerden oluşan bir zincir yapısıdır:
anlık görüntü blokları ve bir sonraki anlık görüntü bloğu,
önceki anlık görüntü bloğunun karması.

Bir kullanıcı hesabının durumu, bakiyeyi ve
hesap zincirinin son bloğunun karması; ek olarak
yukarıdaki iki alan, bir sözleşmeli hesabın durumu şunları içerir:
bunun Merkle kök karması, bir devletin durumunun yapısı
hesap aşağıdaki gibidir:

```
struct AccountState {
// hesap bakiyesi
harita <uint32, uint256> bakiyeleri;
// Sözleşme durumunun Merkle kökü
isteğe bağlı uint256 storageRoot;
// son işlemin hash değeri
// hesap zincirinin
uint256 lastTransaction;
}
```

Anlık görüntü bloğunun yapısı aşağıdaki gibi tanımlanır:

```
struct SnapshotBlock {
// önceki bloğun hash değeri
uint256 prevHash;
// anlık görüntü bilgisi
harita <adres, AccountState> anlık görüntüsü;
// imza
uint256 imzası;
}
```

Aynı anda birden fazla jetonu desteklemek için,

Bakiye bilgilerinin Vite's'e kaydedilme yapısı
hesap durumu bir uint256 değil,
bakiyeye belirteç kimliği.
Anlık görüntü zincirindeki ilk anlık görüntü bloğu denir
oluşumun anlık görüntülerini kaydeden "oluşum anlık görüntüsü"
hesapta bloke edin.
Şekil 6: anlık görüntü zinciri
Anlık görüntü zincirindeki her anlık görüntü bloğu,
Vite defterinin tek çatalına sponds, bu mümkündür
tarafından Vite defterinin fikir birliği sonucunu belirlemek için
anlık görüntü bloğu,
anlık görüntü bloğu.

3.3 Anlık görüntü zinciri ve işlem onayı
çiftleşme

Anlık görüntü zincirini tanıttıktan sonra, doğal güvenlik
Blok kafes yapısının kusurları giderildi. Eğer bir
saldırgan, çift harcama işlemi oluşturmak istiyor.
Vite defterinde hash referansını yeniden oluşturmaya ek olarak,
ayrıca tüm anlık görüntü zincirinde yeniden oluşturulması gerekir.
işlemin ilk anlık görüntü bloğundan sonraki bloklar ve
daha uzun bir anlık görüntü zinciri üretmeniz gerekir. Bu şekilde
saldırı maliyeti büyük ölçüde artacaktır.
Vite'da işlemlerin onay mekanizması
aşağıdaki gibi tanımlanan Bitcoin'e benzer:

Tanım 3.3 (Vite'da İşlem Onayı)
Vite'da, bir işlem anlık görüntü zincirinin anlık görüntüsü ise,
işlem onaylandı., anlık görüntü bloğunun derinliği
ilk anlık görüntüye,
işlem.

Bu tanıma göre, onaylanmış işlemlerin sayısı
anlık görüntü zinciri büyüdüğünde eylemler 1 artar,
ve çift harcama saldırısı olasılığı azalır
anlık görüntü zincirinin artmasıyla. Bu şekilde kullanıcılar
Bekleyerek gerekli onay numarasını özelleştirebilir
özelliğe göre farklı onay numaraları için
senaryo.

Anlık görüntü zincirinin kendisi bir fikir birliği algoritmasına dayanır.
Anlık görüntü zinciri çatallanmışsa, en **uzun** çatal seçilir
geçerli bir çatal olarak. Anlık görüntü zinciri bir
yeni çatal, orijinal anlık görüntü bilgileri alınacak
geri, bu, defterdeki orijinal fikir birliğinin
devrildi ve yerini yeni konsensüs aldı. Bu nedenle,
anlık görüntü zinciri, tüm sistemin temel taşıdır
güvenlik ve ciddiye alınması gerekir.

3.4 Sıkıştırılmış depolama

Çünkü tüm hesap durumlarının her anlık görüntüde kaydedilmesi gerekir
anlık görüntü zincirinde blok varsa, depolama alanı çok büyük olmalıdır,
anlık görüntü zincirlerine sıkıştırma gereklidir.
Şekil 7: Sıkıştırmadan önceki anlık görüntü
Anlık görüntü zinciri depolamasını sıkıştırmanın temel yaklaşımı
yaş alanı artımlı depolama kullanmaktır: anlık görüntü bloğu
yalnızca öncekine göre değiştirilen verileri depolar

anlık görüntü bloğu. Bir hesap için işlem yoksa
iki anlık görüntü arasında, son anlık görüntü bloğu,
hesabın verilerini kaydetmeyin.
Anlık görüntü bilgilerini kurtarmak için,
baştan sona anlık görüntü bloğu ve kapak
mevcut veriler tarafından her anlık görüntü bloğunun verileri.
Anlık Görüntü 2
Bir 1 : s 1 ′
Anlık Görüntü 1
Bir 1 : s 1
A 2 : s 2
Bir 3 : s 3
Anlık Görüntü 3
Bir 2 : s 2 ′ ′
Şekil 8: Sıkıştırma sonrası anlık görüntü
Bir hesabın her anlık görüntüsünün yalnızca son durumu
anlık görüntü kaydedilirken kaydedilir, ara durum
dikkate alınmalıdır, bu nedenle içindeki verilerin yalnızca bir kopyası
anlık görüntü, kaç işlem olursa olsun kaydedilecektir.
İki anlık görüntü arasında bir hesap tarafından oluşturulan öğeler.
Bu nedenle, bir anlık görüntü bloğu S ∗ A bayta kadar sürer.
maksimum. Bunlar arasında, S = sizeof (s i), sayısıdır
her hesap durumu için kullanılan bayt ve A toplam
sistem hesaplarının sayısı. Ortalama aktif oranı ise
hesapların toplam hesaplara oranı a, sıkıştırma oranı 1 − a'dır.

4 Konsensüs

4.1 Tasarımın Amacı

Bir fikir birliği protokolü tasarlarken, tam anlamıyla almalıyız
aşağıdaki faktörlerin hesabı:

• **Performans** . Vite'ın birincil hedefi hızlıdır. İçin
yüksek verim ve düşük gecikme performansı sağlar
sistem, bir fikir birliği algoritması benimsememiz gerekiyor
daha yüksek yakınsama hızıyla.

• **Ölçeklenebilirlik** .Vite, herkese açık bir platformdur.
tüm merkezi olmayan uygulamalar, dolayısıyla Ölçeklenebilirlik aynı zamanda
önemli düşünce.

• **Güvenlik** . Vite'ın tasarım prensibi takip etmiyor
nihai güvenlik, ancak yine de sağlanması gerekiyor
yeterli güvenlik temel çizgisi ve etkili bir şekilde
her türlü saldırı.

Mevcut bazı fikir birliği algoritmalarıyla karşılaştırıldığında,
PoW güvenliği daha iyidir ve bir fikir birliğine varılabilir
kötü niyetli düğümlerin bilgi işlem gücü% 50'nin altındaysa.
Bununla birlikte, PoW'nin kesişme hızı yavaştır ve
performans gereksinimlerini karşılamıyor; PoS ve varyantı
algoritmalar matematiksel problemi çözme adımlarını kaldırır.
lems, kesişen hızı ve tek saldırı maliyetini iyileştirin ve
enerji tüketimini azaltın. Ancak PoS'nin Ölçeklenebilirliği
hala zayıf ve "Tehlikede Olan Hiçbir Şey" sorunu [22]
çözmesi zor; BFT algoritmalarında daha iyi performans vardır.
güvenlik ve performans, ancak Ölçeklenebilirliği bir sorundur,
özel zincir veya konsorsiyum zinciri için genellikle daha uygundur;
DPoS [11] serisi algoritması, sorunu etkili bir şekilde azaltır.
üretme izinlerini sınırlayarak yanlış çatalın anlaşılabilirliği
bloklar. Performans ve ölçeklenebilirlik iyidir. Olarak

sonuç olarak, DPoS güvenlikten biraz fedakarlık eder ve
kötü niyetli düğümlerin sayısı 1 / 3'ten fazla olmamalıdır [23].
Genel olarak, DPoS algoritmasının belirgin avantajları vardır
performans ve ölçeklenebilirlikte. Bu nedenle DPoS'u seçiyoruz
Vite mutabakat protokolünün temeli olarak ve onu genişleterek
buna dayanarak düzgün bir şekilde. Hiyerarşik Delege ile
fikir birliği protokolü ve eşzamansız model, genel
platformun performansı daha da geliştirilebilir.

4.2 Hiyerarşik Konsensus

Vite'ın fikir birliği protokolü HDPoS'dir (Hiyerarşik
Temsilci Teminat Kanıtı). Temel fikir ayrıştırmaktır
fikir birliği işlevi Φ (işlevsel ayrıştırma):

$$\Phi (l\,1 , l\,2 , ..., l\,n) = \Psi (\Lambda\,1 (l\,1 , l\,2 , ..., l\,n),$$
$$\Lambda\,2 (l\,1 , l\,2 , ..., l\,n), ...$$
$$\Lambda\,m (l\,1 , l\,2 , ..., l\,n))$$

(5)

Λ i : 2 L → L, yerel fikir birliği işlevi olarak adlandırılır,
döndürülen sonuç yerel konsensüs olarak adlandırılır; Ψ: 2 L → L,
küresel fikir birliği işlevi olarak bilinen, benzersiz bir

---

yerel fikir birliği içinde bir grup adaydan kaynaklanmaktadır.
nihai fikir birliği sonucu.
Bu ayrılıktan sonra tüm sistemin fikir birliği
iki bağımsız süreç haline geldi:
Yerel fikir birliği, yeniden yapılandırmaya karşılık gelen blokları oluşturur.
arama işlemleri ve yanıt işlemi
kullanıcı hesabı veya sözleşme hesabı ve
defterler.
Küresel fikir birliği, defterdeki verileri anlık olarak görüntüler ve
ates anlık görüntü blokları. Defter çatallanmışsa, şunu seçin:
onlardan biri.

4.3 Blok Oluşturma ve Kısıtlama Hakkı
sus Grubu

O halde, işlem bloğunu oluşturma hakkına kim sahip?
anlık görüntü zincirindeki defter ve anlık görüntü bloğunda mı?
Bir fikir birliğine varmak için hangi fikir birliği algoritması benimsenir?
Vite'ın defter yapısı birden fazla
farklı hesaplara göre hesap zincirleri,
hem blokların üretim hakkını uygun şekilde tanımlayın
defterde hesabın boyutuna göre ve
anlık görüntü bloğunun üretim hakkı bir
tek kullanıcı grubu. Bu şekilde birkaç tane koyabiliriz
bir fikir birliği grubuna hesap zincirleri veya anlık görüntü zincirleri,
ve fikir birliği grubunda,
bloğu oluşturun ve bir fikir birliğine varın.

Tanım 4.1 (Konsensus Grubu) *Fikir birliği grubu*
fikir birliği mekanizmasını açıklayan bir demet (L, U, Φ, P)
hesabın veya anlık görüntü zincirinin bir kısmı., L ∈ A | {A s } ,
bir veya birkaç hesap zincirini veya anlık görüntüyü temsil eder
defterdeki fikir birliği grubunun zincirleri; U *temsil eder*
tarafından belirtilen zincir üzerinde blok üretim hakkına sahip kullanıcı
L *;* Φ *konsensüsün fikir birliği algoritmasını belirtir*

grup; ve P *, fikir birliğinin parametrelerini belirtir*
algoritması.
Bu tanım kapsamında, kullanıcılar fikir birliği grupları oluşturabilir
esnek bir şekilde ve farklı fikir birliği parametrelerini seçin.
ihtiyacı var. Daha sonra, farklı fikir birliği grupları üzerinde duracağız.

4.3.1 Anlık Görüntü Konsensüs Grubu

Anlık görüntü zincirlerinin fikir birliği grubuna anlık görüntü denir
en önemli fikir birliği olan fikir birliği grubu
grup Vite. Anlık görüntünün fikir birliği algoritması Φ
sensus group, DPoS algoritmasını benimser ve karşılık gelen
hiyerarşik modelde Ψ. Temsilci sayısı ve
blok oluşturma aralığı,
P parametresi
Örneğin, anlık görüntü fikir birliği grupları belirleyebiliriz
Aralıklarla anlık görüntü blokları oluşturmak için 25 proxy düğümü ile
1 saniye. Bu, işlemin onaylanmasını sağlar
yeterince hızlı olmak için. 10 kez işlem teyidi elde etmek
en fazla 10 saniye beklemesi gerekir.

4.3.2 Özel Konsensüs Grubu

Özel fikir birliği grubu yalnızca profesyonellere uygulanabilir.
defterlerdeki işlem bloklarının azaltılması ve
özel konsensüs grubunun hesap zinciri. Bloklar olabilir
yalnızca özel anahtarın sahibi tarafından üretilebilir
hesabı. Varsayılan olarak, tüm kullanıcı hesapları özel hesaba aittir.
fikir birliği grubu.
Özel fikir birliği grubunun en büyük avantajı
çatallanma olasılığını azaltmaktır. Çünkü sadece bir kullanıcı
blok üretme hakkına sahiptir, tek çatal imkanı
kullanıcının kişisel olarak çifte harcama saldırısı başlatması veya
bir program hatası.
Özel fikir birliği grubunun dezavantajı şudur:
kullanıcı düğümlerinin paketlenebilmesi için çevrimiçi olması gerekir.
işlem. Bu sözleşme için pek uygun değil
hesabı. Sahibin düğümü başarısız olduğunda, başka hiçbir düğüm
Kontrat ürettiği cevap işlemini değiştirebilir,
bu, hizmet kullanılabilirliğini azaltmaya eşdeğerdir.
dApp.

4.3.3 Delege Konsensus Grubu

Temsilci fikir birliği grubunda, kullanıcı hesabı yerine bir
atanmış proxy düğümleri kümesi, aktarımı paketlemek için kullanılır.
DPoS algoritması aracılığıyla eylem. Her iki kullanıcı hesabı
ve sözleşmeli hesaplar mutabakata eklenebilir
grubu. Kullanıcılar bir dizi ayrı aracı düğümü kurabilir ve
yeni bir fikir birliği grubu oluşturun. Bir de varsayılan var
herkes için işlemlerin paketlenmesine yardımcı olmak için Vite'daki fikir birliği grubu
temsilcisini belirlemeyen diğer hesaplar
ayrı ayrı konsensüs grubu olarak da bilinir.
kamusal fikir birliği grubu .
Temsilci fikir birliği grubu çoğu için uygundur
sözleşme hesapları, çünkü işlemlerin çoğu
sözleşme hesabı, sözleşme yanıt işlemleridir.
daha yüksek kullanılabilirlik ve daha düşük gecikmelerin gerekli olduğu
kullanıcı hesabındaki alacak işlemleri.

4.4 Uzlaşmanın Önceliği

Vite protokolünde, küresel fikir birliğinin önceliği

yerel fikir birliğine göre daha yüksek. Yerel con-
sensus çatallanır, küresel fikir birliği seçiminin sonucu
hakim. Başka bir deyişle, küresel fikir birliği bir kez
nihai sonuç olarak yerel konsensüsün çatallaşması, daha da uzun
gelecekteki hesaplarda belirli bir hesap zincirinin çatlaması oluşur,
küresel fikir birliği sonuçlarının geri alınmasına neden olmayacaktır.
Bu sorun uygulanırken daha fazla dikkat edilmesi gerekiyor
çapraz zincir protokolü.
Çünkü bir hedef zincir yuvarlanabilir
geri, aktarma sözleşmesinin ilgili hesap zinciri
Zinciri eşleştirmenin de buna göre geri dönmesi gerekir. Şurada:
bu anda, röle zincirinin yerel fikir birliği varsa
küresel fikir birliği ile benimsenmiştir, bu imkansızdır
geri alma işlemini tamamlayın, bu da verilerin
röle sözleşmesi ve hedef zincir tutarsız.
9

Sayfa 10

Bu sorunu önlemenin yolu, bir parametre *gecikmesi* ayarlamaktır.
konsensüs grubu parametresinde P,
anlık görüntü fikir birliği grubu yalnızca yerelde anlık görüntü almak için
gecikme bloklarından sonra fikir birliği tamamlanır . Bu büyük ölçüde
röle sözleşmelerinin tutarsızlık olasılığını azaltmak,
ama tamamen önlenemez. Kod mantığında
röle sözleşmeleri, geri alma ile de ilgilenmek gerekir
hedef zincirin ayrı ayrı.

4.5 Eşzamansız Model

Sistem verimini daha da iyileştirmek için şuna ihtiyacımız var:
daha mükemmel bir eşzamansız modeli desteklemek için
fikir birliği mekanizması.
Bir işlemin yaşam döngüsü, işlemin başlatılmasını içerir.
işlem, işlem yazımı ve işlem onayı. İçinde
sistemin performansını iyileştirmek için ihtiyacımız olan
bu üç adımı eşzamansız moda tasarlamak için. Bu
farklı zamanlarda işlemlerin miktarı
Kullanıcılar tarafından başlatılan farklıdır, işlem hızı
sistem tarafından işlenen yazı ve işlem onayı
nispeten sabittir. Eşzamansız mod,
zirveler ve çukurlar böylece genel iş hacmini iyileştirir
sistem.
Bitcoin'in eşzamansız modeli ve
Ethereum basittir: tüm kullanıcılar tarafından başlatılan işlem
doğrulanmamış bir havuza yerleştirilir. Madenci paketleri ne zaman
bir bloğa dönüştürülürse, işlem yazılır ve onaylanır.
Aynı zaman. Blok zinciri büyümeye devam ettiğinde,
işlem sonunda ön ayar onayına ulaşır
güven seviyesi.
Bu eşzamansız modelde iki sorun vardır:

- □ İşlemler, kayıtsız şartsız deftere devam ettirilmez.
sıkılaştırılmış durum. Tanınmayan işlemler istikrarsızdır,
ve hiçbir fikir birliği yok, önleyemez
işlemlerin tekrar tekrar gönderilmesi.
- □ Eşzamansız yazma mekanizması yoktur
ve işlemlerin teyit edilmesi. İşlemler sadece
onaylandığında yazılır ve yazma hızı

onay hızı ile sınırlıdır.
Vite protokolü daha gelişmiş bir asyn-
kronlu model: ilk olarak, işlem bir trans-
bir "istek - yanıt" modeline dayalı eylem çifti;
bu bir transfer veya sözleşme görüşmesidir ve işlem
bir istek işlemi yazıldığında başarıyla başlatıldı
deftere. Ek olarak, yazılı ve teyitli
bir işlem de eşzamansızdır. İşlemler olabilir
öncelikle Vite'ın DAG hesabına yazılacak ve
onay süreci tarafından engellenmeyecektir. İşlem
onay anlık görüntü zinciri ve anlık görüntü aracılığıyla yapılır
eylem de eşzamansızdır.
Bu tipik bir üretici - tüketici modelidir. İçinde
üretim hızı ne olursa olsun işlemin yaşam döngüsü
yukarı akıştaki değişiklikler, aşağı akış,
tam olarak yararlanmak için sabit bir oranda işlem
platform kaynakları ve sistemin verimini iyileştirir.

5 Sanal Makine

5.1 EVM uyumluluğu

Şu anda, Ethereum alanında birçok geliştirici var,
ve birçok akıllı sözleşme Solidity'ye dayalı olarak uygulanır ve
EVM. Bu nedenle EVM uyumluluğunu sağlamaya karar verdik
Vite sanal makinesinde ve orijinal anlambilimde
EVM komut setlerinin çoğu Vite'da tutulur. Çünkü
Vite'ın hesap yapısı ve işlem tanımı farklı
Ethereum'dan, bazı EVM talimatlarının semantiği
yeniden tanımlanması gerekiyor, örneğin, bir dizi talimat almak için
bilgileri engelleyin. Ayrıntılı anlamsal farklılıklar olabilir
Ek A'ya atıfta bulunulmaktadır ..
Bunların arasında en büyük fark, anlambilimidir.
mesaj aramaları. Daha sonra ayrıntılı olarak tartışacağız.

5.2 Olay Odaklı

Ethereum protokolünde bir işlem veya mesaj olabilir
birden çok hesabın durumunu etkiler. Örneğin, bir
sözleşme başlatma işlemi, birden fazla
aynı anda değiştirmek için tiple sözleşmeli hesaplar
mesaj aramaları. Bu değişiklikler aynı anda meydana gelir,
ya da hiç. Bu nedenle, Ethereum'daki işlem
aslında bir tür katı işlemdir.
ACID'in özellikleri (Atomiklik, Tutarlılık, İzolasyon,
Dayanıklılık) [24], bu da önemli bir neden
Ethereum'da genişletilebilirlik eksikliği.
Ölçeklenebilirlik ve performans hususlarına dayalı olarak,
Vite, BASE'i tatmin eden nihai bir tutarlılık şemasını benimsedi
(Temel Olarak Kullanılabilir, Yumuşak durum, Nihai tutarlılık) [25]
anlambilim. Özellikle, Vite'ı Olay Odaklı olarak tasarlıyoruz
Mimarlık (EDA) [26]. Her akıllı sözleşme dikkate alınır
bağımsız bir hizmet olmak ve mesajlar iletilebilir
sözleşmeler arasında ilişkilendirilir, ancak hiçbir devlet paylaşılmaz.
Bu nedenle, Vite EVM'sinde,
sözleşmelerde senkron işlev çağrılarının anlambilimini,
ve yalnızca sözleşmeler arasında mesaj iletişimine izin verin.
Etkilenen EVM talimatları esas olarak **CALL** ve
STATICCALL . Vite **EVM'de** bu iki talimat
hemen infaz edilebilirler ve sonucunu da geri veremezler.

arama. Yalnızca yazmak için bir istek işlemi oluştururlar
deftere. Bu nedenle, Vite'da işlevin semantiği
aramalar bu talimata dahil edilmeyecek, bunun yerine
bir hesaba mesajlar.

5.3 Akıllı Sözleşme Dili

Ethereum bir Turing tam programlama dili sağlar
akıllı sözleşmeler geliştirmek için sağlamlık. Desteklemek
zaman uyumsuz anlambilim, Solidity'yi genişlettik ve
10

Sayfa 11

mesaj iletişimi için bir dizi sözdizimi. Genişletilmiş
Solidity, Solidity ++ olarak adlandırılır.
Solidity sözdiziminin çoğu So-
lidity ++, ancak dışındaki işlev çağrılarını içermez
sözleşme. Geliştirici mesajları tanımlayabilir
anahtar kelime *mesajı* ve mesaj işlemcisini tanımlayın (Mes-
anahtar kelime aracılığıyla sageHandler) *üzerine* uygulamak
sözleşmeler arası iletişim işlevi.
Örneğin, A sözleşmesinin add () çağırması gerekir
iadeye göre durumunu güncellemek için B sözleşmesindeki yöntem
değer. Solidity'de, fonksiyon çağrısı ile uygulanabilir.
Kod aşağıdaki gibidir:

```
pragma katılığı ^ 0.4.0;
sözleşme B {
function add (uint a, uint b) döndürür
(uint ret) {
a + b döndür;
}
}
sözleşme A {
toplam uint;
function invoker (adres adres, uint a,
uint b) {
// A.add () 'ye mesaj çağrısı
uint toplamı = B (addr) .add (a, b);
// dönüş değerini kullan
eğer (toplam> 10) {
toplam + = toplam;
}
}
}
```

Solidity ++ 'da, fonksiyon çağrısı codeuint sum =
B (addr) .add (a, b); artık geçerli değil; bunun yerine,
A sözleşmesi ve B sözleşmesi eşzamansız olarak iletişim kurarak
birbirlerine mesaj göndermek. Kod aşağıdaki gibidir:

```
pragma sağlamlığı ++ ^ 0.1.0;
sözleşme B {
mesaj Ekle (uint a, uint b);
mesaj Toplamı (uint toplamı);
Ayriyeten {
// mesajı oku
uint a = msg.data.a;
uint b = msg.data.b;
adres gönderen = msg.sender;
```

```
// birşeyler yap
uint toplamı = a + b;
// sonucu döndürmek için mesaj gönder
gönder (gönderen, Toplam (toplam));
}
}
sözleşme A {
toplam uint;
function invoker (adres adres, uint a,
uint b) {
// B'ye mesaj çağrısı
gönder (adres, Ekle (a, b))
// gönderdikten sonra her şeyi yapabilirsiniz
// kullanmaktan başka bir mesaj
// geri dönüş değeri
}
Sum.on {
// mesajdan dönüş verilerini al
uint toplamı = msg.data.sum;
// dönüş verilerini kullan
eğer (toplam> 10) {
toplam + = toplam;
}
}
}
```

İlk satırda pragma sağlamlığı ++ 0.1.0 kodlayın; içinde-
kaynak kodun Solidity ++ ile yazıldığını ancak
Solidity derleyicisiyle doğrudan derlenmeyecek
derlenen EVM kodunun,
beklenen anlambilim. Vite özel bir derleyici sağlayacaktır
Solidity ++ 'yı derlemek için. Bu derleyici kısmen ileri
uyumlu: ile çakışan bir Solidity kodu yoksa
Vite semantiği, doğrudan derlenebilir, aksi takdirde
hata rapor edilecektir. Örneğin, sözdizimi
yerel fonksiyon çağrıları, diğer hesaplara transferler kalacak
uyumlu; çapraz sözleşmenin dönüş değerini elde etmek
hem işlev çağrısı hem de para birimi eteri,
derlenecek.

A sözleşmesinde, çağırıcı işlevi çağrıldığında,
Ekle mesajı, B sözleşmesine gönderilecektir;
eşzamansızdır ve sonuç hemen döndürülmez
Bu nedenle, bir mesaj işlemcisi tanımlamak gerekir.
A'da döndürülen sonucu almak için on anahtar sözcüğünü kullanarak ve
durumu güncelleyin.

B sözleşmesinde, Ekle mesajı izlenir. Sonra
işlendiğinde, gönderene bir Toplam mesajı gönderilir.

```
mesaj Sonucu döndürmek için Ekle.
```

Sağlamlık ++ mesajlar derlenmiş olacak **CALL**
talimatlar ve bir istek işlemi eklenecektir
defter. Vite'da, defterler mesaj ara yazılımı görevi görür
sözleşmeler arasında eşzamansız iletişim için. Bu ...
mesajların güvenilir bir şekilde saklanmasını sağlar ve kopyalanmayı önler.
Aynı sözleşmeyle bir sözleşmeye gönderilen birden fazla mesaj
FIFO (İlk Giren İlk Çıkar) garantisi verebilir; tarafından gönderilen mesajlar
aynı sözleşmedeki farklı sözleşmeler garanti etmez

FIFO.
Solidity'deki (Event) olayların
ve Solidity ++ 'daki mesajlar aynı kavram değildir.
Olaylar, EVM günlüğü aracılığıyla dolaylı olarak öne gönderilir.
11

Sayfa 12

5.4 Standart Kitaplık

Ethereum'da akıllı sözleşmeler geliştiren geliştiriciler,
Solidity'deki standart kitaplıkların eksikliğinden sık sık rahatsız oluyordu.
Örneğin, Loopring protokolündeki döngü doğrulaması,
zincir dışında yapılması önemli nedenlerden biri
kayan noktalı hesaplama işlevinin
Katılık, özellikle kayan için n karekök [1] [1]
sayılar.
EVM'de, önceden konuşlandırılmış bir sözleşme,
DELEGATECALL komutu li-
brary işlevi. Ethereum ayrıca birkaç Precompiled sağlar
Kontrat, esas olarak birkaç Hash işlemidir. Ama bunlar
işlevler karmaşık uygulamayı karşılayamayacak kadar basit
ihtiyacı var.
Bu nedenle, bir dizi standart kitaplık sağlayacağız
Solidity ++ 'da dize işleme, kayan nokta operatörü gibi
ations, temel matematiksel işlemler, kaplar, sıralama,
ve bunun gibi.
Performans değerlendirmelerine dayalı olarak bu standart
kütüphaneler yerel bir uzantıda uygulanacaktır (Yerel
Uzantı) yolu ve işlemlerin çoğu
Vite yerel kodu ve işlev yalnızca aracılığıyla çağrılır
DELEGATECALL EVM kodunda talimatı.
Standart kitaplık gerektiği gibi genişletilebilir, ancak
çünkü tüm sistemin durum makinesi modeli
deterministik, rastgele gibi işlevler sağlayamaz
sayılar. Ethereum'a benzer şekilde, sözde simüle edebiliriz
anlık görüntü zincirlerinin karması aracılığıyla rastgele sayılar.

5.5 Gaz

Ethereum'da Gaz için iki ana fonksiyon vardır:
ilki, bilgi işlem kaynaklarını ve depolamayı ölçmektir.
EVM kod yürütme tarafından tüketilen kaynaklar ve ikincisi
EVM kodunun durdurulmasını sağlamaktır. Göre
hesaplanabilirlik teorisi, Turing Üzerine Durma Problemi
makineler tartışılmaz bir sorundur [27]. Bu demek oluyor ki
akıllı bir sözleşmenin olup olmayacağını belirlemek imkansızdır
EVM kodunu analiz ederek sınırlı yürütmeden sonra durduruldu.
Bu nedenle, EVM'deki gaz hesaplaması da
Vite. Ancak, Vite'de Benzin Fiyatı kavramı yoktur. Kullanıcılar
ücreti ödeyerek bir değişim için gazı satın almayın, ancak
bilgi işlem kaynakları elde etmek için kota tabanlı bir model aracılığıyla.
Kotaların hesaplanması daha sonra ayrıntılı olarak tartışılacaktır.
"ekonomik model" bölümü.

6 Ekonomik Model

6.1 Yerel Simge

Platform hesaplama ve depolamayı yeniden ölçmek için
kaynakları çalıştırır ve düğümleri çalıştırmaya teşvik eder, Vite bir yerel
token ViteToken. Temel jeton birimi *vite* , en küçüğü

birim *attov* „1 vite = 10 18 attov'dur.
Anlık görüntü zinciri, güvenliğin anahtarıdır ve
Vite platformunun performansı. Düğümü kışkırtmak için
işlem doğrulamasına, Vite protokolüne katılmak
anlık görüntünün üretimi için dövme ödülünü ayarlar
blok.
Aksine, kullanıcılar yeni belirteçler yayınladığında,
sözleşmeler, VNS alan adlarını 1 kaydettirin ve kaynak edinin
kotalar, ViteToken'ı tüketmeleri veya ipotek etmeleri gerekiyor.
Bu iki faktörün birleşik etkisi altında,
sistem kaynaklarının tahsisini optimize etmeye elverişli.

6.2 Kaynak Tahsisi

Vite ortak bir dApp platformu olduğundan,
bunlara uygulanan akıllı sözleşmeler değişiklik gösterir ve her biri farklı
akıllı sözleşmenin iş hacmi için farklı gereksinimleri vardır
ve gecikme. Aynı akıllı sözleşme için bile performans
farklı aşamalardaki gereksinimler farklıdır.
Ethereum'un tasarımında her işlemin ihtiyacı
rekabet edebilmek için fırlatma sırasında bir gaz fiyatı tayin edilmek
diğer işlemlerle hesap yazmak için. Bu tipik bir
bakiyeyi etkili bir şekilde kontrol edebilen teklif modeli
prensip olarak arz ve talep arasında. Ancak, kullanıcı
mevcut arz ve talebi ölçmek zordur
durum ve diğer şirketlerin fiyatını tahmin edemez
rakipler, bu nedenle pazar başarısızlığı kolayca gerçekleşir. Dahası,
her teklif için rekabet eden kaynaklar,
tek bir işlem ve rasyonel
kaynakların hesap boyutuna göre tahsisi.

6.2.1 Kota Hesaplama

Kota bazlı bir kaynak tahsis protokolü benimsedik
kullanıcıların daha yüksek kaynak kotaları elde etmesini sağlayan Vite'ta
üç şekilde:

- İşlem başlatıldığında PoW hesaplanır;
- Hesapta belirli bir miktar *vite* ipotek ;
- Tek seferde az miktarda *zarı* yok etmek .

Belirli kotalar aşağıdaki şekilde hesaplanabilir:
ing formülü:

$$Q = Q_m \cdot \left(\frac{2}{1 + \exp(-\rho \times \xi_T)} - 1 \right) \quad (6)$$

Bunlar arasında, Q_m bir sabittir ve üst
tek bir hesap kotasının sınırı,
sistemin toplam verimi ve toplam sayısı
hesaplar. $= (\xi_d , \xi_s , cost_f)$ maliyeti temsil eden bir vektördür
kaynak elde etmek için bir kullanıcının: ξ_d PoW zorluğudur
kullanıcının bir işlem oluştururken hesapladığı, ξ_s
olduğu *vite* hesabındaki ipotek dengesi ve ξ_f
kullanıcının ödemek istediği tek seferlik maliyettir
kota artışı. Ξ f'nin farklı olduğu unutulmamalıdır
işlem ücretinden. Bu *vite* doğrudan yok edilecek
madencilere ödenmek yerine.

1 *7.2 adlandırma hizmetine* bakın

Formülde, ρ = (ρ d , ρ s , ρ f) vektörü ,
kota elde etmenin üç yolunun ağırlığı, yani
1 tahrip edilmesi ile elde edilen kota *Vite'nin* eşdeğer olan
ipotekli ρ s / ρ f
vite .
Bu formülden, kullanıcı hiçbirinin
ipotek *vite* ne de tek seferlik maliyeti ödemiyor, bu gerekli
PoW hesaplamak için, aksi takdirde kota olmayacaktır.
Tozu etkili bir şekilde önleyebilecek bir işlem başlatmak
saldırır ve sistem kaynaklarının kötüye kullanılmasını önler.
Aynı zamanda bu formül Lojistik bir fonksiyondur. Bu
kullanıcıların daha düşük kotalar alması nispeten kolaydır, bu nedenle
düşük frekanslı kullanıcıların eşiği; ve yüksek frekans
kullanıcıların bir çok kaynağa yatırım yapması gerekir.
daha yüksek kotalar. Ödedikleri ekstra maliyetler,
tüm kullanıcıların faydaları.

6.2.2 Kaynak Miktar Tayini

Anlık görüntü zinciri küresel bir saate eşdeğer olduğundan,
bir hesabın kaynak kullanımını ölçmek için kullanabilir
doğru. Her işlemde, bir anlık görüntünün Hash'i
blok alıntı yapıldığında anlık görüntü bloğunun yüksekliği
işlemin zaman damgası. Bu nedenle, göre
iki işlem zaman damgası arasındaki fark,
iki işlem arasındaki aralığın
yeterince uzun.
Şekil 9: Genel bir saat olarak anlık görüntü zinciri
Yukarıda gösterildiği gibi, A hesabı 4 işlem üretti
2 zaman aralığında, B hesabı yalnızca 2 oluştururken
işlemler. Bu nedenle, bu dönemde A'nın ortalama TPS'si
B'nin 2 katıdır. Yalnızca bir transfer işlemiyse,
ölçülen hesabın ortalama TPS'si yeterlidir. İçin
akıllı sözleşmeler, her borsanın farklı tüketimi vardır
kaynakların her biri için gaz biriktirmek gerekir.
ortalama kaynak tüketimini hesaplamak için işlem
bir süre için. Ortalama kaynak tüketimi
yüksekliği olan bir hesap zincirindeki son k işlemleri
"n" arasında:
Maliyet k (T n) =
k · ∑ n
i = n − k + 1 gaz i
zaman damgası n - zaman damgası n − k + 1 + 1
(7)
Bunlar arasında, bir T n işlemi için zaman damgası n ,
işlemin zaman damgası, yani işlemin yüksekliği
anlık görüntü bloğunu ifade eder; gaz n için tüketilen yakıt
işlem.
Bir işlemi doğrularken düğüm belirleyecektir
kotanın koşulu karşılayıp karşılamadığı: Maliyet (T) ≤ Q,
ve tatmin olmazsa, işlem reddedilecektir.
Bu durumda, kullanıcıların bir işlemi yeniden paketlemesi,
tek seferlik bir ücret ödeyerek veya belirli bir süre bekleyerek kotalar
işlemde daha yüksek bir anlık görüntü almak için.

6.2.3 Kota Kiralama

Bir kullanıcı bol tutarsa *vite* varlıklarını ancak gerekmez

o kadar çok kaynak kotası kullanır ki, kendi
diğer kullanıcılara kota.
Vite sistemi, özel bir işlem türünü destekler.
bir hesap kaynak kotası kullanma hakkını devretmek. Bunda
işlem, ipotek edilebilecek *vite* sayısı ,
devralanın adresi ve kira süresi olabilir
belirtildi. İşlem onaylandıktan sonra, kaynak
jeton miktarına karşılık gelen kota
atanan kişinin hesabına dahildir. Kira süresi dolduğunda
aşıldığında, kota aktarıcıya hesaplanacaktır.
hesabı. Kiralama süresi birimi saniyedir. Sistem
anlık görüntünün yükseklik farkına dönüştürülecek
blok, bu nedenle bazı sapmalar olabilir.
Kira geliri kullanıcı tarafından elde edilebilir.
Vite sistemi yalnızca bir kota aktarım işlemi sağlar ve
leasingin fiyatlandırılması ve ödenmesi sağlanabilir
üçüncü taraf akıllı sözleşmesi aracılığıyla ..

6.3 Varlık İhracı

Yerel belirteç ViteToken'a ek olarak, Vite ayrıca
kullanıcıların tokenlerini çıkarması. Token sorunu yapılabilir
özel bir işlem olan Mint Transaction aracılığıyla. Hedef
nane işlem adres alanı 0 olduğu *veri* bölgesinin
işlem, belirtecin parametreleri şu şekilde belirtilir:
aşağıdaki gibidir:

```
Nane: {
isim: "MyToken",
toplamTedarik: 99999999900000000000000000,
ondalık sayılar: 18,
sahip: "0xa3c1f4 ... fa",
sembol: "MYT"
}
```

İstek ağ tarafından kabul edildikten sonra, *vite*
darphane işlemine dahil edilen

---

mint işlem ücreti olarak başlatıcı hesabı. Sistem
yeni belirtecin bilgilerini kaydeder ve bir
token_id . Yeni oluşturulan tüm bakiyeler
jetonlar *sahip* adresine eklenecektir , yani
sahip hesabı, jetonun oluşum hesabıdır.

6.4 Çapraz Zincir Protokolü

Dijital varlıkların zincirler arası değer transferini desteklemek için
ve "değer adasını" ortadan kaldırarak, Vite bir Vite Cross tasarladı.
zincir Aktarım Protokolü (VCTP).
Zincirler arası aktarıma ihtiyaç duyan her varlık için
hedef zincir, ona karşılık gelen bir belirteç gereklidir
içinde dolaşan hedef Jetonun makbuzu olarak Vite
ToT (Token of Token) olarak adlandırılan Vite. İçin
Örneğin, Ethereum'daki *eteri* transfer etmek istiyorsanız
Vite hesabına, tanımlayıcısına sahip bir ToT verebilirsiniz:
Vite'da ETH , başlangıçtaki TOT miktarı eşit olmalıdır
toplam *eter* miktarına .
Her hedef zincir için, üzerinde bir Ağ Geçidi Sözleşmesi vardır.
Vite arasındaki haritalama ilişkisini sürdürmek için Vite

işlemler ve hedef zincir işlemleri. Fikir birliği içinde
sözleşmenin bulunduğu grup, sorumlu düğüm
blok oluşturma, VCTP Geçişi olarak adlandırılır. VCTP Geçiş ihtiyaçları
Vite düğümü ve hedef zincirin tam düğümü olmak için
aynı anda ve her iki taraftaki işlemleri de dinleyin. Açık
hedef zincir, ayrıca bir Vite Gateway kurmamız gerekiyor
Sözleşme.
VCTP Relay çalışmaya başlamadan önce, ilgili
Vite'daki ToT, ağ geçidi sözleşmesine aktarılmalıdır.
Bundan sonra, ToT arzı yalnızca
ağ geçidi sözleşmesi ve aşağıdakileri sağlamak için hiç kimse eklenemez:
ToT ile hedef varlık arasında 1 değişim oranı. Şurada:
aynı zamanda hedef zincirdeki varlıklar kontrol edilir
Vite ağ geçidi sözleşmesine göre ve hiçbir kullanıcı bunu kullanamaz.
ToT'nin tam bir kabul rezervine sahip olmasını sağlamak için.
Şekil 10: Çapraz Zincir Protokolü
Yukarıdaki resim, çapraz zincir değerinin bir örneğidir
Vite ve Ethereum arasındaki aktarım. Ne zaman
Ethereum kullanıcısı E1, tokenı
Ethereum'u Vite'a, bir işlem gönderebilir.
Vite ağ geçidi sözleşme adresi V iken, kullanıcının adresi
Parametreye Vite üzerindeki A yerleştirilir. Dengesi
transfer, ağ geçidi sözleşmesi hesabında kilitlenecek
ve ToT rezervinin bir parçası olun. Dinledikten sonra
işlem, VCTP geçiş düğümü karşılık gelen bir
hesap gönderen Vite işlemi, aynısını gönderiyor
ToT miktarını kullanıcının Vite'daki A hesabına aktarır. İçinde
resim, sırasıyla ① ve ② gösterir ki E1 ve E2
Vite hesabı A ve B'ye transfer edin.
kullanıcı aktarım sırasında Vite adresini belirtmez,
sözleşme işlemi reddedecektir.
Ters akış, kullanıcı A
Vite hesabından Ethereum'a aktarımı başlatır
hesap, bir işlem Vite ağ geçidine gönderilecektir
sözleşme, belirli bir ToT miktarına aktarır ve belirtir
işlemdeki Ethereum'un E1 alım adresi.
VCTP geçiş düğümü, ilgili
Ethereum Gateway sözleşmesinde yanıt bloğu ve
Ethereum'un bir işlemini Vite ağ geçidine paketlemek
Ethereum üzerinde sözleşme. Ethereum'da, Vite kapısı-
yol sözleşmesi, bu işlemin başlatılıp başlatılmadığını doğrulayacaktır
güvenilir bir VCTP geçişi ve ardından aynı miktarda *eter tarafından*
Vite ağ geçidi sözleşmesinden hedefe aktarılır
hesap E1.
Tüm zincirler arası röle düğümleri hedefi izleyecek
ağ ve her bir zincirler arası aktarım olup olmadığını doğrulayabilirler.
eylem doğrudur ve fikir birliği içinde fikir birliğine varılır
grubu. Ancak anlık görüntü fikir birliği grubu,
ne hedef zincirin işlemi ne de
iki zincir arasındaki eşleştirme doğrudur. Eğer
hedef ağ geri alınır veya zor çatallanır, eşlenir
Vite sistemindeki işlemler geri alınamaz;
benzer şekilde, Vite'deki çapraz zincir işlemleri devredilirse
geri, hedef ağın ilgili işlemi
aynı anda geri alınamaz. Bu nedenle, ne zaman

zincirler arası işlemler yapmak, uğraşmak gerekir
sözleşme mantığında işlem geri alma. Aynı zamanda
4.4 bölümünde anlatıldığı gibi, bir *gecikme* parametresi ayarlamamız gerekir
çapraz zincir Röle konsensüs grubu için.

6.5 Döngü Protokolü

Döngü protokolü [1], bir ondalık sayı oluşturmak için açık bir protokoldür.
tralize varlık ticaret ağı. Diğer DEX ile karşılaştırıldığında
Çözümler, Loopring protokolü çok partili
çift yetkilendirme teknolojisi sağlayan döngü eşleştirme
önleyici işlemleri önlemek için ogy ve tamamen açık.
Loopring protokolünü Vite'a yerleştiriyoruz.
dijital varlıkların dolaşımını teşvik etmeye elverişli
Vite, böylece tüm değer sistemi dolaştırılabilir. İçinde
bu değer sistemi, kullanıcılar kendi dijital varlıklarını çıkarabilir,
varlıkları VCTP aracılığıyla zincirin dışına aktarın ve

Varlık değişimini sağlamak için Döngü protokolü. Bütün
süreç Vite sistemi içinde tamamlanabilir ve
tamamen merkezi olmayan.
Vite'da Loopring Protokolü Akıllı sözleşmesi (LPSC) bir
Vite sisteminin bir parçası. Varlık aktarımı yetkisi ve
çok partili atomik korumanın tümü Vite'da desteklenmektedir.
Loopring rölesi, tamamen entegre olmak için hala açıktır.
kendi ekosistemi.
Kullanıcılar varlık takas işlemlerini ödemek için *vite* kullanabilir ,
döngü gerçekleştiren Looping madencileri tarafından kazanılan token
Vite platformundaki eşleştirme hala *vite* .

7 Diğer Tasarımlar

7.1 Planlama

Ethereum'da akıllı sözleşmeler, işlemlerle yönlendirilir,
ve sözleşmelerin icrası sadece şu şekilde tetiklenebilir:
bir işlem başlatan kullanıcılar. Bazı uygulamalarda bir zamanlama
bir programlama fonksiyonunun yürütülmesini tetiklemek için gereklidir.
bir saat boyunca sözleşme.
Ethereum'da bu işlev üçüncü sırada elde edilir.
parti sözleşmeleri. 1 , performans ve güvenlik garanti edilmez
anteed. Vite'da, zamanlama planlama fonksiyonunu ekliyoruz
yerleşik sözleşme. Kullanıcılar programlarını kaydedebilirler
Zamanlanmış çizelgeleme sözleşmesindeki mantık. Kamuoyu
sus group anlık görüntü zincirini bir saat olarak kullanacak ve
göre hedef sözleşmeye talep işlemi
kullanıcı tanımlı programlama mantığı.
Solidity ++ 'da özel bir Zamanlayıcı mesajı var.
Kullanıcılar sözleşmede kendi planlama mantığını ayarlayabilir
Timer.on aracılığıyla kod.

7.2 İsim Hizmeti

Ethereum'da, sözleşme bir adres oluşturacaktır.
konuşlandırıldığında bir sözleşme tanımlayın. İki tane
adresli sözleşmelerin tanımlanmasındaki sorunlar:

• □ Bir adres, 20 baytlık bir tanımlayıcıdır.
anlam. Kullanıcılar için düşmanca ve sakıncalıdır.
kullanın.
• □ Sözleşmeler ve adresler bire birdir. Yapamazlar

sözleşme yeniden yönlendirmesini destekleyin.
Bu iki sorunu çözmek için geliştirici
Ethereum, üçüncü şahıs sözleşmesi ENS 2 sağladı .
Bununla birlikte, gerçek senaryoda, adlandırma hizmetlerinin kullanımı
çok sık olacak ve üçüncü taraf sözleşmelerinin kullanımı
adlandırmanın küresel benzersizliğini garanti edemiyoruz, bu nedenle
Vite'da bir ad hizmeti VNS (ViteName Hizmeti) oluşturacak.
Kullanıcılar hatırlaması kolay bir dizi isim kaydedebilirler.
ber ve bunları VNS aracılığıyla gerçek adrese çözümleyin.
İsimler, alan adları şeklinde düzenlenmiştir, örneğin:
vite.myname.mycontract . Üst düzey alan adı,
sistem tarafından belirli amaçlar için saklanmalıdır. Örneğin,
vite.xx , Vite adresini temsil eder ve *eth.xx* bir
Ethereum adresi. İkinci düzey alan adı,
tüm kullanıcılar. Kullanıcı, ikinci düzey alan adına sahip olduğunda,
alt alan adı isteğe bağlı olarak genişletilebilir. Alan adı
ad sahibi, alan adının yönlendirdiği adresi değiştirebilir
herhangi bir zamanda isim, bu nedenle bu işlev sözleşme için kullanılabilir
yükseltme.
Alan adının uzunluğu sınırlandırılmamıştır. VNS'de,
alan adının karması aslında depolanır. Hedef
adres, 256 bitten daha az Vite olmayan bir adres olabilir,
çapraz zincir etkileşimi için kullanılabilir.
VNS'nin smart'dan farklı olduğu unutulmamalıdır.
Ethereum'da paket spesifikasyonu EIP190 3 . VNS
bir ad çözümleme hizmetidir, ad şu adrestedir:
çalışma zamanı ve çözüm kuralları dinamik olarak değiştirilebilir
fied; ve EIP190 bir paket yönetimi özelliğidir,
ad alanı statiktir ve derleme zamanında kurulur.

7.3 Sözleşme Güncellemesi

Ethereum'un akıllı sözleşmesi değişmez.
bir Zamanlar
konuşlandırıldı, değiştirilemez. İçinde bir hata olsa bile
sözleşme güncellenemez. Bu çok düşmanca
geliştiricilere ve dApp'ın sürekli yinelemesini çok
zor. Bu nedenle, Vite'ın bir şema sağlaması gerekir:
akıllı sözleşme güncellemesini destekleyin.
Vite'da sözleşme güncelleme süreci şunları içerir:
A. Sözleşmenin yeni bir sürümünü devralmak için
orijinal sözleşmenin durumu.
B. Sözleşmenin adını şuradaki yeni adresi işaret eder:
VNS.
C. **SELFDE** aracılığıyla eski sözleşmeyi kaldırır
STRUCT talimatı
Bu üç adımın aynı anda tamamlanması gerekiyor,
ve Vite protokolü işlemin atomikliğini sağlar.
Geliştiricilerin, eski sözleşme verilerinin
yeni sürüm sözleşmesinde doğru şekilde işlendi.
Yeni sözleşmenin miras almayacağına dikkat edilmelidir.
eski sözleşmenin adresi. Adres tarafından alıntı yapılmışsa,
işlem yine de eski sözleşmeye gönderilecektir. Bu
çünkü sözleşmelerin farklı versiyonları esasen iki
tamamen farklı sözleşmeler, değiştirilip değiştirilemeyecekleri
sözleşmelerin anlamsallığına bağlı olarak dinamik olarak ya da değil.
Vite sistemlerinde akıllı sözleşmeler aslında bölünmüştür

iki türe ayırırsak, ilki bir dApp'ın arka planıdır,
ve iş mantığı açıklanmıştır; ve ikincisi bir tür
gerçek dünyayı haritalayan bir sözleşme. Bir önceki
bir uygulamanın arka plan hizmetine eşdeğerdir;
1 *Ethereum Alarm Clock* , diğer sözleşmelerin yürütülmesini planlamak için kullanılan üçüncü taraf bir sözleşmedir, bkz. Http: // www.
ethereum-alarm-clock.com/
2 *Ethereum Ad Hizmeti , ad çözümlemesi* için kullanılan üçüncü taraf bir sözleşmedir, bkz. Https://ens.domains/
3 *EIP190* Ethereum Akıllı Sözleşme Paketleme Spesifikasyonu, bkz. Https://github.com/ethereum/EIPs/issues/190

Sayfa 16

bir yükseltme yoluyla sürekli olarak yinelenmesi gerekir;
ikincisi bir sözleşmeye eşdeğerdir ve bir kez
etkisi, hiçbir değişiklik yapılamaz, aksi takdirde ihlaldir
sözleşmenin. İzin verilmeyen böyle bir sözleşme için
değiştirilebilir, içinde *statik* anahtar kelimesi ile dekore edilebilir
Solidity ++, örneğin:

```
pragma sağlamlığı ++ ^ 0.1.0;
statik sözleşme Rehni {
// asla değişmeyecek sözleşme
}
```

7.4 Blok Budama

Bir defterde, herhangi bir işlem değişmezdir ve kullanıcılar yalnızca
değiştirmeden veya silmeden deftere yeni işlemler ekleyin-
tarihsel işlemler. Bu nedenle operasyon ile
sistem, defterler daha da büyüyecek. Eğer
ağa katılan yeni bir düğüm,
en son durum, genesis bloğundan başlayarak ve yineleyerek
tüm tarihsel işlemler. Sistemi çalıştırdıktan sonra
bir süre için hesabın kapladığı alan
kitap ve işlemlerin yeniden yapılması için harcanan zaman,
kabul edilemez hale gelir. Yüksek verimli sistem için
Vite, büyüme oranı Bitcoin'den çok daha yüksek olacak
ve Ethereum için bir teknik sağlamak gerekir.
defterlerdeki blokları kırpmak.
Blok kırpma, geçmiş işlemlerin silinmesine atıfta bulunur.
Defterlerde kullanılamayan ve etkilemeyen
işlemsel durum makinesinin çalışması. Öyleyse hangisi
işlemler güvenle silinebilir mi? Hangisine bağlı
işlemin kullanılacağı senaryo, aşağıdakiler dahil:

• **Kurtarma** . Bir işlemin birincil rolü,
durumu kurtar. Çünkü Vite'de anlık görüntü zinciri mağazaları
hesap durumunun anlık görüntü bilgileri, düğümler
anlık görüntü bloğundan durumu kurtarın. Tüm İşlemler
Anlık görüntü bloğundaki *lastTransaction* önce
kurtarma durumuna göre uyarlanmıştır.
• **İşlemlerin** doğrulanması. Yeni bir **işlemi** doğrulamak için
işlem, borsanın önceki
hesap zincirindeki işlem ve bu bir yanıt ise
işlem, aynı zamanda ilgili
işlem iste. Bu nedenle, özel tasarımda
defterleri sayarken, en az bir son işlem

her bir hesap zincirinde saklanmalıdır. Ayrıca hepsi
açık istek işlemleri uyarlanamaz çünkü
hash'lerine sonraki yanıtla başvurulabilir
işlemler.
• **Kotaları hesaplayın.** Bir işlemin karşılayıp karşılamadığı
kota, kayan ortalamasına bakılarak hesaplanır.
son 10 işlem kaynağı, yani en azından son 9
işlemlerin her bir hesap zincirine kaydedilmesi gerekir.
• **Tarih hakkında bilgi alın** . Düğümün sorgulaması gerekiyorsa
işlem geçmişi, ilgili işlem
sorgu uyarlanmayacaktır.
Farklı kullanım senaryolarına göre her düğüm,
yukarıdaki kırpma katmanından birkaç kombinasyon seçin-
egy. Kırpmanın işlem içerdiğini not etmek önemlidir.
anlık görüntü zincirlerinin sağlam tutulması gerekirken, defterlerdeki işlemler.
Ek olarak, anlık görüntü zincirinde kaydedilen şey,
sözleşme durumunun karması. Hesap kırpıldığında,
anlık görüntünün karşılık gelen durumunun olduğu gibi tutulması gerekir.
Vite verilerinin bütünlüğünü sağlamak için şunlara ihtiyacımız var:
tümünü kaydetmek için ağda bazı "Tam düğümler" tutmak için
Işlem verileri. Anlık görüntü fikir birliği grubu düğümleri dolu
düğümler ve ayrıca borsalar gibi önemli kullanıcılar
ayrıca tam düğüm haline gelebilir.

8 Yönetişim

Merkezi olmayan bir uygulama platformu için, verimli bir yönetici-
nans sistemi, sağlıklı bir ekosizasyonu sürdürmek için gereklidir.
tem. Verimlilik ve adalet ne zaman dikkate alınmalıdır?
yönetişim sistemleri tasarlamak.
Vite'ın yönetişim sistemi iki bölüme ayrılmıştır:
zincir içi ve zincir dışı. Zincir üzerinde bir oylama mekanizmasıdır
protokole dayalı ve zincir dışı,
protokolün kendisi.
Oylama mekanizmasında iki türe ayrılır:
Küresel oylama ve yerel oylama. Küresel oylama şuna dayanmaktadır:
vite kullanıcı tarafından yapılan oylamada olarak haklarını hesaplamak için
ağırlık. Küresel oylama esas olarak
anlık görüntü fikir birliği grubu proxy düğümü. Yerel oy
bir sözleşmeye yöneliktir. Sözleşme uygulandığında, bir
jeton, oylamanın temeli olarak belirlenir. Kullanılabilir
konsensüs grubunun ajan düğümlerini seçmek için
sözleşme yer almaktadır.
İşlemlerin doğrulanmasının yanı sıra, aracı düğümü
Anlık görüntü fikir birliği grubunun,
Vite sistemi Uyumsuzluğunu yükseltmek için. Yetki verilen
konsensüs grubu vekil düğümünün olup olmadığına karar verme hakkı vardır.
potansiyelden kaçınmak için sözleşmenin yükseltilmesine izin vermek
sözleşmelerin yükselmesinden kaynaklanan riskler. Ajan
düğüm, karar verme gücünü yükseltmek için kullanılır.
karar verme verimliliğini artırmak için kullanıcılar
ve yetersizlik nedeniyle karar verme başarısızlığından kaçının
oylamaya katılım. Bu proxy düğümlerinin kendileri
ayrıca fikir birliği protokolü ile sınırlandırılmıştır. Sadece en çok 1 temsilci
düğümler geçilir, yükseltme etkili olur. Eğer bunlar
temsilciler karar verme yetkilerini yerine getirmezler.
kullanıcının beklentilerine göre, kullanıcılar proxy'lerini de iptal edebilir

oylama ile yeterlilik.
Zincir dışı yönetim, topluluk tarafından gerçekleştirilir.
Herhangi bir Vite topluluğu katılımcısı bir iyileştirme önerebilir.
Vite protokolünün kendisi veya ilgili sistemler için ment planı,
buna VEP (Vite Enhancement Proposal) denir. VEP
DPoS protokolüne göre 1 , geçerli çoğunluk toplam ajan düğümlerinin 2 / 3'üdür.

---

toplumda geniş çapta tartışılabilir ve
çözümün uygulanmasına Vite ekolojik tarafından karar verilir
katılımcılar. Protokolün yükseltilip yükseltilmeyeceği
VEP'nin uygulanmasına nihayetinde şu karar verilecektir:
ajan düğümü. Elbette, farklar büyük olduğunda,
ayrıca bir tur toplamak için zincir üzerinde bir oylama turu başlatabilirsiniz.
çok çeşitli kullanıcı görüşleri ve proxy düğümü karar verecek
Oylama sonucuna göre yükseltme yapılıp yapılmayacağı.
Bazı Vite katılımcıları yeterli olmayabilir
fikirlerine oy vermek için *vite* belirteçleri. Ama özgürce yapabilirler
VEP gönderin ve görüşlerini tam olarak ifade edin. Kullanıcılar
oy verme hakkına sahip olmak, sağlığı tam olarak hesaba katmak zorundadır
tüm ekolojinin kendi Vite hakları için ve bu nedenle
tüm ekolojik katılımcıların görüşlerini ciddiye alın.

Gelecekte 9 görev

Anlık görüntü zincirlerinde işlem doğrulaması, önemli bir
sistemin formance darboğazı. Çünkü Vite benimser
eşzamansız tasarım ve DAG hesap yapısı, işlem
Doğrulama paralel olarak yürütülebilir. Ancak, vadesi doldu
farklı işlemler arasındaki bağımlılığa
hesaplar, paralellik derecesi büyük ölçüde kısıtlanacaktır.
İşlem doğrulamasının paralelliği nasıl geliştirilir veya
dağıtılmış bir doğrulama stratejisi benimsemek önemli olacaktır.
gelecekteki optimizasyon için bir yön.
Mevcut HDPoS konsensüsünde bazı eksiklikler bulunmaktadır.
sus algoritması da. Aynı zamanda bir optimizasyon yönüdür
fikir birliği algoritmasını geliştirmek veya uyumlu olmak için
yetkilendirilmiş fikir birliğinde daha fazla fikir birliği algoritması ile
grubu.
Ayrıca sanal makinenin optimizasyonu da
sistem gecikmesini azaltmak ve sys'yi iyileştirmek için çok önemli
tem verimi. EVM'nin basit tasarımı nedeniyle ve
talimat setinin basitleştirilmesi gerekli olabilir
gelecekte daha güçlü bir sanal makine tasarlamak ve
daha fazlasını içeren akıllı bir sözleşme programlama dili tanımlayın
güvenlik açıklarını tanımlama yeteneği ve daha az güvenlik açığı.
Son olarak, Vite çekirdek anlaşmasının yanı sıra yapı-
ekolojik kalkınmayı destekleyen yardımcı tesisler
aynı zamanda önemli bir konudur. SDK desteğine ek olarak
dApp geliştiricileri için dApp'te yapılacak çok iş var
ön plan ekosistem inşası. Örneğin şunları yapabilirsiniz:
Mobilde HTML5'e dayalı bir dApplet motoru oluşturun
geliştiricilerin geliştirmesine izin veren Vite cüzdan uygulaması
ve düşük maliyetle dApp yayınlayın.

10 Özet

Diğer benzer projelerle karşılaştırıldığında,

Vite şunları içerir:
• **Yüksek verim** .Vite, DAG defter yapısını kullanır-
ortogonal işlem par-
kitaba alel; ek olarak, çoklu fikir birliği
HDPoS'de gruplar birbirine bağlı değildir
fikir birliği algoritması ve paralel olarak çalışabilir;
en önemli şey, Vite'ın ara sözleşmesinin
iletişim asenkron modele dayanmaktadır
mesajın. Bunların tümü,
sistemin verimi.
• **Düşük gecikme** .Vite, HDPoS konsensüs algoritmasını kullanır
rotasyon üretimini tamamlamak için işbirliği yapmak
gerek kalmadan proxy düğümü üzerinden engelleme
PoW hesaplayın, blok aralığı azaltılabilir
1 saniye, gecikmeyi azaltmak için faydalıdır.
işlem onayı.
• **Ölçeklenebilirlik** . **Ölçeklenebilirlik** gereksinimlerini karşılamak için-
Vite, tek bir serbestlik derecesini sınırlar.
işlem, AC'deki işlemleri gruplama
hesap boyutuna göre sayarak
tamamlanacak farklı hesapların blok üretimi
farklı düğümler tarafından ve ACID semantiğini kaldırmak için
BASE semantiğine yapılan çapraz sözleşme çağrılarının
mesajda. Bu şekilde, düğümlerin artık
dünyanın tüm durumunu kaydedin ve veriler kaydedilir
tüm dağıtılmış ağda parçalama modunda
• **Kullanılabilirlik** . Vite'ın kullanılabilirlik alanındaki iyileştirmeler
Solid'de standart kitaplık desteği sağlayan clude-
ity ++, mesaj sözdizimini, zamanlamayı işlemeye adanmıştır
sözleşme planlaması, VNS adlandırma hizmetleri, destek
sözleşme yükseltme vb.
• **Değer sirkülasyonu** .Vite, dijital varlık is-
suance, çapraz zincir değer transferi, token değişimi
Loopring protokolüne vb. dayalı olarak bir
tam değer sistemi. Kullanıcının bakış açısından,
Vite, tamamen işlevsel, merkezi olmayan bir borsadır.
• **Ekonomi.** Çünkü Vite, kota bazlı kaynakları benimsiyor
tahsis modeli, ticaret yapmayan hafif kullanıcılar
sıklıkla yüksek ücretler veya gaz ücretleri ödemek zorunda kalmazlar.
Kullanıcılar, değiştirmenin çeşitli yollarını seçebilirler.
hesaplama.
Ekstra kota da aktarılabilir
kota kiralama sözleşmesi yoluyla diğer kullanıcılara
sistem kaynak kullanımının verimliliğini artırmak.

11 Teşekkürler

Saygılarımızla, danışmanlarımıza teşekkür ederiz.
bu makaleye rehberlik ve yardım. Özellikle yapardık
Loopring ekibinin katkılarını takdir etmek ve
Topluluğu bu projeye döndürmek.

Sayfa 18

Referanslar

[1] Daniel Wang, Jay Zhou, Alex Wang ve Matthew Finestone. Loopring: Merkezi olmayan bir token değişim protokolü.

URL https://github.com/Loopring/whitepaper/blob/master/en_whitepaper.pdf .
[2] Satoshi Nakamoto. Bitcoin: Eşler arası elektronik nakit sistemi. 2008.
[3] Gavin Wood. Ethereum: Güvenli bir merkezi olmayan genelleştirilmiş işlem defteri. *Ethereum Projesi Sarı Kağıt* , 151,
2014.
[4] Vitalik Buterin. Ethereum: yeni nesil akıllı sözleşme ve merkezi olmayan uygulama platformu (2013). *URL*
http: // ethereum. org / ethereum.html , 2017.
[5] Chris Dannen. *Ethereum ve Solidity ile tanışın* . Springer, 2017.
[6] Jae Kwon ve Ethan Buchman. Cosmos, dağıtılmış defterlerden oluşan bir ağdır. *URL*
https://cosmos.network/whitepaper .
[7] Anonim.
aelf
-
a
çoklu zincir
paralel
bilgi işlem
blok zinciri
çerçeve.
URL
https://grid.hoopox.com/aelf_whitepaper_en.pdf , 2018.
[8] Anton Churyumov.
Byteball:
Değerin depolanması ve aktarımı için merkezi olmayan bir sistem.
URL
https://byteball.org/Byteball.pdf .
[9] Serguei Popov. Karışıklık. *URL https://iota.org/IOTA_Whitepaper.pdf* .
[10] Colin
LeMahieu.
Raiblock'lar:
Bir
duygusuz
dağıtılmış
kripto para
ağ.
URL
https://raiblocks.net/media/RaiBlocks_Whitepaper__English.pdf .
[11] Anonim.
Yetki verilmiş kanıt kanıtı fikir birliği, sağlam ve esnek bir fikir birliği protokolü.
URL
https://bitshares.org/technology/delegated-proof-of-stake-consensus/ .
[12] Bernardo David, Peter Gazi, Aggelos Kiayias ve Alexander Russell. Ouroboros praos:
Uyarlanabilir şekilde güvenli, yarı
eşzamanlı hisse kanıtı blok zinciri. *URL https://eprint.iacr.org/2017/573.pdf* , 2017.
[13] Anonim. Eos.io teknik teknik inceleme v2.0 *URL*
https://github.com/EOSIO/Documentation/blob/master/TechnicalWhitePaper.md .
[14] Dai Patrick, Neil Mahi, Jordan Earls ve Alex Norta. Dağıtılmış bir akıllı sözleşme değer aktarım protokolleri
mobil uygulama platformu. *URL*
https://qtum.org/uploads/files/cf6d69348ca50dd985b60425ccf282f3.pdf , 2017.
[15] Ed Eykholt, Lucius Meredith ve Joseph Denman.
Rchain platform mimarisi.
URL http: // rchain-
architecture.readthedocs.io/en/latest/ .

[16] Anonim.
Neo tanıtım belgesi, akıllı ekonomi için dağıtılmış bir ağ.
URL http://docs.neo.org/en-
us / index.html .
[17] Anonim. Bizans mutabakat algoritması. *URL*
https://github.com/tendermint/tendermint/wiki/Byzantine-
Konsensüs-Algoritma .
[18] Shapiro Marc, Nuno Preguiça, Carlos Baquero ve Marek Zawirski. Çatışmasız çoğaltılmış veri türleri. *URL*
https://hal.inria.fr/inria-00609399v1 , 2011.
[19] Deshpande ve Jayant V. Kısmi bir siparişin sürekliliği üzerine. *Proc. Amer. Matematik. Soc. 19 (1968), 383-386* , 1968.
[20] Weisstein ve Eric W. Hasse diyagramı. *URL http://mathworld.wolfram.com/HasseDiagram.html* .
[21] Tıknaz
Liu.
Enstantane fotoğraf
Zincir:
Bir
Gelişme
açık
blok kafes.
URL
https://medium.com/@chunming.vite/snapshot-chain-an-improvement-on-block-lattice-
561aaabd1a2b .
[22] Anonim. Sorunlar. *URL https://github.com/ethereum/wiki/wiki/Problems* .
[23] Dantheman. Dpos mutabakat algoritması - eksik teknik inceleme. *URL*
https://steemit.com/dpos/@dantheman/dpos-
fikir birliği algoritması bu eksik beyaz kağıt .
[24] Theo Haerder ve Andreas Reuter. İşlem odaklı veri tabanı kurtarmanın ilkeleri. *ACM Comput. Surv.* ,
15 (4): 287–317, Aralık 1983.

Sayfa 19

[25] Dan Pritchett. Baz: Bir asit alternatifi. *Sıra* , 6 (3): 48–55, Mayıs 2008.
[26] Jeff Hanson. Soa'da olay odaklı hizmetler. *URL*
https://www.javaworld.com/article/2072262/soa/event-driven-
services-in-soa.html .
[27] Michael Sipser. *Hesaplama Teorisine Giriş* . PWS Publishing, ikinci baskı, 2006.

Ekler

Ek A EVM Talimat seti

A.0.1 0s: Durdurma ve cebirsel işlem komut seti

Hayır.

Kelimeler

EVM'de POP PUSH Semantiği

Vite'da Anlambilim

0x00
DUR
0
0
Excute için Durdurun.
Sanme semantiği
0x01
EKLE
2

1
İki işlenen ekleyin.
Aynı anlamlar
0x02
MUL
2
1
İki işlenen çarpılıyor.
Aynı anlamlar
0x03
ALT
2
1
İki işlenen çıkarılıyor.
Aynı anlamlar
0x04
DIV
2
1
İki işlenen bölme
Aynı anlamlar
Bölen 0 ise
sonra 0 döndürür
0x05
SDIV
2
1
Sembol ile bölünmüştür.
Aynı anlamlar
0x06
MOD
2
1
Modül İşlemi.
Aynı anlamlar
0x07
SMOD
2
1
Sembollü modül.
Aynı anlamlar
0x08
EKLEME MODU
3
1
İlk ikisini ekleyin
Aynı anlamlar
3. işlenenler ve modül
0x09
ÇOKLU MOD
3
1
İlk ikisini çarpın
Aynı anlamlar
3. işlenenler ve modül

0x0a
tecrübe
2
1
İki işlenenin karesi.
Aynı anlamlar
0x0b SIGNEXTEND
2
1
Sembol uzantısı.
Aynı anlamlar
19

Sayfa 20

A.0.2 10s: Karşılaştırma ve bit işlem komut seti
Hayır.
Kelimeler
EVM'de POP PUSH Semantiği
Vite'da Anlambilim
0x10 LT
2
1
daha az.
Aynı anlamlar
0x11 GT
2
1
daha büyük.
Aynı anlamlar
0x12 SLT
2
1
sembolden daha az.
Aynı anlamlar
0x13 SGT
2
1
sembolden daha büyük.
Aynı anlamlar
0x14 EQ
2
1
eşittir.
Aynı anlamlar
0x15 ISZERO
1
1
0 ise.
Aynı anlamlar
0x16 VE
2
1
Ve yavaş yavaş.
Aynı anlamlar
0x17 VEYA

2
1
Ya da yavaş yavaş.
Aynı anlamlar
0x18 XOR
2
1
Azar azar.
Aynı anlamlar
0x19 DEĞİL
1
1
Az da olsa.
Aynı anlamlar
0x1a BYTE
2
1
Baytlardan birini al
Aynı anlamlar
ikinci işlenenlerden.
A.0.3 20s: SHA3 komut seti
Hayır.
Kelimeler EVM'de PoP PUSH Semantiği
Vite'da Anlambilim
0x20 SHA3
2
1
Keccak-256 hash değerini hesaplayın. Aynı anlamlar
20

Sayfa 21

A.0.4 30s: Çevresel bilgi talimat seti
Hayır.
Kelimeler
EVM'de POP PUSH Semantiği
Vite'da Anlambilim
0x30
ADRES
0
1
Adres alın.
Aynı anlamlar
cari hesabın
0x31
DENGE
1
1
Dengeyi elde edin
Aynı anlambilim.
bir hesabın.
döndürülen
vite hesap bakiyesi
0x32
MENŞEİ
0

1
Göndereni edinin
Farklı samantikler
orijinal işlemin adresleri
sonsuza kadar 0 döndür
Vite sürdürmez
nedensel ilişki
dahili işlem arasında
ve kullanıcı işlemi.
0x33
ARAYAN
0
1
Adresi al
Aynı anlambilim.
doğrudan arayan.
0x34
CALLVALUE
0
1
Aktarılanı edinin
Aynı anlamlar
aranan işlemdeki tutar.
0x35
CALLDATALOAD
1
1
Parametreyi edinin
Aynı anlamlar
bu çağrıda
0x36
CALLDATASIZE
0
1
Boyutunu elde edin
Aynı anlamlar
parametre
bu aramadaki veriler.
0x37
ARAŞTIRMA
3
0
Aranan parametreyi kopyala
Aynı anlamlar
hafızaya veri.
0x38
KOD BOYUTU
0
1
Boyutu elde edin
Aynı anlamlar
çalışan kodun
mevcut ortam.
0x39
KODEKOPİ

3
0
Koşuyu kopyalayın
Aynı anlamlar
mevcut ortamda kod
hafızaya.
0x3a
GAZ FİYAT
0
1
Gazı elde edin.
Farklı samantikler
mevcut ortamdaki fiyat
, sonsuza kadar 0 döndür.
0x3b EXTCODESIZE
1
1
Kodu edinin
Aynı anlamlar
bir hesabın boyutu.
0x3c
EXTCODECOPY
4
0
Kodunu kopyalayın.
Aynı anlamlar
hafızaya bir hesap
0x3d GERİ DÖNÜŞTÜRME
0
1
Veri boyutunu elde edin
Aynı anlamlar
geri döndü
önceki arama.
0x3e
RETURNDATACOPY
3
0
İade edilenleri kopyala
Aynı anlamlar
veri araması
önceden hafızaya alınmış
hafızaya.

Sayfa 22

A.0.5 40s: Blok bilgi talimatları seti

Hayır.
Kelimeler
EVM'de POP PUSH Semantiği
Vite'da Anlambilim
0x40 BLOCKHASH
1
1
Bir bloğun karmasını elde edin. Farklı anlamsal.

hasımını döndür
karşılık gelen anlık görüntü bloğu.
0x41 PARA TABANI
0
1
Adresi alın.
Farklı anlamsal.
madenci yararlanıcısının
mevcut blokta
sonsuza kadar 0 döndür.
0x42 TIMESTAMP
0
1
Dönüş zaman damgası
Farklı anlamsal.
mevcut bloğun.
sonsuza kadar 0 döndür.
0x43 NUMARA
0
1
Numarayı geri ver
Farklı anlamsal.
veya mevcut blok.
Numarasını döndür
yanıtlama işlemi
hesap zincirinde blok
0x44 ZORLUK
0
1
Zorluğu geri ver
Farklı anlamsal.
bloğun.
sonsuza kadar 0 döndür.
0x45 GASLIMIT
0
1
Gazı geri verin.
Farklı anlamsal.
bloğun sınırlaması
sonsuza kadar 0 döndür.
A.0.6 50s: Stach Bellek Saklama Kontrol akışı çalıştırma komut seti
Hayır.
Kelimeler
EVM'de POP PUSH Semantiği
Vite'da Anlambilim
0x50
POP
1
0
Bir veriyi açın
Aynı anlamlar
yığının tepesinden.
0x51
MLOAD
1

1
hafızadan bir kelime yükleyin.
Aynı anlamlar
0x52
DAHA FAZLA
2
0
Hafızaya bir kelime kaydedin
Aynı anlamlar
0x53
DAHA 8
2
0
Hafızaya bir bayt kaydedin.
Aynı anlamlar
0x54
YUVARLAK
1
1
Depodan bir kelime yükleyin.
Aynı anlamlar
0x55
MAĞAZA
2
0
Bir kelimeyi depoya kaydedin.
Aynı anlamlar
0x56
ATLAMA
1
0
Atlama talimatları.
Aynı anlamlar
0x57
JUMPI
2
0
Koşullu talimatları atlayın. Aynı anlamlar
0x58
PC
0
1
Program sayacının değerini alın.
Aynı anlamlar
0x59
MSIZE
0
1
Bellek boyutu elde edin.
Aynı anlamlar
0x5a
GAZ
0
1
Kullanılabilir gazı elde edin.

Farklı anlamsal.
sonsuza kadar 0 döndür.
0x5b JUMPDEST
0
0
Bir atlama hedefini işaretleyin.
Aynı anlamlar
22

Sayfa 23

A.0.7 60s ve 70s: Yığın çalıştırma talimatları
Hayır.
Kelimeler
EVM'de POP PUSH Semantiği
Vite'da Anlambilim
0x60 PUSH1
0
1
Bir baytlık nesneyi itin.
Aynı anlamlar
yığının üstüne
0x61 PUSH2
0
1
İki baytlık nesneyi itin Aynı anlambilim
yığının üstüne.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
0x7f
PUSH32
0
1
32 bayt nesneyi itin
Aynı anlamlar
(tüm kelime) içine
yığının tepesi
A.0.8 80s: Çoğaltma işlemi talimatları
Hayır.
Kelimeler EVM'de POP PUSH Semantiği
Vite'da Anlambilim
0x80 DUP1
1

2
1. nesneyi kopyala ve
Aynı anlamlar
yığının üstüne itin.
0x81 DUP2
2
3
2. nesneyi kopyala.
Aynı anlamlar
ve yığının üstüne itin.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
0x8f
DUP16
16
17
16. nesnenin kopyasını oluştur
Aynı anlamlar
ve yığının üstüne itin.
A.0.9 90s: İşlem talimatlarını değiştir
Hayır.
Kelimeler
Vite'de EVM Semantiğinde POP PUSH Semantiği
0x90 SWAP1
2
2
1. ve 2. yeri değiştirin
Aynı anlamlar
Yığındaki nesne.
0x91 SWAP2
3
3
1. ve 3. yeri değiştirin
Aynı anlamlar
Yığındaki nesne.
.
.
.
.
.
.
.

.
.
.
.
.
.
.
.
0x9f
SWAP16
17
17
1. ve 17.'yi değiştirin
Aynı anlamlar
Yığındaki nesne.
A.0.10 a0s: İşlem talimatlarını günlüğe kaydet
Hayır.
Kelimeler Vite'de EVM Anlambiliminde POP PUSH Semantiği
0xa0 LOG0
2
0
Günlük kaydını uzatın,
Aynı anlamlar
şema yok.
0xa1 LOG1
3
0
Günlük kaydını uzatın.
Aynı anlamlar
1 şema
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
0xa4 LOG4
6
0
Günlük kaydını uzatın.
Aynı anlamlar
4 şema
23

Sayfa 24

A.0.11 f0s: Sistem çalıştırma talimatları

Hayır.
Kelimeler
EVM'de POP PUSH Semantiği
Vite'da Anlambilim
0xf0
OLUŞTURMAK
3
1
Yeni bir sözleşme oluşturun.
Aynı anlamlar
0xf1
TELEFON ETMEK
7
1
Başka bir kontrat ara.
Farklı anlamsal.
göndermeyi belirt
bir hesaba mesaj
Dönen vale sonsuza kadar 0'dır.
0xf2
KALLCODE
7
1
Kodunu arayın
Aynı anlamlar
başka bir sözleşme
Hesabın durumunu değiştirin.
0xf3
DÖNÜŞ
2
0
Yürütmeyi durdur
Aynı anlamlar
ve dönüş değeri.
0xf4
DELEGATECALL
6
1
Kodunu arayın
Aynı anlamlar
başka bir sözleşme, değiştir
sözleşme, akımı değiştir
hesap durumu tutma
orijinal işlem bilgisi.
0xfa
STATICCALL
6
1
Başka bir kontrat arayın,
Farklı anlamsal.
durumu değiştirmeye izin verme.
göndermeyi temsil eder
bir sözleşmeye mesaj,
durumunu değiştirme
hedef sözleşme.

sonsuza kadar 0 döndür. gerekli sonuç
Aracılığıyla başka bir mesaj göndermek
hedef sözleşme ve geri dönüş.
0xfd GERİ DÖN
2
0
Yürütmeyi durdurun ve.
Aynı anlamlar
durumu kurtar ve değeri döndür
sol gazı geri döndürmenin semantiği yok.
0xfe
GEÇERSİZ
0
0
geçersiz talimatlar.
Aynı anlamlar
0xff
KENDİNİ İMHA
1
0
Yürütmeyi durdurun,
Aynı anlamlar
sözleşme yapmak
silinmeyi beklerken
tüm bakiyeyi iade et.
24