Blockstack

ブロックスタック分散コンピューティングネットワーク

テクニカルホワイトペーパー2.0

2019年 5月

For Blockstack PBC

このホワイトペーパーの一部は、以下ピア・レビューカンファレンスおよび雑誌で発行されています。

- M. Ali, J. Nelson, R. Shea and M. J. Freedman, “Blockstack: A Global Naming and Storage System Secured by Blockchains”, 2016 USENIX Annual Technical Conference, Denver, CO, June 2016.
- J. Nelson, M. Ali, R. Shea and M. J. Freedman, “Extending Existing Blockchains with Virtualchain”, Workshop on Distributed Cryptocurrencies and Consensus Ledgers, Chicago, IL, July 2016.
- M. Ali, J. Nelson, R. Shea and M. J. Freedman, “Bootstrapping Trust in Distributed Systems with Blockchains”, USENIX ;login: Issue: Vol. 41, No. 3, Pages 52-58, Fall 2016.

このホワイトペーパーのバージョン（2.0）は、2017年にリリースされたホワイトペーパー1.0以降の主な変更点を内包しています。以前のホワイトペーパーは、次のようにご参照ください。

- M. Ali, R. Shea, J. Nelson and M. J. Freedman, “Blockstack: A New Internet for De- centralized Applications”, Whitepaper Version 1.1, Oct 2017.

このホワイトペーパーで説明しているシステムと概念のいくつかは、このホワイトペーパーの各著者の以下の博士論文でも論じられています。

- M. Ali, Trust-to-Trust Design of a New Internet, PhD dissertation, Princeton Univer- sity, June 2017.
- J. Nelson, Wide-area Software-defined Storage, PhD dissertation, Princeton Univer- sity, June 2018.

免責条項： ブロックスタック・トークンの、「スタックス・ トークン」または「スタックス」は、デラウェア州の有限責任会社であるBlockstack Token LLCによって現在開発している仮想通貨で、そのウェブサイトはwww.stackstoken.comをご覧ください。

このホワイトペーパーは、スタックス・トークンの提供または販売、またはスタックスを購入するためのメカニズムを規定するものではありません。 スタックス・トークンまたは関連する商品の提供販売は、スタックス・トークンの提供する最終文書に基づいてのみ行われます。

この通知は、1933年証券法に基づく規則Aに基づく「事前の試み」資料と見なされる場合があります。規則Aに基づく申し出を完遂する義務を負うことを保証するものではありません。当社は、証券取引委員会（SEC）に当社が提出したオファリング・ステートメントを「適格」と承認した後にのみ、販売を行う計画です。 そのオファリング・ステートメントの情報は、現在提供している情報よりも正確になり、重大な違いが生じる可能性があります。投資決定の前に、SECに提出された書類をご覧ください。

金銭や供応は要求せず、領収致しません。証券購入のオファーは一切受け入れられず、会社がSECに提出したオファリング・ステートメントがSECによって適格承認されるまで、事前に購入した金額の一部の変換もできません。適格承認前の投資の申し出や、それに似た提案の一切は、いかなる義務も負う事なく、撤回または取り消しが可能である点ご注意ください。

投資意向の表示は、いかなる義務や責務も内包しません。

スタックス・トークンのオファーに投資の興味をお持ちの方には、そのオファーに関連して公に提出されたオファーのステートメントを再度検討お願い致します。そのコピーはwww.sec.govで確認可能です。

ブロックスタックは、ブローカー、ディーラー、または投資顧問役として、証券取引委員会（SEC）、金融業界規制当局（FINRA）、またはその他の金融規制当局に登録されているわけでもなく、監督されておらず、金融に関するライセンスも有していません．また、財務上のサービスやアドバイスを提供するライセンスも持ち合わせておりません。

*このホワイトペーパーは、元文である英語版の翻訳版であります。この翻訳版と英語版の間に内容上の違いが生じる場合は、原文である英文版の内容が優先するものと致しますので、ご注意ください。

ブロックスタック分散コンピューティングネットワーク

Muneeb Ali　Jude Nelson　Aaron Blankstein
Ryan Shea　Michael J. Freedman*

https://blockstack.org
ホワイトペーパー Version 2.0.4
2019年 5月 20日

要旨

インターネットアプリケーションの基盤である、通常のクラウドコンピューティングや従来のクライアント/サーバーモデルでは、サーバー側でのみ、動作を維持することによって、アプリケーション拡張ができました。近頃は、大手テクノロジ企業が、すべてのユーザーデータを保存しています。その結果、単一障害点の発生や制御不能、大量のデータの漏えい、ユーザーのプライバシーの侵害など、いくつかの問題が生じています。

本稿では、ブロックスタック・分散コンピューティングネットワークを紹介します。ブロックスタックは、安全でプライベートなアプリケーションを構築するため、従来のクラウドコンピューティングに切り替わるフルスタックを提供します。ブロックスタックによって実現される分散アプリケーションと従来のインターネットアプリケーションとの主な違いは、ほとんどのビジネスロジックとデータ処理が、アプリケーションプロバイダによってホストされる集中型サーバーではなく、クライアント上で実行されることです。多くの点で、分散コンピューティングへの移行は、1980年代のメインフレームからデスクトップコンピューティングへの移行のようでもあります。

ブロックスタックは、エンドツーエンドの設計原則によって、ネットワークのコアをできるだけシンプルに保ちながら、複雑さをエッジ（ユーザーデバイスとユーザー制御ストレージ）にプッシュします。私たちのネットワークの基盤となっているのはスタックス・ブロックチェーンです。[1]これは、（a）分散アプリケーションを拡

張し、（b）開発者がネットワーク上で高品質のアプリケーションを構築できるよう動機付けるように設計されています。スタックス・ブロックチェーンは、新しいブロックチェーンを安全にブートストラップするため、新しいTunableProof-of-Workシステムを使用しています。当社の新しいスマートコントラクト言語は、スマートコントラクトのセキュリティと予測可能性を最適化し、すべての取引について静的分析を適用しています。

私たちのアーキテクチャの重要な要素は、ガイアと呼ばれる非常に拡張性の優れた高性能分散型ストレージシステムです。ユーザーは自分のプライベートデータロッカーを自分のブロックスタック・クライアントソフトウェアに接続し、アプリケーションはユーザーデータをデータロッカーに直接書き込みます。ブロックスタック

Authと呼ばれる汎用IDおよび認証システムでは、アプリケーションごとに個別のサインアップの必要がなくなり、暗号認証よりも安全性が低いことが知られているパスワードベースのログインも不要になります。

* プリンストン大学のコンピュータサイエンス教授およびBlockstack PBCの技術顧問。

私たちのSDKと開発者ツールは、ブロックスタック・アプリケーションの開発を従来のインターネットアプリケーションの開発よりも簡単で、開発者はサーバーやデータベースの駆動に悩む必要はありません。 ブロックスタックは、2019年初頭の時点で、100を超える独立したアプリケーションを構築しており、ブロックスタックのアーキテクチャは、アプリ開発者から長年の経験とフィードバックをもって進化しています。このホワイトペーパーは、2017年初のホワイトペーパーを大きく改訂されたものです。

1. はじめに

40年以上前に設計されたインターネットは、単なる研究プロジェクトから、世界中のほぼすべてのデジタルインタラクションに関わるものへと成長しました。インターネットのアプリケーションレイヤーサーバーインフラストラクチャは、インターネットアプリケーションの急成長をサポートするために飛躍的に進化しました。それに反し、コアとなる下位インターネットプロトコルは、1990年代以来その性質を保っています。

インターネットアプリケーションを構築するための主なモデルは、90年代に普及したクライアント/サーバーモデル[1]です。このモデルは短期的には恵みと言えるが、長期的には悪影響をもたらすものであります。それはWeb自体の大成功つながりましましたが、Webサービスがリモートサーバにますます依存するようにしました。クラウドコンピューティングは、根本的にクライアント/サーバーモデルの進化です。近頃、クラウド・プロバイダーは、ユーザーの個人情報保存、アプリケーションロジックと計算の実行、アクセス認証情報の管理などを行っています。

この10年間で、クラウドコンピューティングの悪影響が出始めています。それは、クライアント/サーバーモデルに依存しながら、ソフトウェアを構築するという点で、システム全般に疑問を持たせています。大規模なデータ侵害[2]、ユーザーのプライバシーの侵害[3]、データの移植性の欠如、および大手企業への広範な不信[4]は、クライアント/サーバーモデルの基本設計によるものです。人間社会におけるコンピューティングの重要性が増していることを考えると、時代遅れのコンピューティングモデルに私たちの生活を規定させることはできません。

クラウドコンピューティングの次の進化は、これらの集中プラットフォームへの依存を減らすために、より強力なクライアントデバイス、エッジコンピューティング、そしてグローバルな接続性を利用することでしょう。分散コンピューティングにおいて、この進化はすでに進んでおり、メインフレームの後デスクトップが登場して件以来、コンピューティング業界にとって最も重要な技術シフトとなりえるでしょう。分散コンピューティングが、ソフトウェアの構築と使用のやり方を変えることができます。開発者に働きかけるため一連の新しいツールを与え、ソフトウェ

アはユーザーを保護し、他の何よりもユーザーの利益を最重点にするとのことで、ユーザーとソフトウェアとの関係を変わります。ブロックスタックは、従来のクラウドコンピューティングに代わるフルスタックコンピューティングネットワークを設計、開発、および成長させるためのオープンソースの取り組みです。

ブロックスタックは、従来のインターネットのアプリケーションレイヤーを再構想し、分散型アプリケーションに新しいネットワークを提供します。ブロックスタックに構築されたアプリケーションは、ユーザーが自分のデータを直接所有し統制することを可能にします[5]。 ブロックスタックは従来のインターネットトランスポートレイヤと基盤になる通信プロトコルを使いながら、アプリケーションレイヤーに集中するポイントを取り除きます。私たちは、エンドツーエンドの設計原理[6]、[7]に従って、ネットワークのコアを単純に保ちながら、クライアントに複雑さを加えます。アプリケーションを拡張するために、グローバルステートの変化を最小限に抑え、クラウドストレージに匹敵するパフォーマンスを提供する信頼性の高い分散ストレージシステムを提供します。さらに、当社のフルスタックアプローチでは、分散型アプリケーションの構築に携わるすべての開発者に必要なデフォルトオプションのスタックコンポーネントを提供します。ブロックスタックはモジュール式であり、開発者はそれを簡単にカスタマイズして代替のテクノロジを統合できます。

このホワイトペーパーは、2017年以前のホワイトペーパーを大幅に改訂したものであり、プロダクション展開からの教訓およびアプリケーション開発者からのフィードバックを参考にした、デザイン革新を取り入れています。2016年ピアーレビューを経た出版物[8、9、10]の一部も古いものとなっています。最新のブロックスタック設計については、このペーパーを参照することをお勧めします。このホワイトペーパーでは、分散型アプリケーションを拡張し、開発者に高品質のアプリケーションを構築させるためのインセンティブシステムが追加された、新しいスタックス・ブロックチェーンの設計について紹介します（セクション2）。セキュリティと予測可能性を最重点におくクラリティという名の新しいスマートコントラクト言語を紹介します（セクション3）。ガイアの分散ストレージシステムの設計（セクション4）、認証プロトコル（セクション5）、開発者ツール（セクション6）の概要を説明し、アプリケーション開発者が現在ブロックスタックを使用している様子（セクション7 ）を強調しています。

1.1. 分散コンピューティングの概要

分散システムは、基盤となるインフラストラクチャーを単一エンティティーが管理できない、特定タイプの分散システムであり、ノードがネットワークに参加すればインセンティブをえる仕組みです。分散ネットワークへの最近の関心は、Bitcoinホワイトペーパーの公開から始まりました[11]。ブロックチェーンと仮想通貨は、現代

の分散システムで重役を果たしています。ブロックチェーンと仮想通貨の背景については、[12]を参照することをお勧めします。

近頃は、さまざまなタイプの分散システムがあります。現在最大のブロックチェーンネットワークであるBitcoinの主な目的は、当時点でのBitcoinの所有権を追跡して確認する処理過程の構築です。Ethereum[13]の目的はより一般的です。スマートコントラクトと分散アプリケーションを可能にするための「ワールドコンピュータ」の構築を目指しています。Filecoin[14]は分散ファイルホスティングとストレージのためのネットワーク構築を試みています。対照的に、ブロックスタックは、ブロックチェーンレイヤーの最小限状態を維持しながら、ロジック処理を安全にこなせるプライベートアプリケーションを具現することに焦点を合わせて、分散コンピューティングのためのフルスタックを実現することです。

1.2. 設計目標

ブロックスタックの設計は、以下の特性を生かしています。

1. **使いやすさ：** 分散型アプリケーションは、エンドユーザーにとって現在のインターネットアプリケーションと同じくらいの使いやすさを提供すると予測します。さらに、分散型アプリケーションは、今日のクラウドコンピューティングでの開発と同じくらい簡単に開発できると予測します。
2. **スケーラビリティ：**分散型アプリケーションは、数億から数十億のユーザーのサポートを可能にします。そのため、ネットワーク（ブロックチェーンを含む）は、ユーザーとアプリケーションの数に合わせて拡張する必要があります。
3. **ユーザーコントロール：**分散コンピューティングベースのアプリケーションは、デフォルト状態でユーザーに制御権が属する必要があります。そのため、アプリケーションによって運営されるサーバーに頼る代わりに、ユーザーは演算機能とストレージを提供する必要があります。

これらの設計目標を念頭に置いて、ブロックスタックは、「重い」ブロックチェーンと「ワールドコンピューター」の設計思想を用いた現代の分散コンピューティングアプローチとは異なる設計思想を追求します[13、15、16、17]。17].

ブロックチェーンレイヤーでの最小限のロジックとステート: スケーラビリティを実現するために、ブロックスタックは、「軽量」ブロックチェーンレイヤーでアプリケーションロジックとデータを最小限に抑えます。アプリケーションロジックとストレージにブロックチェーン操作を適用することは、「オフチェーン」アプローチよりも本質的に低速です。広範囲のネットワークおよびデバイスにわたってステートを同期させて検証する必要があるため、そのような操作のスループットには大きな制限があります。制限

要因は、一般的なネットワークノードで利用可能なグローバル接続およびメモリ/ストレージの基本的な帯域幅、すなわち物理的制限（対プロトコル制限）です。

ローカルステートの変化とグローバルステートの変化: ブロックスタック・ネットワークは、ブロックスタック上に構築されたアプリケーションがスケーラブルであることを保証するため、フルスタックアプローチを使用します。このスケーラビリティは、アプリケーション内の相互作用の結果、いつでも起りうるローカルステートの変化とグローバルステートの変化を反映するためです。よって、当社のストレージシステム（ガイア、セクション4）および認証プロトコル（セクション5）は、当社のネットワークの基本提供のコンポーネントとなっています。これらを使用すると、アプリケーションはユーザのプライベートデータロッカーとの通信が可能になって、ブロックチェーントランザクションに関わることなくユーザーを認証することができます。スタックス・ブロックチェーンは、グローバルなステート遷移を一貫した手順で分散的に調整するのみに使用されます。

信頼性の高いクラウドに似たストレージとピアストレージ: ブロックスタック上に構築されたアプリケーションは、ユーザーと共にデータを保存し（プライベートデータロッカーを使用して）、ユーザーデータを保存したり、サーバー側で認証情報にアクセスしたりする必要はなくなります。このアプローチでは、ユーザーがデータを管理できるだけでなく、開発者の困難も軽減できます。開発者は、サーバーやデータベースを実行したり、クラウドインフラストラクチャ料金をユーザーに代わって支払ったりする必要がなくなります。さらに、ピアツーピアストレージ固有の信頼性とパフォーマンスの問題[18]を回避でき、従来のクラウドストレージプロバイダを分散型の広域ファイルシステムに再利用できます。ブロックチェーンレイヤーはユーザーのデータロッカーへのポインタのみを保存します。

開発者向けのフルスタックSDK: ブロックスタックは「フルスタック」アプローチを採用し、分散アプリケーションの開発に必要なすべてのレイヤーのデフォルトのオプションを提供します。開発者SDKは、ブロックチェーンやその他の技術の要する複雑さを排除しています。アプリケーション開発者は、SDKのインタフェースを使用して自分のアプリケーションを簡単に構築できます（セクション6）。開発者スタックのさまざまなレイヤーはモジュール式であり、必要に応じて他の技術とともに使用できます。

現代の分散型コンピューティングアプローチとこれらの違いに加えて、私たちのスマートコントラクト言語は、スマートコントラクトのセキュリティと予測可能性を最適化するための独自の設計思想で構築されています（詳細はセクション3を参照）。

1.3. アプリケーションのための新しいモデル

ブロックスタックは、開発者にアプリケーションを構築するための新しいモデルを提供します。これにより、アプリケーションは分散化され、ユーザーは基本的に管理権を有します。

1. **不透明なデータベースがない**: クライアント/サーバーモデルでは、サーバーは大量のユーザーデータを保存してクエリを実行する必要があるため、データベースはあらゆるアプリケーションの中核部分でした。分散コンピューティングでは、開発者は最初からデータをホストしないため、データベースの保守と保護について心配する必要はありません。開発者は主に自分のアプリロジックに焦点を当てています。ユーザーはアプリをダウンロードし、自分のプライベートデータロッカーをプラグインします。データベースが使用される場合、それは古いインターネット上の「検索インデクサー」としてのみ機能し、共用データに索引付けする機能は同じです。誰でも基礎となる（分散化された）データを使用して、これらの索引を作成できます。
2. **サーバーを必要としない**: クライアント/サーバーモデルでは、すべてのユーザーの計算がサーバー側で実行されるため、アプリケーションはサーバーを追加することで拡張されます。分散コンピューティングでは、アプリケーションはクライアント側で実行され、新しいユーザーはそれぞれ（アプリ開発者に頼るのではなく）ネットワークに計算能能力とストレージ容量を提供します。各ユーザーはアプリを使用するために必要なストレージとコンピューティングのリソースを持ってくるため、開発者はアプリケーションコードをホストするための最小限のインフラストラクチャーを提供するだけで済みます。
3. **スマートコントラクト**: クライアント／サーバーモデルでは、グローバルステートの変化は、ネットワーク内に唯一権限を持つ中央サーバーによって調整されます。分散コンピューティングでは、これらのステートの変化はオープンブロックチェーン上で実行されるスマートコントラクトを通じて発生します。
4. **分散認証**: 従来のインターネットでは、ユーザーは信頼できる認証プロセスを使用して認証を受けていました。アプリケーションがユーザーデータベースを管理している場合、アプリケーションはパスワードを使用してユーザーを認証します。アプリケーションが、GoogleやFacebookなどのサードパーティのIDサービスに依存している場合、そのIDサービスからアサーションを取得するためにOAuth[19]プロトコルが使用されました。もちろん、これらすべてのアプローチはユーザー自身からプロセスの制御を剥奪するものです。分散コンピューティングでの認証は、ブロックチェーンに固定された特定のユーザーネームの制御権を暗号的にサインすることで、ユーザーのクライアントによって実行されます。どのアプ

リケーションでもこれらの証明を独立的に分類検証できます。

5. **ネイティブトークン:** 従来のインターネットアプリケーションでは、通常、クレジットカードなどのサードパーティのサービスを使用して支払い処理が行われました。デジタルトークンは、ブロックスタックやEthereumなどの分散コンピューティングプラットフォームのネイティブアセットです。ユーザーはこれらのトークンを直接所有しており、それらを使用してデジタルアセットやスマートコントラクトを登録したり、スマートコントラクトを実行するための支払いしたりできます。このようなネイティブトークンの使用を、サブスクリプションサービスの構築または他のアプリ機能を自動化することで、プログラムすることもできます。そのようなプログラム可能なトークンは従来のインターネットアプリ開発者には利用できないものでした。

1.4. 分散コンピューティングのレイヤー

ブロックスタックの分散型コンピューティングネットワークは、従来インターネット設計での「アプリケーションレイヤー」であります。しかし、ブロックスタック・ネットワーク自体は、分散型アプリケーションの実装に必要なコンポーネントをまとめて提供する複数のシステムで構成されています。

1. **スタックス・ブロックチェーン**: ブロックスタック・ネットワークの基盤は、ユーザーが一般的なユーザーネームなどのデジタルアセットを登録および管理し、スマートコントラクトを登録/実行することを可能にするスタックス・ブロックチェーンです。ユニバーサルユーザーネームなどのデジタルアセットによって、ユーザーは自分のデータストレージなどを制御できます。他にも、プライベートデータロッカーのアクセス資格情報をユニバーサルユーザーネームとリンクさせることができます。
2. **ガイア**: ガイア・ストレージシステムは、アプリケーションがプライベートデータロッカーとやり取りできるようにするユーザー制御のストレージシステムです。ユーザーは、これらの暗号化されたデータロッカーをクラウドプロバイダー、ローカルディスク、またはリモートストレージでホストできます。重要なのは、ユーザーが根幹となるプロバイダの選択が可能であることです。ガイア上のデータは暗号化され、ユーザーの暗号化キーによってクライアントサイドで署名されます。ユーザーのデータロッカーは、スタックス・ブロックチェーンの情報ですぐ見つかります。
3. **ブロックスタック認証**: ブロックスタック認証プロトコルは、アプリケーションとの分散認証用のプロトコルです。このプロトコルにより、ユーザーは自分が所有するIDを使用して認証し、そのユーザーのアプリケーションデータを保存するためにどのガイア・ロケーションを使用するべきかについて情報提供します。
4. **ブロックスタック・ライブラリとSDK**: ソフトウェアスタックの最上位には、アプリケーション開発者とユーザーがブロックスタック・ネットワークのさまざまなコンポーネントについて議論できる、開発者ライブラリとSDKがあります。たとえば、ブロックスタック・クライアントソフトウェアを使用すると、ユーザーは自分のIDを登録および管理できます。 ブロックスタックの開発者ライブラリは、開発者が従来のWebアプリケーションを作成するのと同じくらい簡単にブロックスタック・アプリケーションを作成も可能にします。

2. スタックブロックチェーン

ブロックスタック・ネットワークの基本レイヤーはスタックス・ブロックチェーンです。 スタックス・ブロックチェーンは、ネットワークに対するグローバルコンセンサスと調整レイヤーを提供し、スタックス・トークンと呼ばれるブロックスタック・ネットワークのネイティブトークンを実装します。スタックトークンは、ユーザーがユニバーサルユーザーネーム、ソフトウェアライセンス、ストレージロッカーへのポインタなどのデジタルアセットを登録するときに「資源」として消費されます。

このセクションでは、スタックス・ブロックチェーンの高度な設計について説明します。 これらの設計がどのように実装され、進化しているかについての詳細は、

さまざまなコンポーネントのためのスタック改善提案（SIP）を読むことをお勧めします[1]。

[1] https://github.com/blockstack/blockstack-core/tree/develop/sipから入手できます。

更新予定の本稿では、スタックス改善プロセスを通じて、より多くのSIPが受け入れられるようになりました。 スタックス・ブロックチェーンには、次の設計上の決定事項が組み込まれています。

1. リーダー選出のための調整可能なProof-of-Workメカニズム。
2. 従来のブロックチェーンのハッシュパワーを再利用するためのProof-of-Burn マイニングアルゴリズム。
3. ランダムグラフを使用する新しいピアネットワーク（Atlas）がピアの接続性を調べ、合意を達成するのに必要なデータ量を削減。
4. 不完全チューリングで解釈されたスマートコントラクト言語であるクラリティ。

ブロックチェーンバージョン: 現在のスタックス・ブロックチェーンは「バージョン1」で、これは基本的な機能をデプロイするための最初の実装です。 Stacks blockchain　v1はBitcoinネットワークを使用してコンセンサスアルゴリズムを実装し、転送操作などのスタックス・トークン操作をサポートします。 Stacks blockchain v1は、Blockstack Naming System [8]のようなユースケース用のスマートコントラクトを実装しています。バージョン1の実装と機能の詳細については、Githubで利用可能な実装[20]をご覧ください。このセクションの残りの部分では、スタックス・ブロックチェーンの「バージョン2」の設計について説明します。Stacks blockchain v2は、新しいコンセンサスアルゴリズムとスマートコントラクト言語の全機能を実装しており、バージョン1へのメジャーアップグレードとなります。

2.1. リーダー選出

ブロックスタックの第一世代のブロックチェーンはLayer-1（L1）上で動作し、各トランザクションはL1Bitcoinトランザクションと1対1で対応していました。これを行う理由は、ブロックスタックのブロックチェーンを再編成することの難しさが、Bitcoinのブロックチェーンを再編成するのと同じくらい難しいことを確実にするためです。これは、Namecoin[8]のようなより小さなブロックチェーンネットワークに生じたセキュリティ問題から学んだ教訓でもあります。

スタックス・ブロックチェーンは、リーダー選出プロセスに調整可能なProofメカニズムを使用しています。調整可能なProofメカニズムは、複数のメカニズムから入力を受け取り、各入力に与えられる相対重量を調整できるリーダー選出システムです。例えば、調整可能なProofメカニズムは、ネイティブProof-of-Workアルゴリズムに追加された、他のところで確立されているブロックチェーンのハッシュパワーを再利用する機能を組み合わせることができます。調整可能なProofで目指すこと

は、新しいブロックチェーンを安全にブートストラップし、ネイティブのProof-of-Workメカニズムの使用へと徐々に移行することです。現在、調整可能なProofメカニズムは、（a）ネイティブProof-of-Work、および（b）他の仮想通貨に使うProof-of-Burnの2つのコンポーネントで構成されています。

当初、Proof-of-Burnはより重いものでした。Proof-of-Burnは、マイニングプロセスへの参加の関心を表明するため、仮想通貨を燃やすに使われました。リーダーとして選出されるために、候補者は基礎となる仮想通貨（すなわち、Bitcoin）を燃やし、リーダーのブロック内の最初の一連のトランザクションにコミットします。このコミットメントは、リーダーのフォーク選択としても機能します。ブロックのコンセンサスハッシュには、前のブロックヘッダーを内包する必要があります。複数競合するフォークが発生した場合、フォークを失うことで「採掘」を選択したリーダーは、ブロック報償や取引手数料、燃やされた仮想通貨を取り戻すことはできません。

スタックス・ブロックチェーンで使用されているProof-of-Burnメカニズムにより、次のことが可能になります。

高い検証スループット：処理されたスタックトランザクションは、基礎となる「Burn Chain」（すなわちビットコイン）のトランザクション処理レートから切り離されます。Proof-of-Burn選出を使用すると、スタックス・トランザクションのブロック全体が、Burn Chain内の新しいブロックをブロックごとに確認できます。

低遅延ブロック：1人のリーダー選出によって、私達のProof-of-Burnコンセンサスアルゴリズムは現在のリーダーが彼らのスタックス・ブロックのmempoolからの新しいトランザクションを直ちに含めることを可能にします。このブロックストリーミングモデルでは、ブロック内のユーザーへ数秒以内にトランザクションが合併されたことを知らせることができます。

オープンなリーダーシップセット： Proof-of-Burn選出により、誰でもリーダーになることができます。このメカニズムは、スタックス・ブロックチェーンがオープンブロックチェーンであることを保証します（固定されたリーダーシップセットに依存するクローズドブロックチェーン、またはクローズドセットとして機能する代行証明システムとは異なる）。さらに、シングルリーダー選出を実施することにより、当社のコンセンサスアルゴリズムは、リーダーになる予定のリーダーが調整を加える必要がないことが確実になります。

ハードウェアなしでマイニング参加可能：リーダーとして参加するに必要な作業には、従来のProof-of-Workマイニング方式ではなく、仮想通貨の焼却を要します。これによって、マイニングハードウェアを持たずとも、リーダーとしての参加も可能です。最低限の報償量でもかまわないなら、焼却した仮想通貨を収穫できれば誰でもマイニングに参加できます。

フェアなマイニングプール：スタックス・ブロックチェーンは、フェアなマイニングプールをネイティブでサポートしています。ネットワークに参加している人なら誰でも、特定のリーダーの選出を支援するために仮想通貨を焼却ことができます。そのような「ユーザーサポート焼却」を行うユーザーは、リーダーと同等の割合でスタックス・ブロックの報酬を受領します。

フェイルオーバー機能：この設計により、バーンチェーンが不安定になったり、その他スタックチェーンのマイニングも不安定になった場合、スタックチェーンは他のburn chainを利用できます。
当社のProof-of-Burn機能の詳細については、[21]をご覧ください。スタックス・ブロックチェーンに十分なネイティブハッシュパワーが存在する限り、将来Proof-of-

Burn機能を使う機械はあまり多くないかも知れません。

2.2. 調整可能なProof

スタックス・ブロックチェーンには、コンセンサスアルゴリズムにProof-of-Burn機能に加えて、ネイティブProof-of-Work（PoW）機能を含めています。この組み合わせにより、SIP-001で説明されているProof-of-Burns選出システムと供に、チェーンのセキュリティ責任を共有することができます。このネイティブProof-of-Work とProof-of-Burnsの組み合わせは、私たちのシステム上、調整可能なProofの現在仕様の実装で、ネイティブのProof-of-Workマイニングの導入を信頼できるものにします。それは、Proof-of-Workマイニングの際、Proof-of-Workのセキュリティ監視が弱まっても、Proof-of-Burnsがセキュリティを保つ仕組みになります。この調整可能な側面は、burnchainが劣化した時、マイグレーションへの道を提示することもあります。 また調整可能なProofは、調整可能な方法でその他のProof-of-WorkやProof-of-Stake の研究と、それらを過去数年間少しずつ披露することを可能にしています。

リーダー選出におけるProof-of-Workは、リーダー候補が彼らのProof-of-Workナンスを任意に内包することを可能にする機能を持ちます。ナンスの生成に投入された作業量は、候補の焼却量としてカウントされます。最初は、ネイティブのProof-of-Workに5%の上限があります。より多くの事項が具体化される次第、このセクション（および対応するSIP）は更新される予定です.

2.3. Atlasピアネットワーク

Atlasピアネットワークは、各ピアがネットワーク内の他のどのピアが追跡されているかを追跡し、各ピアがネットワーク内のすべてのデータの完全なレプリカの保存を試みるゴシッププロトコルを実装する連想メモリ・ピアネットワークです。ネットワークの容量は、スタックス・ブロックチェーンによってレート制限をされています。データセット内の新しいエントリは、スタックス・ブロックチェーン上のトランザクションに関連付ける必要があります。アトラスピアネットワークは、スタックス・ブロックチェーンのサブシステムとして機能します。ノードがネットワークに参加したり離脱したりする問題を回避するため、非構造化ピアネットワークとして設計されています[18、22]。さらに、すべてのノードはすべてのデータのレプリカを保存し、データのインデックスはスタックス・ブロックチェーンから入手できるため、新しいAtlasノードは、他のピアからどのデータを保存するかを事前に知っているので、迅速にデータにシンクできる特徴を持ちます。これは、一般的にピアツーピアネットワークのノードでは利用できません。

Atlasネットワークは、スタックス・ブロックチェーンの「拡張ストレージ」サブシステムとして機能します。私たちの設計アプローチは、スタックス・ブロック

チェーン自体と直接作用することにできるだけ頼らず、最小限のデータを保存することです。ブロックスタック・NamingSystem（BNS）スマートコントラクト[8]など、ブロックスタック上の多くのアプリケーションでは、不変性を持って、タイムスタンプ付きのデータを保存するメカニズムが不可欠です。BNSでは、ユーザーネームと、そのユーザーのプロファイルとアプリケーションデータを見つけるために使用されるルーティング情報を関連付けるため使用されます。ほとんどのブロックチェーンでは、この種類のデータはブロックチェーンそのものに直接保存されます。それと違ってアトラスピアネットワークは、代わりにハッシュをブロックチェーン（スペースが高価な場合）に保存し、それらのハッシュに対応するデータを交換するための個別のピアリングネットワークを実装する方式を採択しています。

2.4. スタックトークンの使用

スタックス・ブロックチェーンによって実装されたネイティブのスタックス・トークンは、ブロックスタック・ネットワーク上でいくつかの基本操作を使用可能にします。

1. **デジタルアセットを登録するための資源**：スタックス・トークンは、ユーザーネーム、ドメインネーム、ソフトウェアライセンス、Podcast、その他さまざまな種類のデジタルアセットを登録するために使用されます。
2. **スマートコントラクトを登録/実行するための資源**： スマートコントラクトを実行するには、スマートコントラクトの正確性の検証と実行のコストをまかなうため、資金が必要です。スタックトークンは、スマートコントラクトをスタックブロックチェーンに保存するためのコストをカバーするためにも使用されます。
3. **トランザクション手数料**：タックス・トークンは、スタックス・ブロックチェーンにトランザクションを内包させるためのトランザクション手数料を支払うために使用されます。
4. **アンカー付きアプリチェーン**： ブロックスタックで非常にポピュラーになったアプリのために、私たちのブロックチェーンは、アプリがスタックス・ブロックチェーンの上でそのブロックチェーンを初期化することができるスケーラビリティオンランプを持っています。そのような「アプリチェーン」は、彼らの採掘と進歩のためにスタックスを焼却します。

上記のリストは完結したわけではありません。ブロックスタック・ネットワークが成熟するにつれて、ネットワーク参加者がスタックス・トークンの他の用途を発見して発明することを期待しています。現在、トークン所有者が「App Mining」と

呼ばれる開発者向けインセンティブプログラムに参加できる「App Staking」メカニズムを積極的に研究しています[23]。

3. クラリティという名のスマートコントラクト言語

スタックス・ブロックチェーンは、デジタルアセットのプログラム制御を可能にするスマートコントラクトの起動と実行をサポートします。このクラリティと名づけた新しいスマートコントラクト言語は、セキュリティと予測可能性を最適化します。これにより、以前のスマートコントラクトシステムとは異なる、いくつかの重要な設計目標が実現されました。

1. 実行時間とスペースの要求と相まって、この言語は迅速で正確な静的分析を事前許可しなければなりません。これをサポートするために、この言語は単一のトランザクションの実行に対して完全ではありません。 しかし、トランザクションの全履歴を取得すれば、言語はチューリング完了の状態になります。
2. スマートコントラクトは、コンパイルされたものではなく、当社のVMによって解釈されるべきです。 開発者によって書かれたコードは、ブロックチェーンに直接デプロイする必要があります。

上記の2つの特性を達成するため、スマートコントラクト向けに特別に設計された新しいLISP-variantを作成しました。 クラリティのデザインに関するより詳細な議論については、SIP-002 [24]をご覧ください。

1. 言語の概要

クラリティは他のLISP系列の言語（例えばScheme）と似ていますが、以下の違いがあります。

1. 再帰は禁止されており、ラムダ関数はありません
2. Looping はmap、filter、またはfoldによってのみ実行できます。
3. アトミック型は、ブーリアン、整数、固定長バッファ、およびプリンシパルに制限されます。
4. アトミック型のリストに対する追加のサポートもありますが、言語で唯一の長さ可変のリストは関数入力として現れます（すなわち、appendやjoinのようなリスト操作に対するサポートはありません）。ネーム付きで、タイピングされたタプルもサポートしています。
5. 変数はletバインディングによってのみ生成され、setのような関数のmutatingはサポートされません。

6.defineステートメントを使用してコードを単純化するために、定数と関数を定義することができます。ただし、これらは純粋に構文上のものです。定義がインライン化できない場合、契約は違法であると拒否されます。このように定義された関数は、特定のスマートコントラクトで定義された他の関数によってのみ呼び出されるので、これらの定義も非公開です

7.public-define文によって指定された機能が、パブリック機能です。これらの関数への引数はそれらの型を指定しなければなりません。

スマートコントラクトは以下のことを可能にします。

1.他のスマートコントラクトからパブリック機能を呼び出します。これらのスマートコントラクトはハッシュによって識別され、呼び出し元のスマートコントラクトが発行された時点ですでに存在する必要があります。これは、再帰の違法性と相まって、従来のスマートコントラクトプラットフォームでよく見られる攻撃方法である機能の再入を防ぐためです。

2.デジタルアセットを所有および管理します。スマートコントラクトは、公開キーまたはマルチシグネチャアドレスと同じく一次プリンシパルです。

各スマートコントラクトには独自のデータスペースがあります。このデータスペース内のデータはマップに格納されます。これらのストアは、型付きタプルを別の型付きタプルに関連付けます（型付けされたキーバリューストアとほとんど同じです）。テーブルデータ構造とは対照的に、マップは与えられたキーを正確に一つの値に関連付けるだけです。

スマートコントラクトは、他のスマートコントラクトマップからデータを取得できます。ただし、スマートコントラクトだけが独自のマップ内のデータを直接更新できます。

次の2つの理由から、他のデータ構造ではなくデータマップの採択にいたりました。

1.データマップの単純さにより、VM内での単純な実装と、機能に関するより簡単な推論の両方が可能になります。与えられた関数定義を調べることによって、どのマップが変更されるのか、そしてそれらのマップの中でさえ、どのキーが呼び出しに影響を受けているかが明らかです。

2.データマップのインタフェースは、マップ操作の戻り値の型が固定長であることを保証します。これは、スマートコントラクトの実行時間、コスト、およびその他のプロパティの静的分析に必要です。

2. チューリングの不完全性と静的解析

チューリングではない完全な言語を作成することは、設計上の重要な考慮事項でした。ブロックチェーンという過酷な環境でのプログラミングには、これから多くの利点があります。

1. 不完全チューリングは、静的分析に与えられたトランザクションを構築するコストを与えさせることを可能にするネットワークには、トランザクションへの料金を事前に正確に知らせることができます。よって、クライアントもトランザクションのブロードキャストにかかるコストを知らされ、ユーザーにこれらの情報を手軽に伝えられることになり、クライアントの動作も改善されます。
2. 不完全チューリングは、静的分析を単一トランザクションが他のどの契約を呼び出すことができるかなど、重要なプロパティをすばやく判断できるようにします。これによって、クライアントは特定のトランザクションから起りうる副作用についてユーザーに警告できるため、ユーザーエクスペリエンスが向上します。
3. 改良された正確な静的分析は、スマートコントラクトが開始される前、起こりうるすべての障害や手違いについて、プログラマーが気楽に分析できるようにします。

基本的に、スマートコントラクトプログラミングを他の形式のプログラミングと同じように扱うのは間違いだと思います。ブロックチェーンの性質は、スマートコントラクトの詳細性を非常に重要なものと捉えます。そして、プログラミングの容易さと引き換えに得た、人間と機械のスマートコントラクトに対する増進した理解は良い取引になったと思われます。スマートコントラクトの従来の実用性がこれを証明します。

完全チューリングスマートコントラクトの歴史は基本的にスマートコントラクトのバグの歴史です。

クラリティでは、スマートコントラクトをブロードキャストする前に実行された静的分析によって、次のような情報が得られます。

1. 入力サイズの関数として与えられたトランザクションをブロードキャストするためのコスト。

2. 特定のテーブルを変更できるトランザクションのセット。

将来の研究開発では、スマートコントラクトコードの証明を自動的にチェックする機能など、さらに高度な分析機能のサポートが可能になる見込みです。

3. 解釈言語とコンパイル言語

クラリティにおける2番目の重要な設計上の決定は、コンパイルされた言語ではなく、解釈された言語を採択したことです（たとえば、WASMへのコンパイル）。コンパイラを採択しないという私たちの設計上の決定は、現代のアプローチとの根本的な違いです。この設計決定の主な理由は、実装のバグについて推論する能力です。

バグは現実上の事実であり、たとえ最高のコーディング標準があっても、バグは避けられません。これは他のコードにも当てはまるのと同様にスマートコントラクトバグ（ブロックチェーン）にも当てはまります。スマートコントラクトのバグは、扱いが複雑です。さまざまなブロックチェーンコミュニティが「コードは法律です」という哲学を遵守しており、その人たちにとっては、ブロックチェーンに献呈された規則は究極なる真実の源です。スマートコントラクトを作成している開発者は、その意図をソースコードで表現しますが、コンパイルはその意図を実際のルールに変換します。これは、コンパイラのバグのために、実際のルールが開発者の意図とは異なる状況になります。これは厄介な状況につながります。 開発者の意図がより重要であるのか、それともルールがより重要であるのかについて人々は議論しています。スタックス・ブロックチェーンでは、コンパイルステップを削除し、開発者の意図がルールから逸脱しないように開発者の意図を直接ブロックチェーンにコミットすることで、この状況を回避します。

スマートコントラクト言語（つまりVM）の実装にバグがある場合を考えてみましょう。スマートコントラクト言語がインタプリタを使用している場合、バグ修正は比較的簡単に適用できます。世界のコントラクトコードはすべてブロックチェーン自体にあります。インタプリタにバグ修正を適用して、ブロックチェーンを生成から再開することができます（すべてのトランザクションを再適用する）。

しかし、スマートコントラクト言語がコンパイルされていて、バグがVMではなくコンパイラにある場合は、改善策はそれほど明白ではないため、より議論の余地があります。これは、コンパイラのバグにより、生成されたコード（ブロックチェーンで最終的にブロードキャストされるコード）が開発者の意図したコードとは異なる動作を起こす可能性があるためです。暗号コミュニティで見られる「コードは法律です」という哲学では、この状況はもっと複雑です。開発者によって書かれたコードは正しいですが、ブロックチェーン上で生成されたトランザクション自体は間違っています。すべての開発者のソースコードを収集して再コンパイルするのは現実的ではありません。特に、ソースコードが変更されていないことを確認できない場合は特にそうです。実際には、そのような状況では、ブロックチェーンに公開されているコードがほとんどの場合、真実の根本的な原因であると考えられます。そ

の場合、開発者は自分のソースコードではなく、そのコードについて推論し検証する必要があります。高水準の解釈された言語を使用することは、正しいスマートコントラクトの実行を確実にするために最も重要であると考えます。

4. ガイア：ユーザー統制ストレージ

ブロックスタックは、ガイア・ストレージシステムを使用してデータ制御権をユーザーに属します。これは、アプリケーションがプライベートデータロッカーとの相互作用を可能にするユーザー制御のストレージシステムです。これらのデータロッカーは、クラウドプロバイダーまたは他のデータストレージプロバイダーでホストされます。重要なのは、ユーザーがどのプロバイダを使用するかを選択できることです。 ガイア上のデータは暗号化され、ユーザーが制御する暗号化キーによる署名がされています。論理的には、ガイアはファイルを保存するため、マウントできるワイドエリアファイルシステムとして機能します。

ガイア・ストレージシステムでは、ユーザーはデータを保存するガイア・ストレージの場所を指定します。 ガイアの場所への「ポインタ」のみがスタックス・ブロックチェーン（およびAtlasサブシステム）に保存されます。ユーザーがブロックスタック認証プロトコル5を使用してアプリケーションやサービスにログインすると、その位置をアプリケーションにパスします。この情報を使用して、アプリケーションデータをユーザーが指定したストレージに保存されるように、アプリケーションは指定されたガイア・ストレージロッカーと通信する方法を学びます。

ガイアの設計思想では、エンドユーザーが基盤となるクラウドプロバイダーを信頼するという負担をなくすため、従来のクラウドプロバイダーとインフラストラクチャーを再利用します。私たちはクラウドストレージプロバイダ（Amazon S3、Google Cloud Storage、あるいは単なるローカルディスクなど）を「ダムドライブ」として扱い、それらに暗号化または署名されたデータを保存します。クラウドプロバイダーはユーザーのデータ似触れることはできず、暗号化されたデータBLOBのみに接触可能です。

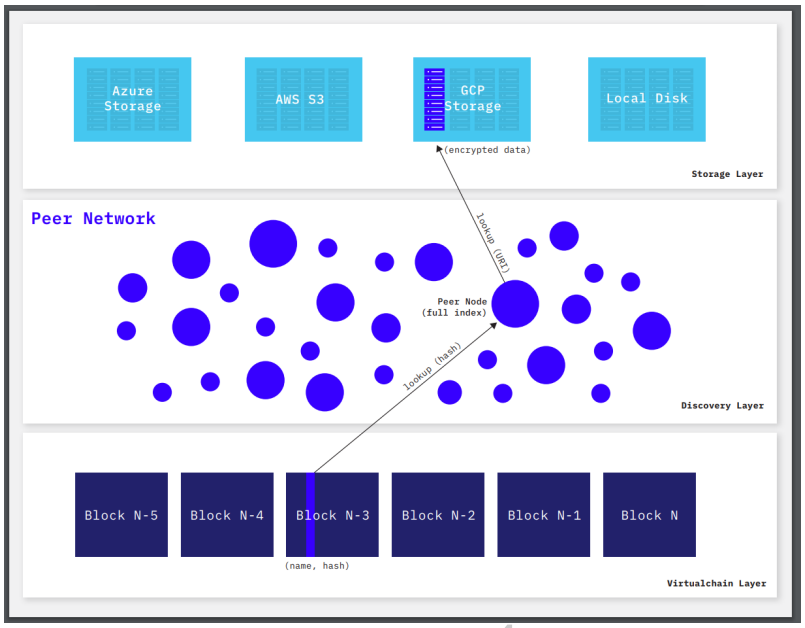

図1：ガイアの概要とデータ検索の手順。

さらに、関連する公開キーまたはデータハッシュはスタックス・ブロックチェーンを通じて検出可能であるため、クラウドプロバイダーはユーザーのデータを改ざんすることはできません。

ガイア・サーバーにデータを書き込むには、そのサーバー上の適切な場所にデータをPOSTします。これらのPOSTは、そのような書き込み要求が署名付き認証トークンを持っていることを確認することで、サーバーに検証されます。このトークンは、書き込まれる特定のバケットを制御する秘密キーによって署名されています。ユーザーが使用する可能性があるアプリケーションごとに別々のバケットをプロビジョニングするために、ユーザーはそれらのアプリケーションごとに別々の秘密キーを導出します。これらの秘密キーはそれぞれ、ガイア・サーバー上の特定のバケットへのアクセスのみを許可します。

ガイアでは、ユーザーのブロックチェーン検証済みルーティング情報に、署名されたJSONオブジェクト（ユーザーネームのオーナーキーによって署名された）を表すURLが含まれています。この署名付きJSONオブジェクトには、ユーザーのガイア・データロッカーを表すURLが含まれています。アプリケーションがユーザーのガイア・データロッカーの場所を知るなら、通常のHTTPリクエストを使用してその場所からファイルを要求するだけです。別のユーザーによって作成されたファイルを検索するため、アプリケーションはこれらの順次検索を完全にクライアント側で実行できます。これは初期の検索に待ち時間のペナルティを課しますが、このルーティング情報の多くはブラウザ（またはネイティブアプリケーション）によってローカルに

キャッシュされるため、以降の検索はインターネット上の従来のデータフェッチと同じくらいの速度になります。

図1にガイアの概要を示します。 werner.idのようにネームのデータを検索します。

次のように動作します。

1. スタックブロックチェーンでネームを検索して、（ネーム、ハッシュ）のペアを取得します。

2. ブロックスタックのAtlasピアネットワークでハッシュ（ネーム）を検索して、ネームのルーティング情報ファイル取得します。

3. ルーティングファイルからユーザーのガイア URLを取得し、そのURLを検索してストレージバックエンドに接続します。

4.指定されたガイア・サービスからデータをGET / PUTし（必要ならば、そしてアクセス権を持っていればそれを解読し）、それぞれの署名またはハッシュを検証する。

上記のステップ1と2は、/ v1 / names / <name>エンドポイント。ブロックスタック・coreを1回呼び出すものです。
これらの反復的読み書きは、開発者ライブラリで自動的に処理されます。

パフォーマンス: 私たちのアーキテクチャの目標は、クラウドプロバイダー上に構築された従来のインターネットアプリケーションに匹敵する性能を与えることです。まず、意味のあるセキュリティとフォールトトレランスの利点をえるため、平均的な制御への障害の中心的な要素を排除することで、読み取り/書き込みパフォーマンス性能は少し低下しますが、通常のユーザーにとっては重要でも顕著でもないほどで、多少の性能低下を受け入れる価値があります。ガイアの読み書きのパフォーマンスは、基盤となるストレージに匹敵する速度でファイルを読み書きすることを実証しました。 ガイアは、暗号化のため、ファイルあたりオーバーヘッドが無視できる一定のストレージスペースを追加します（およそ5%大きいファイルになる）。暗号化にはCPUのオーバーヘッドがありますが、ファイルサイズの違いは非常に小さいため、読み取り/書き込みのネットワークパフォーマンスは、基盤となるストレージサービスに直接アクセスする時とほぼ同じです。

システムのスケーラビリティ: 私たちのアーキテクチャのストレージレイヤーはスケーラビリティを邪魔するわけではありません。現代のクラウドストレージシステムは非常にスケーラブルです[25]。また、Atlasネットワークは個々のユーザーファイルやファイルチャンクをインデックス化するのではなく、ユーザーのストレージバックエンドへのポインタをインデックス化するため、スケーラビリティに優れています。ストレージバックエンドは大量のデータの読み書きを処理します。アトラスネットワークは、（a）ユーザーが自分のストレージバックエンドまたは公開キーマッピングを変更または更新しているとき、または（b）新しいユーザーがシステム

に登録しているときにのみ関与します。システム。新しいドメイン/ユーザーを登録するときは、ルーティングファイルのハッシュをブロックチェーンでアナウンスする必要があります。ブロックチェーンは、（Atlasネットワークと比較して）スケーラビリティの邪魔となる可能性がありますが、どのユーザーにとっても非常にまれにしか経験できないことです。さらに、オフチェーンのネームレジストラを使用すると、数十万のユーザーを単一のブロックチェーントランザクションにレジスト可能で、一日数百万のユーザーの登録をサポートできるので、従来のクラウドベースのプラットフォームでの1日あたり新規ユーザー登録の数と変わりません。） ガイアを実際に何十億ものユーザーに拡張すれば、スケーラビリティにそれなりの問題が生じうるでしょう。しかし現時点ではさほど差し支えはなく、その課題を解決するのは、現在進行中の研究と将来のタスクになります。

5. 認証

ユーザーアカウントはインターネットアプリケーションを使用するために不可欠です。 ブロックスタックは、パスワードなしでもすべてのアプリケーションにわたって機能するユニバーサルユーザーネームを提供します。パスワードベースの認証の代わりに、ユーザーは暗号化公開キーを使用して認証します。ローカルで実行されているソフトウェアクライアントは、それぞれのアプリケーションからのサインイン要請とサインの検証要請を処理します。

当社の認証プロトコルであるブロックスタックAuthは、アプリをユーザーのガイア・ハブやアプリ固有の秘密キーにも接続します。これは、ユーザーとアプリケーションがともにユーザーデータを保存し、他のユーザーによって作成されたデータを検証するために使用されます。

5.1. シングルサインオン

ブロックスタックAuthは、認証に公開キー暗号を使用します。ユーザーは、署名付きデータを生成および保存する機能をアプリケーションに付与するためにアプリケーションにサインインします。他のユーザーもこれを読んで認証することができます。これは、サインインしたユーザーが正当であることを他のユーザーに証明します。

ブロックスタックでは、サインインの目的は、認証データを生成および保存するために十分な情報をアプリケーションクライアントに提供することです。つまり、認証機能はオーセンティケーターアプリの形でユーザーのコンピューター上でのみ実行できます。すべてのネームがスタックス・ブロックチェーンに登録されているため、すべてのアプリケーションおよび認証アプリは常に（1）存在するすべての

ネーム、（2）すべての公開キーおよびガイア・ハブの最新情報を持ちます。これにより、サーバーサイドのアイデンティティプロバイダが不要になります。

アプリケーションクライアントは、ユーザーデータを認証するためにスタックス・ブロックチェーンピアと通信可能である必要があるだけです。これを行うために、ユーザーはサインインの際、自分の優先スタックピアのネットワークアドレスをアプリケーションに提供します。

ユーザーは「ログイン」ボタンをクリックしてブロックスタック・アプリケーションにサインインします。アプリケーションは（blockstack.js SDKを介して）サインインの要求と共にユーザーを自分のBlockstack認証アプリにリダイレクトします。ユーザーには、サインインに使用するブロックスタックIDの選択とそのリストが表示されます。IDを選択すると、オーセンティケーターはユーザーをアプリケーションに戻し、アプリケーションに3つの情報をパスします。

1. ユーザーのユーザーネーム（まだネームがない場合は、公開キーのハッシュ）。
2. ユーザーのデータを暗号化して署名するためのアプリケーション固有の秘密キーは、ユーザーのマスタープライベートキーにより生成され、サインインおよびアプリケーションのHTTP オリジンに使用されます。
3. 他のユーザーとそのデータを検索するために使用する、ユーザーのガイア・ハブと優先スタックス・ブロックチェーンピアへのURL。

これを行うことで、ユーザーは自分のユーザーネームを提示し、自分のデータを見つけて保存できる場所についてアプリケーションに指示します。それから、独自のストレージまたはIDソリューションなしで、アプリケーションは固有データを永続的に読み込み、書き込みができ、他のユーザーのアプリケーションの固有データにアクセスできます。

サインアウトするのは、単にアプリケーションのローカルステートを消去し、それによってWebブラウザとクライアントにアプリケーション固有のプライベートキーを忘れさせることになります。

6. ブロックスタックのライブラリとSDK

公益法人であるBlockstack PBCは、オープンソースの貢献者とともに、ブロックスタックのコアプロトコルと開発者ライブラリを開発します。開発者ライブラリは開発者がブロックスタック・ネットワークでアプリケーションを構築することをよ

り簡単にし、ブロックスタック・クライアントはユーザーがブロックスタック・ネットワークのさまざまなコンポーネントやアプリケーションと相互作用することを可能にしています。

6.1. 開発者ライブラリ

ブロックスタックは、分散アプリケーションの開発を開発者にとってできるだけ簡単にするように設計されています。 スタックス・ブロックチェーンと分散ストレージの複雑さは、大体アプリ開発者と隔離されており、開発者はアプリロジックだけに集中することができます。ブロックスタック・オープンソースリポジトリには、JavascriptWeb SDK（blockstack.js）と、iOSおよびAndroid用のモバイルSDKといったさまざまなプラットフォーム向けの開発者ライブラリが含まれています。これらのライブラリはすべてMITライセンスの条項に基づいて提供されており、https：//github.com/blockstackから入手できます。

これらのライブラリは、認証プロトコルの実装、ガイア・サーバーとの直接のやり取り、およびスタックス・トランザクションの生成に必要なすべてのAPIとコードを提供します。これらのライブラリを使用することで、開発者は従来のアプリケーションを開発しているのと同じくらい簡単にユーザーのセキュリティとプライバシーを尊重する分散型アプリケーションを作成が可能です。

Radiks 複雑なソーシャルグラフデータを発信するためのアプリケーションの場合、Radiksシステムは、それらのデータを元にして索引を作成するのに最も適したサーバーおよびクライアントライブラリです。
Radiksライブラリを使用すると、開発者はアプリ内のフィールド値でクエリできるクロスユーザー構造化データコレクションを作成できます。これにはインデックスとクエリを処理するサーバーサイドコンポーネントが必要ですが、決定的にはユーザーの信頼できるコンピューティングベースの一部ではありません。データ暗号化テキストと、インデックスを介したクエリの構築および回答に必要なメタデータのみが表示されます。 Radikに関するより詳しい情報は[26]をご覧ください。

6.2. ユーザーソフトウェア

アプリケーション開発者はSDKやライブラリを使用してブロックスタック・ネットワークとやり取りできますが、ユーザーもネームの登録、ガイア・サーバーの指定、アプリケーション認証などの機能を実行するに必要です。ブロック-スタックエコシステムは現在、ユーザーとネットワークの疎通をサポートする2つのオープンソースプロジェクトを提供しています。

1. **ブロックスタック・ブラウザー：**これは現在オーセンティケーターアプリのリ

ファレンスオープンソース実装であり、ユーザーは利用可能なブロックスタック・プリケーションを閲覧し、ユーザーネームを登録し、そしてアプリケーションで認証することができます。デスクトップにローカルにインストールする場合と、Webにデプロイされたフォームの場合に利用できます。

2. **Blockstack CLI：** これは、パワーユーザーや開発者がブロックスタックのプロトコルとやり取りできるようにするコマンドラインユーティリティです。認証機能を提供するだけでなく、ユーザーは生のトランザクションを生成し、ガイアを使って高度なデータ管理タスクに取り組むことができます。

6.3. ドキュメントとコミュニティのリソース

ブロックスタック・オープンソースコミュニティは、GithubおよびHttps：// docsblockstack.orgで入手可能なチュートリアル、APIドキュメント、およびシステム設計ドキュメントを管理しています。

ブロックスタックのユーザーと開発者は、以下の公式コミュニティリソースを利用できます。

- **Github:** すべてのソフトウェア開発はGithub（https：//github.com/ blockstack）を通じて行われます。
- **フォーラム:** パワーユーザーと開発者は、ブロックスタックフォーラム（https：// blockstack.orgforum）で技術的な質問に答えたり、アイデアを共有したり、互いに助け合ったりします。
- **Slack:** ブロックスタック・コミュニティはhttps://blockstack.slack.comで、リアルタイムチャットができるパブリックSlackグループを開設しています。

7. アプリとサービス

2019年初めの時点で、ブロックスタック上に100以上のアプリケーションが構築されています。開発者はさまざまな種類のアプリケーションを構築しており、増加中のブロックスタック・アプリケーションのリストはapp.coをご覧ください。ブロックスタックはモジュール式であるため、さまざまなアプリケーションがさまざまなコンポーネントを個別に利用できます。以下に、いくつかのユースケースの概要を提供します。

ブロックスタック上の現在のオフィス向けアプリケーション[27、28、29、30]は、ブロックスタックAuthおよびガイア・ストレージを使用して、ユーザーが文書を作成、編集、および共有できるようにしています。ユーザーが互いの文書を見つけやすくするために、これらのアプリはブロックスタック・プロファイル検索インデク

サーを利用しています。プロファイルのセットがグローバルに表示され、検出可能であるため、誰でもプロファイル検索インデクサーを展開して実行できますので、この検索インデクサーは分散化されています。

ブロックスタック・エコシステムには、ソーシャルアプリケーションも多数含まれています[31、32、33、34]。これらは通常、ユーザーが他のユーザーのデータを効率的に発見して取得できるようにするため、Radiksサーバー展開と組み合わせてブロックスタック・ Authを使用します。例えば、アプリは暗号化されたメッセージを多数のユーザーにルーティングするために専用のリレーチャネルを使用します[31]。 ブロックスタック上のパブリッシュおよびストレージアプリケーション[35、36、37、38、39、40]は、ユーザーデータを保存するだけでなく、従来のHTTP URLを介して非ブロックスタックユーザーと共有するためにもガイアを使用します。

開発者のインセンティブ　　スタックス・ブロックチェーンは、アプリケーション開発者がネットワーク上で高品質のアプリケーションを公開することによってスタックス・トークンを「マイニング」できる新しいマイニングの概念を広げます。 App Miningと呼ばれるこのメカニズムは、ネットワーク上で高品質のアプリケーションを掲載させるためのインセンティブメカニズムとして設計されています。 App Miningプログラムは現在、Blockstack PBCによって何人かの独立したレビュー担当者と共に運営されています。開発者は、月に1回レビューを受けるアプリケーションを送信し、アプリのランク付けメカニズムにおけるアプリのパフォーマンスに基づいて支払いを受け取ることができます。アプリは、優れたアプリと評価するそれぞれ独自の基準を持つ、独立した審査員団によって評価されます。アプリケーションの総合スコアはその順位を決定し、ランキングと支払いは毎月行います。App　Miningプログラムの詳細はこのペーパーの範囲外であり、詳細についてはhttps://app.co/miningをご覧ください。

8. まとめ

ブロックスタックは、分散アプリケーションを構築するための開発者にフルスタックを提供する分散コンピューティングネットワークです。現在までに、100を超える分散型アプリケーションがネットワーク上に構築されています。 ブロックスタックを使用すると、開発者はサーバーやデータベースを駆動する必要がなくなります。アプリはユーザーの制御下にあるプライベートデータロッカーのデータを書き込みます。この分散型ストレージシステムは、従来のクラウドストレージに匹敵するパフォーマンスを提供し、暗号化/復号化にわずかなオーバーヘッドを加えるだけです。この認証プロトコルは暗号認証より安全性が低いことが知られているパスワードベースのログインの必要性を取り除き、ユーザーは、さまざまなサービスやアプ

リケーションを単一のアカウントを利用できるため、新しいアカウントやサービスを繰り返して作成する必要がなくなります。当社の開発者ライブラリは、このプラットフォームでの分散型アプリケーションの開発を従来のインターネットアプリケーションの構築と同じくらい簡単にします。

本稿では、ブロックスタックの最新設計について紹介しました。 2016年と2017年の初期の実装以来、ブロックスタックのコアデザインは、製品展開から学んだ教訓、および分散アプリケーションの開発者からのフィードバックを反映し進化しつつあります。以前の（2017）ホワイトペーパーとの主な変更点は、（a）新しいブロックチェーンの安全なブートストラップのため、調整可能なproofメカニズムを使うスタックス・ブロックチェーンの説明 、および（b）新しいスマートコントラクト言語の説明です。それはセキュリティとスマートコントラクトの予測可能性に焦点を当てています。 ブロックスタックをオープンソースとしてリリースし、一般に公開されています[41]。

謝辞

長年にわたって多くの人々がブロックスタックの設計と実装に貢献してくれました。Larry Salibra氏、Ken Liao氏、Guy Lepage氏、Patrick Stanley氏、John Light氏の初期貢献とアイデアについて、 Hank Stoever氏、Shreyas Thiagaraj氏、Matthew Little氏の開発者ライブラリとSDKの開発協力について、 Jeff Domke氏、Mark Hendrickson氏、Thomas Osmonson氏、Jasper Jansz氏、Mary Anthony氏の製品設計および開発者向けのドキュメント作成についてJesse Wiley氏、[redacted]Tim Wells氏のインフラストラクチャー開発について、Brittany Laughlin氏、Diwaker Gupta氏の優れた助言とフィードバックについて、誠に感謝しております。

その他、ブロックスタックへの貢献している方については、//github.com/blockstackをご覧ください。事実上、数えきれないほど多くの貢献者に貢献いただき、感謝の言葉もございません。ブロックスタック・オープンソースコミュニティへのすべての支援に感謝申し上げます。

参考文献

[1]S. Sulyman, “Client-server model,” IOSR Journal of Computer Engineering, vol. 16, pp. 57–71, 01 2014.

[2]N. Perlroth, “Yahoo says hackers stole data on 500 million users in 2014,” Sept. 2016. http://nyti.ms/2oAqn0G.

[3]K. Granville, “Facebook and cambridge analytica: What you need to know as fallout widens,” Mar. 2018. https://nyti.ms/2HP4Dr3.

[4]R. Mcnamee, “I mentored mark zuckerberg. i loved facebook. but i can’t stay silent about what’s happening.,” Jan. 2019. http://time.com/5505441/mark-zuckerberg-mentor-facebook- downfall/.

[5]“Blockstack website,” 2019. http://blockstack.org.

[6]J. H. Saltzer, D. P. Reed, and D. D. Clark, “End-to-end arguments in system design,” ACM Trans. Comput. Syst., vol. 2, pp. 277–288, Nov. 1984.

[7]D. D. Clark and M. S. Blumenthal, “The end-to-end argument and application design: The role of trust,” Federal Comm. Law Journal, vol. 63, no. 2, 2011.

[8]M. Ali, J. Nelson, R. Shea, and M. Freedman, “Blockstack: A global naming and storage system secured by blockchains,” in Proc. USENIX Annual Technical Conference (ATC ’16), June 2016.

[9]J. Nelson, M. Ali, R. Shea, and M. J. Freedman, “Extending existing blockchains with virtualchain,” in Workshop on Distributed Cryptocurrencies and Consensus Ledgers (DCCL’16), (Chicago, IL), June 2016.

[10]M. Ali, J. Nelson, R. Shea, and M. J. Freedman, “Bootstrapping trust in distributed systems with blockchains,” USENIX ;login:, vol. 41, no. 3, pp. 52–58, 2016.

[11]Satoshi Nakamoto, “Bitcoin: A peer-to-peer electronic cash system,” tech report, 2009. https://bitcoin.org/bitcoin.pdf.

[12]A. Narayanan, J. Bonneau, E. Felten, A. Miller, and S. Goldfeder, Bitcoin and Cryptocurrency Technologies: A Comprehensive Introduction. Princeton, NJ, USA: Princeton University Press, 2016.

[13]V. Buterin, “A next-generation smart contract and decentralized application platform,” tech. rep., 2017. https://github.com/ethereum/wiki/wiki/White-Paper.

[14]“Filecoin: A Cryptocurrency Operated File Network,” tech report, 2014. http://filecoin.io/filecoin.pdf.

[15] https://eos.io.

[16]T. Hanke, M. Movahedi, and D. William, "Dfinity technology overview series consensus system rev. 1," 2018. https://dfinity.org.

[17]"Ethereum 2.0 specifications," 2019. https://github.com/ethereum/eth2.0-specs.

[18]Eng Keong Lua, J. Crowcroft, M. Pias, R. Sharma, and S. Lim, "A survey and comparison of peer- to-peer overlay network schemes," IEEE Communications Surveys Tutorials, vol. 7, pp. 72–93, Second 2005.

[19]“Oauth.” https://oauth.net.

[20]“Blockstack Core: Stacks blockchain v1,” 2018. https://github.com/blockstack/blockstack-core/tree/v20.0.8.1.

[21]“SIP 001: Burn Election,” 2019. https://github.com/blockstack/blockstack-core/blob/ develop/sip/sip-001-burn-election.md.

[22]I. Stoica, R. Morris, D. Karger, M. F. Kaashoek, and H. Balakrishnan, “Chord: A scalable peer-to-peer lookup service for internet applications,” in Proceedings of the 2001 Conference on Applications, Technolo- gies, Architectures, and Protocols for Computer Communications, SIGCOMM ’01, pp. 149–160, 2001.

[23]“App mining.” https://app.co/mining/.

[24]“SIP 002: Smart Contract Language,” 2019. https://github.com/blockstack/blockstack- core/blob/develop/sip/sip-002-smart-contract-language.md.

[25]“Google Cloud Storage SLA.” Retrieved from https://cloud.google.com/storage/slain May 2017.

[26]“Radiks.” https://github.com/blockstack-radiks.

[27]J. E. Hunter, “Graphite docs,” Feb. 2019. https://app.graphite-docs.com. [28]D.

Travino, “Noteriot,” Feb. 2019. https://note.riot.ai/.

[29]“Forms.id,” Feb. 2019. https://forms.id. [30]“Blockusign,” Feb.

2019. https://blockusign.io.

[31]P. Bhardwaj and A. Carreira, “Stealthy,” Feb. 2019. https://www.stealthy.im. [32]A.

Sewrathan, R. Adjei, and F. Madutsa, “Afari,” Feb. 2019. https://afari.io. [33]T. Alves,

“Recall,” Feb. 2019. https://app.recall.photos/.

[34]T. Alves, “Travelstack,” Feb. 2019. https://app.travelstack.club.

[35]J. E. Hunter, “Graphite publishing,” Feb. 2019. https://publishing.graphitedocs.com. [36]“Decs,” Feb.

2019. https://app.decs.xyz.

[37]“Sigle,” Feb. 2019. https://app.sigle.io. [38]“Xorbrowser,” Feb.

2019. https://xorbrowser.com. [39]“Mevaul,” Feb. 2019. https://

mevaul.com/. [40]“Xordrive,” Feb. 2019. https://xordrive.io.

[41]“Blockstack source code release v20.0.8,” 2019. http://github.com/blockstack/blockstack- core.