# Plasm Project Draft.

Takumi Yamashita info@stake.co.jp

Sep 17, 2019

**概要**

Plasm Project のミッションはすべての開発者にスケーラブルな分散アプリケーションの開発メソッドを提供することを通して新しい Web のあり方 "**Web3.0**" を実現することです。このドキュメントでは Plasm Project の目的、ブロックチェーン業界で必要とされる背景と発展に寄与する理由、そして、Plasm Project を通して実現する社会についての概要を記載します。さらに、Plasm Project を達成するための 3 つのプロダクト『Plasm』『PlasmChain』『Plasma as a Service』についての設計概要を記載します。

## 1 Introduction

私達はブロックチェーンを活用することで Web3.0 の実現を目指しています。既存の社会では権力者による情報や富の独占と、自分達に有利なルールを作ってきた歴史があります [1]。また、公平な仕組みを謳っていてもそこに透明性が無く信頼に依存して成り立っている仕組みと言えるでしょう。そのような既存社会に対してブロックチェーンは、分権的なガバナンスを用いることでプラットフォーマーが管理しない透明性と公平性のある Trustless なシステムを実現します。何故なら、ブロックチェーンは誰もが閲覧、検証、運用可能なシステム上でフォールトトレラント性と高い改ざん耐性を実現できるシステムだからです [2]。

ブロックチェーンのプロトコルを作るとして実際にユーザーにその恩恵を届けるためにはアプリケーションが必要になります。アプリケーションとは一種の OS とユーザーを繋ぐインターフェイスであるからです。ブロックチェーン上のアプリケーションを一般に Decentalized application(Dapps) と言います。現在、様々なブロックチェーンでスマートコントラクトやチェーンコードといった形で Dapps が開発、デプロイされてユーザーに提供されています。しかしながら、ブロックチェーンの分散冗長化された仕組み上 Dapps の処理性能は決して高いものではありません。最も規模の大きいスマートコントラクトを搭載したブロックチェーンである Ethereum[3] のトランザクションスループットは秒間 15 トランザクションです [4]。一方で世界中に多くのユーザーを保有する VISA や Alipay ではそれぞれ秒間 1700 トランザクション [5]、256,000 トランザクションを処理しています [6]。多くのユーザーが Dapps の恩恵を受けるためには現状の処理性能があまりに不十分であることが分かります。そこでブロックチェーンでは様々なスケーリングソリューションが考え出されました。

ブロックチェーンのスケーリングソリューションはいくつか存在します。例えば、Segwit などのブロックサイズを圧縮するソリューション [7]、ユーザー同士がオフチェーンでいくつかの取引をまとめて行い最終状態のみをブロックチェーンに記述するステートチャネル [8]、複数のノードでトランザクションの分散処理を行うシャーディング [9]、そして別のチェーンにトランザクション処理を行わせてルートハッシュのみをメインチェーンに保存する Plasma などがあります [10]。私達はその中でもまず、メインチェーンの外でトランザクションを処理するレイヤー 2 ソリューションに焦点を当てました。レイヤー 1 は Ethereum や Bitcoin と

言ったパブリックブロックチェーンのことを指します。それらはトランザクションが飽和している問題を抱えています [11]。このことから、10 年後のブロックチェーンの使用法は今までとは大きく異なり、レイヤー 1 がトラストレイヤー、レイヤー 2 がトランザクションレイヤーとして使用されることになると予想されます。

私達がその中でも Plasma に焦点を置いた理由は、Plasma がメインチェーンの処理性能に最も依存しないスケーリングソリューションだからです。Plasma では単一のオペレータと呼ばれる運営者がサイドチェーンの運営を行います。つまり、合意形成プロセスの不要な中央集権的管理方法で多くのトランザクションを処理することができます。それは、既存の中央集権的システムで使われているスケーリングソリューションをそのまま転用できることを意味するため分散台帳では不可能な高い処理性能を実現することができるのです。Plasma のアプローチは全ての分散台帳に飛躍的な処理性能の向上を行えるためこの先、必要不可欠な技術になっていくと言えるでしょう。

Plasma が高いスケーラビリティを誇ることは分かりました。しかしながら、Plasma の運用には未だいくつかの問題を抱えています。一つは Plamsa を使ったアプリケーション (Plapps) で出来ることに制限がある点です。Plasma で出来ることは “Predicate により記述可能” と言われています [12]。そしてこの “Predicate” で記述可能な範囲の定義はは現在研究の最中です。もう一つは Plasma は複数のコンポーネントにより構成されている複雑なシステムであるという点です [13]。単純にスマートコントラクトを記述しデプロイしただけでは Plapps を構築することは出来ません。具体的には Plapps は親チェーンコントラクト、子チェーン、オペレータ、ユーザーの 4 つのコンポーネントから成り立っておりそれぞれについて処理を記述しなければならなりません。私達はこれらの問題を Plasm Project を通して解決していきます。**Plasm** では Predicate を正しく記述できるように標準規格を設けてライブラリ化します。**PlaaS** は複数のコンポーネントを簡単にデプロイできるためのクラウドサービスを構築します。Plasm Project が提供するプロダクトを通して開発者が Plapps を開発する過程を手助けします。

私達はこれらのシステムを Polkadot[14] を軸に展開しようと考えています。Polkadot は複数の異なるブロックチェーンを束ねるプロトコルです。Polkadot によって今まで独立だった複数のブロックチェーンが透明で分権的なガバンスのもと相互運用性を得ることができます。さらに、Substrate[15] と呼ばれるブロックチェーンを作るためのフレームワークがあります。Polkadot が束ねるチェーンや Polkadot 自体も Substrate を使って作られます。一つの理想的なブロックチェーンですべての用途を賄うことは現実的ではありません。ブロックチェーンを使う理由はユーザーごとに多様であり、すべてのニーズを満たすガバナンスの構築ないし可用性を維持することが難しいからです。それ故に、今まで 900 を超えるパブリックブロックチェーンが作られてきました。Polkadot と Substrate は人々が用途に応じてブロックチェーンを作っていく時代にさらに拍車を掛けていくでしょう。私達はこの大きな流れに賭けることにしました。当然、Polkadot においても Plasma は必要不可欠な役割です。故に私達は Plasma を展開する先に Polkadot × Substrate を選択しました。

## 2 Plasm Project

Plasm Project はすべての開発者にスケーラブルな分散アプリケーションの開発メソッドを提供するためのプロジェクトです。私達はこれを以下の 3 つのサービスにより実現します。全体図を図 1 に示します。

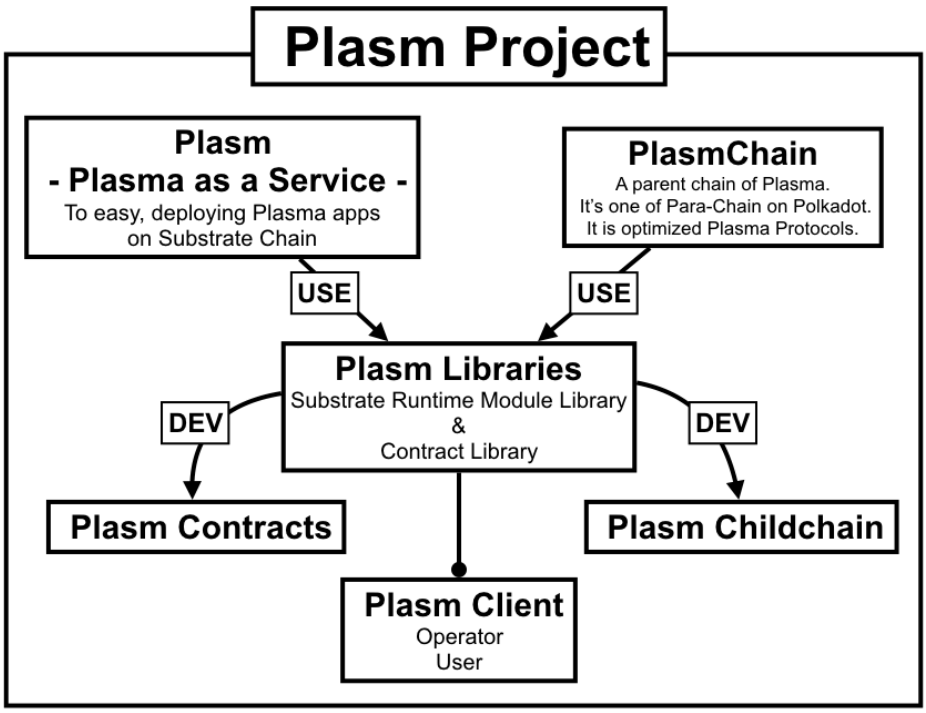


図 1　Plasm Project.

## 2.1　Plasm Libraries

Plasm は Substrate で Plasma チェーンを扱うためのライブラリです。Plasma のコア機能を実装する Rust Library である Plasm。Plasma のクライアント側の機能を実装する Typescript Library である Plasm-Client が用意されています。

## 2.2　PlasmChain

Polkadot ネットワーク上に Plasma Application をデプロイするためのスタンダードなメインチェーンとして機能するパラチェーンです。Plasma を扱うための独自のモジュールが用意されています。

## 2.3　PlaaS - Plasm as a Service -

Plasma アプリケーションを簡単にデプロイするための Platform as a Service です。メインチェーンを選んで GUI を通してカスタマイズした子チェーンをデプロイ、管理する機能が提供されています。

次の各セクションでそれぞれのサービスについての概要と技術詳細を解説していきます。

# 3 Plasm Libraries

Plasm Libraries は Substrate で Plasma チェーンを扱うためのライブラリです。Plasm Libraries は当初 SRML(Substrate Runtime Module Library) を開発することのみを想定していました。しかし、ユーザーがより自由に Plasma アプリケーションをデプロイするための仕組みにするために方針を変更し Plasm ライブラリを複数のコンポーネントに分けることにしました。それに伴い、Plasm は現在 Plasma の標準実装規格として Plasma Group Spec(PGSpec) を実装しています。しかし、PGSpec は本来 Ethereum を親チェーンとすることが想定されています。そこで、Plasm では PGSpec を Substrate 向けに適した設計に改変します。図 2 が Plasm PGSpec の全体構成となります。

Substrate 上のスマートコントラクトである ink![17] を用いて Plasma の仕様とデフォルト実装を開発することになりました。これを **Plasm Contract** と呼びます。

Plasm では子チェーンのデータベースも扱います。これは子チェーンのデータベースを実装とそのエンドポイントを実装します。これを **Plasm Childchain** と呼びます。

オペレータとユーザー役は適切に子チェーンのエンドポイントを叩くクライアントアプリケーションとして実装されます。SubstrateNode を叩くクライアントアプリケーションは **Plasm Client** と呼びます。

## 3.1 Plasm Contract

`https://github.com/stakedtechnologies/Plasm/tree/master/contracts`

Plasm Contract は `contract` SRML 上で展開される ink! スマートコントラクトによって記述されます。Plasm Contract ライブラリを用いることで開発者は標準規格通りの Plasma 親チェーン Contract を開発することが出来ます。Plasm Contract は標準規格を示した Trait と抽象化されたデータ構造、PGSpec 規格に基づいて実装された Plasma Cash[18] のデフォルト実装を提供します。

Plasm Contract の全体像は図 3 のようになっています。

各コンポーネントは以下のような役割を担います。

### 3.1.1 Commitment

Plasma 子チェーンのブロックヘッダー（MerkleRoot）を保持します。オペレータによる新しいブロックの作成、ブロックにある状態が含まれているかの検証を行います。

標準実装では MerkleIntervalTree を使ったブロックの包含証明を用いることを前提とした実装が提供されています。図 4 に MerkleIntervalTree の構造を示します。

### 3.1.2 Deposit

PlsmaChain に預けられた資産を管理します。預けられた資産は PlasmaChain で取引され、その正当な所有者のみが親チェーンに Exit できる仕組みを提供します。これは PlasmaExitGame の大部分のロジックを含みます。 標準実装では PlasmaChain のトークンに依存しない ExitGame の実装が提供されています。図 5 にある状態についてのチェックポイントに焦点をあわせた ExitGame の状態遷移図を示します。チェックポイントはある NFT について指定されたブロック高以前の状態遷移は全て正しいことを保証するものです。

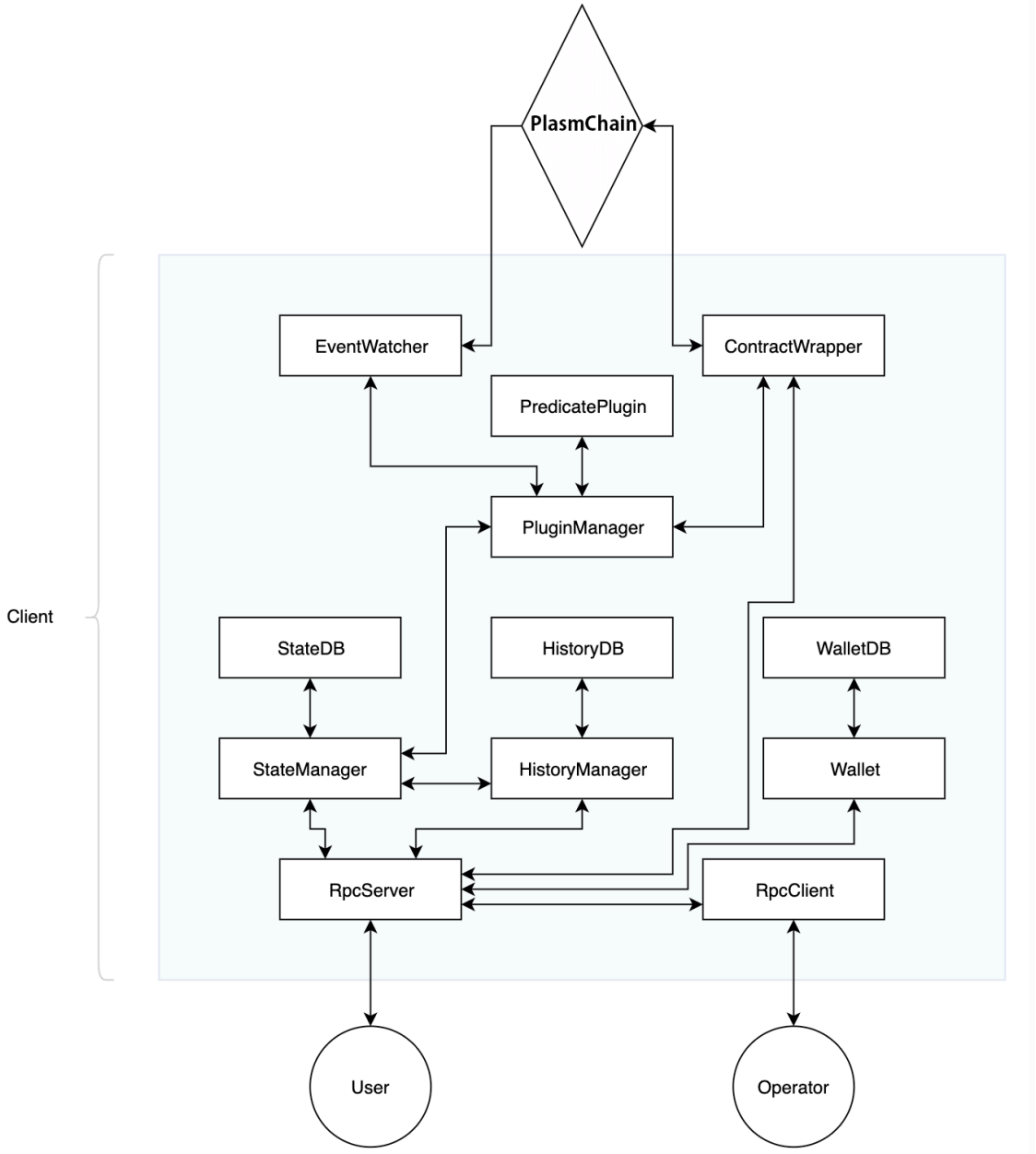


図 2　Plasma Group Spec optimized for Substrate[16]

### 3.1.3　Predicate

独自の ExitGame の規則を定義します。最も基本的なものはある Exit を無効にするロジックである DeprecateExit を記述します。通常ならば、このロジックはトランザクションが無効化されたことを証明するために使用されます。Predicate は一般化された Plasma Cash です。故に、Predicate を用いて Plasma Cash を表現することが出来ます。標準実装では状態オブジェクトとして所有者の情報のみを持つ Ownership Predicate と呼ばれる Predicate を提供しています。状態遷移には遷移前の状態オブジェクトの所有者の署名

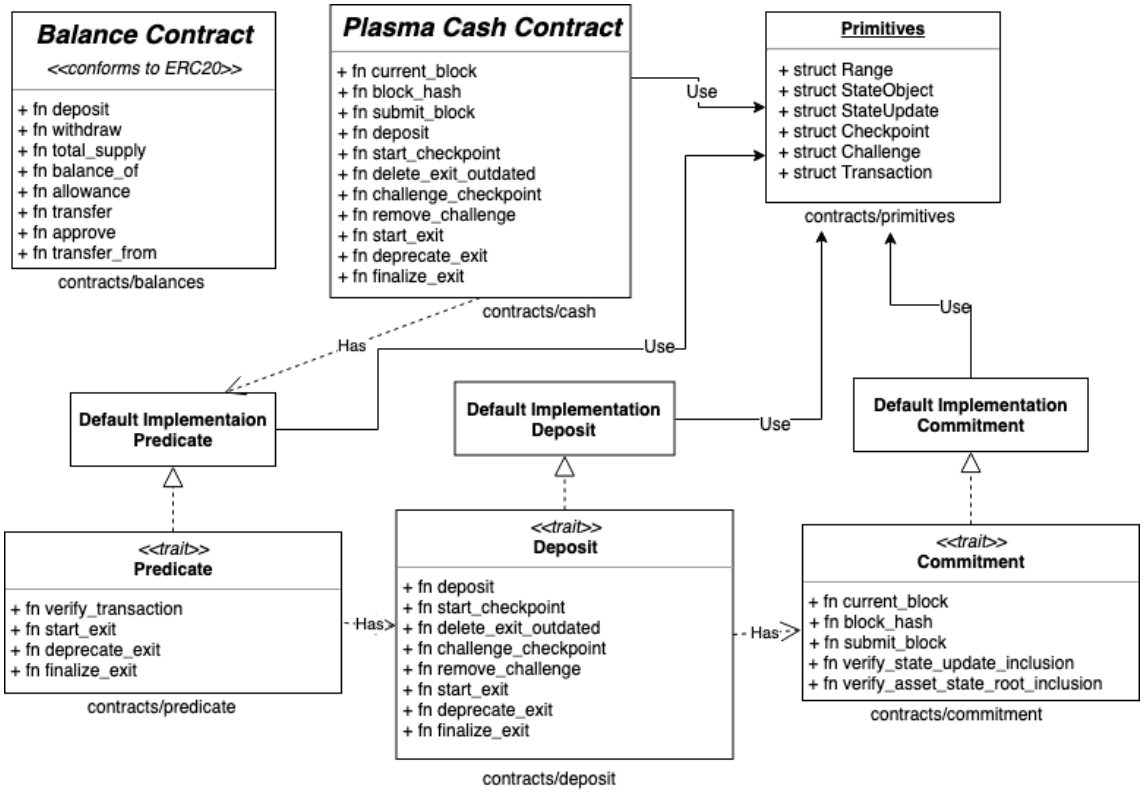


図 3　Plasm Contract Archtecture.

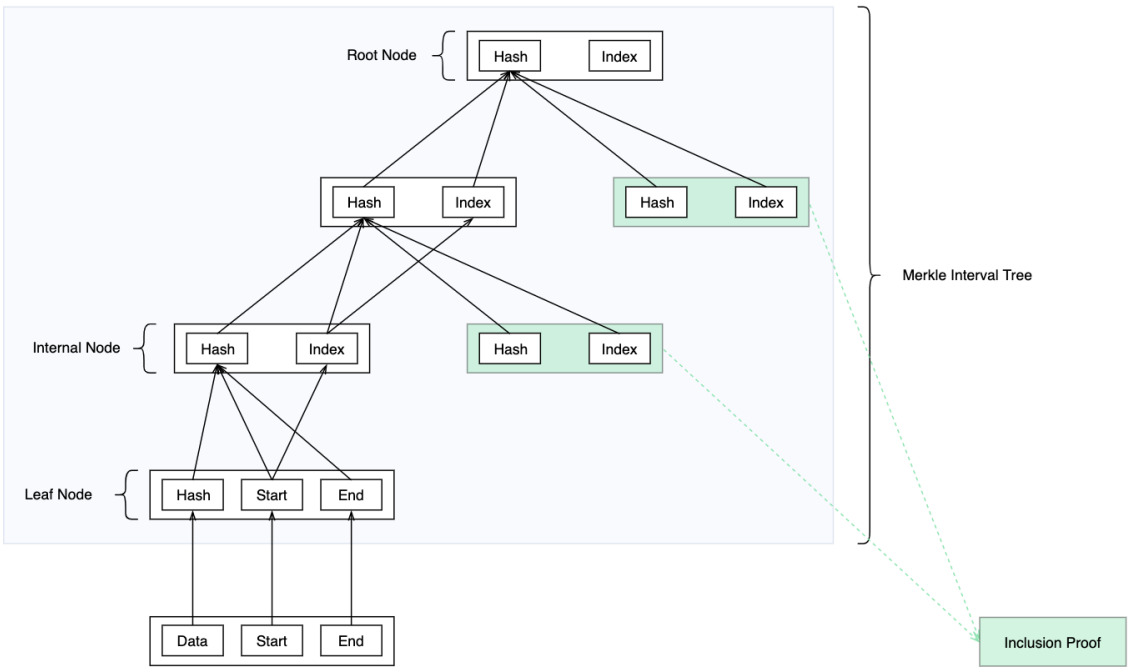


図 4　Merkle Interval Tree figure from Plasma Group[19]

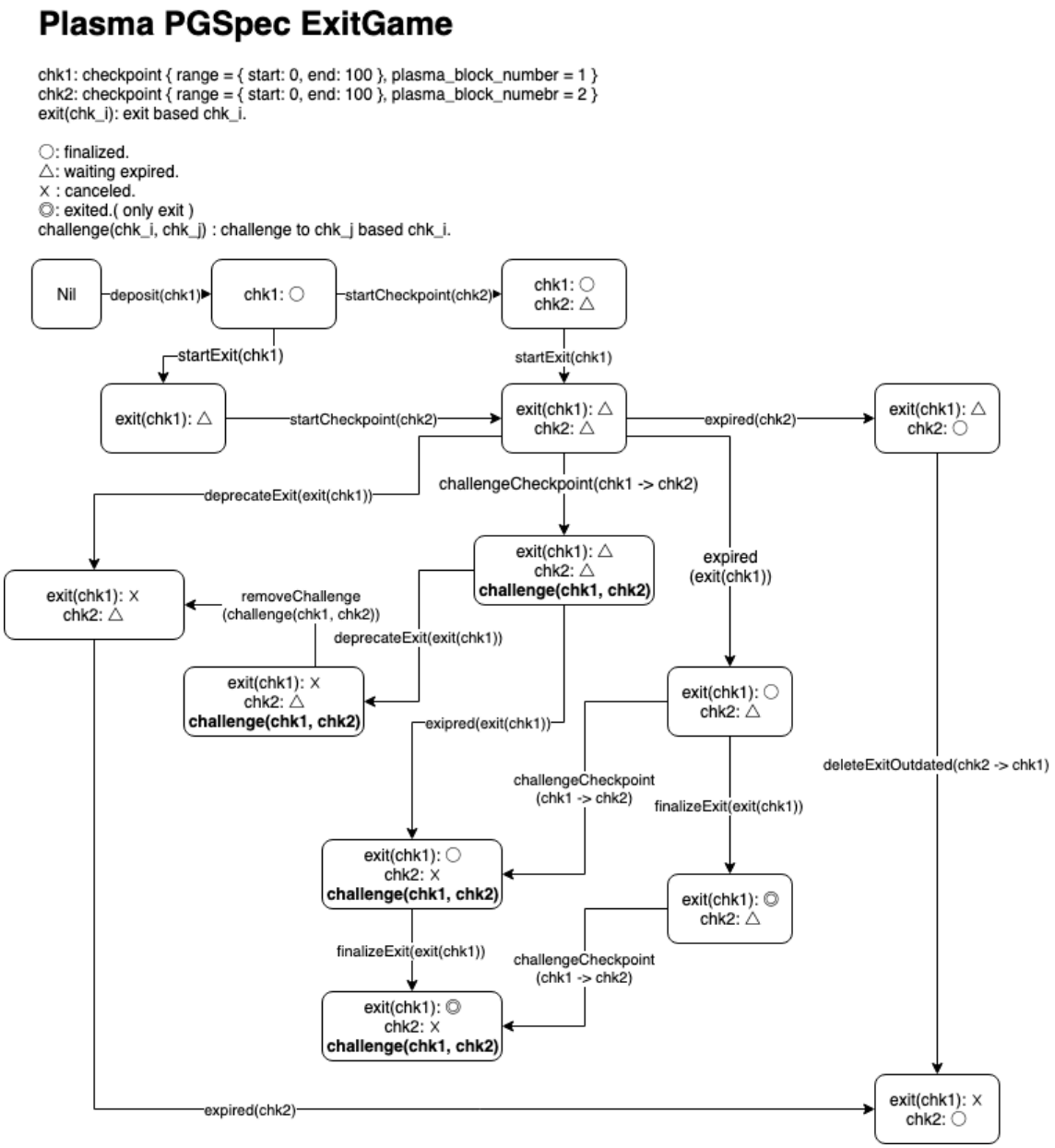


図 5　ExitGame state machine figure

が必要です。これは Plasma Cash に必要な機能を完全に提供します。

### 3.1.4　Primitive

Plasma Contract で共通して使う構造体と特性について Primitive ライブラリとして提供します。

#### 3.1.5 Balance Contract

Plasma Contract で扱う資産は ERC20 に準拠したものを用意する必要があります。Plasm は Substrate の基軸通貨を ERC20 準拠なコントラクトとして扱うために Balance Contract を標準で用意する予定です。これは Ethereum における WrappedETH のようなものだと思ってください。

#### 3.1.6 Plasma Cash Contract

標準実装として PlasmCash に相応する機能を持った Plasma Contract が実装されています。Predicate の標準実装である Ownership Predicate と Deposit, Commitment の標準実装を組み合わせることでこれを実現しています。

### 3.2 Plasm Childchain

Plasm Childchain は全体図の Client において保存するデータベースとそれに対するアクセスエンドポイントを提供します。これは Cryptoecnomics Lab[20] が提供する Plasma Rust Framework [21] を利用して Substrate 用に実装される予定です。クライアントは自分に必要なモジュールのみを保持します。

### 3.3 Plasm Client

`https://github.com/stakedtechnologies/plasm-client`

Plasm Client ではオペレータとユーザーのクライアントアプリケーションとそれらが扱う共通ライブラリを提供します。

## 4 PlasmChain

PlasmChain は Polkadot Network 上の Plasma Defauct Standard Chain として機能する Parachain になることを想定しています。一般に Plasma アプリケーションをデプロイする時のデフォルトチェーンとしてこの Plasma Chain を選びます。PlasmChain はランタイムに Plasma Specific な機能を搭載することで Plasma アプリケーションに新たな可能性と快適なユーザービリティを提供します。また、PlasmChain では革新的なトークン発行アルゴリズムを用いて User/Developer ファーストなトークンエコノミーを設計しています。それでは、Plasm Chain が提供するユニークなランタイムアルゴリズムについて見ていきましょう。

### 4.1 PlasmChain Token Design

ここでは PlasmChain のトークン設計について解説します。PlasmChain では Lockdrop[22] と呼ばれるトークン発行の仕組みを利用します。Lockdrop は機会費用を担保にトークンを発行する仕組みであり多くのユーザーが低リスクでトークンを受け取ることができる革新的なトークン発行メカニズムです。これは Edgeware がトークン発行の仕組みとして考え出した Lockdrop を拡張したものになります。図 6 に Lockdrop の概要図を示します。

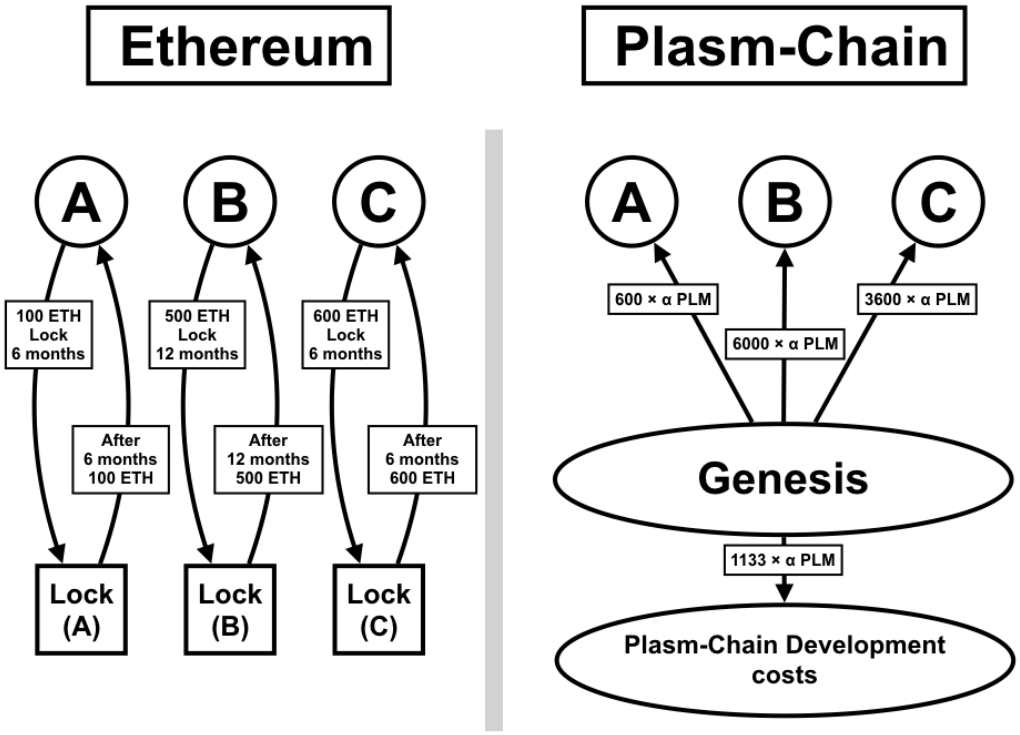


図 6　About Lockdrop

#### 4.1.1　Lockdrop Overview

私達は第一回目の Lockdrop を Ethereum の機会費用を担保に行います。そのため説明では Ethereum による Lockdrop を前提に行います。しかしながら、Lockdrop は TimeLock 機能のあるあらゆるチェーンで代替可能なアルゴリズムであることに注意してください。

1. Ethereum のトークンホルダーは Ethereum 上の Lock Contract に所持している ETH を期間を指定して Lock します。
2. PlasmChain の Genesis Block は、Lockdrop を行ったトークンホルダーに対してそれぞれ `Lock` **された** `ETH` **の総量** × `Lock` **期間** × $\alpha$ だけ PlasmToken を付与します。
3. PlasmChain の Genesis Block は、私達に対して Lockdrop で**ユーザーに配布したトークン総量** × `1/9` の PlasmToken を付与します。
4. Lockdrop を行ったトークンホルダーは Lock 期間を過ぎると Lock した ETH が使用できるようになります。

Ethereum のトークンホルダーは Lock した Ethereum の量と期間分だけ機会費用を払っています。PlasmToken はこの機会費用を担保にしてトークンを発行することでトークンの価値を担保します。また、

PlasmChain のトークンの発行量は決まっていません。これは Genesis ブロック以降にも Lockdrop によるトークン配布をより公平に行うためです。具体的には以下のような方法で複数のトークン配布を行います。

#### 4.1.2 Multi-Lockdrop

PlasmChain はトークンの総発行量がジェネシスブロックで決まりません。正確には、3 度の Lockdrop の後にトークンの総発行量が決定します。

Lockdrop を複数回に分ける理由は 2 つあります。1 つは先行者利益が肥大化することを避けるためです。1 度の Lockdrop で全てのトークンを分配すると、仮に最初の Lockdrop の参加者が少なかった場合にトークンの大半を保有するホルダーが現れるかもしれません。そして、そのようなことが発生した後にロールバックを行うことは Chain の信頼性を損ないます。PlasmChain は予め提示したルールに基づいてこのような問題を避ける施策を打つ必要があります。この問題を解決するために私達はジェネシスブロックでトークンの総発行量を決定しないアルゴリズムを考案しました。2 つ目は PlasmChain が安全にスケール/分権化するために序盤はある程度管理可能で実験的な振る舞いが出来るようにするためです。Blockchain の強固なセキュリティシステムは大量のノード参加者と広く分散化されたトークンホルダーが存在することで成り立っています。起動直後からこのセキュリティを担保することは極めて難しいです。序盤は信頼できる機関がある程度管理可能な状態であるほうが好ましいでしょう。そこで 3 度の Lockdrop を設けることにより、全てのトークン発行前に不用意なトークン価値の高騰を抑えることで管理コストを下げます。私達は最終的に PlasmChain を完全なパブリックチェーンとすることを前提として安定して動作するまでの準備期間を設けるために複数回の Lockdrop を行います。

各 Lockdrop についてトークンホルダーが獲得できる PlasmToken の量を決定するための式に使われる定数 $\boldsymbol{\alpha}$ を決定します。i 番目の Lockdrop で使われる $\boldsymbol{\alpha}$_i について$\boldsymbol{\alpha}$_i > $\boldsymbol{\alpha}$_(i+1) です。その i 番目の Lockdrop でユーザーーに配られる PlasmToken の総量は 、i 番目の Lockdrop に参加するトークンホルダーの数を n_i、i 番目の Lockrop における j 人目の Lock した ETH の量を LockedEth_{i,j} 、ETH を Lock する期間を LockedTime_{i,j} とすると次の式で表されます。

$$\sum_{j=1}^{n_i}(LockedEth_{i,j} \times LockedTime_{i,j}) \times \alpha_i$$

PlasmChain では合計で 3 回 Lockdrop を行います。よって最終的に発行されるトークンの総量はブロック生成報酬を除くと以下になります。

$$\sum_{i=1}^{3}\sum_{j=1}^{n_i}(LockedEth_{i,j} \times LockedTime_{i,j}) \times \alpha_i \times \frac{10}{9}$$

仮にトークンの発行総量をジェネシスで決定したとします。その場合、各 Lockdrop について総量の何割を配布するかを決定します。各 Lockdrop について i 番目の Lockdrop の参加者の数は i-1 番目の参加者の数よりも多いことを仮定すると先行者利益が想定以上に莫大なものになる危険性があります。そのため、一定の先行者利益を保証するがそのような問題の発生を防ぐために上記の方法を考案しました。

#### 4.1.3 Advantage

- ICO ではありません。私達はトークンホルダーの資産を預かりません。

- ユーザーは Ethereum の機会費用を担保にトークンを得ます。一定期間の間、暗号通貨を使えなくなりますが指定した期間の後にその通貨は返却されます。しかしながら、現在多くの通貨は投機的価値しか持たず長期的ホールドを前提にしているトークンホルダーにとってこれはデメリットになりません。
- Lock した Ethereum は一定期間を終えた後、トークンホルダーに返却されます。仮に PlasmChain で悪意のある攻撃を受けたとしても Lockdrop 参加者は ETH を持ち続けることができます。
- Lockdrop を行う方法は、私たちのロックドロップ・コントラクトへ送金することで Lockdrop を行うことができます。どんなアカウントでもハードウェアまたはソフトウェアのウォレット（例：Trezor や Metamask など多数）からこれを実行することができます。さらに言えば、任意の ETH ホルダーが参加することもできます。

## 4.2 Consensus Algorithm

PlasmChain は Polkadot の RelayChain で採用されているコンセンサスアルゴリズムを使います。このアルゴリズムは大きく 3 ステップに分かれています。i) Nominator は Validator を選出する NPoS[23]。 ii) Validator はトランザクションを検証しブロックを生成する Babe[24]。iii) 配布されたブロックをファイナライズする GRANDPA[25]。また、初期の Validator ノードを限定することで十分な Validator がいない状態において悪意のある Validator が闊歩するリスクを減らすことができます。ブロックの生成報酬は Validator と支援者である Nominator に分配されます。また、PlasmChain の価値向上の貢献者にもこの報酬は支払われます。この貢献者に対する報酬の支払いを以下で説明します。

## 4.3 Plasm Treasury

PlasmChain のブロック生成報酬の 50% は PlasmChain の価値を向上させたアプリケーション製作者に当てられます。PlasmChain では Plasma Application の Operator に対しても Stake することができます。これを Plapps Nominator と呼びます。図 7 にアプリケーション製作者に対する報酬割当の概要図を示します。
次の変数を定義します。

- `BlockReward`：ブロック生成報酬
- `n`：Plasma Operator の数
- `m_i`：i 番目の Plasm Operator に対する Plapps Nominator の数
- `PlappsNominator_{i,j}`：i 番目の Plasm Operator に対する j 番目の Plapps Nominator
- `stake_{i,j}`：i 番目の Plasm Operator に対する j 番目の Plapps Nominator の Stake したトークンの量

この時、この Stake について PlappsNominator_{i,j} はブロック生成時に以下の報酬が得られます。ここで `PlappsNominator_{i,j} = PlappsNominator_{k,l} (i ≠ k)` があり得ることに注意してください。

$$BlockReward \times \frac{stake_{i,j}}{\sum_i^n \sum_j^{m_i} stake_{i,j}} \times 0.1$$

次に Stake を受けた Plasm Operator はブロック生成時に以下の報酬が得られます。

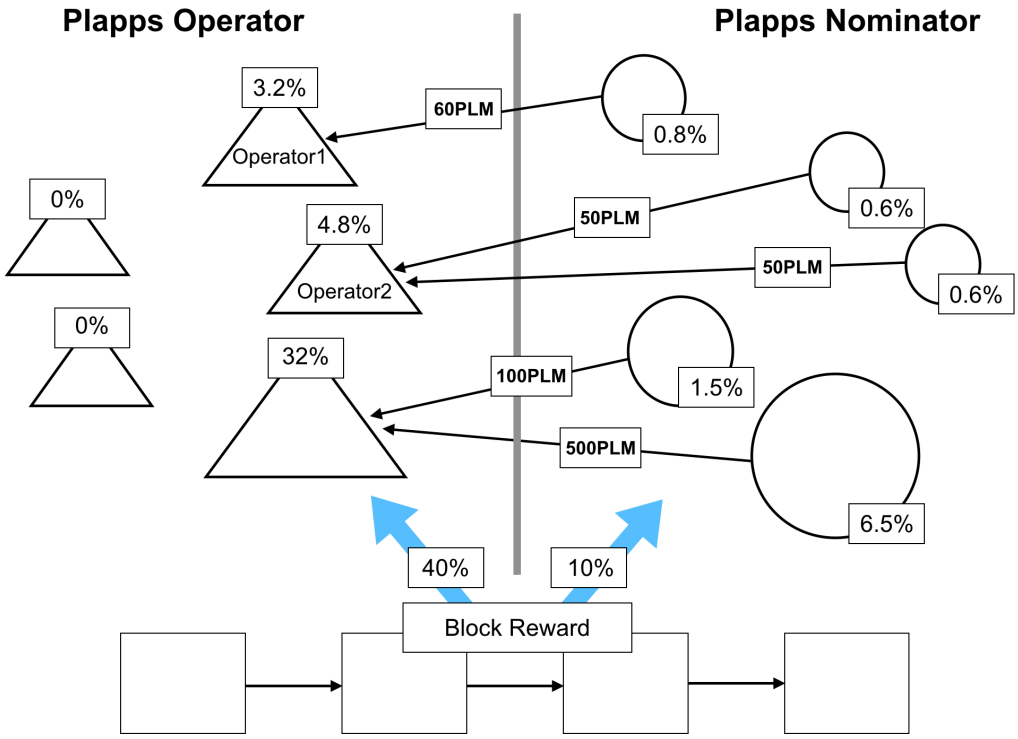


図 7　plasm application reward

$$BlockReward \times \frac{\sum_j^{m_i} stake_{i,j}}{\sum_i^n \sum_j^{m_i} stake_{i,j}} \times 0.4$$

Plapps Nominator は選んだ Plapps Operator に関わらず Stake 量に比例した報酬を得ることができます。一方、Plasm Operator は Stake された量に比例し、エコシステム内で Stake された総量に反比例した報酬を得ることができます。これにより、Plapps Nominator は単純に PlasmToken の価値を向上させると思われる Application に対して Stake するインセンティブが発生します。また、Operator は Stake を受けることで半永続的に PlasmChain のブロック生成報酬を受取ることができます。これは Dapps 開発者のマネタイズが難しい問題に対する革新的な解決策となることを期待しています。

しかしながら、Operator はいくつかの不正ができます。ところが Plasma のシステムは Operator の不正を申告する機構が既に存在しています。Operator が不正を行った場合、この Plasma の faild proof を転用することで Operator に対するブロック生成報酬を Slash し、またその Plapps Nominator の Stake しているトークンも一定量 Slash されます。つまり、Plapps Nominator は Operator が不正をしたときに Stake したトークンを失うリスクを負います。さらに、大口の StakeHolder が自作自演で Operator を運営して Stake する可能性があります。このような極めて悪質なケースは Governance によって Stake した全てのトークンが Slash されることになるでしょう。

このような Slash による罰則を加えると不安定な Operator に対して Stake するインセンティブが減少し

てしまいます。全てのユーザーーが安定して稼働しているアプリケーションに対して Stake することになるでしょう。Operator のローンチ後、$k(\leq 3)$ ヶ月未満の Opeartor に対する Stake には別途でボーナス報酬を得る権利を加えます。これは $l(\geq 12)$ ヶ月後以降に以下の権利を $k$ ヶ月間行使できます。ここで `x^{old}` は Stake した時点での `x` の状態を表す。

$$BlockReward \times \frac{\sum_j^{m_i} stake_{i,j}}{\sum_i^n \sum_j^{m_i} stake_{i,j}} \times \frac{stake_{i,j}^{old}}{\sum_i^{n^{old}} \sum_j^{m_i^{old}} stake_{i,j}^{old}} \times 0.4$$

のブロック生成報酬を得る。つまり、$l$ **ヶ月**後の Plasm Operator が得ている報酬に比例した額の報酬を得る権利があります。この権利を行使された際は Plasm Operator が貰える報酬はその分だけ減少します。故に Plasm Operator は Nominate を受けることを拒絶する権利があります。$k$, $l$ について Plapps Operator が最初に指定することができます。

## 4.4 Operator Trading

Operator Trading は Plasma アプリケーションのオペレータを売買する仕組みです。上記の Plasm Treasury の仕組みにより、Operator は恒常的に利益を得ることができます。よりよく Operator の運営を行えるならばその Operator を別の相手に譲ることも一つの案でしょう。当然、Operator は常時売買されるわけではありません。Operator は自身に対してついた値に対して妥当だと思われる値をつけた相手に対してその権利を譲るでしょう。Operator の権利を譲り受けた対象は Operator としての報酬を受け取ることができます。この時、実際に運用を交替する必要はありません。しかし、Operator を売り渡した側は既にその Oeprator を誠実に運用するインセンティブを失うため必然的に譲渡先の新しい Operator の保有者が Operator の運用を行うことになるでしょう。

### 4.4.1 Conclusion

私達は PlasmChain に以上の特殊な仕組みを導入することで Plasma Application の構築に特化したブロックチェーンを構築します。流動性のあるトークン発行を低コストで行う仕組み、安全に分権化されたブロックチェーンシステムを拡張する仕組み、そして Plasma application の開発者に報酬を与える仕組みの 3 つの仕組みが PlasmChain を繁栄させることでしょう。

# 5 PlaaS

Plasm as a Service(以下 PlaaS) は『Plasm』を内部で用いることで Plasma アプリケーションを誰でも簡単にデプロイできるようにするためのサービスです。図 8 に PlaaS のホーム画面を示します。

私達が開発している Substrate のライブラリ『Plasm』は Substrate Chain が Plasma を扱えるようにするためのライブラリです。これを用いることで、Plasma アプリケーションをデプロイ可能なパラチェーンを展開することが可能になります。スマートコントラクトでは性能の限界があり実用化に至らないようなケースは非常に多いため Plasma アプリケーションの潜在的需要は非常に高いと言えるでしょう。

しかしながら、Plasma アプリケーションを開発・運用することは容易ではありません。

全体構成図を見て分かる通り Plasma は複数のコンポーネントから成り立っています。これらの構成要素すべてを 1 から実装するのは大変な作業であり通常のコントラクトを実装するより遥かに高い開発コストが発

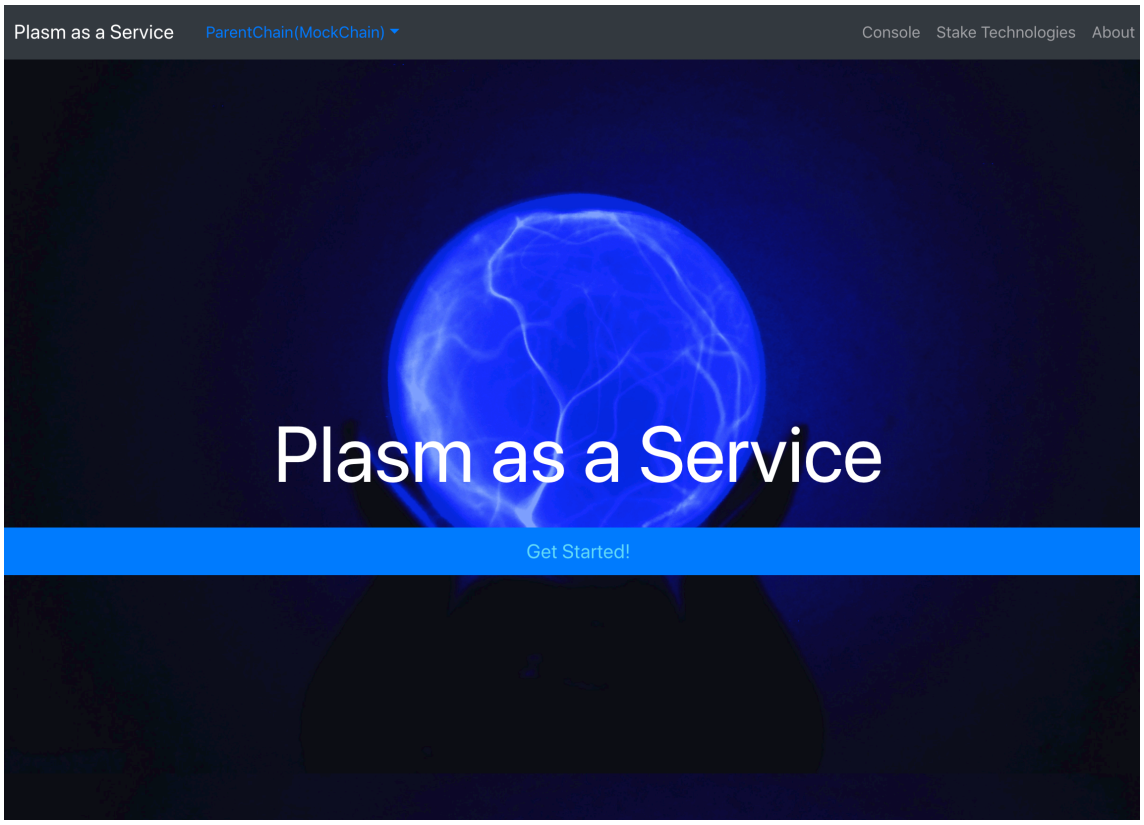


図 8　Plasma as a Service.

生するでしょう。Plasma 特有の状態管理を踏まえての開発は高度な技術力と Plasma への理解、そして工数を要します。また、オペレータと呼ばれる Plasma アプリケーションの管理者が必要となり長期的な管理コストも発生します。それらの問題を解決するために我々は GUI を用いて Plasma アプリケーションをより簡単にデプロイできるサービスを提供します。

## 5.1　機能

PlaaS は Plasma アプリケーションを簡単にデプロイ・管理することができます。具体的には以下のステップで Plasma アプリケーションを作成していきます。

1. **GetStart**: Plasma アプリケーションを作成するスタート画面へとジャンプします。
2. **Generate Plasma App**: Plasma アプリケーションを作成する際に使うテンプレートを選択します。デフォルトでは Payment のみが可能な Plasma Cash が用意されています。また、Customizable な Plasma アプリケーションを作るためのテンプレートも用意されてく予定です。
3. **Generate Settings**: Plasma アプリケーションの設計を記述していきます。選んだテンプレートに応じて要求される項目を GUI 上で設定していきます。
4. **Deploy a Contract and Childchain**: 3 で設計した Plasma アプリケーションを実際にデプロイします。この時に使用される fee などの支払いは PlaaS が自動で行ってくれます。このフェーズでは、

1) 親チェーンに対する Contract のデプロイ。2) 子チェーンの DB を持つサーバのデプロイ。3) オペレータアカウントのデプロイ。4) ジェネシスアカウントの作成。を行います。

5. **Console**: 4 でデプロイした Plasma アプリケーションの状態や設計を確認・操作することができます。具体的には 1) 親チェーンの Contract の状態の確認。2) 子チェーンの DB の状態の確認。3) オペレータアカウントの操作。4) 4 で生成したジェネシスアカウントを含む登録したアカウントの操作。ができます。

以上の機能により、Plasma のアプリケーション作成者は簡単に Plasma アプリケーションを作成し管理することが可能になります。

### 5.2 Demo

デモアプリケーションを試して Plasma アプリケーションを数分で作成していく過程を体験してください！（※実際にはこのデモでアプリケーションのデプロイはされていません。）
https://mock.d3dg769h9ndpcf.amplifyapp.com

## 6 Conclusion

Plasm Project を通して私達はすべての開発者にスケーラブルな分散アプリケーションの開発メソッドを Polkadot 上で展開します。複数の異なるブロックチェーンを束ねる Polkadot に優秀なスケーリングソリューションである Plasma を展開できる恩恵は計り知れません。私達は典型的な Plasma アプリケーションを PlaaS を使ってものの数分でデプロイできるようになります。複雑な Plasma アプリケーションもカスタマイザブルな Plasm ライブラリを用いることで標準規格に準拠した正しいアプリケーションを開発することができます。また、私達はユーザースペシフィックなユースケースについてサポートするでしょう。PlasmChain ではデプロイされた Plasma アプリケーションは任意で市場に出すことができます。そのアプリケーションが保有するエコノミーを売買することでさながら M&A のようなやり取りが可能になります。故に、Plasma アプリケーションはもはやただのアプリケーション以上の価値を持っていると言えるでしょう。私達は Plasma アプリケーションが栄えた新しい Decentalized な世界観を楽しみにしています。

## 7 謝辞

これらの技術の組み合わせは Plasm ライブラリを支援して頂いた Web3 Foudnation、Substrate/Polkadot の開発元である Parity technologies, Plasma R&D の最先端を担う Plasma Group と Cryptoecnomics Lab の協力なくしては実現できません。彼らに深い謝辞を申し上げます。

## 参考文献

[1] Problem of capitalism : Robert B. Reich, "Saving Capitalism: For the Many, Not the Few" https://www.jstage.jst.go.jp/article/lecgsa/15/0/15_139/_pdf/-char/en

[2] Mastering BitCoin : https://github.com/bitcoinbook/bitcoinbook

[3] Ethereum : https://www.ethereum.org/

[4] Ethereum Transaction Throughput : https://www.coindesk.com/information/will-ethereum-scale

[5] Visa Transaction Throughput: https://hackernoon.com/the-blockchain-scalability-problem-the-race-for

[6] ALIPAY Transaction Throughput : https://www.barrons.com/articles/alibaba-records-25-3-billion-in-singles-day-sales-1510538618

[7] Segwit : https://github.com/bitcoin/bips/blob/master/bip-0141.mediawiki

[8] StateChannel : Jeff Coleman, Liam Horne, and Li Xuanji "Counterfactual: Generalized State Channels", June 12, 2018, https://l4.ventures/papers/statechannels.pdf

[9] Sharding : Loi Luu, Viswesh Narayanan, Chaodong Zheng, Kunal Baweja, Seth Gilbert, Prateek Saxena, "A Secure Sharding Protocol For Open Blockchains", https://www.bubifans.com/ueditor/php/upload/file/20181015/1539597837236127.pdf

[10] Plasma : Joseph Poon, Vitalik Buterin, "Plasma: Scalable Autonomous Smart Contracts" August 11, 2017https://plasma.io/plasma.pdf

[11] Ethereum limits: StopAndDecrypt, "The Ethereum-blockchain size has exceeded 1TB, and yes, it' s an issue" https://hackernoon.com/the-ethereum-blockchain-size-has-exceeded-1tb-and-yes-its-an-issue-2b650b5f4f62

[12] About Predicate : https://docs.plasma.group/projects/spec/en/latest/src/02-contracts/predicate-contract.html

[13] Plasma Components : https://docs.plasma.group/projects/spec/en/latest/src/05-client-architecture/introduction.html

[14] Polkadot : DR. GAVIN WOOD, "POLKADOT: VISION FOR A HETEROGENEOUS MULTI-CHAIN FRAMEWORK" https://polkadot.network/PolkaDotPaper.pdf

[15] Substrate : https://www.parity.io/substrate/

[16] PGSpec : https://docs.plasma.group/projects/spec/en/latest/src/05-client-architecture/introduction.html

[17] ink! : https://github.com/paritytech/ink/wiki

[18] Plasma Cash https://ethresear.ch/t/plasma-cash-plasma-with-much-less-per-user-data-checking/1298

[19] MerkleIntervalTree : https://docs.plasma.group/projects/spec/en/latest/src/01-core/merkle-interval-tree.html

[20] Cryptoeconomics Lab https://www.cryptoeconomicslab.com/

[21] Plasma Rust Framework https://github.com/cryptoeconomicslab/plasma-rust-framework

[22] Lockdrop https://blog.edgewa.re/full-details-on-the-edgeware-lockdrop

[23] NPoS https://research.web3.foundation/en/latest/polkadot/NPoS/

[24] BABE https://research.web3.foundation/en/latest/polkadot/BABE/Babe/

[25] GRANDPA https://research.web3.foundation/en/latest/polkadot/GRANDPA/