

# PULSAR 白皮书

## 目录

Pulsar 白皮书.....	1
.1 区块链概述.....	3
.1.1 历史与发展.....	3
.1.2 局限性.....	4
.2 Pulsar.....	6
.2.1 愿景.....	6
.2.2 整体架构.....	6
.3 关键技术.....	10
.3.1 DS-POW 混合共识.....	10
.3.2 DAG.....	18
.4 应用场景.....	29
.4.1 数据交易.....	29
.4.2 商品溯源.....	30
.4.3 车联网.....	31
.4.4 供应链金融.....	32
.4.5 身份认证.....	33
.4.6 股权登记转让.....	34
.5 路线图.....	36
.6 免责声明.....	37
.7 团队介绍.....	38
.8 参考文献.....	41

## .1 区块链概述

### .1.1 历史与发展

2008 年，中本聪 (Satoshi Nakamoto) 发表了一篇名为《比特币：一种点对点式的电子现金系统》(Bitcoin: A Peer-to-Peer Electronic Cash System) 的论文。该论文首次描述了一个可实现的去中心化匿名数字货币系统，其设计思想成为区块链技术的基石。2009 年 1 月，比特币系统上线，正式拉开了数字货币时代的序幕。

去中心化的数字货币概念，早在几十年以前就被提出来了。1980 和 1990 年代的匿名电子现金协议，大部分是以乔姆盲签技术 (Chaumian Blinding) 为基础的。这些电子现金协议提供具有高度隐私性的货币，但是这些协议都没有流行，因为它们都依赖于一个中心化的中介机构。1998 年，戴伟 (Wei Dai) 的 B-Money 首次引入了通过解决计算难题和去中心化共识创造货币的思想，但是该建议并未给出实现的具体方法。2005 年，芬尼 (Hal Finney) 引入了“可重复使用的工作量证明机制”(Reusable Proofs of Work) 的概念，它同时使用 B-Money 的思想和 Adam Back 提出的哈希现金 (Hashcash) 难题来创造密码学货币。但是，这种概念依赖于可信任的计算作为后端。

比特币的创新是引入这样一个理念：将一个非常简单的基于节点的去中心化共识协议与工作量证明机制结合在一起。节点通过工作量证明机制获得参与到系统的权利，约每十分钟将交易打包到“区块”中，从而创建出不断增长的区块链。拥有大量计算力的节点有更大的影响力，但要获得超过整个网络一半以上的计算力会非常困难。尽管比特币的共识模型非常简单，但是长期的实践证明它已经足够好用。

作为去中心化的电子现金系统，比特币使用非图灵完备的脚本语言控制交易。通过限制脚本的功能，能够避免恶意脚本的攻击和有安全缺陷的脚本造成的影响，在一定程度上提升了电子现金交易的安全性。然而，这也制约了比特币在通用领域的发展和应用。

2013 年，Vitalik Buterin 详细阐述了“下一代智能合约和去中心化应用平台”(A Next-Generation Smart Contract and Decentralized Application Platform)，

即以太坊（Ethereum）。以太坊继承了比特币共识模型的核心部分，使用图灵完备的交易脚本，这构成了智能合约系统的基础。智能合约以及大量基于智能合约的去中心化应用（DApp）极大的繁荣和发展了数字货币领域和区块链技术及其应用。

## .1.2 局限性

### .1.2.1 交易处理能力

区块链系统的交易处理能力一直是制约其应用的瓶颈。交易处理能力低下是由系统内在的机制所决定的。在分布式计算系统中，两个因素根本上决定了系统的处理能力：

- 节点通信的网络时延。
- 节点状态一致性的开销。本质上由两个因素决定：
  - ✓ 处理交易顺序的节点数量，节点越多，开销越大。
  - ✓ 参与确认交易的节点数量，节点越多，开销越大。

中心化的分布式计算系统运行于可信网络和可信节点，在某个时间窗口内，处理交易序列和确认交易的节点都只有一个。因此节点状态的一致性开销很小，仅依赖于网络通信延迟。去中心化的分布式计算系统的处理交易序列和确认交易的节点数量往往都是多个，处理节点状态一致性需要大量额外的计算与状态同步，开销很大。综上所述，在拥有同等计算资源的系统中，去中心化模型的交易处理能力一定低于中心化模型。

在提高区块链系统处理能力的众多方法中，一类是通过减少处理交易顺序的节点和验证交易的节点来达到提高性能的目的，代表系统为 BitShares、Steemit 和 EOS 等。这些系统部分牺牲了去中心的特性，可看作是多中心化系统，通过社区来约束这些处理与确认交易的少数“超级”节点。从实践上看，这类系统确实提高了系统的处理能力，但离中心化系统仍然有几个数量级的差距。同时“超级”节点及社区治理也产生了很多负面的影响。

另一类系统通过优化交易处理机制来提升性能，例如以 DAG（Directed Acyclic Graph，有向无环图）为基础的交易模型，可以将交易处理并行化，从而

大幅提高系统吞吐量。但是，目前这类系统大多还处于早期试验阶段。

### .1.2.2 用户参与度

对于互联网应用系统，用户参与度不仅直接决定流量经济，而且也影响应用系统功能和质量的提升。用户参与度是一个应用系统持续发展最重要的参考指标。对于区块链系统，用户参与度还直接影响了系统的安全性。因此，如何提高区块链系统的用户参与度是系统设计之初要重点考虑的问题。

区块链系统的用户参与度包括全节点和轻节点的参与度，分别表示用户成为全节点和轻节点的意愿。全节点用户参与度的抑制因素主要是计算力和数据存储成本，促进因素则是 Token 激励制度（挖矿收益）。采用 POS 共识机制相对于 POW，摈弃了算力的竞争，成本更低，因而能够提升全节点的参与度。

轻节点的持有区块的部分数据，对区块数据增长不敏感，运行成本低，可以直接运行于移动终端。但轻节点客户端在功能上受限，只能发起交易，查询账户余额，交易数据等。在单纯 POW 系统上很难进一步提高轻节点用户参与度。对于使用 POS 共识机制的系统，可以通过股权委托的方式使轻节点客户端参与挖矿，从而显著提升用户参与度。

综上所述，POS 共识机制能够获取最大的用户参与度。但单纯的 POS 容易产生中心化倾向，在技术上也极易发生区块临时分叉。因此，应考虑设计一种混合的共识机制来消除 POS 内在的一些缺陷。

## .2 Pulsar

### .2.1 愿景

Pulsar 致力于成为一个具有高性能的完全去中心化的区块链基础平台系统。该系统具有安全可信、高性能与低能源消耗、功能完整且扩展性强的特性。通过图灵完备的智能合约和功能强大的 RPC 接口，使得应用开发者在 Pulsar 上能方便快速的开发满足用户需要的 DApp。

Pulsar 致力于成为一个有现实社会价值的区块链基础平台系统。在设计之初我们充分考虑了落地应用的需求，希望通过插件扩展模式让 Pulsar 低成本的适配通用商业和各类行业细分领域的应用。

Pulsar 技术团队致力于建设一个全球化的开发社区。通过委员会制度和适当的激励制度，吸引全世界优秀的开发者参与到 Pulsar 的基础平台功能的完善和应用开发。

Pulsar 团队致力于建设一个全球化的用户社区。通过社区互动了解用户的真实需求；通过与开发社区的互动让用户的需求能够得到及时的反馈；通过投票机制让用户真正决定系统后期的发展与规划。

### .2.2 整体架构

Pulsar 在以太坊架构的基础上使用创新的共识机制，总体分为四层：基础层、核心层、服务层、应用层。基础层提供区块链系统最基础的组件，主要包括 P2P 网络、数据库、密码学算法库等。核心层实现区块链系统的核心逻辑，包括区块链数据与状态的管理和全新的共识机制（DS-POW 和 DAG 模块）。服务器层提供对外提供服务，包括虚拟机的实现、RPC 接口以及智能合约。应用层则面向最终用户提供可信、安全、快捷的区块链应用，主要以 DApp 的形式为用户提供服务。整体的框图如图 .2 1 整体架构示意图所示。

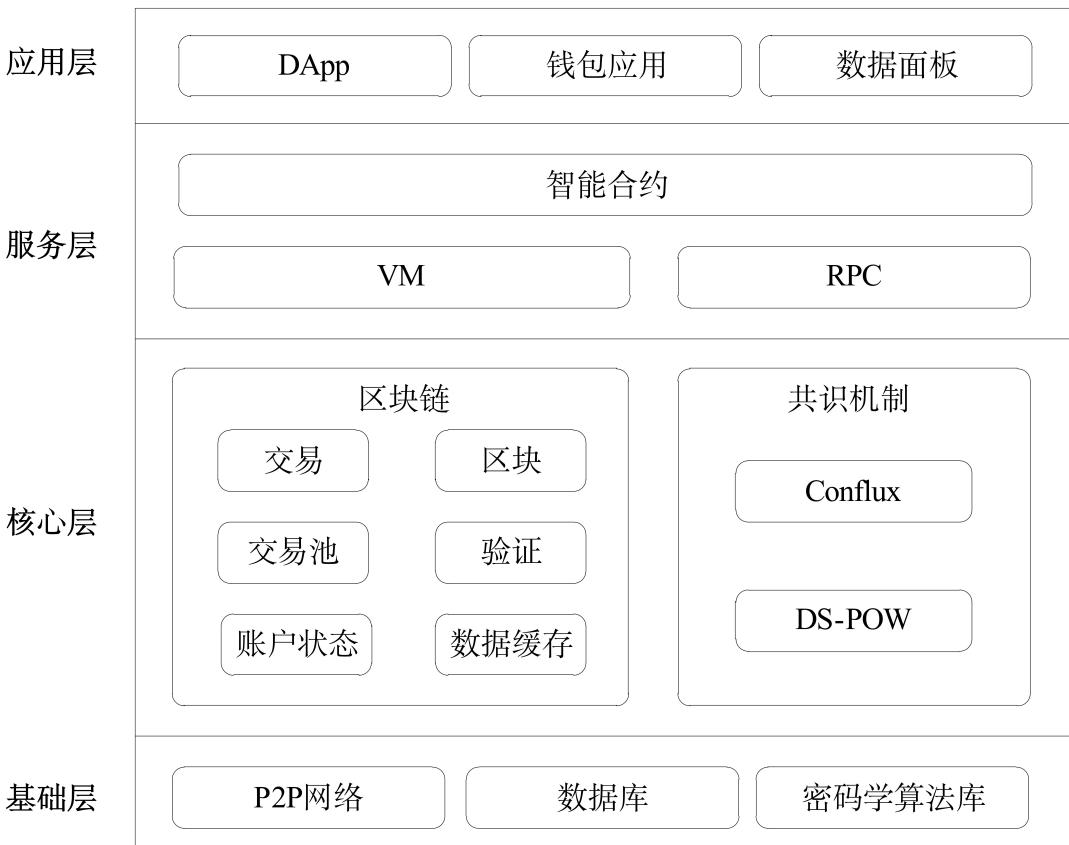


图 .2M 整体架构示意图

### .2.2.1 基础层

- P2P 网络

P2P (Peer-to-Peer) 网络实现了节点之间的通信，包括节点发现与管理、数据同步等功能。P2P 网络中各节点的地位平等，同时执行服务器和客户端的功能。资源的所有权和控制权被分散到网络中的每一个节点，从而把对中间服务器的依赖减少到最小。

- 数据库

Pulsar 选择 LevelDB 数据库来支持区块链系统重要数据的本地持久化存储。LevelDB 是一种开源的 No-SQL 数据库，使用 LSM (Log Structured Merge) 机制，具有很高的随机写、顺序读/写性能，适合区块链系统的存储。

- 密码学算法库

密码学在区块链系统中有三大功能，分别是保证区块链数据的完整性、机密性以及数据传输的安全性。在 Pulsar 中，创新的引入了门限群签名的算法，让多

重签名的应用场景更加丰富。

### .2.2.2核心层

- 区块链

区块链的本质是一个分布式账本，包括交易管理、区块管理、区块链状态管理以及账户状态管理等。交易管理包括交易数据的产生、验证及打包处理；区块管理包括区块的生产、传播、验证、维护等；区块链状态管理包括节点上区块的接收、主链的管理；账户状态管理则是对节点上所有账户的世界状态进行一致性维护。

- 共识机制

区块链是一个历史可追溯、不可篡改、能解决多方互信问题的分布式系统。因此其必然面临着数据一致性问题，我们称解决一致性问题的方法为共识机制。

在 Pulsar 中，我们使用 DS-POW 和 DAG 相结合的共识机制。DS-POW 机制通过股权和算力混合竞争的方式，有效地避免了算力垄断和股权垄断，同时还解决了 POW 和 POS 中的其它缺点。DAG 则是在区块 DAG 的基础上，允许矿工并行打包区块，大幅提高了系统的 TPS，解决了当前区块链系统中所存在的性能瓶颈。

### .2.2.3服务层

- 智能合约

智能合约本质是运行在区块链上的一段代码，由事件驱动且具有状态。其作用是按照合约参与者的意图，正确执行一组相对复杂的、带有触发条件的逻辑。部署和运行合约都需要消耗 gas，gas 的收取和合约的复杂度成正比。随着合约的执行，gas 会逐步消耗。如果合约执行完毕，剩余的 gas 会退回到调用者，如果 gas 消耗完，则合约回滚到执行之前的状态。

- 虚拟机

虚拟机提供智能合约的运行环境，它使用沙箱机制将智能合约的运行与外部环境隔离。

- RPC

Pulsar 提供多种 RPC 的编程接口，包括 JSON RPC、HTTP JSON RPC、HTTP RESTful 三种形式的 API。非智能合约类型的应用使用 RPC API 与服务层进行交互。

#### .2.2.4 应用层

应用层包含了运行在区块链上的各种 DApp，这些应用一般由第三方开发。作为公链平台，我们主要在应用层提供两个基础的应用：钱包和区块数据面板。

## .3 关键技术

### .3.1 DS-POW 混合共识

#### .3.1.1 概述和基本定义

POW 已经在比特币系统中运行近 10 年时间，经过了时间的考验，是目前最为安全稳定的共识算法。然而，由于近年来大部分算力掌握在几大矿池手中，业界对大矿池算力垄断的担忧日趋严重。另外 POW 还有如下问题：对资源的巨大消耗引起了环保人士的批评；很低的 TPS（即每秒处理交易量 Transactions Per Second, TPS）；将大多数人排除在挖矿之外导致很低的参与度，偏离了去中心化的设计目标。为了解决这些问题，Sunny King 和 Scott Nadal（通过 Peercoin, PPC）提出了 POS 的共识方案。POS 摒弃了算力竞争，能耗更低，让大部分用户可以参与进去，但同时也带来了另一方面的问题，如大股东股权垄断、区块高分叉倾向、无利害关系攻击（Nothing-at-Stake）、长程攻击（Long-range-attack）等。因此，作为一种改进，Daniel Larimer（通过 Bitshares, BTS）又提出了 DPOS 共识。DPOS 采用股东选举代理，再由代理轮流出块的方式维持系统的运转，可以同时保证低能耗、高 TPS 和不易分叉。但是 DPOS 的问题在于其代理相对固定，这造成了一种寡头垄断的倾向，可以看成一种多中心化系统，中心化程度远高于 POW 和 POS，普通用户无法对抗代理联合作恶。

综合以上共识机制的优劣，在基格链的共识机制中创新地使用了 POW+POS 相结合的方式，我们定义为 DS-POW（Delegated Stake - POW）。它有机地结合了以上共识的优点，通过股权结合算力综合竞争的方式，有效避免了算力或股权垄断等其它问题。DS-POW 与后述的 DAG 相结合能大幅提高系统的 TPS，非常适合作为去中心化公链平台的共识机制。

DS-POW 中主要的定义如下：

- **难度目标值：**挖矿时难度的表现形式。其大小与可行解范围成正比，与难度成反比。难度是通过难度目标值起作用的。POW 和 POS 都有各自的难度目标值，结合起来可以计算出综合难度目标值。
- **权重：**分为 POW 和 POS 的权重，分别表示 POW 和 POS 难度目标值对

挖矿的影响。

- **矿工**: 区块的产出者。分普通矿工和代理矿工。
- **普通矿工**: 无需注册，只进行 POW 挖矿。因此其可行解范围仅受 POW 难度目标值影响，不受 POS 难度目标值影响。
- **代理矿工**: 需要在链上注册，并公布一个自己的代理授权账户。股东可以向该账户发起股权授权交易以进行 DS-POW 挖矿。注册给代理矿工的总股权决定了该代理矿工挖矿的 POS 难度目标值。
- **股东**: 通过股权获取收益的用户。股东锁定自己的一部分资产作为股权并注册授权给代理矿工用于挖矿，代理矿工在挖矿时根据授权给其的所有股东的股权来确定 POS 难度目标值，进而确定可行解范围。

### .3.1.2 基本运行机制

DS-POW 的基本运行机制为：股东锁定自己的一部分资产作为股权并注册授权给代理矿工用于挖矿，代理矿工在挖矿时根据授权给其的所有股东的股权来确定可行解范围。授权到某个代理矿工的股权越多，其可行解范围越大，挖矿的综合难度越小；反之，授权到其的股权越少，其可行解范围越小，挖矿的综合难度越大，如果某个代理矿工没有任何股权授权，就相当于一个普通的 POW 矿工。挖矿成功后，其挖矿收益分为两部分：POW 收益和 POS 收益，其中 POW 收益是矿工挖矿的奖励，将直接发放给矿工，而 POS 收益是股东的股权奖励，将直接发放给股东。

股东在注册授权机制中锁定作为股权的资产，只是提供给矿工代理挖矿权限，并不会将资产本身转移给矿工。当股东不想继续挖矿或者想更换矿工时可自行取消授权并解锁资产，因此无需担心代理矿工腐败或其它作恶行为。

DS-POW 中有两种主要角色，其主要行为分别如下：

- **代理矿工**
  - ✓ 注册委托账户。
  - ✓ 等待股东授权。
  - ✓ 挖矿并分配收益。
- **股东**

- ✓ 给代理矿工的委托账户授权。
  - ✓ 等待代理矿工挖矿并分配股权收益。
  - ✓ 取消授权。

### 3.1.3 工作流程

DS-POW 挖矿的流程如图 .3K1 DS-POW 挖矿流程:

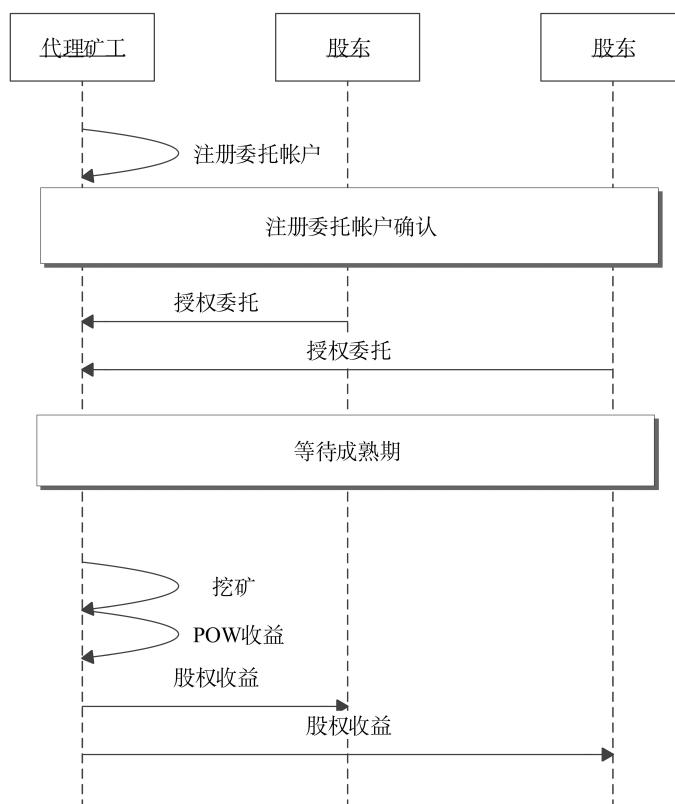


图 .3M DS-POW 挖矿流程

(1)代理矿工注册委托账户。矿工在链上注册为委托账户，注册成功后，股东可以向该账户发起委托授权交易，委托该代理矿工为自己挖矿。

(2)股东授权委托。股东锁定自己的一部分资产作为股权，并通过授权委托交易授权给代理矿工用于挖矿。该交易将建立委托账户与股东锁定的股权之间的关联。一个委托账户可接受多个股东的授权，委托账户的总股权为所有授权给该委托账户股权的总和。此时股东授权的股权为不成熟的股权，需要等待 N1 个区块之后才可用于挖矿。

(3)等待 N1 个区块高度后，此时授权的股权可用于挖矿。

(4)代理矿工挖矿。代理矿工使用 POW 方式进行挖矿获取 POW 收益，同时也会根据委托账户的股权余额给股东发放 POS 收益。挖矿时使用该代理矿工的综合难度。关于综合难度的计算与调整详见下一小节。

(5)代理矿工挖到区块后，将区块广播到全网。全网其它节点会验证区块的合法性。除了基本的交易验证外，还包括 POS 验证。全网验证通过后，DS-POW 挖矿完成。POS 验证如下：

- 该矿工的委托账户总股权是否合法（等于所有股东授权股权的总和）。
- 该矿工的本地 POS 难度目标值是否合法（由委托账户的总股权占全网总股权的比例确定）。
- 该区块是否如实按股东授权的股权和规定收益率将收益发放给所有股东。
- 该矿工的综合难度目标值是否合法（由全网 POW 难度目标值、矿工的本地 POS 难度目标值和当前全网的 POW/POS 权重共同确定）。

股东已经授权给代理矿工挖矿后，如果不想继续挖矿或者想更换矿工，或者有已锁定资产的使用需求，也可以取消授权。取消授权的操作完全由股东自行决定，不受代理矿工制约，取消之后代理矿工的委托账户就解除了与那部分股权的关联，并且那部分资产会在成熟期后解锁，变为可用状态。这样可以在股东和矿工之间形成一种制衡，避免了算力或股权垄断的系统风险。

股东取消授权的流程如图 .3L2 股东退出挖矿：

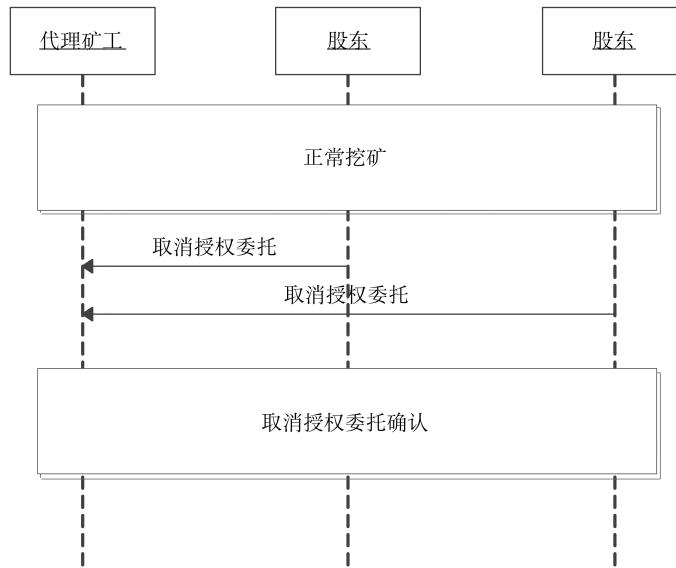


图 .3M2 股东退出挖矿

- (1) 股东发起取消授权交易。该交易将取消锁定股权与委托账户的关联。
- (2) 等待 N2 个区块高度后，取消授权交易成熟，此时授权的股权资产被正式解锁。

### .3.1.4 挖矿难度

DS-POW 挖矿时所使用的难度为综合难度。挖矿难度是通过难度目标值起作用的。

对于全网，有如下公式描述：

$$\text{Target}_{total} = \text{Target}_{pow} + \text{Target}_{pos}$$

其中：

$$\text{Target}_{pow} = \text{Target}_{total} * \text{Weight}_{pow}$$

$$\text{Target}_{pos} = \text{Target}_{total} * \text{Weight}_{pos}$$

$$\text{Weight}_{pow} + \text{Weight}_{pos} = 1$$

对于每个代理矿工，满足以下公式：

$$\text{Target}_{local} = \text{Target}_{pow} + \text{Target}_{pos\_local}$$

$$\text{Target}_{pos\_local} = \text{Target}_{pos} * \frac{\text{Stake}_{local}}{\text{Stake}_{total}} * \text{Count}(\text{Miner}_{delegate})$$

显然，它们的关系满足以下等式：

$$\text{Sum}(\text{Target}_{pos\_local}) = \text{Target}_{pos} * \text{Count}(\text{Miner}_{delegate})$$

$$\text{Average}(\text{Target}_{pos\_local}) = \text{Target}_{pos}$$

$\text{Target}_{total}$  代表全网综合难度目标值。

$\text{Target}_{pow}$  和  $\text{Target}_{pos}$  分别代表 POW 和 POS 的难度目标值。

$\text{Weight}_{pow}$  和  $\text{Weight}_{pos}$  分别代表 POW 和 POS 的权重。它们的比例会自动根据算力和币值比重动态调整，从而使 POW 和 POS 的影响各趋近于一半。

$\text{Target}_{pos\_local}$  代表代理矿工本地的 POS 难度目标值。

$\text{Target}_{local}$  代表代理矿工本地的综合难度目标值。

$\text{Stake}_{local}$  代表矿工的所有受托股权。

$\text{Stake}_{total}$  代表全网的所有股权。

$\text{Count}(\text{Miner}_{delegate})$  代表所有代理矿工的数量。

$\text{Sum}(\text{Target}_{pos\_local})$  代表所有代理矿工目标值的总和。

$\text{Average}(\text{Target}_{pos\_local})$  代表所有代理矿工目标值的平均值。

### 3.1.5 问题与解决方案

DS-POW 是 POW+POS 的有机结合，既能有效利用各自的优势，又能避免各自的劣势。但 DS-POW 可能遇到以下问题：

被委托的单个代理矿工停机问题。会影响到其所有的委托者。在此种情况下，股东可以撤销对其的委托，再委托给其它矿工。因此，该问题可以平滑解决，并不会有影响。

代理矿工算力竞争力不足问题，导致一直无法在出块竞争中获胜。此时委托的股东倾向于撤销对其的委托，再委托给算力竞争力更强的矿工。极端情况下，

可能会导致大量股东集中委托给大矿工形成寡头垄断。但是事实上，在没有出现算力可能形成垄断的几大矿池的前提下，其实无须担心该问题。而一旦出现有矿池可能形成垄断的倾向，出于对垄断现象的担心，大部分股东更倾向于选择不同的代理矿工，从而避免了垄断，这就自然解决了该问题。

### .3.1.6 波霎公链燃料 PUL 的总量及挖矿方式

总量设定为 21.5 亿左右，POW 和 POS 各占一半，减半周期为 2 年。

按 15 秒的出块间隔，则每年的区块数为  $60*60*24*365/15=2102400$ ，预计减半周期设为 2 年，即为 4204800 个区块。每个区块的初始 POW 奖励设为 128 个，则第一个减半周期的总产出为  $128*4204800=538214400$ ，POW 总量上限为 1076428800，即 10.76 亿，POW+POS 总产量为 2152857600，约 21.5 亿。

#### Pos 和 pow

在波霎链中有成熟周期的设计，且验证区块头是并行的，因此区块头中不能有对父区块头的依赖，否则无法验证。在发放奖励时，为了保持稳定，计算总产量也只计算已成熟周期的产量，这样每个成熟周期内使用的剩余 POS 空间都是相同的。由于没有每个区块实时更新剩余空间，会导致在单个周期内实际产出略多，但由于公式自带修正效果，对最终产量上限基本无影响

每次发放时的收益为

此处的 N 为发完 1 倍的收益所需要的次数，即预期利率的倒数。

如果预期年化 100%，则  $N=60*60*24*365/15=2102400$

在此公式下，当每次发放收益之后即会更新总量，从而使剩余 POS 空间发生变化。长期来看无论剩余空间如何变化，总产出总是会无限趋于总量上限。有两个接口可以查询当前 POS 和 POW 的产量，如下：

- 获取指定区块高度上 Pow 的总产出

```
eth.getPowTotalSupply(blockNumber)
```

blockNumber 为指定区块高度（可选），若不指定，则默认为当前最新高度

- 获取指定区块高度上 Pos 的总产出

```
eth.getPosTotalSupply(blockNumber)
```

blockNumber 为指定区块高度（可选），若不指定，则默认为当前最新高度

## 申请代理矿工

申请成为代理矿工需要全节点钱包，申请成为代理矿工的条件只需要满足一个条件：

- 有效余额大于 100 万

申请方法：

```
eth.sendTransaction({from:addr1,
txType:1,
delegateFee: delegatefee })
```

addr1 为要注册的代理矿工地址；

delegatefee 为代理矿工手续费费率

## 委托和撤销资金到代理矿工

### 1. 第一种方法

- 委托资金到代理矿工

```
eth.sendTransaction({from:stakeHolderAddr,
to:minerAddr,
value: stakeValue,
txType:2 })
```

stakeHolderAddr：股东地址；

- minerAddr: 矿工地址;
- stakeValue: 委托金额
- 股东撤销资金委托
 

```
eth.sendTransaction({from:stakeHolderAddr,
                          to:minerAddr,
                          txType:3 })
```

stakeHolderAddr: 股东地址  
 minerAddr: 矿工地址

委托挖矿成功后会自动获取到收益，同时委托金额将会被锁定。可以随时取消委托，取消挖矿后委托金额才会被解锁。

### .3.2 DAG

#### .3.2.1 概述

从 2009 年比特币上线至今，尽管出现了许多新思想与新技术，如共识机制创新，密码学应用创新等，但在主流成熟的系统中，区块的逻辑先后顺序一直是以链表的形式表现。具体来讲，每个区块包含一个验证合法的交易集合，区块之间通过哈希值引用形成父子关系确定区块的逻辑先后顺序，从而确定了包含在区块中所有交易的顺序。这也是区块“链”形象表述的来源。

所有节点要到达完全一致的区块顺序则要求系统中的生产新区块的节点（矿工）都必须看到相同的祖先区块集合。理论证明，在完全开放的网络环境下这是难以实现的。实践中，区块链系统只要能达到最终一致性即可。

比特币通过两种途径来达到最终一致性：

- 通过计算哈希难题（即 POW，工作量证明）控制整个网络中产生新区块的速率以保证区块数据能被同步到大部分网络节点。
- 在出现链分叉时，选择最长链作为唯一合法链。

比特币的解决方法简单可靠但性能低下，包括交易处理与交易确认的速度。一方面，选择最长链意味着强制丢弃分叉链（即使它们的交易数据并无冲突），

这导致部分交易处理结果被浪费；另一方面，较长的区块打包时间直接限制了交易确认的时间。同时当前区块可能被丢弃，因此交易有可能是无效的，一般认为经过 6 个区块后交易数据才是不可逆的。

链的分叉导致比特币网络算力发生分化，削弱了系统抗算力攻击的能力。以太坊在继承比特币的优点之上，使用 GHOST 算法选择主链，增强了系统抗算力攻击的能力。但以太坊并没有计入分叉区块数据内合法的交易。因此本质上，以太坊也只有一条链的交易数据作为有效交易，这导致以太坊的交易处理能力同样不高。

Pulsar 的 DAG 是基于区块顺序 DAG 的共识机制，在完全开放的去中心化分布式系统中具有高性能，易伸缩的特性。相较于传统的区块链系统在每个区块产生时确定交易逻辑顺序，DAG 计入所有分支区块内的合法交易，延迟对交易逻辑顺序的排列。DAG 使得节点能在一定程度上并行地处理交易，因而能大幅度提高整个系统的交易处理性能。

DAG 对系统性能的提升程度依赖下列几个因素：

- 区块的分支数量：在一定的前提下，分支越多，性能提升越明显，但分支数量多也意味着对节点计算资源的需求增大；分支越少，当减少为 1 时，相当于最长链选择算法。
- 分支区块内交易冲突的概率：当所有分支区块的交易都不存在冲突时，使得性能提升最大化；当所有分支都存在冲突的交易时，相当于最长链选择算法。
- 分支区块内交易重复的程度：当所有分支区块的交易都不重复时，使得性能提升最大化；当所有分支区块交易都重复时，相当于使用 GHOST 选择算法。

图 .3K3 是以上三个因素的图形化描述。

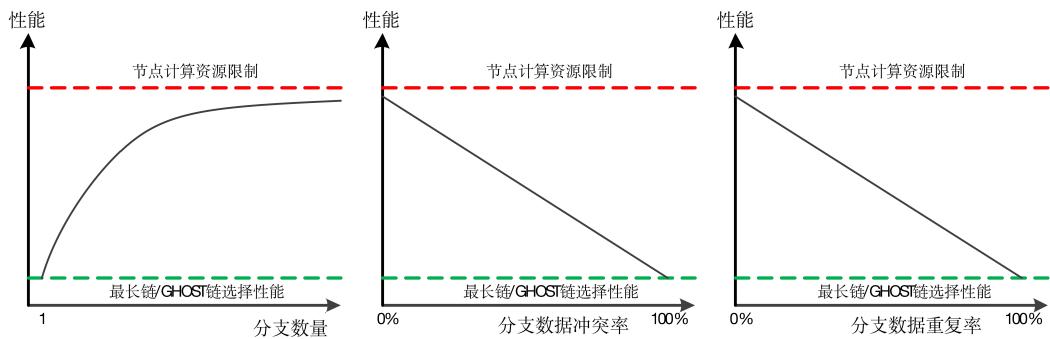


图 .3K3 性能曲线示意图

在 Pulsar DAG 共识机制中，所有节点根据 GHOST 算法选择一条区块分支作为轴链（Pivot Chain），轴链的最后一个区块作为当前节点要生成新区块的父区块。新区块会指向父区块，该指向称为父边（Parent Edge）。同时节点还需要找到并指向所有其它区块分支的最后一个区块，这些指向称为引用边（Reference Edge），引用边表示了区块的逻辑先后顺序。所有的区块通过父边和引用边形成一个 DAG 的拓扑结构。

轴链是一条从创世区块开始，只包含父边的一条链。通过轴链可以确定交易的全序关系。因此，只要轴链是稳定的，那么所有交易的顺序就是稳定不可逆的。

### .3.2.2 原理

#### (1) 概述

DAG 从创世块开始，创世块决定了链的初始状态。和传统的区块链相比，DAG 主要区别是由区块和边组成的是 DAG 结构而不是线性结构。由于网络的延迟，每个节点的 DAG 结构可能有所不同。为了使所有节点对最终的区块和交易全序达成一致，DAG 维护了每个节点的本地 DAG 状态。

- P2P 网络

所有节点是通过 P2P 网络连接的，如图 .3L4 DAG 架构示意图顶部所示。节点在任何时候发起一笔交易（如图 .3K4, Tx9）或产生新区块（如图 .3K4, Block B），就通过该网络广播到其他节点。

- 待处理交易池

每个节点都维护了一个待处理交易池，如图 .3L4 DAG 架构示意图底部所示。池中保存了该节点收到但未打包到任何区块中的交易。节点在任何时候从网

络中收到交易都会先加入到它的交易池中（如图 .3K4, Tx9）。当节点收到一个新区块（无论是自己产生还是从其他节点收到）后，它会将区块中的交易从交易池中移除（如图 .3K4, Tx1、Tx2、Tx3、Tx5）。由于网络延迟或恶意节点的存在，并发区块中可能包含了重复或冲突的交易，这个问题会通过共识协议解决。

- 区块生成器

每个节点都会运行一个生成有效新区块的区块生成器，交易池中的交易会被打包到新区块中（如图 .3K4 底部所示）。基格链采用 DS-POW 机制，区块生成器通过解决一个数学难题来生成新区块，该难题的难度和股权相关。同时，通过动态调整难度，使网络保持一个稳定的区块产出率。

- 本地 DAG 状态

每个节点都维护了一个本地 DAG 状态，它包含了节点知道的所有区块。在 DAG 中，每个区块通过边连接了之前的多个区块，这种状态就是 DAG（如图 .3K4 中部所示）。节点收到新区块时，就会更新本地的 DAG 状态。

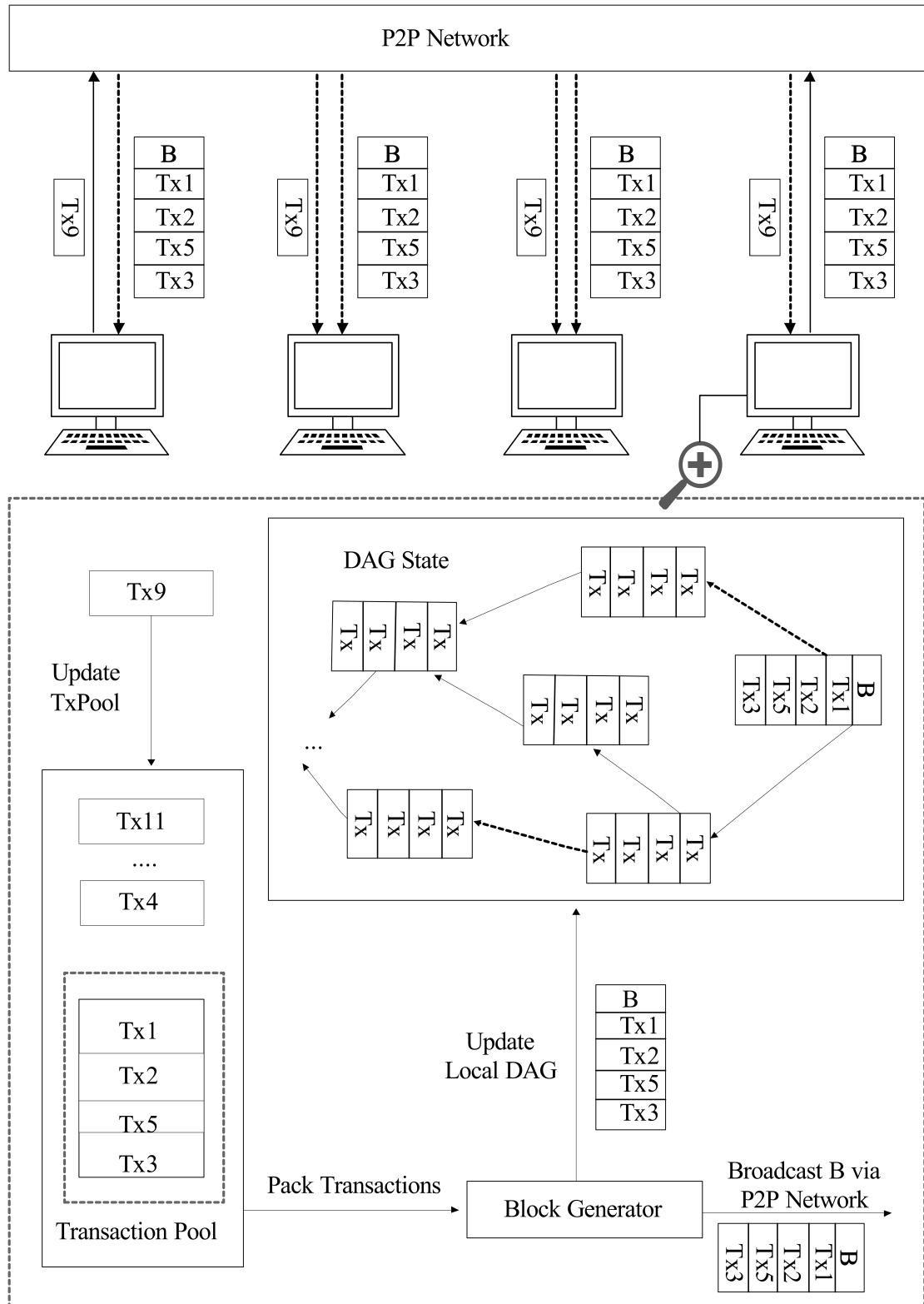


图 .3M4 Pulsar DAG 架构示意图

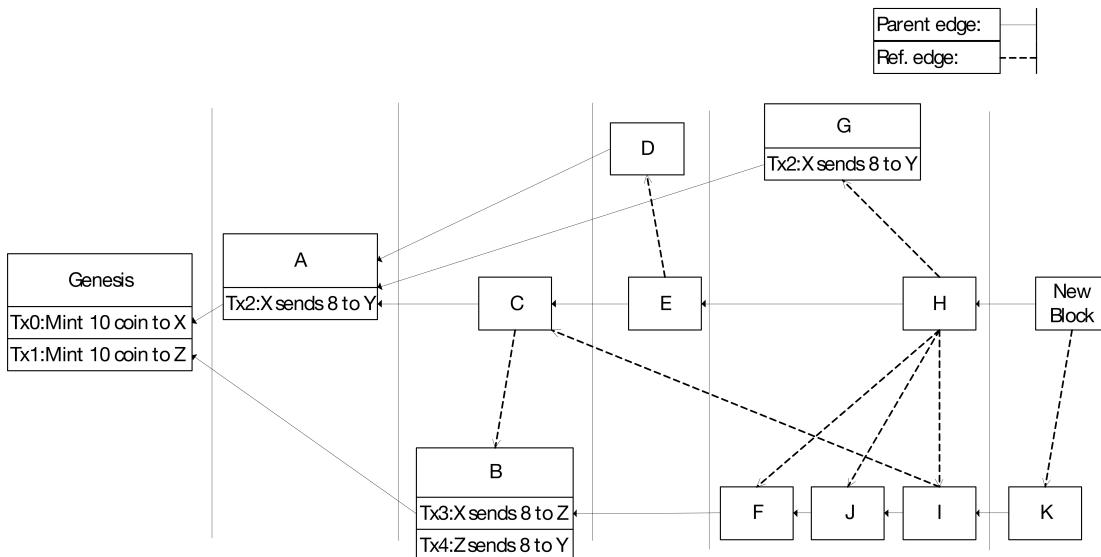


图 .3M5 DAG 状态示意图

## (2)共识协议

Pulsar DAG 共识协议基于节点的本地 DAG 状态，目的是让所有节点对区块的全序达成一致。因此，通过区块的顺序和其内部交易的顺序，就能推导出所有交易的全序，从而有效地解决了冲突和重复的交易。DAG 只选顺序最前的为有效交易，其后或重复的交易会被丢弃。

### ● DAG 和边

为了表示区块间的关系，定义了两种边：

- ✓ 父边 (Parent Edge)

除了创世块外，每个区块都有且仅有一条指向父区块的边（如图 .3K5，实线箭头），称之为父边。父边代表一种选择关系，表示当前区块选择了父区块的历史交易。

- ✓ 引用边 (Reference Edge)

除创世块外，每个区块可以有多条指向其他区块的边（如图 .3K5，虚线箭头），称之为引用边。引用边代表区块产生的先后顺序，指向了在当前区块产生之前的区块。

### ● 轴链

在 DAG 结构中，每个区块都可能有向其它区块的引用边，连上这些引用边一起，而这些被引用的区块会增加当前区块的权重。在协议中会选择从根节点到当前分支节点中权重最重的路径作为轴链。这和比特币选择最长链是有明显区别

的。通过 GHOST 选择算法，从创世块开始选出唯一的轴链。

- 添加新区块

节点在任何时候加入新块时，它首先会根据本地的 DAG 状态计算出轴链并将轴链的最后一个块作为新区块的父块。这样就使得该条轴链权重更大，因此其它节点同样选择其作为轴链的可能性更大。然后，找出所有末端的无入边的区块并且从新块到它们连一条引用边。

- Epoch

由于有父边、引用边和轴链，可以将所有的区块分为不同的 epoch。轴链上的每个块都对应了一个 epoch。轴链上对应的区块，可以通过父边和引用边到达本 epoch 内其它所有区块。每个 epoch 的区块不包括其它 epoch 的区块。

- 区块全序

DAG 通过如下规则对区块进行排序：首先根据区块所处的 epoch 排序，然后再根据它们的拓扑顺序对每个 epoch 内的区块进行排序。如果同一个 epoch 中的两个区块没有直接的先后关系，那么通过两个区块的 id (哈希值) 大小来决定先后顺序。

- 交易全序

DAG 是根据区块的全序对交易进行排序。同一个块中的交易是通过它们在块中的先后顺序决定。如果交易冲突或重复，那么会保留顺序最前的交易，丢弃之后的交易。

### (3) 安全分析

安全性对任何区块链系统都很重要，我们将分析潜在攻击策略以及说明为什么面对这些攻击时 DAG 是安全的。如图 .3K5 所示，假设攻击者想重置 B 区块中的交易 Tx4，那么它需要改变区块的全序。一个简单的想法是直接将创世块作为 B 区块的父块。但是，B 区块没有子块，它不会被选为轴链上的块，并且由于 epoch 的定义，新区块必须等轴链上的块引用它。因此，B 区块仍然属于未来的某个 epoch，之前的区块全序不会改变。

诚实的节点始终工作在轴链上，会使得轴链变得更长更重。所以如果攻击者想重置轴链，则必须拥有超过 50% 的算力。

### .3.2.3 实现

本节描述 DAG 的具体实现。

#### (1) 算法基本定义

对任意时刻，DAG 中用户的本地状态为一个图，可描述为  $G = \langle B, g, P, E \rangle$ 。

其中：

- $B$  为  $G$  中的区块集合。
- $g \in B$  为创世区块。
- $P$  为将区块  $b$  映射到父区块  $P(b)$  的函数， $P(g) = \perp$ 。其中， $\perp$  表示创世区块的父区块，此处仅用于表示，实际并不存在。
- $E$  为图  $G$  中引用边和父边的集合。
- $e = \langle b, b' \rangle \in E$  为从区块  $b$  到区块  $b'$  的有向边，其中区块  $b'$  先于区块  $b$  产生。

对任意区块，始终有一条父边指向其父区块，即  $\forall b \in B, \langle b, P(b) \rangle \in E$ 。DAG 中所有节点使用同一个确定性哈希函数，每个区块使用该函数能求取唯一的哈希值，即满足  $\forall b \neq b', \text{Hash}(b) \neq \text{Hash}(b')$ 。

相关函数定义如下：

- `Chain()` 返回从创世区块到给定区块沿唯一父边形成的链。
- `Child()` 返回给定区块的子区块的集合。
- `Sibling()` 返回给定区块的兄弟区块的集合。
- `Subtree()` 返回给定区块在亲本树中的子树的集合。
- `Before()` 返回在给定区块之前生成并与其有直接关系的区块的集合，即在此之前产生，与其有父边和引用边连接区块的集合。
- `Past()` 返回在给定区块之前产生的区块的集合。

函数的形式化定义如下：

- $G = \langle B, g, P, E \rangle$
- $\text{Chain}(G, b) = \begin{cases} g, & b=g \\ \text{Chain}(G, P(b)), & \text{其他情况} \end{cases}$

- $\text{Child}(G, b) = \{b' | P(b') = b\}$
- $\text{Sibling}(G, b) = \text{Child}(G, P(b)) - \{b\}$
- $\text{Subtree}(G, b) = (\bigcup_{i \in \text{Child}(G, b)} \text{Subtree}(G, i)) \cup \{b\}$
- $\text{Before}(G, b) = \{b' | b' \in B; < b', b > \in E\}$
- $\text{Past}(G, b) = (\bigcup_{i \in \text{Before}(G, b)} \text{Past}(G, i)) \cup \{b\}$
- $\text{TotalOrder}(G) = \text{DAGOrder}(G, \text{Pivot}(G, g))$

在本节的剩余部分，将使用有序表来表示链及串行化序列。“o”表示两个有序表的级联。

## (2)轴链选择

给定一个 DAG 状态  $G$ ,  $\text{Pivot}(G, g)$  返回从创世块  $g$  开始的轴链上的最后一个区块。该算法选择当前时刻权重最大的分支即为当前的轴链。

$\text{Pivot}(G, g)$  的伪代码定义如下:

*Input:* The local state  $G = \langle B, g, P, E \rangle$ , and a starting block  $b \in B$

*Output:* The last block in the pivot chain for subtree of  $b$  in  $G$

**if**  $\text{Child}(G, b) = \emptyset$  **then**

**return**  $b$

**else**

$s \leftarrow \perp$

$w \leftarrow -1$

**for**  $b' \in \text{Child}(G, b)$  **do**

$w' \leftarrow |\text{Subtree}(G, b')|$

**if**  $w' > w$  **or**  $w' = w$  and  $\text{Hash}(b') < \text{Hash}(s)$  **then**

$w \leftarrow w'$

$s \leftarrow b'$

---

**return**  $Pivot(G, s)$

### (3) 共识主循环

共识主循环处理两种类型的事件：

第一种类型的事件是通过底层的网络从其它节点接收 DAG 更新信息。节点更新本地的 DAG 状态，同时通过网络转发接收到的信息。

第二种类型的事件是本地区块生成器如何生成新区块 b。在此情况下，节点将 b 加入到本地 DAG 中，并按照以下步骤更新 G。首先，设置本地 DAG 中轴链的最后一个区块为区块 b 的父区块。然后，节点找出本地 DAG 中所有没有入边的区块，并创建一条从 b 到这些区块的引用边。最后，节点通过网络将更新后的 G 广播到其他节点。

共识主循环过程的伪代码描述如下：

**Local State**: A graph  $G = \langle B, g, P, E \rangle$

**while** (*Node is running*) **do**

**upon event** Received  $G' = \langle B', g, P', E' \rangle$  **do**

$G'' \leftarrow \langle B \cup B', g, P \cup P', E \cup E' \rangle$

**if**  $G \neq G''$  **then**

$G \leftarrow G''$

*Broadcast the updated G to other nodes*

**upon event** Generated a new block  $b$  **do**

$a \leftarrow Pivot(G, g)$

$E' \leftarrow E \cup \{ \langle b, t \rangle \mid \forall b' \in B, \langle b', t \rangle \notin E \}$

$G \leftarrow \langle B \cup \{b\}, g, P[b \rightarrow a], E' \rangle$

*Broadcast the updated G to other nodes*

为简化描述，在伪代码中的表述为广播整个图到网络。在实际 DAG 实现中，DAG 广播并转发的是每个单独的区块，这样可以避免不必要的网络传输。

#### (4)全序列

本文定义 DAGOrder()为所有区块的排序算法。给定一个本地状态 G 和轴链上一个区块 a, DAGOrder(G, a)返回 epoch a 及其之前的所有区块的列表。通过 DAGOrder(), 也可以方便的定义求取整个本地状态 G 的全序函数 TotalOrder(G)。

DAGOrder()的伪代码如下:

*Input:* The local state  $G = \langle B, g, P, E \rangle$  and a block  $a$

*Output:* A list of blocks  $L = b_1 ob_2 o \dots ob_n$ . Where  $b_1 = g$  and  $\forall 1 \leq i \leq n, b_i \in B$

$a' \leftarrow P(a)$

**if**  $a' = \perp$  **then**

**return**  $a$

$L \leftarrow DAGOrder(G, a')$

$B_\Delta \leftarrow Past(G, a) - Past(G, a')$

**while**  $B_\Delta \neq \emptyset$  **do**

$G' = \langle B_\Delta, g, P, E \rangle$

$B'_\Delta \leftarrow \{x | |Before(G', x)| = 0\}$

Sort all blocks in  $B'_\Delta$  in order as  $b'_1, b'_2, \dots, b'_k$ .

such that  $\forall 1 \leq i \leq j \leq k, b_i \in B, Hash(b'_i) \leq Hash(b'_j)$

$L \leftarrow L ob'_1 ob'_2 o \dots b'_k$

$B_\Delta \leftarrow B_\Delta - B'_\Delta$

**return**  $L$

该算法首先会递归的对之前所有的 epoch 中的区块进行排序(例如 epoch P(a) 和之前的 epoch)。然后该算法对 a 所在的 epoch 中的所有区块进行拓扑排序，并将结果添加到列表。当发生冲突时，使用唯一的 id 大小排序来解决冲突。

## .4 应用场景

### .4.1 数据交易

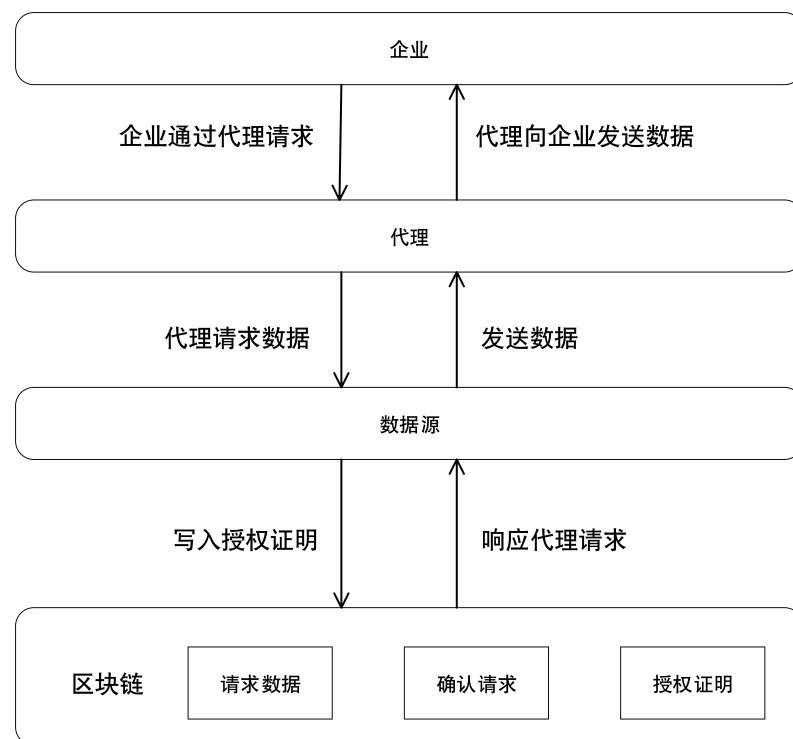


图 .4K1 区块链数据交易解决方案

数据是互联网、AI、大数据经济中最重要的组成部分，分析数据会产生许多重大发现。而对于数据收集能力有限的企业，数据交易将是一个促进企业创新、增加收入的重要手段。目前，市场上数据交易存在透明度低、非法倒卖、易被篡改等问题，导致了数据交易的规模受限。

由于区块链的去中心化、不可篡改和安全性，能够让参与主体之间建立信任，推进数据交易的可持续增长。数据的交易、授权和所有权都会记录在区块链上，数据的所有权可以被确认，精细化的授权可以规范数据的使用。同时，数据的流向完全透明，从采集到分发的每一步都记录在区块链上，数据可溯源，数据源可控，从而加强了数据质量。因此，基于区块链的去中心化数据交易平台，可以形成规模巨大的全球化数据交易场景。

在物联网领域中，由于分布着大量的设备、传感器，会收集到海量的数据。去中心化网络数据交易能够进行实时和精细化交易，可以成为物联网领域中数据交易的媒介。同时，增加了信任度与透明度，很好地支持了物联网领域的生态建

设。最后，去中心化交易如能克服扩展性差、交易成本高、交易速度慢等缺点，就能加速推进商用化进程。

## 4.2 商品溯源

商品的溯源防伪仍是目前社会和企业的主要难题。很多商品虽然有相应的防伪标识，但因为人为因素在整个供应链中参与过多，商品的真实可靠性仍然无法保证。区块链技术依托其具有的数据不可篡改、交易可追溯等特性，可以很好的解决供应链体系内各参与方可能产生的纠纷，实现有效的追责和商品防伪。

通过区块链技术，可以让每个商品的静态、动态信息在生产制造企业、仓储企业、物流企业、各级分销商、零售商、电商、消费者以及政府监管机构中共享、共识。具体如下：

- 信息记录：记录每个商品的关键信息到区块链中，利用区块链公开不可篡改的属性，防止数据伪造。
- 信息追踪：每个商品对应唯一标识码，即“一物一码”。通过智能手机、传感器等终端设备对商品的标识码进行自动识别，追踪商品的转移关系。
- 多方参与：通过区块链共享的特性，企业能可靠的掌握周边企业、相关交易、相关商品以及最终消费者的情况，同时能提供监管方介入的接口，有利于政府和市场的监管。

商品溯源采用区块链技术的示意图如图 .4K2:

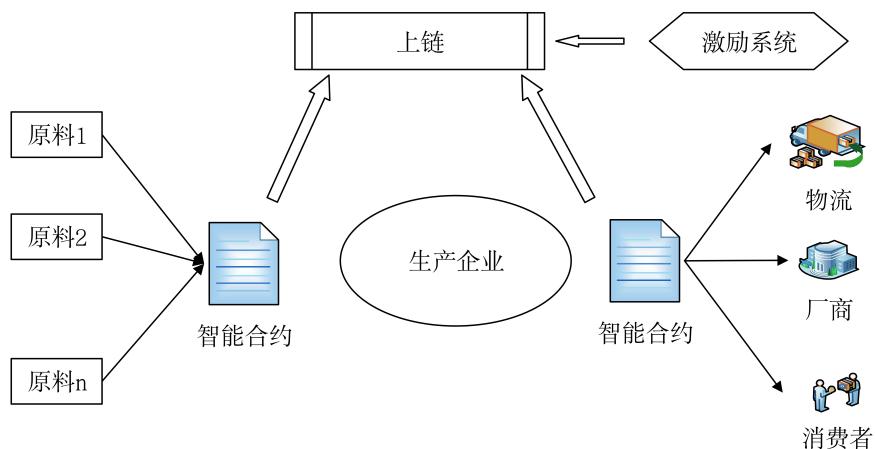


图 .4K2 商品溯源示意图

### 4.3 车联网

在物联网的时代，用区块链实现信息准确共享，构建新经济模式是新的发展方向。车联网是基于网络连接、车载传感器数据收集，配合云端的设备管理、大数据分析等技术的融合，并加上大量的应用创新而形成的一种综合性技术。由于各种技术的突飞猛进，使得汽车的经营模式发生重大转变。从传统的单一产品服务到多方共同维护，再到将来的价值链。由此可见，汽车领域中技术服务的市场会持续增大。

具体来说，车联网有如下特点：

- 数据来源广

大量的车载设备、传感器、网络终端，使得数据收集分析更加快速便捷，呈分布式状态特点。

- 参与度高

不仅仅是传统车主，包括了车厂、4S 店、保险、二手市场、车辆管理部门、执法部门、应用开发商等多方参与。

- 利益不一致且无单一可信方

如用户、保险、4S 店之间存在一定的博弈。虽然有仲裁，但是处理的过程往往是漫长的。

- 客观取证

如事故记录等客观事实会被多方采用。

由此可见，区块链技术所具有的技术恰恰符合车联网的上述特点。通过区块链防篡改、可追溯的特性，可以记录车辆完整的生命周期并与所有的参与方共享，实现去中心化的消息互通。与此同时，结合智能合约、链上链下等技术，可以实现各种流程的自动化，大大提升效率。

数据隐私性是区块链技术在车联网领域中的关键挑战，这需要通过技术层面和非技术层面同时加以解决。技术上可以通过加密技术，保证数据的隐私性。非技术层面，可让用户选择是否同意数据共享等。当然，区块链的应用还有其它诸多挑战，存在着很大的进步空间。

## 4.4 供应链金融

区块链天然的去中心化、共享、防篡改的特性加速了信息的安全分发、呈现、传输和处理。从区块链中收益最多的是那些参与者之间信任度低、交易记录安全性和完整性高的行业，金融行业正是其中之一。有报告显示，分布式账本技术每年可为金融行业节省成本 50-70 亿美元。这种成本的降低主要来自于区块链对现有业务的改进，如对账流程的优化、用户身份认证效率的提升、支付价值链的改善等。

供应链金融是区块链在金融领域的最佳应用场景之一。供应链金融具有系统性、结构性的业务理念，决定了信息流是供应链金融风控的关键。风控的基础和难点在于如何获取准确、全面、有效的数据。通过区块链技术可以为参与供应链的众多企业和金融机构搭建一座可信的信息桥梁，即从源端获取信息，通过区块链保证端到端的信息数据透明、不可篡改，各参与方通过区块链系统实现资源、物流、资金等信息的共享。金融机构根据企业背景、实时运营数据进行决策，通过区块链系统，能大大缩短资料数据收集、校验和评估的时间，降低风险成本，提升决策的精准性和效率。同样，企业也能够通过供应链金融更快的获取资金，获取更方便的服务。

区块链技术在供应链金融中应用具体体现在以下两个方面：

- 通过不可篡改的特性，记录供应链金融中相关企业以及企业周边的资金流、物流、产品流等过程，将数据完整记录在案。
- 通过智能合约等技术手段，为企业间的“合同信任”添加新的保障措施。

通过智能合约，能简化企业之间的相互担保、风险共担、合约履行等流程，能大大降低违约纠纷等问题带来的成本。

供应链金融采用区块链技术的示意图如图 4K3：

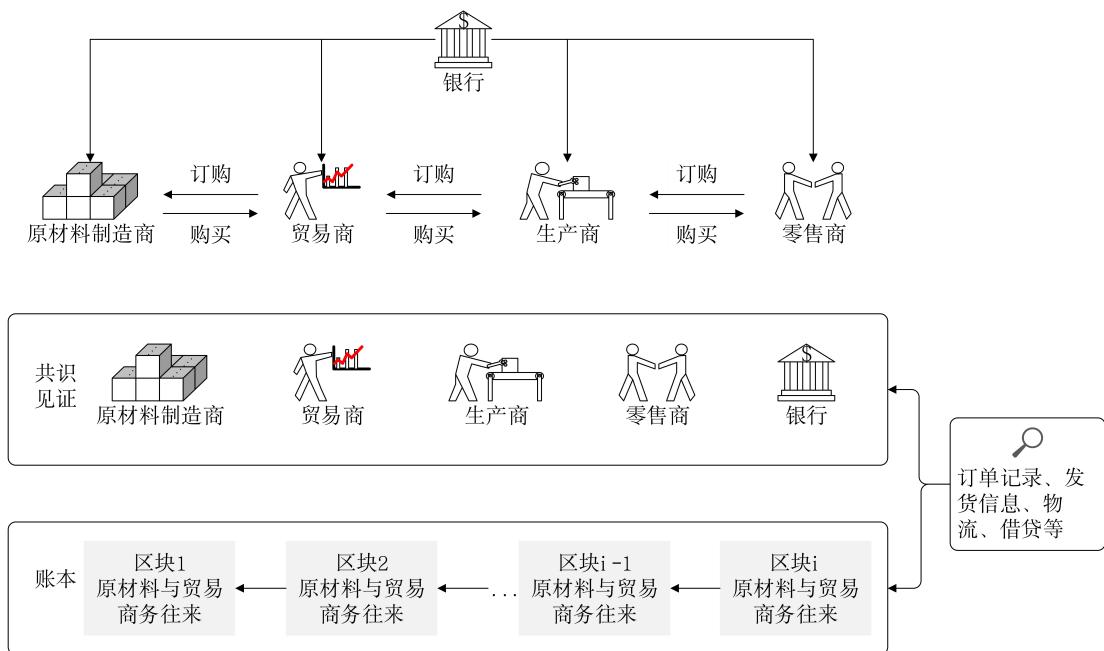


图 .4K3 供应链金融示意图

## 4.5 身份认证

身份及接入管理服务是区块链技术应用的一个重要领域，不仅如此，由于区块链技术具有高可靠性、可追溯和可协作等特质，使得其在身份及接入管理服务的应用领域具备成为基础技术的潜力。

伴随着数字化进程的加速，身份及接入管理服务的应用领域将越来越广泛，包括互联网、物联网、社会和经济生活等。在这些应用领域中，身份及接入管理服务的典型作用是保障具备合法身份的用户或设备可以安全、高效的接入和享受服务。

身份及接入管理服务在各个应用领域中所处的位置至关重要，但目前该服务也一直面临着隐私泄露、身份欺诈以及碎片化等问题，给用户、设备和系统均带来极大的挑战。

区块链技术的引入和发展，为解决上述问题提供了新的思路。将区块链技术应用到身份及接入管理服务中，将有可能形成一种协作的、透明的身份管理方案，有助于企业、组织更好的完成身份管理和接入认证。

Pulsar 区块链技术在身份认证接入管理服务的应用将依托软件、硬件和区块链平台等一系列的配套措施，为企业、政府、组织、个体提供安全、高效的身份

管理服务，示例如图 .4K4。

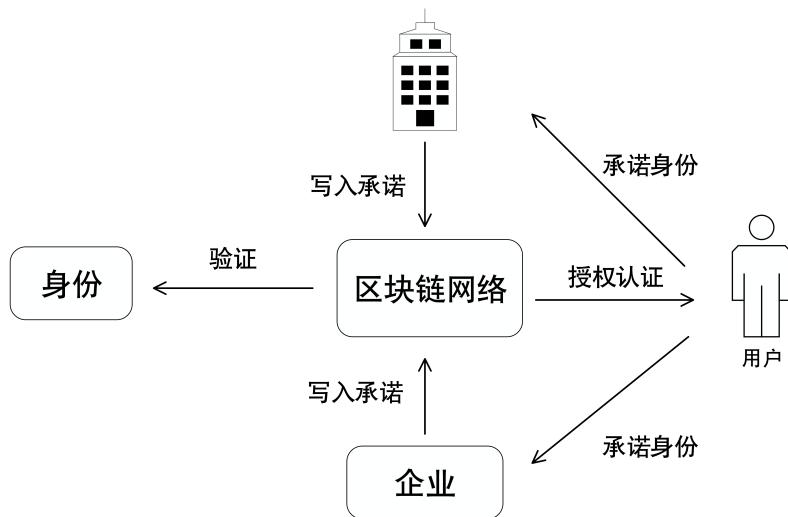


图 .4K4 身份认证示意图

## .4.6 股权登记转让

将区块链技术用于对股权、债券等资产进行加密，有助于完善登记与流转服务。尤其是区块链的去中心化、不可篡改的特性，能够较大幅度提高资产跨领域流通效率，降低交易成本。

目前的股权登记仍然需要人工进行，股东名册维护较为繁琐，历史交易维护复杂，追踪困难。传统股权交易是在双方信用的基础上，建立双边授信后才可交易，信用风险由交易双方承担，而交易平台则集中承担市场交易参与者的信用风险。

区块链在股权登记方面的优势主要体现为以下几点：

去中心化不可篡改的记录，适用于股权债券等资产的登记。

不需要中心化的信任，便于加密资产的转让和交易。

共享的信息披露记录，易于符合相关部门监管的要求。

股权登记采用区块链技术应用的示意图 .4K5:

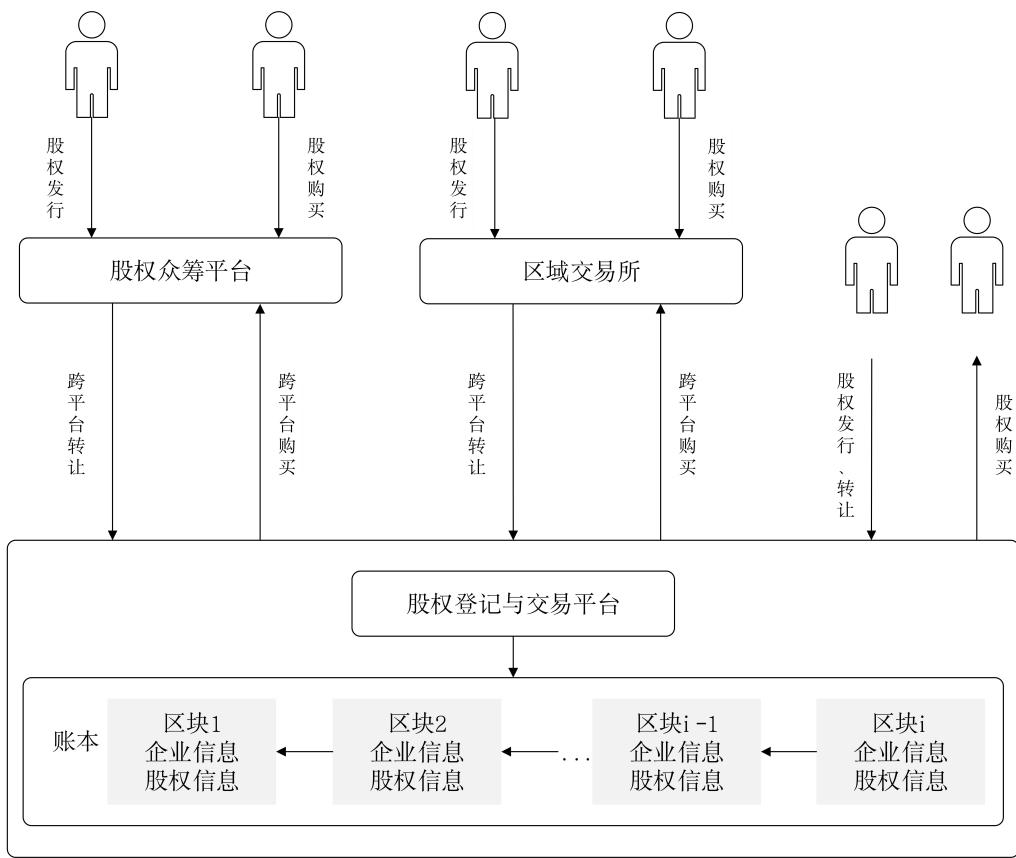


图 .4K5 股权登记示意图

## .5 路线图

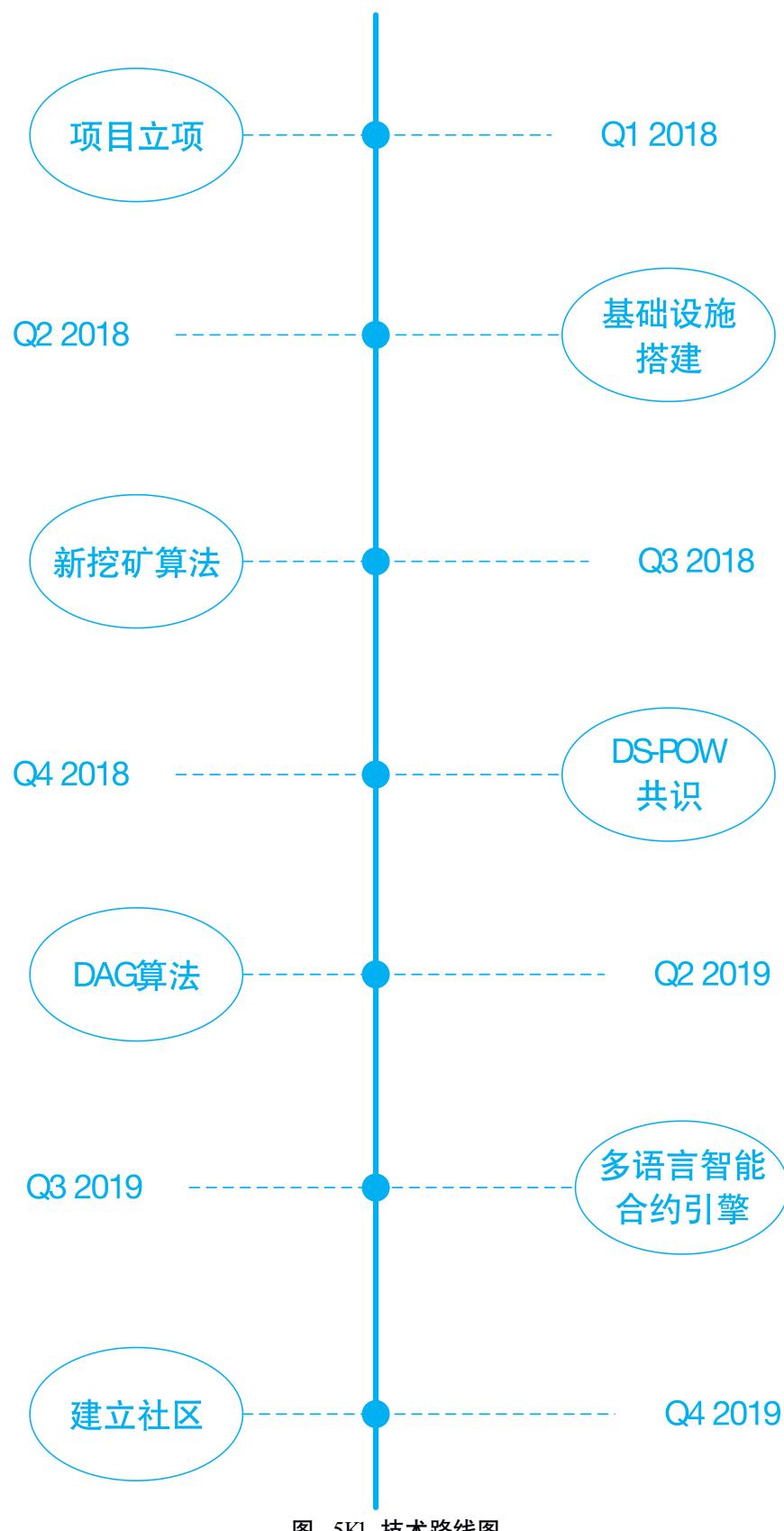


图 .5K1 技术路线图

## .6 免责声明

本白皮书仅作为传达信息之用，内容仅供参考，不构成任何投资买卖建议或相关邀约。本文档不构成也不应被理解为提供任何买卖的行为，或邀请买卖任何形式证券的行为，更不是任何形式上的合约或者承诺。

项目团队将不断进行合理的尝试，以确保白皮书中信息的真实准确。开发过程中，系统可能会进行更新。文档的部分内容可能会随着项目的进展在新版的白皮书中进行相应的调整，相应的修改或新版白皮书会通过官方网站进行发布。请参与者务必及时获取最新版白皮书，并根据其具体内容做出相应的决策。

团队将不遗余力的实现白皮书中所提及的目标，但团队不作任何承诺。

## .7 团队介绍

### **Jammy**

在信永中和元都集团工作近 8 年，曾担任审计经理和财务总监，长期深耕于财务领域，对中小企业各个行业的财税问题有一定研究。

### **Sabrina**

在 PwC 的金融咨询部工作近三年，主要服务于金融机构风险管理咨询项目。曾参与 AI 模型交易对手预警模型项目、反洗钱产品评估项目以及风险数据集市建设项目等。

### **Marvin han**

纳斯达克上市公司技术总监，致力于前后端技术和架构领域、曾多次参与大型银行和互联网公司核心系统的架构设计和开发 Ella 在泸州老窖成都公司任职 6 年，担任人事行政经理，组织了 6 年泸州老窖春季糖酒会的展会工作。曾经用 4 个月的时间成功搭建了西安、成都、上海 3 个办事处，完成了基本人员招聘，完成了组织架构的搭建。4 年多互联网公司行政人事管理经验，对人员招聘和绩效考核有一定的见解。

### **Ella**

在泸州老窖成都公司任职 6 年，担任人事行政经理，组织了 6 年泸州老窖春季糖酒会的展会工作。曾经用 4 个月的时间成功搭建了西安、成都、上海 3 个办事处，完成了基本人员招聘，完成了组织架构的搭建。4 年多互联网公司行政人事管理经验，对人员招聘和绩效考核有一定的见解。

### **Clearlee**

曾任 ThunderSoft 软件开发工程师，有 5 年项目开发经验，熟悉汇编,java,c/c++ 开发语言，对系统底层，web 开发和移动端开发有较深入研究。

### **Joe**

多年来一直从事移动互联网开发，曾就职于世界 500 强企业任 iOS 资深开发工程师，负责技术指导，项目开发。工作期间获得多种奖励荣誉。擅长 APP 的架构、性能优化等工作。

### **Lee williams**

7 年测试行业经验，曾就职于 Lenovo, 暴风等上市企业，曾担任测试主管、测试架构师等职务，主导过多个大型项目的测试工作，有丰富的软件测试经验。精通功能测试、web、接口、性能、APP 自动化测试。

#### **kate UI**

资深设计，4 年教学经验，曾在某高校负责数字媒体专业核心课程教学，并带领设计团队与人民教育出版社合作获得多项国家级大奖。有着非常完整的大型项目经验。

#### **Ricky Jerret**

拥有 10 余年的股票证券、金融行业从业经验。曾在香港 Interactive BrokersLtd. 任职 负责量化交易，管理亚洲市场的量化投资组合，并带领团队开发算法交易平台。

#### **Nancy**

从事 ui 设计 7 年，服务过区块链、金融、电商、科技、游戏等行业，积累了丰富的实战经验，创新思考，给予有效的视觉策略支持。

#### **Able**

5 年 Java 互联网研发经验，曾参与大型电商平台、社交产品等多家互联网公司的开发，曾担任电商项目研发中心组长，具有丰富的项目经验，擅长分布式、微服务等技术领域。

#### **Harlen**

8 年运维老油条，曾就职于中国信息通信研究院云计算与大数据研究所，担任运维工程师一职。致力于海量业务的技术运营保障工作，对架构规划、性能调优、用户体验提升等运维增值服务感兴趣。

#### **Karl Thomason**

具有 12 年的分布式数据库系统和 P2P 网络架构设计经验，研究方向包括区块链共识、性能和生态系统。在技术和研究方面都有丰富经验。

#### **Brian**

国内一线互联网公司从业经历，负责过多个领域的全栈项目开发，纵跨直播平台、新媒体管理系统、区块链等多个行业领域，对于微服务框架、服务治理等技术颇有研究，自创多种优化流程。

**Anson**

互联网行业老兵，有前后端开发经验，熟悉 python、go、nodejs 等多种语言，目前国内某独角兽互联网大厂工作，专注于解决区块链扩容问题，关注最新区块链技术研究，探索区块链在身份认证、物联网、安全与隐私方面的应用。

**Benson Leo**

具有多年的分布式数据库系统、区块链开发和架构设计经验，包括领导 FaceBook 开发者，在技术研究方面都有经验丰富，经常在区块链和金融科技活动中发表演讲。

**Mani Wood**

毕业于 UCLA(加州大学洛杉矶分校)商业金融专业硕士。曾任职于全球电商巨头美国亚马逊总部，负责网络与电商营运工作，拥有近十年移动互联网产品架构及数据运营经验。回国后加入国际投行摩根士丹利负责企业资产管理及信息咨询，为中国多家大型企业提供并购及海外上市服务

## .8 参考文献

- [1] Nakamoto S. Bitcoin: A peer-to-peer electronic cash system[J]. 2008.
- [2] Etherum White Paper. <https://github.com/ethereum/wiki/wiki/White-Paper>.
- [3] Ethereum Yellow Paper: <https://ethereum.github.io/yellowpaper/paper.pdf>.
- [4] SOMPOLINSKY, Y., AND ZOHAR, A. Secure high-rate transaction processing in bitcoin. In International Conference on Financial Cryptography and Data Security (2015).
- [5] Conflux Paper: <https://arxiv.org/pdf/1805.03870>
- [6] Gossip:  
[https://www.stat.berkeley.edu/users/aldous/260-FMIE/Papers/shah\\_GA.pdf](https://www.stat.berkeley.edu/users/aldous/260-FMIE/Papers/shah_GA.pdf)
- [7] SOMPOLINSKY, Y., AND ZOHAR, A. Secure high-rate transaction processing in bitcoin. In International Conference on Financial Cryptography and Data Security (2015), Springer, pp. 507–527.
- [8] SOMPOLINSKY, Y., LEWENBERG, Y., AND ZOHAR, A. Spectre:Serialization of proof-of-work events: confirming transactions via recursive elections, 2016.
- [9] BUTERIN, V., AND GRIFFITH, V. Casper the friendly finality gadget. arXiv preprint arXiv:1710.09437 (2017).
- [10] Wood G. Ethereum: A secure decentralised generalised transaction ledger[J]. Ethereum project yellow paper, 2014, 151: 1-32.
- [11] Narayanan A, Bonneau J, Felten E, et al. Bitcoin and Cryptocurrency Technologies: A Comprehensive Introduction[M]. Princeton University Press, 2016.
- [12] Delmolino K, Arnett M, Kosba A, et al. Step by step towards creating a safe smart contract: Lessons and insights from a cryptocurrency lab[C]//International Conference on Financial Cryptography and Data Security. Springer, Berlin, Heidelberg, 2016: 79-94.
- [13] Vigna P, Casey M J. The age of cryptocurrency: how bitcoin and the blockchain are challenging the global economic order[M]. Macmillan, 2016.
- [14] GILAD, Y., HEMO, R., MICALI, S., VLACHOS, G., AND ZELDOVICH, N. Algorand: Scaling byzantine agreements for cryptocurrencies. In Proceedings of the 26th Symposium on

Operating Systems Principles (2017), ACM, pp. 51–68.

[15] EYAL, I., AND SIRER, E. G. Majority is not enough: Bitcoin mining is vulnerable. In International conference on financial cryptography and data security (2014), Springer, pp. 436–454.

[16] KOGIAS, E. K., JOVANOVIC, P., GAILLY, N., KHOFFI, I., GASSER, L., AND FORD, B. Enhancing bitcoin security and performance with strong consistency via collective signing. In 25th USENIX Security SymPOSium (USENIX Security 16) (2016), pp. 279–296.

[17] IBM Blockchain for supply chain.

<https://www.ibm.com/blockchain/supply-chain>.