Aptos Blockchain：安全，可扩展和可升级的Web3基础设施

2022年8月11日
V1.0

翻译: deepl+kk德米安

摘要

作为一种新的Internet基础设施，块链的兴起导致开发人员以快速增长的速度部署了成千上万的分散式应用程序。 遗憾的是，由于频繁的中断，高昂的成本，低吞吐量限制和众多的安全问题，块链的使用还不普遍。 为了在web3时代实现大规模采用，块链基础设施需要遵循云基础设施的路径，作为一个值得信赖，可扩展，经济高效且不断改进的平台，用于构建广泛使用的应用程序。

我们推出Aptos块链，其设计以可扩展性，安全性，可靠性和可升级性为主要原则，以应对这些挑战。 Aptos块链是由全球350多名开发人员在过去三年中开发的[1]。 它在协商一致，智能合同设计，系统安全，性能和权力下放方面提供了新的创新。 这些技术的结合将为将web3带给大众提供一个基本的构建块：[1.]

- 首先，Aptos块链在本地集成并在内部使用移动语言，以实现快速，安全的事务执行[2]。 Move Prover是一种正式的验证工具，用于验证以Move语言编写的SMART合约，它为合约不变和行为提供了额外的保障。 这种对安全性的关注使开发人员能够更好地保护他们的软件免受恶意实体的攻击。
- 其次，Aptos数据模型实现了灵活的密钥管理和混合保管选项。 这与签署前的交易透明性和实用的轻量客户端协议一起，提供了更安全，更可信的用户体验。
- 第三，为了实现高吞吐量和低延迟，Aptos块链采用了流水线和模块化方法来处理事务的关键阶段。具体而言，事务分发，块元数据排序，并行事务执行，批处理存储， 和分类账认证都同时运行。此方法充分利用所有可用的物理资源，提高硬件效率并实现高度并行执行。
- 第四，与其他并行执行引擎通过要求读取和写入数据的前期知识来打破事务原子性不同，Aptos块链不会对开发人员施加此类限制。 它可以通过任意复杂的事务有效地支持原子性，从而为实际应用程序实现更高的吞吐量和更低的延迟，并简化开发。
- 第五，Aptos模块化架构设计支持客户端灵活性，并针对频繁和即时升级进行优化。 此外，为了快速部署新技术创新并支持新的web3使用案例，Aptos blockchain提供嵌入式的链式变更管理协议。

[1]法律免责声明：本白皮书及其内容不是出售或索取购买任何代币的要约。 我们发布本白皮书的唯一目的是接收公众的反馈和评论。本文档中的任何内容均不应被理解或解释为对Aptos封锁链或其令牌(如果有)将如何发展，使用或累积价值的保证或承诺。 Aptos仅概述其当前计划，这些计划可能会自行更改，其成功与否取决于其无法控制的许多因素。 此类未来陈述必然涉及已知和未知风险，这可能导致未来期间的实际绩效和结果与本白皮书中所述或暗示的情况有重大差异。 Aptos公司不承担更新其计划的义务。 由于实际结果和未来事件可能存在重大差异，因此无法保证白皮书中的任何陈述都是准确的。 请勿过度依赖未来的陈述。

- 最后，Aptos块链正在尝试未来的方案，以扩展到单个验证器性能之外：其模块化设计和并行执行引擎支持验证器的内部分片，并且同构状态分片提供了水平吞吐量可扩展性的潜力，而不会增加节点运算符的复杂性。

1 简介

在Web2版本的Internet中，信息传递，社交媒体，金融，游戏，购物， 和音频/视频流由控制直接访问用户数据的集中式公司(例如Google，Amazon，Apple和Meta)提供。 这些公司使用针对目标用例优化的应用程序特定软件开发基础设施，并利用云基础设施将这些应用程序部署到用户。 云基础设施提供对虚拟化和/或物理基础设施服务的访问，例如租用的虚拟机(VM)和在全球数据中心内运行的裸机硬件(例如AWS，Azure和Google Cloud)。 因此，构建可扩展到数十亿用户的Web2 Internet服务从未像现在这样简单。 但是，Web2要求用户明确信任集中式实体，这一要求已日益受到社会的关注。

为了消除这一关切，新的互联网时代已经开始：web3. 在网络3版的互联网中，出现了锁链，提供分散的，不可改变的分类账，使用户能够安全可靠地相互互动，所有这一切都不需要对控制中间人或中央实体的信任。 与Web2 Internet服务和应用程序依赖云基础设施作为构建块的方式类似，分散式应用程序可以使用块链作为分散式基础架构层，以覆盖全球数十亿用户。

然而，尽管目前存在许多封锁，但尚未广泛采用web3 [3]。 虽然技术在继续推动行业发展，但现有的封锁链不可靠，对用户收取高额交易费用，吞吐量限制较低，由于安全问题经常遭受资产损失，并且无法支持实时响应。 与云基础设施如何使Web2服务达到数十亿相比，块链还没有使web3应用程序实现同样的功能。

2 Aptos愿景

Aptos的愿景是提供一个能将主流应用引入web3的块链，并使分散应用程序生态系统能够解决实际用户问题。我们的使命是通过提供灵活的模块化模块化块链体系结构，提高块链可靠性，安全性和性能方面的一流水平。 此架构应支持频繁升级，快速采用最新技术进步以及对新的和新兴使用案例的一流支持。

我们设想一个由使用它的社区管理和运营的分散，安全和可扩展的网络。 当全球范围内的基础设施需求增长时，块链的计算资源会横向和纵向扩展以满足这些需求。 随着新的使用案例和技术进步的出现，网络应经常无缝升级，而不会中断用户。 基础设施问题应逐渐消失。 开发人员和用户将可以访问许多不同的选项，包括密钥恢复，数据建模，智能合同标准，资源使用权衡，隐私， 和可组合性。 用户知道他们的资产是安全的，随时可用的，并且可以以几乎低廉的费用进行访问。 任何人都可以安全，轻松和不可改变地与世界各地的不受信任方进行交易。 Blockchains和云基础设施一样无处不在。

为了实现这一愿景，必须取得重大的技术进步。 我们在过去三年中构建，开发，推进和部署Diem块链(Aptos块链的前身)的经验证明，网络可以在不中断客户端的情况下不断升级其协议[4]。 在2020年初，Diem Mainnet部署到了十多个节点运营商以及多个钱包提供商。 在接下来的一年中，我们的团队发布了两项重大升级，更改了共识议定书和核心框架。 两个升级都已完成，用户无需停机。 在Aptos BlockChain的启发下，我们对技术堆栈进行了一系列彻底的改进，同时还将安全，透明和频繁的升级作为核心功能进行了整合。 我们特别强调交易处理的新方法(如第7节所述)以及权力下放和网络治理的新方法。

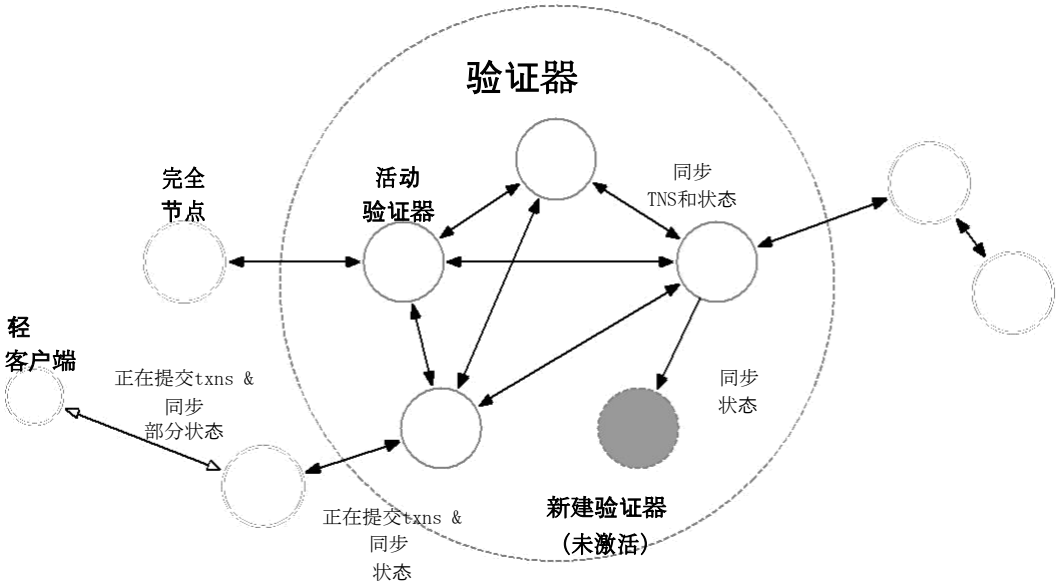

图1：Aptos生态系统的组成部分。

随着Aptos块链的不断改进和发展，我们将发布此白皮书的更新版本，其中包含我们的协议和设计选择的最新迭代。 在本文档的其余部分中，我们将介绍Aptos封锁链的当前状态以及未来的计划。

3 概述

Aptos块链(如图1所示)由一组验证器组成，这些验证器使用拜占庭容错(BFT)(利益相关方验证共识机制)联合接收和处理用户的事务。 令牌持有人在其选定的验证器中锁定或标记令牌。 每个验证者的协商一致投票权重与所占的数量成比例。 验证者可以处于活动状态并参与协商一致。 同样，如果验证器没有足够的利害关系来参与，从验证器集转出，在同步封锁链状态时选择脱机，或者由于历史性能不佳而被视为不参与协商一致协议，则验证器也可能处于非活动状态。

客户端是系统中需要提交交易或查询封锁链的状态和历史的任何部分。 客户端可以选择下载并验证所查询数据的验证器签名的证明。 完整节点是从验证器或网络中的其他完整节点复制事务和封锁链状态的客户端。 他们可以根据需要选择修剪事务历史记录和封锁链状态以回收存储。 Light客户端仅维护当前的验证器集，并且可以安全地查询部分块链状态，通常是从完整节点查询。 钱包是轻量客户端的常见示例。

为了满足安全，快速，可靠和可升级的web3基础设施的需求，以实现广泛采用，Aptos块链基于以下核心设计原则构建：

- 通过新的智能合约编程语言，快速，安全地执行，以及简单的可审计性和机械分析性，移动[5]。Move起源于Aptos块链的前身，并随着该项目的发展而继续推进。
- 通过批量，流水线和并行的事务处理方法实现极高的吞吐量和低延迟。
- 全新的并行事务处理，可通过Block-STM高效地支持任意复杂事务的原子性，与现有的并行执行引擎不同，它需要读取和写入数据位置的前期知识。
- 通过快速的权重验证器设置轮换和声誉跟踪优化性能和分散管理。

- 可升级性和可配置性作为一流的设计原则，以支持新的使用案例和最新技术。
- 模块化设计，支持严格的组件级测试，以及适当的威胁建模和无缝部署，所有这些都确保了高度安全和可靠的操作。
- 水平吞吐量可扩展性，同时保持分散，其中分片是向用户公开的一流概念，并且是编程和数据模型的固有概念。

第4节 介绍开发人员如何在Aptos块链中与Move交互。 第5节 介绍逻辑数据模型。 第6节 详细介绍Aptos封锁链如何通过强大的验证方法实现安全的用户体验。 第7节 介绍了有关流水线，批处理和并行化的关键性能创新。 第8节 详细介绍了不同客户端与其他节点同步状态的各种选项。 第9节 介绍了我们的社区所有权和治理计划。 最后，第10节 讨论了在保持权力下放的同时今后的业绩方向。

4 移动语言

Move是一种新的智能合约编程语言，强调安全性和灵活性。 Aptos块链使用MOVE的对象模型来表示其分类帐状态(参见第5.5节)，并使用MOVE代码(模块)来编码状态转换的规则。 用户提交的事务可以发布新模块，升级现有模块，执行模块内定义的输入功能，或者包含可以直接与模块的公共接口交互的脚本。

移动生态系统包含编译器，虚拟机和许多其他开发工具。 移动受Rust编程语言的启发，该语言通过线性类型等概念在语言中明确数据所有权。 迁移强调资源稀缺性，保护和访问控制。 移动模块定义每个资源的生命周期，存储和访问模式。 这可确保Coin等资源不会在没有适当凭证的情况下生成，不会重复花费，也不会消失。

Move利用字节码验证器来确保类型和内存安全，即使存在不受信任的代码。 为了帮助编写更可信的代码，Move包括一个正式的验证器，即Move Prover [6]，能够根据给定的规范验证Move程序的功能正确性，该规范使用与Move集成的规范语言来表示。

除了用户帐户和相应的帐户内容之外，分类帐状态还包含Aptos块链的链式配置。 此网络配置包括一组活动验证器，staking属性以及Aptos块链中各种服务的配置。 Move对模块升级能力和全面的可编程性的支持可实现无缝配置更改，并支持升级Aptos块链本身(这两组升级已多次执行，在专用主网中实现零停机)。

Aptos团队通过支持更广泛的web3使用案例，进一步增强了移动。 如第5.5节后面所述，Aptos块链实现了精细的资源控制。 这不仅支持执行的并行化，而且还实现了与访问和改变数据相关的几乎固定的成本。 此外，Aptos块链还提供基于精细存储的表支持，允许在单个帐户中进行大规模数据集(例如，大量的NFT集合)。 此外，Aptos支持完全在链上表示的共享或自主帐户。 这使复杂的分散式自治组织(DAOS)可以协作共享帐户，并将这些帐户用作混合资源集合的容器。

5 逻辑数据模型

Aptos封锁链的分类帐状态代表所有帐户的状态。 分类帐状态使用与系统已执行的事务数对应的无符号64位整数进行版本控制。 任何人都可以将事务提交到Aptos blockchain以修改分类帐状态。执行事务时，将生成事务输出。 事务输出包含零个或多个操作，用于处理分类帐状态(称为写入集)，结果事件的矢量(参见第5.1.1节)，消耗的气体量和执行的事务状态。

5.1 交易

已签名的事务包含以下信息：

- 事务验证器：发送方使用包含一个或多个数字签名的事务验证器来验证事务是否经过验证。
- 发件人地址：发件人的帐户地址。
- 有效负载：有效负载指的是链上现有的输入函数，或者包含要作为内联字节码(称为脚本)执行的函数。 此外，一组输入参数以字节数组编码。 对于点对点交易，输入内容包含收件人的信息和转移给他们的金额。
- 煤气价格(以指定货币/煤气单位计)：这是发送方愿意为执行交易支付的每单位煤气的金额。 GAS是支付计算，网络和存储费用的一种方式。 气体单位是计算的抽象测量，没有固有的实际值。
- Maximum gas amount（最大气体量）：最大气体量是允许事务在中止前消耗的最大气体单位。账户必须至少有汽油价格乘以最大汽油量，否则交易将在验证过程中被丢弃。
- 序列号：事务的序列号。 这必须与执行事务时发件人帐户中存储的序列号匹配。 成功执行事务后，帐户序列号将递增，以防止重放攻击。
- 过期时间：事务失效之前的时间戳。
- 链ID：标识此事务有效的封锁链，为用户提供进一步保护以防止签名错误。

在每个版本I中，状态更改由元组(TI，Oi，Si)表示，包含事务，事务输出， 和生成的分类帐状态。给定确定性函数应用，分类帐状态$Si-_1$的事务TI的执行将产生事务输出Oi和新分类帐状态Si。 即，应用($Si-_1$，TI)→Oi，Si。

5.1.1 活动

事件在执行事务期间发出。 每个移动模块都可以定义自己的事件，并选择在执行时何时发出这些事件。 例如，在投币转账期间，发件人和收件人的帐户将分别发出SentEvent和ReceivedEvent。 此数据存储在分类帐中，可通过Aptos节点查询。 每个注册的事件都有一个唯一的键，该键可用于查询事件详细信息。

向同一事件键发出的多个事件将生成事件流，每个条目包含从0开始的顺序增加的数字，类型和数据的事件列表。 每个事件都必须按某种类型定义。 可能存在由相同或类似类型定义的多个事件，特别是在使用泛型时。 事件具有相关数据。 对于移动模块开发人员，一般原则是包括所有必要的数据，以便在执行更改数据并发出事件的事务之前和之后了解对基础资源的更改。

事务只能生成事件，不能读取事件。 此设计允许事务执行仅是当前状态和事务输入的功能，而不是历史信息(例如，以前生成的事件)。

5.2 帐户

每个帐户都由唯一的256位值(称为帐户地址)标识。 当从现有帐户发送的事务调用CREATE_ACCOUNT(addr) MOVE函数时，将在分类帐状态下创建一个新帐户(参见第5.5节)。 当交易尝试将Aptos令牌发送到尚未创建的帐户地址时，通常会发生这种情况。 为方便起见，Aptos还支持转移(从，到，金额)功能，如果在转移之前帐户不存在，则会隐式创建帐户。

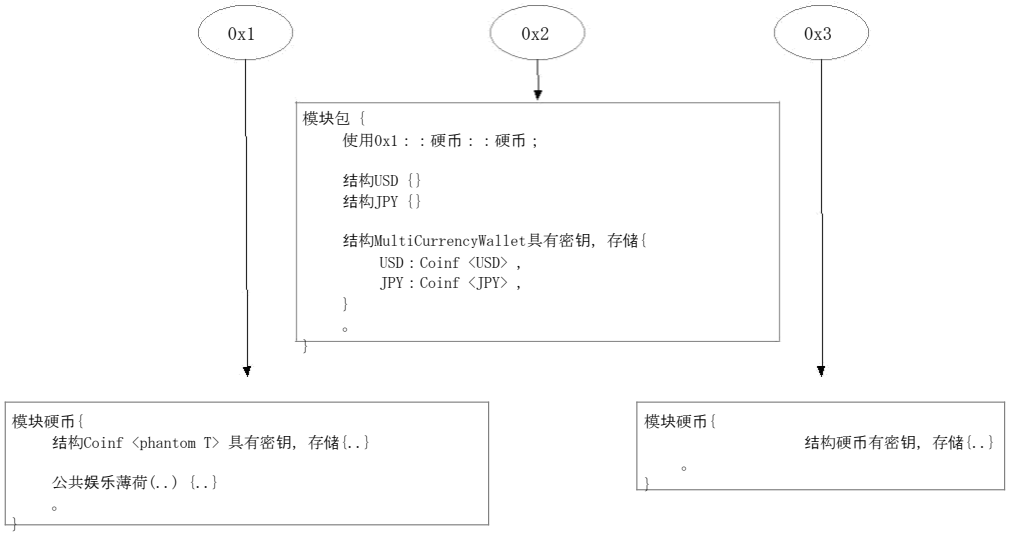

图2：链式移动模块示例。

要创建新帐户，用户首先生成签名密钥对：(VK，SK)。 接下来，给定签名方案的新帐户地址是使用公共验证密钥VK的加密哈希H派生的，该密钥与签名方案标识符(SSID)连接：其中addr = H(VK，SSID)。

在地址addr创建新帐户后，用户可以使用专用签名密钥SK对要从地址addr的帐户发送的事务进行签名。用户还可以轮换SK，以主动更换SK或响应可能的妥协。 这不会更改帐户地址，因为帐户地址在创建过程中仅从公共验证密钥派生一次。

Aptos封锁链不会将帐户与真实身份关联。 用户可以通过生成多个密钥对来创建多个帐户。 由同一用户控制的帐户彼此之间没有固有的链接。 但是，单个用户仍然可以在一个钱包中管理多个帐户，以便进行简单的资产管理。 这种灵活性为用户提供了假名，同时我们还在未来版本中试用了保护隐私的原语。 单个用户或一组用户拥有的多个帐户也提供了增加执行并发性的渠道，如第7.4节所述。

5.3 移动模块

移动模块包含声明数据类型(结构)和过程的移动字节码。 它由声明模块的帐户地址和模块名称一起标识。例如，图2中第一个货币模块的标识 符为0x1::cin。 一个模块可以依赖于其他链上模块，如图2中的钱包模块所示，从而实现代码重用。

一个模块必须在一个帐户中唯一命名，即每个帐户最多可以使用任何给定名称声明一个模块。例如，图2中地址0x1处的帐户 无法声明另一个名为COIN的模块。 另一方面，地址0x3处的帐户可以声明一个名为cin的模块，该模块的标识符为0x3::cin。 请注意，0x1 : : 硬币 : : 硬币和0x3 : : 硬币 : 硬币是不同的类型，不能互换使用，也不能共享通用模块代码。 相反，0x1 : : 硬币 : : : 硬币<0x2 : : 钱包 : : : USD>和0x1 : 硬币 : : 硬币<0x2 : 钱包 : : : JPy>是同一通用类型的不同实例，不能互换使用，但可以共享通用模块代码。

模块被分组到位于同一地址的包中。 此地址的所有者将软件包作为整个链发布，包括字节码和软件包元数据。 软件包元数据决定软件包是可以升级还是不可变。 对于可升级的软件包，在允许升级之前会执行兼容性检查：不能更改现有的入口点功能，也不能将任何资源存储在内存中。 但是，可以添加新的功能和资源。

Aptos框架由Aptos块链的核心库和配置组成，定义为常规的可升级模块包(参见第9.2节)。

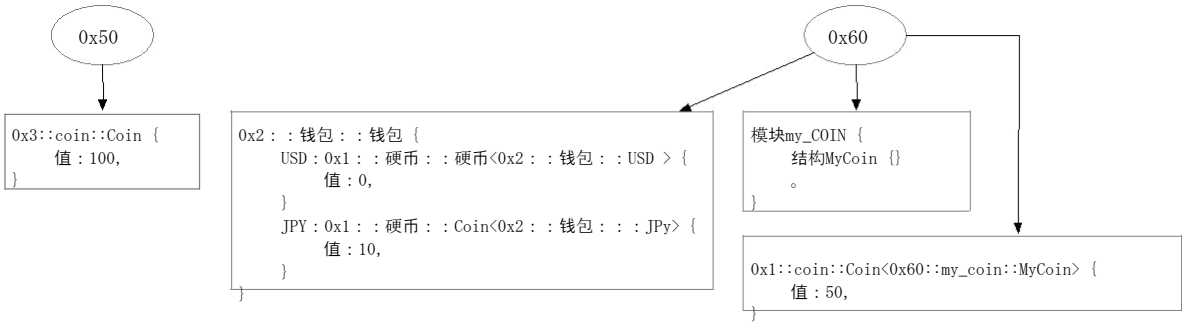

图3：链数据示例。

5.4 资源

与模块类似，帐户地址也可以具有与其关联的数据值。 在每个帐户地址中，这些值按其类型键入，每个类型中最多有一个属于帐户的值。 图3 提供了一个示例。 地址0x50包含单个值，0x3::c币::Coin为完全限定类型。 0x3是存储币形模块的地址，COIN是模块的名称，而Coin是数据类型的名称。也允许使用通用类型的值，将不同的实例视为不同的类型。 这对于可扩展性非常重要，允许不同的实例化共享相同的功能代码。

用于更改，删除和发布值的规则在定义数据类型的模块中进行编码。 Move的安全和验证规则可防止其他代码或实体直接创建，修改或删除其他模块中定义的数据类型的实例。

在一个地址下最多有一个每种类型的顶级值，最初可能会限制声音。 但是，在实践中这不是问题，因为程序员可以将其他数据定义为内部字段的包装类型，从而避免任何限制。 图3中的电子钱包结构 是如何使用包装类型的示例。

还应注意的是，并非所有数据类型都可以在链上存储。 要使数据实例符合顶级值的条件，数据类型必须具有键功能。 同样，嵌套值需要存储性能。 具有两种功能的数据类型也称为资源。

5.5 分类帐状态

从移动虚拟机(移动VM)的角度来看，每个帐户都由一组值和键值数据结构组成。 这些数据结构称为表条目，以二进制规范序列化格式(BCS)存储。 这种数据布局使开发人员能够编写智能合同，从而可以有效地处理在大量帐户中复制的少量数据，以及在少量帐户中存储的大量数据。 移动模块的存储方式与帐户数据类似，但位于独立的名称空间下。 起源分类帐状态定义了初始帐户集及其在块链初始化时的关联状态。

在发布时，Aptos封锁链将由单个分类帐状态表示。 但是，随着采用率的增加和技术的发展，Aptos将增加碎片数量以提高吞吐量(即，启用多个分类帐状态)，并支持跨碎片移动或访问资产的交易。 每个分类帐状态都将为特定分页维护所有的链上资产，并提供相同的帐户模型和细化的关键值数据存储，提供接近固定的存储访问成本。

6 安全的用户体验

要接触数十亿互联网用户，web3用户体验必须安全且可访问。 在下面的章节中，我们将介绍Aptos块链为实现这一目标而提供的几项创新。

6.1 交易生存能力保护

签署交易意味着签署人授权由封锁链进行和执行交易。 有时，用户可能无意中签署交易，或者没有充分考虑他们的交易可能被操纵的所有方式。 为了降低这种风险，Aptos封锁链限制了每笔交易的可行性，并保护签署人免受无限有效性的影响。 Aptos封锁链目前提供三种不同的保护-发件人的序列号，事务到期时间和指定的链标识符。

- 每个发件人的帐户只能提交事务的序列号一次。 因此，发件人可以观察到，如果当前帐户序列号是≥事务t的序列号，则t已提交或t将永远不会提交(因为t使用的序列号已被另一事务使用)。
- 块链时间以高精度和高频率(通常为次秒)前进，详见第7.3.1节。 如果封锁链时间超过事务t的到期时间，则同样，t已提交或将永远不会提交。
- 每个事务都有一个指定的链标识符，以防止恶意实体在不同的阻止链环境(例如，跨测试网和主网)之间重放事务。

6.2 基于移动的密钥管理

如第5.2节中所述，Aptos帐户支持密钥轮换，这是一项重要功能，可帮助降低与私钥泄露，远程攻击以及可能破坏现有加密算法的未来改进相关的风险。 此外，Aptos账户也足够灵活，可以实现新的混合监管模式。 在一种此类模型中，用户可以将帐户私钥的轮转权委托给一个或多个保管人和其他受信任实体。 然后，移动模块可以定义一个策略，使这些受信任实体能够在特定情况下轮转密钥。例如，实体可能由许多受信任方持有的k-out－of－n multi-SIG密钥表示，并提供密钥恢复服务以防止用户密钥丢失(例如，20%的比特币目前锁定在不可恢复的帐户[7]中)。

此外，虽然许多钱包支持各种密钥恢复方案，例如备份私有密钥到云基础设施，多方计算和社交恢复，但通常在实施时不需要封锁链支持(即，脱链支持)。 因此，每个钱包都需要实施自己的密钥管理基础结构，相关操作对用户变得不透明。 相比之下，在Aptos块链层支持密钥管理功能，可以完全透明地处理所有与密钥相关的操作，并使实施具有丰富密钥管理功能的钱包变得更加简单。

6.3 签署前交易透明度

如今，钱包对其签署的交易几乎没有透明度。 因此，用户通常很容易被骗签署恶意交易，这些交易可能会窃取资金并产生破坏性后果。 即使是要求枚举每个事务访问的所有链上数据的块链也是如此。因此，目前很少有用户防护措施，使用户容易受到各种攻击。

为了解决这一问题，Aptos生态系统为交易执行前提供服务：这是一项预防措施，在签署前向用户(以人类可读的形式)说明其交易的结果。 将其与已知的以前攻击历史记录和恶意智能合同配对将有助于减少欺诈。 此外，Aptos还使钱包能够在执行过程中对事务施加限制。 违反这些限制将导致事务被中止，以进一步保护用户免受恶意应用程序或社会工程攻击。

6.4 实用的轻量客户端协议

仅依靠API提供程序的TLS/SSL证书来建立块链客户端和服务器之间的信任，并不能充分保护客户端。 即使存在有效的证书，钱包和客户端也无法保证所呈现数据的真实性和完整性

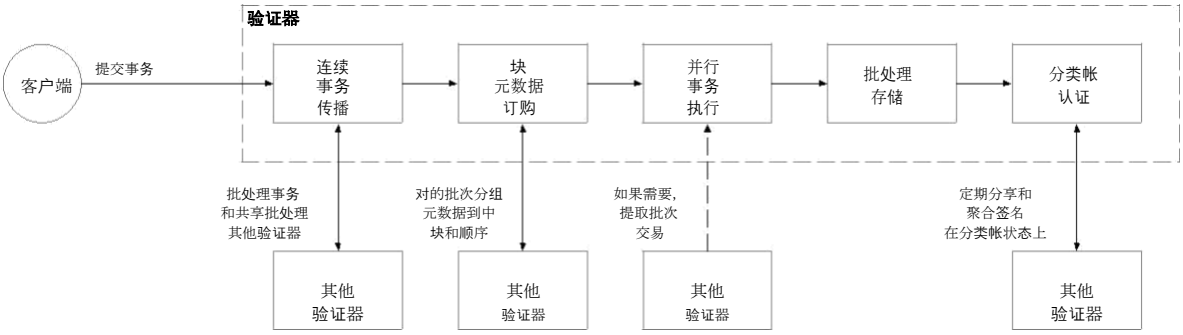

图4：事务处理生命周期。 所有阶段都是完全独立的，并且可以单独并行。

。 因此，API提供商可能会返回不正确或恶意的封锁链数据，欺骗第三方并执行双重支出攻击。

为了防止这种情况，Aptos提供了状态证明和轻量客户端验证协议，可供钱包和客户端用于验证不受信任的第三方服务器提供的数据的有效性。 此外，通过利用第7.5.2节中基于时间戳的状态证明，轻量级客户端可以始终确保严格限制帐户状态的新鲜度(例如，在几秒钟内)，并且只需要跟踪网络配置的更改(纪元更改)或使用当前受信任的检查点(航点)来保持最新[8]。 Aptos封锁链将高频时间戳和廉价的状态验证相结合，为客户端提供了更高的安全保障。

此外，Aptos节点还提供丰富的高性能存储接口，可进一步微调，以允许订阅针对链上特定数据和帐户的校样。 轻量客户端可以利用此功能来保留最少的可验证数据，而无需运行完整节点或处理大量事务。

7 流水线，批处理和并行事务处理

为了最大限度地提高吞吐量，增加并发性并降低工程复杂性，Aptos块链上的事务处理被划分为不同的阶段。 每个阶段都是完全独立的，可单独并行的，类似于现代的超标量处理器体系结构。 这不仅提供了显著的性能优势，而且使Aptos块链能够提供新的验证器-客户端交互模式。 例如：

- 当特定事务已包括在一批持久事务中时，可以通知客户。 持续有效的交易很可能会在紧急情况下进行。
- 当订购了一批持久的事务时，可以通知客户。 因此，为了减少确定执行的事务输出的延迟，客户端可以选择在本地执行事务，而不是等待验证器远程完成执行。
- 客户机可以选择等待验证程序执行认证的事务，并对认证的结果执行状态同步(例如，请参见第8节)。

Aptos模块化设计有助于加快开发速度并支持更快的发布周期，因为更改可以针对单个模块，而不是单个单片体系结构。 同样，模块化设计还提供了一种结构化的路径，使验证器能够扩展到单台计算机之外，从而提供对其他计算，网络和存储资源的访问。 图4 显示了不同处理阶段的事务生命周期。

7.1 批处理

批处理是一项重要的效率优化，是Aptos块链中每个操作阶段的一部分。 在事务分发期间，每个验证者将事务分组为批次，在协商一致期间，批次合并为块。 执行，存储和

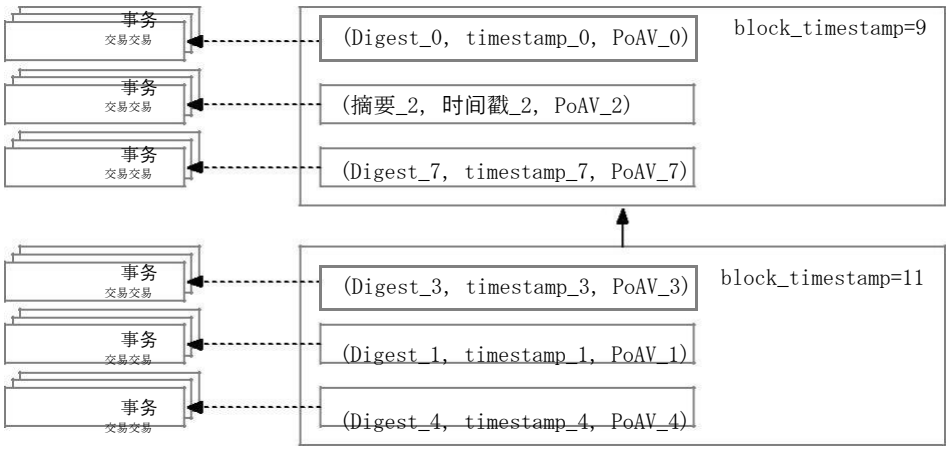

图5：块元数据排序独立于事务分发。

分类账认证阶段也会分批工作，以便提供重新排序，减少操作(例如重复计算或签名验证)以及并行执行的机会。

将事务分组为批可能会导致少量延迟，例如，在执行分发前等待200毫秒以积累一批事务。 但是，批处理可根据最大等待时间和最大批处理大小轻松配置，从而使分散式网络能够自动优化延迟和效率。批处理还允许高效的收费市场优先处理交易，避免来自过于狂热的客户的意外拒绝服务(DoS)攻击。

7.2 持续的交易传播

根据Narwhal & Tusk的主要见解[9]，Aptos块链中的交易传播与共识脱钩。 验证程序会不断地将大量事务传送到彼此之间，同时利用所有可用的网络资源。 验证器v分发的每个批次都将保持不变，批次摘要上的签名将发送回v。根据第7.3节中定义的协商一致要求，批次摘要上的任何2f + 1个权重签名构成可用性证明(PoAv)。 这样的证明可以保证至少有f + 1个权重的诚实验证器已存储了该批，因此所有诚实验证器都能够在执行前对其进行检索。

无限持续的大量事务可能会导致验证器存储空间耗尽并崩溃，从而打开DoS攻击媒介。 为了防止这种情况发生，每批事务都有一个关联的时间戳。 批处理上的时间戳可以在每个验证器上有效地收集垃圾。 此外，还设计了单独的每个验证器配额机制，以保护验证器即使在最极端的情况下(例如在潜在的拜占庭攻击下)也不会用完空间。 批次还具有规模限制，在同意保持稳定存储之前已验证这些限制。 最后，对重复数据删除和缓存事务进行的几项优化可降低存储成本，并确保与并行执行引擎的性能集成。

7.3 块元数据排序

一个常见的误解是共识速度缓慢，因此是阻止链吞吐量和延迟的主要瓶颈。 Aptos封锁链的一个关键创新是将非协议相关任务从协商一致阶段中分离出来，例如交易发布，交易执行/存储和分类账认证。通过将事务传播与协商一致阶段分离，可以在极低的带宽下进行订购(仅限块元数据和校样)，从而实现高事务吞吐量和最小化延迟。

如今，Aptos封锁链利用了DiemBFTv4 [10]的最新迭代，这是一种反应最优化的BFT共识协议。 在常见情况下，共识只需要两次网络往返(全球往返时间通常少于300毫秒)，并通过领先的声誉机制根据故障验证器进行动态调整[11]。 连锁领导者的声誉

机制将提升已在窗口中成功提交块的验证器，并降级未参与的验证器。 这种新颖的机制显著提高了分散式环境中的性能，相应地提供了基础设施以提供适当的激励措施，并快速将失败的验证器对吞吐量和延迟的影响降至最低。

DiemBFTv4可在部分同步时保证活跃性，并确保异步时的安全性，其中验证器总电桩为≥3f + 1，具有最多f个电桩加权故障验证器。 自2019年以来，DiemBFTv4已在多个迭代中通过数十个节点运算符和多钱包生态系统进行了广泛测试。 我们还在试验我们最近的研究(例如Bullshark [12])和其他协议，这些协议依赖块历史记录和相关通信来确定块元数据排序和最终性。

如图5所示，由领导者提议协商一致块和建议书时间戳，并由其他验证者同意。 请注意，每个协商一致块仅包含批元数据和打样。 在块中不需要实际交易，因为PoAV确保在订购后的执行阶段可以提供各批交易(参见第7.2节)。 验证员可以在验证证明并满足块元数据标准(例如，建议时间戳≤块到期时间)后，对领导者的建议进行投票。

7.3.1 Blockchain时间

Aptos块链对每个建议的块采用一个近似的，商定的物理时间戳，并相应地对该块内的所有事务采用该时间戳。 此时间戳可用于许多重要的用例。 例如：

- 智能合同中与时间相关的逻辑。 举例来说，发展商希望在星期四中午之前，把所有竞投的投标书都编入电子编码。
- 随着oracles发布链上数据，需要准确且可信的链上时间戳来关联事件并处理来自实际数据的延迟。
- 客户可以辨别他们对封锁链的最新程度。 出于安全原因，为了避免过时数据和远程攻击，客户端应能够访问帐户状态更新时的高精度时间戳。
- 使用值得信赖的时间戳审核封锁链可提供与非连锁事件的强大关联，例如确保合法实施的支付符合预期要求。
- 事务到期时间基于最近的提交时间戳。 作为客户交易的附加保障措施，客户可以选择交易的到期时间，如第6.1节所述。

Aptos封锁链就一个封锁范围内所有交易的时间戳提供以下保证：

- 时间在块链中单一增加。 也就是说，如果块B1 <块B2，则B1.T ime < B2.T ime。
- 如果一个事务块与timestamp T一致，则至少f +1个诚实的验证者已确定T是过去的。 一个诚实的验证者只会在块本身的时钟≥timestamp T时投票。 请参阅第7.2节。
- 如果一个事务块具有协商一致的签名仲裁和时间戳T，则诚实的验证器在其自己的时钟≥时间戳T之前不会向其他验证器提供此类块。

每个已提交块上的最新时间戳都会更新，并用作该块中所有事务的时间戳。 当网络同步时，每个网络往返都会提交一个事务块，并提供快速更新和高度可靠的时钟。 如果需要，可以确定事务块内更精细的排序。

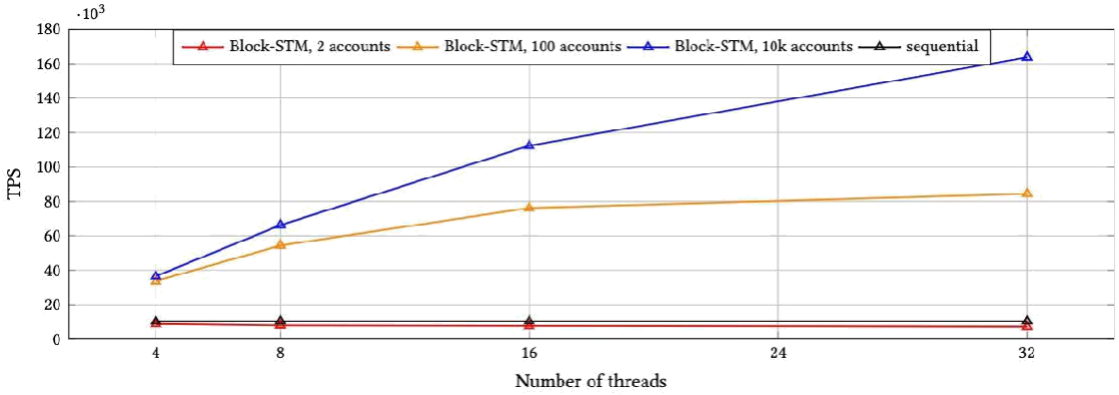

图6：数据块STM（仅限组件）基准比较具有不同竞争级别的物理内核数量。

7.4 并行事务执行

一旦对协商一致块元数据进行排序，事务就可以由任何验证程序，完整节点或客户端执行。 至少2f + 1个股权加权验证者已有效地保留了建议批次的交易。 由于交易发布是持续的，随着时间的推移，其他诚实的验证者将收到交易批次。 如果诚实的验证者在到达执行阶段时尚未收到所订购批次的事务，则可以从2f +1权重加权验证者处下载这些事务，前提是至少有f + 1权重验证者（≥一半的加权PoAV签署人）诚实。

任何阻止链的一个重要目标是尽可能多地实现并行执行。 Aptos块链从数据模型和执行引擎两个方面向前推进了这一方向。

7.4.1 并行数据模型

移动数据模型本身支持数据和模块的全局寻址。 数据和帐户中没有重叠冲突的事务可以并行执行。考虑到Aptos块链使用的流水线设计，对一组事务重新排序可以减少冲突数量，从而改善并发性。

即使事务修改了同一组链上值，大部分事务执行过程仍可并行化。 Aptos块链引入了一个新概念，即增量写入，它描述了对帐户状态的修改，而不是修改的帐户状态(例如，增加一个整数，而不是简单地确定最终值)。 所有事务处理都可以并行完成，然后按正确的顺序对冲突值应用增量写入，以确保确定性结果。

随着时间的推移，Aptos块链将继续以改善并发性(例如利用读/写提示)的方式增强数据模型，同时改进人机工程学，使开发人员更自然地创建，修改和编写链上值。 Move提供了在语言级别以及通过平台特定功能进行这些改进的灵活性。

7.4.2 并行执行引擎

Block-STM并行执行引擎检测和管理有序事务集的冲突，并进行乐观的并发控制，以便在给定特定顺序[13]的情况下实现最大的并行性。

同时优化执行各批事务，并在执行后验证。 验证失败会导致重新执行。 BLOCK－STM使用多版本数据结构来避免写入冲突。 所有对同一位置的写入操作都与其版本一起存储，版本中包含其事务ID和优化重新执行写入事务的次数。 当事务TX读取内存位置时，它从多版本数据结构中获取通过按预设顺序显示在TX之前的最高事务写入此位置的值以及关联的版本。

BLOCK－STM已集成到Aptos模块链中。 为了了解数据块- STM性能的全部潜力，我们对非普通的对等移动事务(即每个事务8次读取和5次写入)进行了实验，作为一个独立的，仅执行(而非端到端)基准，并使用内存数据库。 在图6中，我们介绍了我们的Block-STM执行结果。 每个数据块包含10k事务，而帐户数量决定了冲突和争用的级别。

在低争用情况下，Block-STM通过32个线程实现了16倍的连续执行加速，而在高争用情况下，Block-STM实现了8倍以上的加速。 Block-STM是块链空间中其他并行执行引擎的独特之处，它能够动态而透明地(无需用户的任何提示)从任何工作负载中提取固有的并行性。 与需要读取或写入数据位置的前期知识的并行执行环境相比，Block-STM可以同时支持更复杂的事务。 此属性可减少事务并提高其效率，降低成本，并为用户提供更低的延迟。 也许最重要的是，将原子事务拆分为多个较小的事务会打破具有复杂状态结果的单个事务的全语或全语义。 将表现式事务语义与Block-STM中的并行执行相结合，使开发人员能够充分利用这两个领域。

请注意，块元数据排序步骤不排除在并行执行阶段对事务重新排序。 可以跨一个或多个块对事务重新排序，以优化并行执行的并发性。 唯一的要求是重新排序必须在所有诚实的验证器中具有确定性。 针对并行执行进行优化以及在重新排序中添加随机化可提高性能，并可能抑制用于盈利验证器事务重新排序的最大可提取值(MeV)技术。 此流水线设计还可以采用“先下单后揭晓”的MeV耐受策略。

块- STM和事务重新排序是增加执行并行性的补充技术。 它们可以与事务读/写访问提示结合使用，以获得额外的并发性。

7.5 批存储

并行执行阶段会为组中的所有事务生成写入集。 这些写入集可以存储在内存中以获得最大执行速度，然后用作要执行的下一个块或块集的缓存。 任何重叠写入只需写入稳定存储一次。 如果验证器在存储内存中的写入集之前出现故障，它只需从块元数据排序阶段恢复并行执行。 将写入集的批存储与并行执行步骤分离，确保并行执行可以高效运行。 总之，批处理写入集可减少存储操作的数量，并利用更高效，更大的I/O操作。

为写入集高速缓存保留的内存量可按计算机手动配置，并提供自然的背压机制。 如果需要针对特定I/O和内存环境进行调整，则批处理的粒度可能与并行执行块的粒度不同。

7.6 分类账认证

在管道中的这一点上，每个验证器都计算了已提交事务块的新状态。 但是，为了有效地支持经过验证的轻型客户端和状态同步，Aptos块链对分类帐历史记录和分类帐状态执行分类帐验证。 Aptos封锁链的一个关键区别是，分类账认证不在交易处理的关键路径上，如果需要，甚至可以完全在带外运行。

7.6.1 分类账历史记录认证

验证器将事务及其执行输出附加到全局验证的分类帐数据结构。 事务输出的一部分是状态写入集，由对可通过MOVE访问的全局状态所做的更改组成。 此数据结构的短验证器是对分类帐历史记录的绑定承诺，其中包括新执行的一批事务。 与事务执行类似，此数据结构的生成具有确定性。

每个验证器都对生成的数据库的新版本的短验证器进行签名。 验证器彼此共享其最近的一组签名短验证器，并将仲裁签名短验证器集合在一起，还将最近的仲裁签名短验证器彼此共享。

使用此集合签名，客户端可以信任数据库版本根据协议的BFT属性表示完整，有效和不可逆的分类帐历史记录。 客户端可以查询任何验证器(或数据库的任何第三方副本，如完整节点)来读取数据库值，并使用验证器和所需数据的证明来验证结果。

7.6.2 定期状态认证

可通过MOVE访问的整个全局状态可在历史的任何时刻总结为简短的验证器，类似于分类帐历史记录的摘要。 由于全局状态的随机访问性质(与仅附加的分类帐历史记录不同)，维护此身份验证的成本很高。 但是，在一个大批次中更新数据结构时，我们可以并行计算更新，还可以利用每个状态值更改时必须更新的零件之间的任何重叠。 Aptos块链特意仅定期验证全局状态以减少重复的共享更新。

在确定性和配置的时间间隔内，网络会发出状态检查点事务，这些事务在输出时包含全局状态验证器。 此类版本表示状态检查点。 两个检查点之间的差距越大，每笔交易更新状态验证数据结构的摊销成本就越低。

使用状态检查点，用户可以以不可信的方式读取其中的任何状态值，而不存储所有全局状态。此功能对于增量状态同步，跨验证器共享存储，无状态验证器节点和存储受限的轻型客户端等应用程序非常有用。

但是，由于状态检查点是定期的，因此要获得分类帐状态特定版本的证明，则需要对缺少的状态交替执行额外的事务执行，或者需要从已验证的分类帐历史记录中获得包含这些检查点的证明。

状态检查点与分类帐历史记录中的特定事务版本相关联，因此与第7节中提到的事务批次相关联的时间戳相关联。 使用时间戳，轻量级客户端可以了解经过验证的状态值的最近时间。 如果没有时间戳，简单的客户端证明只能确保以前的状态的有效性，而以前的状态可能是很旧的，这几乎不能保证相关性。 此外，状态打样的时间戳对于跟踪历史访问和审核目的(例如计算令牌保留中令牌的平均每小时余额)是必需的。

可以根据以前的状态检查点和之后事务输出中的状态交替推导出状态检查点。 因此，持续的状态检查点到稳定的存储不需要位于事务处理的关键路径上。 此外，在坚持国家检查站时也存在有利的批处理效应。 在内存中缓存最近的状态检查点(或者它们之间的增量)，并仅将定期状态检查点转储到稳定存储，可以大大减少存储带宽的消耗。 选择保留检查点的方式不会影响经过验证的数据结构的计算。 因此，这是每个节点的选择：节点操作员可以在内存容量和存储带宽之间配置适当的平衡。

8 状态同步

Aptos块链旨在为生态系统中的所有参与者提供高吞吐量，低延迟的系统。 因此，块链必须提供有效的状态同步协议，以便将块链数据传播，验证和保留给轻型客户端，完整节点和验证器[14]。 此外，同步协议还必须能够容忍网络中的资源限制和异构性，同时考虑到不同的用户和使用情况。 例如，它必须允许归档完整节点验证并保留整个块链历史记录和状态，同时使轻量客户端只能有效地跟踪Aptos块链状态的一小部分。

为了实现此属性，Aptos块链利用验证程序，完整节点和其他复制程序提供的经过身份验证的分类帐历史记录和经过验证的状态证明(请参阅第7.6.1节)，提供灵活且可配置的同步协议。 具体而言，网络中的参与者可以选择不同的同步策略，以针对其使用案例和要求进行优化。

例如，在全节点的情况下，Aptos允许多种同步策略，包括能够处理自时间开始以来的所有事务或完全跳过封锁链历史记录，并使用航点仅同步最新的封锁链状态。 对于轻量客户端，策略包括同步部分封锁链状态(例如，特定帐户或数据值)和启用

已验证状态读取，例如，已验证帐户余额提取。 在所有情况下，Aptos都允许参与者配置要获取，处理和保留的数据的数量和期限。

通过采用灵活且可配置的状态同步方法，Aptos可以适应各种客户端要求，并在未来继续提供新的，更高效的同步策略。

9 社区所有权

Aptos封锁链将由广泛而多样的社区拥有，运营和管理。 本机Aptos令牌将用于交易和网络费用，协议升级和链式/脱链式流程的管理投票，以及通过利益相关方证明模型确保封锁链的安全。 未来的出版物将对Aptos标记经济学进行完整描述。

9.1 交易和网络费

所有Aptos交易都有一个气体单位价格(在Aptos令牌中指定)，使验证员可以优先处理网络中价值最高的交易。 此外，在管道模式的每个阶段，都有多种机会放弃低价值交易(允许封锁链在系统容量下有效运作)。 随着时间的推移，将会部署网络费用，以确保使用Aptos块链的成本与硬件部署，维护和节点操作的实际成本成比例。 此外，开发人员将有机会设计计算，存储和网络之间具有不同成本权衡的应用程序。

9.2 网络管理

Aptos块链上的每一项重大功能更改和改进都将经历几个阶段，包括建议，实施，测试和部署。 这种结构为相关方和利益相关方提供反馈，分享顾虑和提供建议创造了机会。 最后阶段，即部署，通常分两步完成。 首先，将向每个节点部署具有新功能的软件版本，其次，将启用该功能，例如通过功能标志或链上配置变量。

节点操作员进行的每个软件部署必须向后兼容，以确保新软件可与受支持的版本互操作。 部署新软件版本的过程可能需要多天时间，以考虑不同时区的操作员和任何外部问题。 一旦升级了足够数量的节点，则可以通过同步点(如商定的块高度或时代变化)来启用新功能。 在紧急情况下(例如，不可避免的停机时间)，可以通过节点操作员手动和强制更改来启用，在最坏的情况下，可以通过网络中的硬派生来启用。

与其它块链相比，Aptos块链对其链式配置进行编码。 每个验证程序都能够与块链的当前状态同步，并根据当前的链上值自动选择正确的配置(例如，协商一致协议和Aptos框架版本)。 由于此功能，Aptos块链中的升级是无缝和即时的。

为了使启用过程具有灵活性和可配置性，Aptos封锁链将支持链式管理，其中令牌持有人可以根据其固定的令牌权重进行投票。 链式投票协议是公开的，可验证的，并且可以是即时的。 链式管理还可以支持在不部署软件的情况下实现非二进制结果。 例如，链式领导者选举协议参数可以通过链式管理进行修改，而已知同步点将无法处理动态修改，因为所有更改都必须提前知道。

随着时间的推移，可以在整个升级管理过程中部署链式管理。
例如：

1. 令牌持有人在向新的抗量子签名方案过渡的过程中进行了连锁投票。
2. 开发人员实施并验证新的签名方案，并创建新的软件版本。
3. 验证员将其软件升级到新版本。

4. 令牌持有人在链上投票以启用新签名方案，链上配置将更新，更改将生效。

作为一个开放源代码项目，Aptos blockchain将依赖强大的社区反馈，并使用链式管理来管理适当的流程。 在某些情况下，可能仍需要启用脱机升级，但随着时间的推移，这种支持将会最小化。

9.3 利益攸关的共识

要参与Aptos封锁链上的事务验证，验证者必须具有最少的必需数量的Staked Aptos令牌。 在交易发布期间，制动金额按比例影响2f + 1个股权加权PoAv，以及在块元数据排序期间的投票权重和领导者选择。 验证者决定自己和各自的烧杯者之间的奖励划分。 Stakers可以选择任意数量的验证者，将其令牌用于预先商定的奖励分配。 在每个时代结束时，验证员及其各自的stakers将通过相关的连锁移动模块获得奖励。

任何拥有足够的电桩的验证器操作员都可以自由地加入Aptos块链。 所有参数(包括所需的最低风险)均可通过第9.2节中所述的链式启用流程进行设置。

10 性能

如第7节所述，Aptos块链能够通过其并行，批处理优化和模块化事务处理管道实现最佳吞吐量和硬件效率。 其他性能计划(如协商一致升级，增量写入，事务提示和关键路径缓存)将随着时间的推移继续提高吞吐量和效率。

如今，块链吞吐量通常以每秒事务数来衡量。 然而，鉴于交易和基础设施的成本和复杂性范围广泛，这是一种不精确的系统比较方法。 事务延迟也同样存在缺陷，因为提交至最终结果的起点和终点在不同的实验中各不相同。

此外，一些系统要求对交易输入和产出有更深入的了解，并迫使逻辑交易被分成较小，较不复杂的交易。 拆分事务会导致用户体验不佳，并人为地影响延迟和吞吐量，而不考虑开发人员要完成的任务。 相比之下，Aptos方法是让开发人员能够自由地进行构建，不受限制，并测量实际使用案例(而非综合事务)的吞吐量和延迟。

Aptos块链将继续优化各个验证器的性能，并尝试将更多的验证器添加到网络中的缩放技术。 这两个方向都有不同的权衡。 任何具有并行执行功能的块链都可以通过要求更强大的硬件或甚至将每个验证器构建为单个计算机群集来支持其他并发性。 但是，全球验证器的数量存在实际限制，这与验证器操作人员的成本和复杂性相称。 云服务中无服务器数据库的兴起和普及说明了很少有实体可以高效地部署和维护这些类型的复杂分布式系统。

10.1 同质状态分区

最初，Aptos封锁链将以单一分类账状态启动。 随着时间的推移，Aptos网络将采用独特的方法实现横向可扩展性，同时仍保持权力下放。 这将通过多个共享分类帐状态来实现，每个状态都提供了一个同类API和共享作为一流的概念。 Aptos令牌将用于所有碎片的交易费用，桩账和治理。

数据可以通过同构网桥在碎片之间传输。 用户和开发人员可以根据自己的需要选择自己的分片方案。例如，开发人员可以在现有的共享区内建议新的共享区或群集用户，以实现高的共享区内连接。 此外，碎片可能具有不同的系统特征。 一个碎片可以使用进行计算优化

SSD和另一个可以针对计算特性较低的大型硬盘进行优化。 通过在不同的碎片之间提供硬件灵活性，开发人员可以利用适合其应用程序的系统特性。

总之，同构状态分片提供了水平吞吐量可扩展性的潜力，允许开发人员在分片间使用单一通用状态进行编程，并使钱包能够轻松地为其用户合并共享数据。 这提供了单个统一移动智能合同平台的显著性能优势和简单性。

参考资料

[1] “Aptos-core”，2022。［在线］。 可查阅：https://github.com/aptos-labs/aptos-core

[2] “移动”2022。［在线］。 可查阅：https://github.com/move-language/move

[3] D. Matsumoka, C. Dixon, E. Lazzarin和R. Hackett。 (2022)介绍2022加密状态报告。［在线］。 可查阅：https://a16z.com/tag/state-of-crypto-2022/

[4] Z. Amsden, R. Arora, S. Bano, M. Baudet, S. Blackshear, A. Bothra, G. Cabrera, C. Catalini, K. Chalkia, E. Cheng, A. Ching, A. Chursin, G. Danezes, G. D. Giacomo, D. L. Dill, 丁, 杜琴科, 高, 加奥, 加里洛, M. Gorven, P. Hayes, J. M. Hou, Y. Hu, K. Hurley, K. Lewi, C. Li, Z. Li, D. Malkhi, S. Margulis, B. Maurer, P. Mohassel, L. de Naurois, V. Nikolaenko, T. Nowacki, O. Orlov, D. Perelman, A. Poott, B. Proctor, S. Qadeer, Rain, D. Russi, B. Schwab, S. Sezer, A. Sonnino, H. Venter, L. Wei, N. Wernervet, B. Williams, Q. Wu, X. Yan, T. Zakian和R. Zhou, “the libra blockchain”，2019。［在线］。 可查阅：https://developers.diem.com/papers/the-diem-blockchain/2020年05月26日 . pdf

[5] S. Blackshear, E. Cheng, D. L. Dill, V. Gao, B. Maurer, T. Nowacki, A. Poott, S. Qadeer, D. R. Rain, S. Sezer, T. Zakian和R. Zhou, “Move：A language with programmable resources”，2019。［在线］。 可查阅：https://developers.diem.com/papers/diem-move-a-language-with-programmable-resources/2019年06月18日 . pdf

[6] D. Dill, W. Grieskamp, J. Park, S. Qadeer, M. Xu, 和E. Zhong, “使用移动程序快速可靠地正式验证智能合同”，载于Tools and Algorithms for the Construction and Analysis of Systems, D. Fisman and G. Rosu, Eds. Cham: Springer International Publishing, 2022, pp. 183-200.

[7] N. Popper。 (2021)丢失密码将百万富翁从比特币财富中锁定。［在线］。 可查阅：https://www.nytimes.com/2021/01/12/technology/bitcoin-passwords-wallets-fortunes.html

[8] Diem团队, “在重新配置的系统中对已提交信息进行状态同步和验证”，2020年。［在线］。 可查阅：https://github.com/aptos-labs/aptos-core/blob/main/documentation/tech-papers/lbft-verification/lbft-verification.pdf

[9] G. Danezis, L. Kokoris-Kogias, A. Sonnino和A. Spiegelman, 《Narwhal and tusk：一个基于DAG的成员库和有效的bft共识》，载于第十七届欧洲计算机系统会议的议事录，ser. 欧洲系统22. 美国纽约州纽约市：计算机械协会，2022年，第34-50页。［在线］。 可查阅：https://doi.org/10.1145/3492321.3519594</s>349.2321万 351.9594万

[10] Diem团队, “Diembft v4：State machine replication in the diem blockchain”，2021。［在线］。可查阅：https://developers.diem.com/papers/diem-consensus-state-machine-replication-in-the-diem-blockchain/2021年08月17日 . pdf

[11] S. Cohen, R. Gelashvili, L. Kokoris Kogias, Z. Li, D. Malkhi, A. Sonnino和A. Spiegelman, “请注意你的领导人”，Corr, vol. ABS/210.0.096万，2021。［在线］。 可查阅：https://arxiv.org/abs/2110.00960

[12] A. Spiegelman, N. Giridharan, A. Sonnino和L. Kokoris-Kogias, ″Bullshark: Dag bft protocols made inactued″, 载于第20届计算机和通信安全会议(CCS)议事录。 CCS’22. 美国加利福尼亚州洛杉矶：计算机械协会，2022年。

[13] R. Gelashvili, A. Spiegelman, Z. Xiang, G. Danezis, Z. Li, Y. Xia, R. Zhou和D. Malkhi, “Block-STM：Scaling blockchain execution by turning ordering curse to a performance bessing, ”2022。［在线］。 可查阅：https://arxiv.org/abs/2203.06871

[14] J. Lind, “状态同步的演变：每秒处理100k以上事务的路径，次秒Latin Aptos”，2022年。［在线］。 可查阅：https://medium.com/aptoslabs/52e25a2c6f10