Aptos区块链：安全、可扩展和可升级的 Web3 基础架构

译者：MoveWorld社区 — G_xbt

版本：v1.0

摘要

区块链作为一种新的互联网基础设施的兴起，使得开发者以快速增长的速度部署了数以万计的去中心化的应用程序。不幸的是，由于频繁的中断、高成本、低吞吐量的限制、以及众多的安全问题，使得区块链的使用还没有普遍化。为了在web3时代实现大规模应用，区块链基础设施需要遵循云计算基础设施的道路，去成为一个可信赖的、可扩展的、具有成本效益的、并且可以不断改进的平台，用于构建广泛使用的应用程序。为此，我们提出了Aptos区块链，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。

在过去三年中，Aptos区块链由全球350多名开发者共同开发[1]。它在共识、智能合约设计、系统安全、性能和去中心化方面提供了新的思路和新颖的创新。这些技术的组合将提供一个基础的框架，从而把web3带给大众：[1]

- 首先，Aptos区块链原生集成并在内部使用Move语言来实现快速安全的交易执行[2]。Move验证器，一个用Move语言编写的智能合约的正式验证器，为合约的不变量和行为提供了额外的保障。这种对安全的关注使开发者能够更好地保护他们的软件免受恶意实体的侵害。

[1] 法律声明：本白皮书及其内容不是出售任何代币的要约，也不是征求购买任何代币的要约。我们发布本白皮书只是为了接受公众的反馈和意见。本文件中的任何内容都不应被理解为对Aptos区块链或其代币（如果有）如何发展、利用或累积价值的保证或承诺。Aptos只概述了其目前的计划，这些计划可能会酌情改变，其成功与否将取决于其控制之外的许多因素。这种未来声明必然涉及已知和未知的风险，这可能导致未来时期的实际表现和结果与我们在本白皮书中描述或暗示的有实质性的差异。Aptos不承担更新其计划的义务。不能保证白皮书中的任何陈述将被证明是准确的，因为实际结果和未来事件可能会有很大的不同。请不要过分依赖未来的声明。

• 其次，Aptos数据模型能够实现灵活的密钥管理和混合托管选项。这一点，加上签署前的交易透明度和实用的轻型客户端协议，提供了一个更安全和更值得信赖的用户体验。

• 第三，为了实现高吞吐量和低延迟，Aptos区块链在交易处理的关键阶段采用了流水线和模块化方法。具体来说，交易传播、区块元数据排序、并行交易执行、批量存储和分类账认证都是同时进行的。这种方法充分利用了所有可用的物理资源，提高了硬件效率，并实现了高度并行执行。

• 第四，与其他并行执行引擎不同，这些引擎通过要求预先了解要读和写的数据来打破交易的原子性，Aptos区块链没有对开发者施加这种限制。它可以有效地支持任意复杂事务的原子性，为现实世界的应用实现更高的吞吐量和更低的延迟，并简化了开发。

• 第五，Aptos模块化架构设计支持客户的灵活性，并对频繁的即时升级进行了优化。此外，为了快速部署新的技术创新和支持新的web3用例，Aptos区块链提供嵌入式链上变更管理协议。

• 最后，Aptos 区块链正在试验未来的计划，以超越单个验证者的性能：其模块化设计和并行执行引擎支持验证者的内部分片，同质状态分片提供了水平吞吐量可扩展性的潜力，而不会增加节点运营商的额外复杂性。

1 导言

在web2版本的互联网中，消息、社交媒体、金融、游戏、购物和音频/视频流等服务由控制用户数据的中心化公司提供（例如，谷歌、亚马逊、苹果和Meta）。这些公司使用针对目标用例优化的特定应用软件开发基础架构，并利用云基础架构将这些应用部署给用户。云基础设施提供对虚拟化和/或物理基础设施服务的访问，如租用的虚拟机（VM）和在全球数据中心内运行的裸机硬件（如AWS、Azure和谷歌云）。因此，建立能够扩展到数十亿用户的web2互联网服务从未像今天这样容易。然而，web2 要求用户明确信任中心化实体，这一要求越来越受到社会关注。

为了解决这个问题，一个新的互联网时代已经开始：web3。在互联网的 web3 版本中，区块链已经出现，以提供分散的、不可变的分类账，使用户能够安全可靠地相互交互，所有这些都不需要信任拥有控制能力的中介或中心化实体。与web2互联网

服务和应用程序依赖云基础设施作为构建模块的方式类似，去中心化的应用程序可以使用区块链作为去中心化的基础设施层，来给世界各地的数十亿用户提供服务。

然而，尽管当今存在许多区块链，但尚未广泛采用 web3 [3]。虽然技术不断推动行业发展，但现有的区块链并不可靠，对用户收取高额的交易费用，拥有有低吞吐量的限制，由于安全问题经常遭受资产损失，并且不能支持实时响应。与云计算基础设施让web2服务达到数十亿的规模相比，区块链还没有使web3应用做到这一点。

2 Aptos的愿景

Aptos 的愿景是提供一个区块链，可以为 web3 带来主流采用，并授权去中心化应用程序生态系统来解决现实世界的用户问题。我们的使命是通过提供灵活和模块化的区块链架构，推动区块链可靠性、安全性和性能方面的最先进技术的发展。该架构应支持频繁升级、快速采用已经取得进步的最新技术，以及对新兴用例的一流支持。

我们设想的是一个去中心化的、安全的、可扩展的网络，由使用它的社区管理和运营。当全世界的基础设施需求增长时，区块链的计算资源就会横向和纵向扩展以满足这些需求。随着新的用例和技术进步的出现，网络应该在不干扰用户的情况下频繁地无缝升级。基础设施的问题应该淡出背景。开发人员和用户将可以拥有许多不同的选项，用于密钥恢复、数据建模、智能合约标准、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、始终可用的，并且可以以接近成本的费用进行访问。任何人都可以安全、轻松、不可逆转地与全世界不受信任的各方进行交易。区块链就像云基础设施一样无处不在。

为了实现这一愿景，必须在技术上取得重大进步。我们在过去三年中构建、开发、推进和部署Diem区块链（Aptos区块链的前身）的经验证明，一个网络可以不断地升级其协议而不影响其客户[4]。Diem主网在2020年初被部署到十几个节点运营商与多个钱包供应商。在接下来的一年里，我们的团队发布了两次重大升级，改变了共识协议和核心框架。两次升级都是在不影响用户的情况下完成的。通过Aptos区块链，我们对技术栈进行了一系列彻底的改进，同时还将安全、透明、频繁的升级作为核心功能，这是受Diem区块链的启发。 特别是，我们强调了交易处理的新方法（如第7节所述）和去中心化和网络治理的新方法。

随着Aptos区块链的不断完善和发展，我们将发布本白皮书的最新版本，包括我们协议和设计选择的最新迭代。在本文件的其余部分，我们描述了Aptos区块链的现状以及未来计划。

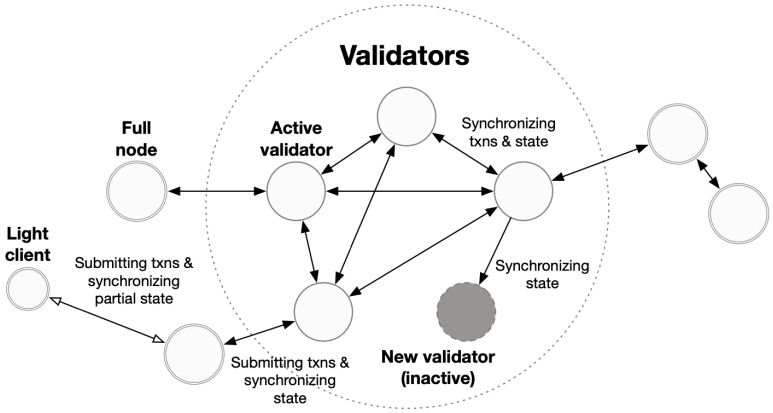

Figure 1: Components of the Aptos ecosystem.

3 概述

如图1所示，Aptos区块链由一组验证器组成，这些验证器使用拜占庭容错（BFT）、股权证明共识机制共同接收和处理用户的交易。代币持有者将代币锁定在他们所选择的验证器中，或将代币投入其中。每个验证器的共识投票权重与投在其中的金额成正比。验证者可以是积极的，并参与到共识中。同样，如果验证者没有足够的权益参与被轮换出验证者集、在同步区块链状态时离线了，或者由于历史性能不佳而被共识协议认为不参与，验证者也可能处于非活跃状态。客户端是整个系统的一部分，其职责是提交交易、查询区块链的状态和历史。客户端可以选择下载并验证验证者签名的查询数据证明。 全节点是一种客户端，从验证者或网络中的其他完整节点复制交易和区块链状态。他们可以根据需要选择修剪交易历史和区块链状态，以减小储存的大小。轻客户端只维护当前的验证器集合，并可以安全地查询部分区块链状态，我们通常的选择是从全节点查询。钱包是轻型客户端的一个常见例子。

为了满足安全、快速、可靠和可升级的web3基础设施的需求，以便广泛采用，Aptos区块链建立在以下核心设计原则之上。

• 通过一种新的智能合约编程语言 Move [5]，快速、安全地执行以及简单的可审计性和机械分析性。 Move 起源于 Aptos 区块链的前身，并随着该项目的发展而不断进步。

• 通过批处理、流水线和并行化的事务处理方法实现极高的吞吐量和低延迟。

• 新颖的并行事务处理通过 Block-STM 有效地支持任意复杂事务的原子性，这与现有的并行执行引擎不同，后者需要预先了解要读取和写入的数据位置。

• 通过快速的权益权重验证器集轮换和声誉跟踪来优化性能和去中心化。

• 可升级性和可配置性作为一流的设计原则，以拥抱新用例和最新技术。

• 模块化设计可实现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，同时保持去中心化，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原生概念。

第 4 节解释了开发人员如何与 Aptos 区块链中的 Move 交互。第 5 节描述了逻辑数据模型。第 6 节详细介绍了 Aptos 区块链如何通过强大的验证方法实现安全的用户体验。第 7 节描述了围绕流水线、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 节描述了我们的社区所有权和治理计划。最后，第 10 节讨论了在保持去中心化的同时未来的性能方向。

4 Move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 区块链使用 Move 的对象模型来表示其账本状态（参见第 5.5 节），并使用 Move 代码（模块）对状态转换规则进行编码。用户提交的交易可以发布新模块、升级现有模块、执行模块内定义的入口功能，或者包含可以直接与模块公共接口交互的脚本。

Move 生态系统包含编译器、虚拟机和许多其他开发工具。Move 受到 Rust 编程语言的启发，它通过线性类型等概念在语言中明确数据的所有权。Move模块定义了每个资源的生命周期、存储和访问模式。这确保了像 Coin 这样的资源不会在没有适当凭证的情况下产生，不会被双重花费，也不会消失。

即使存在不受信任的代码，Move 也利用字节码验证器来保证类型和内存安全。为了帮助编写更受信任的代码，Move 包含了一个正式的验证器，即 Move Prover [6]，它能够根据给定的规范验证 Move 程序的功能正确性，该规范以集成到 Move 中的规范语言制定。

除了用户账户和相应的账户内容，账本状态还包含 Aptos 区块链的链上配置。此网络配置包括一组活跃的验证者、质押属性以及 Aptos 区块链中各种服务的配置。Move 对模块可升级性和全面可编程性的支持可实现无缝配置更改，并支持升级 Aptos 区块链本身（两组升级已多次执行，私有主网上的停机时间为零）。Aptos 团队进一步增强了 Move，支持更广泛的 web3 用例。如后面 5.5 节所述，Aptos 区块链实现了细粒度的资源控制。这不仅支持执行的并行化，而且还实现了与访问和修改数据相关的近乎固定的成本。此外，Aptos区块链提供了建立在细粒度存储之上的表支持，允许在单个帐户中使用大规模数据集（例如，大量NFT集合）。此外，Aptos支持完全在链上表示的共享或个人帐户。这使得复杂的去中心化自治组织（DAO）能够协作地共享账户，以及将这些账户作为异质资源集合的容器。

5 逻辑数据模型

Aptos区块链的账本状态代表了所有账户的状态。账本状态使用一个无符号的64位整数，对应于系统所执行的交易数量，进行版本划分。任何人都可以向Aptos区块链提交交易以修改账本状态。在执行一个交易时，会产生一个交易输出。一个交易输出包含零个或多个操作来操纵账本状态（称为写入集），一个由此产生的事件向量（见第5.1.1节），消耗的气体量，以及已执行的交易状态。

5.1 交易

一个已签名的交易包含以下信息：

• 交易验证器：发送方使用包含一个或多个数字签名的交易验证器来验证交易是否经过验证。

• 发送者地址：发送者的账户地址。

• 有效负载：有效负载要么指链上现有的入口函数，要么包含要作为内联字节码（称为脚本）执行的函数。此外，一组输入参数被编码为字节数组。对于点对点交易，输入包含接收者的信息和转移给他们的金额。

• Gas 价格（以指定的货币/Gas 单位表示）：这是发送者愿意为执行交易而为每单位 Gas 支付的金额。 Gas 是一种支付计算、网络和存储费用的方式。气体单位是计算的抽象度量，没有内在的现实世界价值。

• 最大气体量：最大气体量是事务在中止前允许消耗的最大气体单位。账户必须至少有天然气价格乘以最大天然气金额，否则交易将在验证期间被放弃。

• 序号：交易的序号。这必须与交易执行时存储在发件人帐户中的序列号匹配。成功执行交易后，帐户序列号将增加，以防止重播攻击。

• 过期时间：交易停止有效的时间戳。

• 链id：标识此交易有效的区块链，提供进一步保护

用于用户防止签名错误。

在每个版本 i，状态变化由元组 (Ti,Oi,Si) 表示，分别包含交易、交易输出和结果账本状态。给定一个确定性函数 Apply，执行具有账本状态 Si-1 的交易 Ti 会产生交易输出 Oi 和一个新的账本状态 Si。即，$Apply(Si-1, Ti) \rightarrow \langle Oi, Si \rangle$。

5.1.1 事件

事件是在交易执行过程中发出的。每个Move模块可以定义自己的事件，并选择在执行时发出这些事件。例如，在代币转移过程中，发送方和接收方的账户都会分别

发出SentEvent和ReceivedEvent。这些数据存储在账本中，可以通过Aptos节点进行查询。每个注册的事件都有一个唯一的密钥，这个密钥可以用来查询事件的细节。

向同一事件键发的多个事件生成事件流、事件列表，每个条目包含从0开始的顺序递增的数字、类型和数据。每个事件必须由某种类型定义。每个事件必须由某种类型定义。可能存在由相同或类似类型定义的多个事件，尤其是在使用泛型时。事件具有关联数据。对于Move模块开发人员来说，一般原则是在执行更改数据和发出事件的事务之前和之后，包括理解底层资源更改所需的所有数据。交易只能生成事件，不能读取事件。这种设计使交易的执行只成为当前状态和交易输入的函数，而不是历史信息（例如，以前产生的事件）。

5.2 账户

每个帐户由一个唯一的256位值标识，该值称为帐户地址。当从现有账户发送的交易调用create_account（addr）Move函数时，将在账本状态下创建一个新账户（见第5.5节）。当交易试图向尚未创建的帐户地址发送Aptos令牌时，通常会发生这种情况。为了方便起见，Aptos还支持transfer(from,to,amount)功能，如果在转账之前账户不存在，则该功能会隐式创建账户。

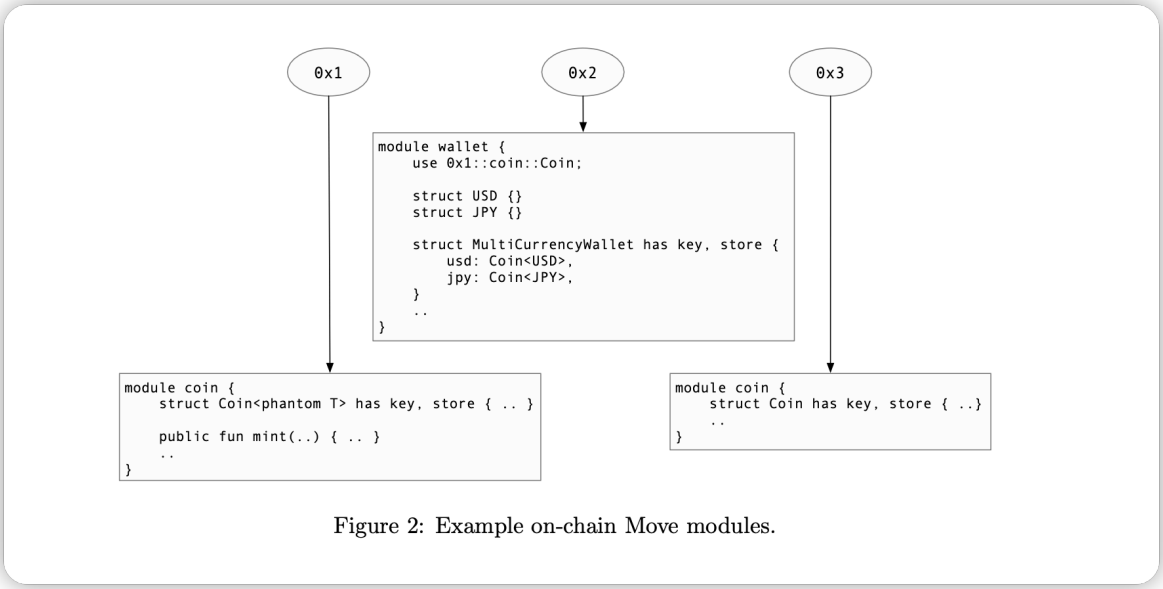

Figure 2: Example on-chain Move modules.

为了创建一个新账户，用户首先生成一个签名密钥对:(vk，sk)。接下来，使用与签名方案标识符 (ssid) 连接的公共验证密钥 vk 的加密哈希 H 导出给定签名方案的新帐户地址： 其中 addr = H(vk, ssid)。

在地址 addr 创建新帐户后，用户可以使用私有签名密钥 sk 对要从 addr 帐户发送的交易进行签名。用户还可以轮换 sk，以主动更改 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期间仅从公共验证密钥派生一次。

Aptos 区块链不会将账户与真实世界的身份相关联。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内在联系。但是，单个用户仍然可以在单个钱包中管理多个帐户，以进行简单的资产管理。这种灵活性为用户提供了匿名性，同时我们为未来的版本试验了保护隐私提供了基础。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通道，如第 7.4 节所述。

5.3 Move 模块

Move 模块包含声明数据类型（结构）和过程的 Move 字节码。它由声明模块的帐户地址和模块名称来标识。例如，图 2 中第一个货币模块的标识符是 0x1::coin。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声明一个具有任何给定名称的模块。例如，图 2 中地址 0x1 的帐户无法声明另一个名为 coin 的模块。另一方面，地址为 0x3 的账户可以声明一个名为 coin 的模块，该模块的标识符为 0x3::coin。请注意，0x1::coin::Coin 和 0x3::coin::Coin 是不同的类型，不能互换使用，也不能共享公共模块代码。

模块被分组到位于相同地址的包中。该地址的所有者将包作为一个整体在链上发布，包括字节码和包元数据。包元数据决定了一个包是可以升级还是不可变的。对于可升级包，在允许升级之前执行兼容性检查：必须更改现有的入口点函数，并且不能将资源存储在内存中。但是，可以添加新的功能和资源。Aptos 框架由 Aptos 区块链的核心库和配置组成，被定义为可定期升级的模块包（参见第 9.2 节）。

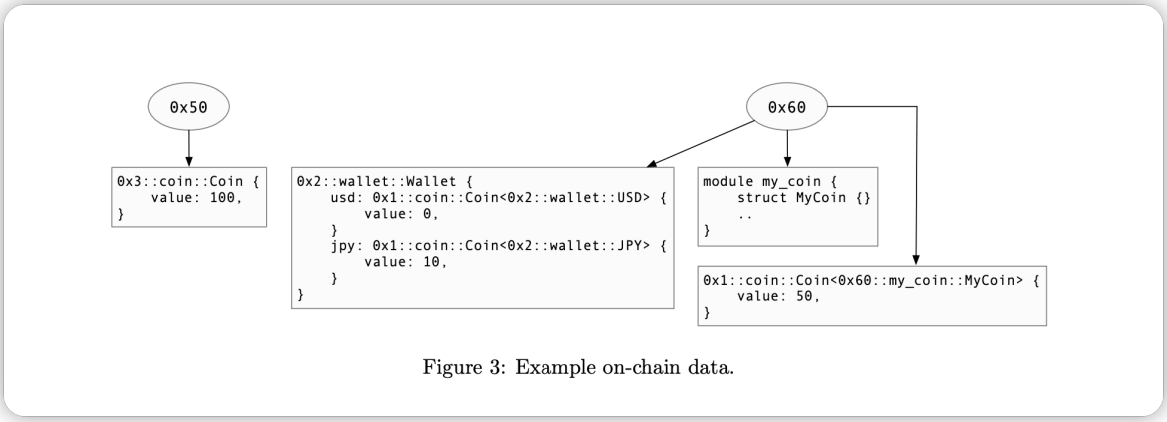

Figure 3: Example on-chain data.

5.4 资源

与模块类似，帐户地址也可以具有与之关联的数据值。在每个帐户地址中，值按其类型作为键，每种类型最多有一个值属于该帐户。图 3 提供了一个例子。地址 0x50 保存单个值，其中 0x3::coin::Coin 是完全限定类型。0x3是coin模块存放的地址，coin是模块名称，Coin是数据类型名称。泛型类型的值也是允许的，不同的实例被视为不同的类型。这对于可扩展性至关重要，允许不同的实例共享相同的功能代码。

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。在一个地址下最多具有每个类型的一个顶级值可能起初听起来很有限。但是，这在实践中不是问题，因为程序员可以将包含其他数据的包装器类型定义为内部字段，从而避免任何限制。图 3 中的 Wallet 结构是如何使用包装器类型的示例。

还应注意，并非所有数据类型都可以存储在链上。为了使数据实例符合顶级值，数据类型必须具有键能力。同样，嵌套值也需要存储能力。具有这两种能力的数据类型也称为资源。

5.5 账本状态

从 Move 虚拟机（Move VM）的角度来看，每个账户由一组值和键值数据结构组成。这种数据布局使开发人员能够编写智能合约，这些合约可以有效地处理跨大量账户复制的少量数据，以及存储在少数账户中的大量数据。Move模块的存储方式与帐户数据类似，但在独立的命名空间下。创世账本状态定义了初始账户集及其在区块链初始化时的关联状态。

在发布时，Aptos 区块链将由一个单一的账本状态表示。然而，随着采用率的提高和技术的发展，Aptos 将增加分片的数量以增加吞吐量（即启用多个分类帐状态）并支持跨分片移动或访问资产的交易。每个账本状态都将维护特定分片的所有链上资产，并为相同的账户模型提供细粒度的键值数据存储，为存储访问提供近乎固定的成本。

6 安全的用户体验

为了覆盖数十亿互联网用户，web3 用户体验必须安全且易于访问。在以下部分中，我们描述了 Aptos 区块链提供的几项创新，旨在实现这一目标。签署交易意味着签署者授权该交易被区块链承诺和执行。

6.1 交易生存期保护

签署交易意味着签署者授权该交易被区块链承诺和执行。为了降低这一风险，Aptos区块链限制了每笔交易的生存周期，并保护签名者免受无限生存周期的影响。目前Aptos区块链提供了三种不同的保护—发个人交易的序列号、交易到期时间和指定的链上标识符。

• 一笔交易的序列号对每个发件人的账户只能准确地承诺一次。因此，发送者可以观察到，如果当前账户序列号≥交易t的序列号，那么要么t已经被承诺，要么t永远不会被承诺（因为t使用的序列号已经被另一个交易消耗）。

• 如第7.3.1节所述，区块链时间以高精度和高频率（通常为亚秒）推进。如果区块链时间超过了交易t的到期时间，那么同样地，要么t已经被提交，要么t永远不会被提交。

• 每笔交易都有一个指定的链上标识符，以防止恶意实体在不同的区块链环境（例如，跨越测试网和主网）之间重放交易。

6.2 基于Move的密钥管理

如第5.2节所述，Aptos账户支持密钥轮换，这是一个重要的功能，可以帮助减少与私钥泄露、远程攻击和可能打破现有加密算法的未来进展有关的风险。此外，Aptos账户也很灵活，可以实现新的混合托管模式。在一种这样的模型中，用户可以将轮换帐户私钥的能力委托给一个或多个保管人和其他受信任的实体。然后，Move 模块可以定义一个策略，使这些受信任的实体能够在特定情况下轮换密钥。例如，实体可能由许多受信任方持有的 k-out-of-n 多重签名密钥表示，并提供密钥恢复服务以防止用户密钥丢失（例如，目前 20% 的比特币被锁定在不可恢复的账户中[7]）。

此外，虽然许多钱包支持各种密钥恢复方案，例如将私钥备份到云基础设施、多方计算和社会恢复，但它们通常在没有区块链支持的情况下实施（即脱链）。结

果，每个钱包都需要实现自己的密钥管理基础设施，相关操作对用户来说变得不透明。相比之下，在 Aptos 区块链层支持密钥管理功能提供了所有与密钥相关的操作的完全透明性，并使实施具有丰富密钥管理的钱包变得更加简单。

6.3 签署前交易透明度

如今，钱包对他们签署的交易几乎没有透明度。因此，用户往往很容易被骗签署恶意交易，从而窃取资金并造成灾难性后果。即使对于需要枚举每笔交易访问的所有链上数据的区块链也是如此。因此，目前几乎没有用户保护措施，使用户容易受到各种各样的攻击。

为了解决这个问题，Aptos 生态系统为交易预执行提供服务：一种预防措施，在签署之前向用户（以人类可读的形式）描述他们的交易结果。将此与已知的先前攻击历史和恶意智能合约相结合将有助于减少欺诈。此外，Aptos 还使钱包能够在执行期间规定对交易的限制。违反这些约束将导致交易被中止，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 实用的轻客户端协议

仅仅依靠 API 提供商的 TLS/SSL 证书在区块链客户端和服务器之间建立信任并不能充分保护客户端。即使存在有效证书，钱包和客户也无法保证提供给他们的数据的真实性和完整性。

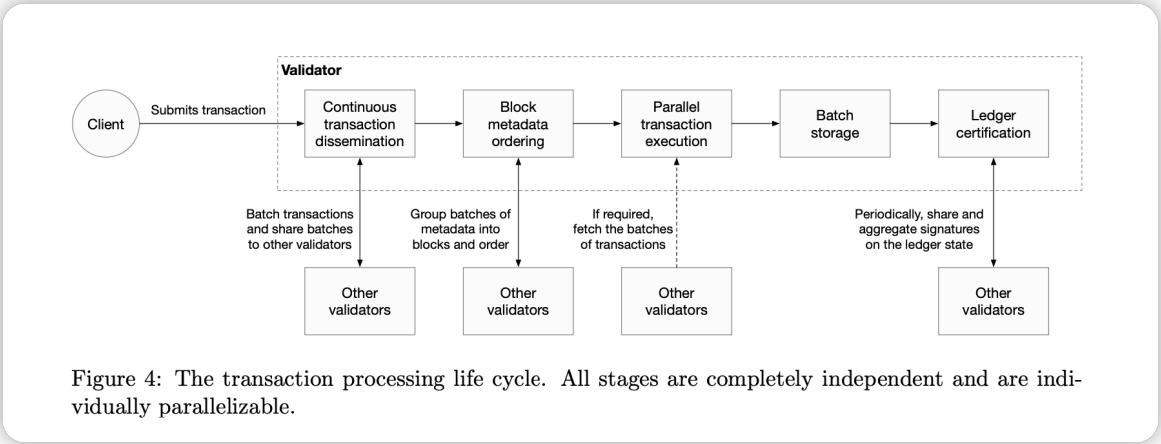

Figure 4: The transaction processing life cycle. All stages are completely independent and are individually parallelizable.

因此，API 提供商可能会返回不正确或恶意的区块链数据，从而欺骗第三方并执行双花攻击。

为了防止这种情况，Aptos 提供状态证明和轻客户端验证协议，钱包和客户端可以使用这些协议来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 7.6.2 节中基于时间戳的状态证明，轻客户端始终可以确保帐户状态的新鲜度（例如，在几秒钟内）的严格限制，并且只需要跟踪网络配置的变化（纪元变化）或使用当前受信任的检查点（航路点）来保持最新状态 [8]。通过结合高频时间戳和廉价的状态证明，Aptos 区块链为客户提供了更高的安全保障。

此外，Aptos节点还公开了丰富的高性能存储接口，可以进一步微调这些接口，以允许订阅针对链上特定数据和帐户的证据。这可以被轻型客户利用来保留最小的可验证数据，而不需要运行一个完整的节点或处理大量的交易。

7 流水线、批处理和并行交易处理

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 区块链上的交易处理被划分为不同的阶段。每个阶段都是完全独立且可单独并行化的，类似于现代的超标量处理器架构。这不仅提供了显着的性能优势，而且使 Aptos 区块链能够提供验证者-客户端交互的新模式。例如：

• 当特定的交易被包含在一批持久化的交易中时，客户可以得到通知。持久有效的交易极有可能立即被提交。

• 当一批持久化的交易被订购时，客户端可以被告知。因此，为了减少确定已执行交易输出的延迟，客户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。

• 客户端可以选择等待验证者的认证交易执行，并对认证的结果进行状态同步（例如，见第8节）。

Aptos的模块化设计有助于提高开发速度，支持更快的发布周期，因为变化可以针对单个模块，而不是单一的单体架构。同样，模块化设计也提供了一个结构化的路径，可以将验证器扩展到单台机器之外，提供额外的计算、网络和存储资源。图4显示了交易生命周期的各个处理阶段。

7.1 批处理

批处理是一项重要的效率优化，是 Aptos 区块链中每个操作阶段的一部分。在交易传播过程中，每个验证者将交易分组为批处理，在共识过程中将批次组合成块。执行、存储和账本认证阶段也分批工作，以提供重新排序、减少操作（例如，重复计算或签名验证）和并行执行的机会。

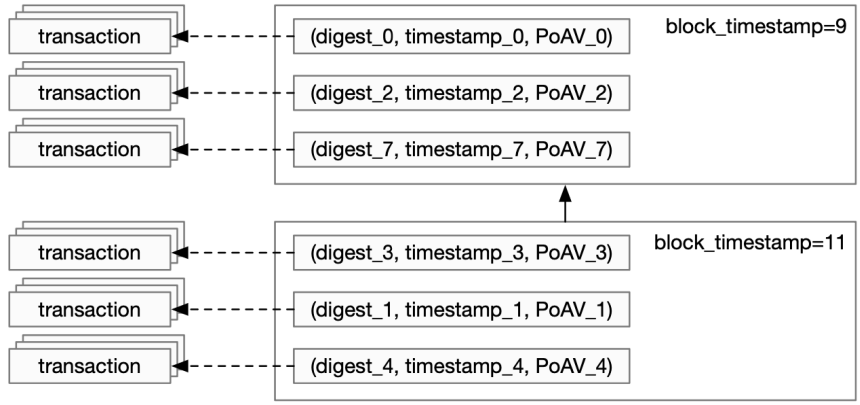

Figure 5: Block metadata ordering occurs independently from transaction dissemination.

将交易分组可能会导致少量延迟，例如，在执行分发之前等待 200 毫秒来累积一批交易。批处理还允许有效的费用市场对交易进行优先排序，并避免来自非常频繁发送交易的客户的意外拒绝服务 (DoS) 攻击。

7.2 交易的持续传播

根据Narwhal & Tusk [9]的主要见解，Aptos区块链中的交易传播与共识脱钩。验证器不断地将成批的交易流向对方，并发地利用所有可用的网络资源。验证器 v 分发的每个批次都被持久化，并且批次摘要上的签名被发送回 v。根据第 7.3 节中定义的共识要求，批处理摘要上的任何 2f + 1 个权益加权签名形成可用性证明 (PoAv)。这样的证明保证至少有 f + 1 个权益加权诚实验证者存储了该批处理，因此所有诚实验证者都能够在执行之前检索它。

无限持久的交易批处理可以通过导致验证器耗尽存储和崩溃来打开 DoS 攻击向量。为了防止这种情况，每批交易都有一个相关的时间戳。批处理上的时间戳允许在每个验证器上进行有效的垃圾收集。此外，一个单独的每个验证者配额机制旨在保护验证者即使在最极端的情况下也不会耗尽空间，例如潜在的拜占庭攻击。批处理还具有大小限制，这些限制在同意持久存储到稳定存储之前经过验证。最后，对重复数据

删除和缓存事务进行的多项优化可降低存储成本并确保与并行执行引擎的高性能集成。

7.3 块元数据排序

一种常见的误解是共识速度很慢，因此是区块链吞吐量和延迟的主要瓶颈。Aptos 区块链的关键创新之一是将非协议相关的任务从共识阶段解耦出来，例如交易传播、交易执行/存储和账本认证。通过将交易传播与共识阶段分离，可以在非常低的带宽下进行排序（仅块元数据和证明），从而实现高交易吞吐量和最小化延迟。

今天，Aptos 区块链利用了 DiemBFTv4 [10] 的最新迭代，这是一种乐观响应的 BFT 共识协议。常见情况下的共识只需要两次网络往返（全球往返时间通常小于 300 毫秒），并通过领导者信誉机制动态调整到有故障的验证者 [11]。链上领导者信誉机制提升在窗口中成功提交区块的验证者，并降级未参与的验证者。这种新颖的机制显着提高了去中心化环境中的性能，相应地为适当的激励提供了基础设施，并迅速将失败的验证器对吞吐量和延迟的影响降至最低。

DiemBFTv4 保证了部分同步下的活跃性，并确保了异步下的安全性，其中总验证者权益 ≥ 3f + 1，最多 f 个权益加权错误验证者。自 2019 年以来，DiemBFTv4 已经在多次迭代中经过了数十个节点运营商和多钱包生态系统的广泛测试。我们还在试验我们最近的研究（例如 Bullshark [12]）和其他依赖区块历史和相关通信来确定区块元数据排序和最终性的协议。

共识块和提案时间戳由领导者提出并由其他验证者同意，如图 5 所示。请注意，每个共识块仅包含批处理元数据和证明。区块中不需要实际交易，因为 PoAV 确保交易批次在订购后的执行阶段可用（参见第 7.2 节）。验证者可以在验证证明并且满足块元数据标准（例如，提议时间戳≤块过期时间）后对领导者的提议进行投票。

7.3.1 区块链时间

Aptos区块链为每个提议的区块以及相应的该区块内的所有交易采用近似的、商定的物理时间戳。这个时间戳支持许多重要的用例。例如：

• 智能合约中的时间依赖逻辑。例如，开发人员希望编码必须在周四中午之前收到拍卖的所有投标。

• 随着预言机发布链上数据，需要准确且可信的链上时间戳来关联事件并处理来自现实世界数据的延迟。

• 客户可以辨别他们在区块链方面的最新情况。出于安全原因，为避免过时数据和远程攻击，客户端应该能够访问帐户状态更新时间的高精度时间戳。

• 使用可信时间戳审计区块链可以提供与链下事件的强相关性，例如确保合法执行的支出符合预期要求。

• 交易到期基于最近提交的时间戳。作为对客户端事务的额外保护，客户端可以选择事务的到期时间，如第 6.1 节所述。

Aptos 区块链为区块内所有交易的时间戳提供以下保证：

• 在区块链中时间是单调递增的。也就是说，如果块 B1 < 块 B2，则 B1.Time < B2.Time。

• 如果一个交易块的时间戳为T，那么至少f+1诚实验证器已经确定T在过去。诚实的验证器只会在自己的时钟停止时对块进行投票≥ 时间戳T。见第7.2节。

• 如果一个事务块具有与时间戳T一致的法定签名，则诚实的验证器在其自己的时钟之前不会将此类块提供给其他验证器≥ 时间戳T。

在每个提交的块上更新最近的时间戳，并将其用作该块中所有交易的时间戳。当网络同步时，每个网络往返都会提交一个交易块，并提供快速更新和高度可靠的时钟。如果需要，可以确定交易块内更精细的排序粒度。

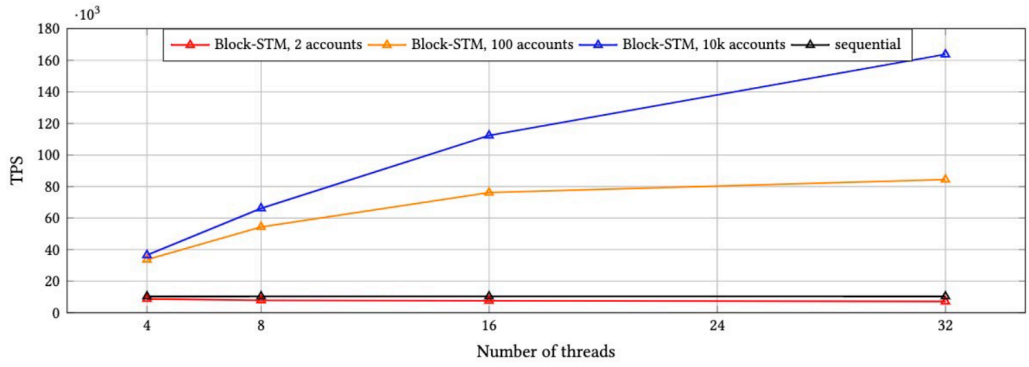

Figure 6: Block-STM (component-only) benchmarks comparing the number of physical cores with different levels of contention.

7.4 并行交易执行

一旦对共识块元数据进行排序，任何验证器、完整节点或客户端都可以执行交易。至少有 2f + 1 个权益加权验证者对提议的批次进行了真正持久的交易。由于交易传播是连续的，额外的诚实验证者将随着时间的推移接收交易批次。如果诚实的验证人在到达执行阶段时还没有收到订单批次的交易，它可以从 2f+1 个权益加权验证人下载它

们，知道至少有 f+1 个权益加权验证人（≥一半的权益加权验证人）权益加权 PoAV 签名者）是诚实的。任何区块链的一个重要目标是实现尽可能多的并行执行。 Aptos 区块链从数据模型和执行引擎两方面向前推进。

7.4.1 并行数据模型

Move数据模型本身支持数据和模块的全局寻址。数据和账户没有重叠冲突的事务可以并行执行。鉴于 Aptos 区块链使用的流水线设计，重新排序一组交易可以减少冲突的数量，从而提高并发性。

即使交易修改了同一组链上值，大部分交易执行过程仍然可以并行化。 Aptos 区块链引入了一个新概念，delta writes，它描述了对账户状态的修改，而不是修改后的账户状态（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，然后以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 区块链将继续以提高并发性（例如，利用读/写提示）并改善人体工程学的方式来增强数据模型，使开发人员更自然地创建、修改和组合链上值。 Move 提供了灵活性，可以在语言级别以及特定于平台的功能上进行这些改进。

7.4.2 并行执行引擎

Block-STM 并行执行引擎检测和管理一组有序交易的冲突以及乐观并发控制，以在给定特定顺序的情况下实现最大并行度 [13]。

批量交易以并行方式乐观地执行，并在执行后得到验证。不成功的验证会导致重新执行。 Block-STM 使用多版本数据结构来避免写-写冲突。所有对同一位置的写入都与它们的版本一起存储，其中包含它们的交易 ID 和写入事务被乐观地重新执行的次数。当交易 tx 读取一个内存位置时，它从多版本数据结构中获取按预设顺序出现在 tx 之前的最高交易写入该位置的值以及相关版本。

Block-STM 已经集成到 Aptos 区块链中。为了了解 Block-STM 性能的全部潜力，我们将非平凡的点对点移动事务（即每个事务 8 次读取和 5 次写入）作为隔离的、仅执行的（不是端到端的）进行了实验) 使用内存数据库进行基准测试。在图 6 中，我们展示了我们的 Block-STM 执行结果。每个区块包含 10k 笔交易，账户数量决定了冲突和争用的程度。

在低争用情况下，Block-STM 在 32 个线程的顺序执行中实现了 16 倍的加速，而在高争用情况下，Block-STM 实现了超过 8 倍的加速。与区块链空间中的其他并行执行引擎不同，Block-STM 能够动态且透明地（无需用户提示）从任何工作负载中提取固有的并行性。与需要预先了解要读取或写入的数据位置的并行执行环境相比，BlockSTM 可以同时支持更复杂的事务。此属性导致更少但更高效的事务，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

会破坏具有复杂状态结果的单个事务的全有或全无语义。在 Block-STM 中将表达性事务语义与并行执行相结合，使开发人员能够两全其美。

请注意，块元数据排序步骤不排除在并行执行阶段重新排序交易。交易可以跨一个或多个块重新排序，以优化并行执行的并发性。唯一的要求是所有诚实验证者的重新排序必须是确定性的。优化并行执行以及在重新排序中添加随机化可以提高性能，并可能阻止最大可提取价值 (MEV) 技术用于有利可图的验证者交易重新排序。“订购然后显示” MEV 抗性策略也可以纳入这种流水线设计中。

Block-STM 和交易重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，以实现额外的并发性。

7.5 批量存储

并行执行阶段导致组中所有交易的写入集。这些写入集可以存储在内存中以获得最大的执行速度，然后用作下一个块或要执行的块集的缓存。任何重叠写入只需要写入稳定存储一次。如果验证器在存储内存写入集之前失败，它可以简单地从块元数据排序阶段恢复并行执行。将写入集的批量存储与并行执行步骤分离，确保并行执行能够高效运行。总之，批处理写入集减少了存储操作的数量，并利用了更高效、更大的 I/O 操作。

可以为每台机器手动配置为写入集缓存保留的内存量，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，批处理的粒度可以不同于并行执行块的粒度。

7.6 账本认证

在管道中的这一点上，每个单独的验证器都已经计算了已提交交易块的新状态。然而，为了有效支持经过验证的轻客户端和状态同步，Aptos 区块链对账本历史和账本状态实施了账本认证。 Aptos 区块链的一个关键区别是账本认证不在交易处理的关键路径上，如果需要，甚至可以完全带外运行。

7.6.1 账本历史证明

验证器将交易连同它们的执行输出一起附加到全局认证的分类帐数据结构中。交易输出的一部分是状态写入集，包括对 Move 可访问的全局状态所做的更改。该数据结构的短验证器是对账本历史的绑定承诺，其中包括新执行的一批交易。与交易执行类似，这种数据结构的生成是确定性的。

每个验证器将短验证器签名到生成的数据库的新版本。验证者彼此共享他们最近的一组已签名的短身份验证器，共同聚合 quorum-signed 短身份验证器，并且还彼此共享最近的 quorum-signed 短身份验证器。

根据协议的 BFT 属性，使用这种集体签名，客户可以相信数据库版本代表了完整、有效和不可逆的分类帐历史。客户端可以查询任何验证器（或数据库的任何第三方副本，例如完整节点）以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

Move 可访问的整个全局状态可以在历史的任何时间点汇总到一个简短的身份验证器，类似于分类帐历史的摘要。由于全局状态的随机访问性质（与仅追加的分类帐历史不同），维护此身份验证的成本很高。然而，在大批量更新数据结构时，我们可以并行计算更新，并且还可以利用每个单独状态值更改时必须更新的部分之间的任何重叠。Aptos 区块链故意只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，网络发出状态检查点事务，其中包括全局状态验证器作为其输出的一部分。这样的版本被称为状态检查点。两个检查点之间的差距越大，每笔交易更新经过状态验证的数据结构的摊销成本就越低。

使用状态检查点，可以以一种无需信任的方式从中读取任何状态值，而无需存储所有全局状态。这种能力对于增量状态同步、跨验证器的分片存储、无状态验证器节点和存储受限的轻客户端等应用程序非常有用。

但是，由于状态检查点是周期性的，因此要获得特定版本的账本状态的证明，需要为丢失的状态交替执行额外的交易，或者从经过身份验证的账本历史中获得包含它们的证明。

状态检查点与分类账历史中的特定交易版本相关联，因此与第 7 节中提到的与交易批次相关的时间戳绑定。通过时间戳，轻客户端可以了解已证明状态值的新近度。如果没有时间戳，轻量级客户端证明只能确保可能是很久以前的先前状态的有效性，这几乎不能保证相关性。此外，状态证明的时间戳对于跟踪历史访问和审计目的是必要的，例如计算代币储备中代币的平均小时余额。

状态检查点可以基于先前的状态检查点和之后的交易输出中的状态变化来派生。因此，将状态检查点持久化到稳定存储并不需要处于事务处理的关键路径上。此外，在持久化状态检查点时也存在有益的批处理效果。在内存中缓存最近的状态检查点（或者更确切地说是它们之间的增量）并仅将周期性状态检查点转储到稳定存储中可以大大减少存储带宽的消耗。选择持久化检查点的方式不会影响经过身份验证的数据结构的计算。因此，这是每个节点的选择：节点运营商可以在内存容量和存储带宽之间进行适当的权衡。

8 状态同步

Aptos 区块链旨在为生态系统中的所有参与者提供一个高吞吐量、低延迟的系统。因此，区块链必须提供一个有效的状态同步协议来传播、验证和持久化区块链数据到轻客户端、全节点和验证者 [14]。此外，同步协议还必须能够容忍网络内的资源限制和

异构性，考虑到不同的用户和用例。例如，它必须允许归档完整节点验证和保存整个区块链历史和状态，同时还允许轻客户端有效地跟踪 Aptos 区块链状态的一小部分。

为了实现这一特性，Aptos 区块链利用验证器、完整节点和其他复制器提供的经过验证的账本历史记录和经过验证的状态证明（参见第 7.6.1 节）来提供灵活且可配置的同步协议。具体来说，网络中的参与者可以选择不同的同步策略来优化他们的用例和需求。

例如，在全节点的情况下，Aptos 允许多种同步策略，包括处理自时间开始以来的所有交易或完全跳过区块链历史并使用航点仅同步最新区块链状态的能力。在轻客户端的情况下，策略包括同步部分区块链状态，例如特定帐户或数据值，并启用验证状态读取，例如验证帐户余额获取。在所有情况下，Aptos 都允许参与者配置数据的数量和年龄以获取、处理和保留。

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需求，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 区块链将由一个广泛而多样化的社区拥有、运营和管理。原生 Aptos 代币将用于交易和网络费用、协议升级和链上/链下流程的治理投票，以及通过权益证明模型保护区块链。 Aptos 代币经济学的完整描述将在未来的出版物中发布。

9.1 交易和网络费用

所有 Aptos 交易都有一个 gas 单价（在 Aptos 代币中指定），允许验证者优先考虑网络中最高价值的交易。此外，在流水线模型的每个阶段，都有多个丢弃低价值交易的机会（允许区块链在系统容量最大的情况下高效运行）。随着时间的推移，将部署网络费用，以确保使用 Aptos 区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在计算、存储和网络之间进行不同成本权衡的应用程序。

9.2 网络治理

Aptos 区块链上的每一项重大功能更改和改进都将经历几个阶段，包括提案、实施、测试和部署。这种结构为相关方和利益相关者创造了提供反馈、分享关注和提出建议的机会。最后阶段，部署，通常分两步完成。首先，具有新功能的软件版本将部署到每个节点，其次，将启用该功能，例如，通过功能标志或链上配置变量。

节点运营商部署的每个软件都必须向后兼容，以确保新软件与支持的版本可互操作。部署新软件版本的过程可能会持续数天，以解决不同时区的运营商和任何外部问题。一旦升级了足够数量的节点，新功能的启用可以由同步点触发，例如商定的块

高度或时代变化。在紧急情况下（例如，当停机时间不可避免时），可以通过节点运营商的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分叉来启用。

与其他区块链相比，Aptos 区块链在链上对其配置进行编码。每个验证者都能够与区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，共识协议和 Aptos 框架版本）。由于此功能，Aptos 区块链中的升级是无缝且即时的。

为了为启用过程提供灵活性和可配置性，Aptos 区块链将支持链上治理，代币持有者可以就其质押的代币权重进行投票。链上投票协议是公开的、可验证的并且可以是即时的。链上治理还可以支持在没有软件部署的情况下实现非二元结果。例如，链上领导者选举协议参数可以通过链上治理进行修改，而预先知道的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，链上治理可以部署在整个升级管理过程中。举个例子：

1. 代币持有者在链上投票决定过渡到新的抗量子签名方案。

2. 开发者实施并验证新的签名方案并创建新的软件版本。

3. 验证者将他们的软件升级到新版本。

4. 代币持有者在链上投票启用新的签名方案，链上配置更新，变更生效。

作为一个开源项目，Aptos 区块链将依赖强大的社区反馈，并使用链上治理来管理适当的流程。在某些情况下，可能仍需要启用链下升级，但随着时间的推移将最小化。

9.3 权益证明共识

要参与 Aptos 区块链上的交易验证，验证者必须拥有最低要求数量的质押 Aptos 代币。质押数量成比例地影响交易传播期间的 2f + 1 质押加权 PoAv，以及块元数据排序期间的投票权重和领导者选择。验证者决定他们自己和各自的质押者之间的奖励分配。质押者可以选择任意数量的验证者来质押他们的代币，以获得预先商定的奖励分配。在每个 epoch 结束时，验证者及其各自的质押者将通过相关的链上 Move 模块获得奖励。

任何拥有足够权益的验证者运营商都可以自由加入 Aptos 区块链。所有参数，包括所需的最低权益，都可以通过第 9.2 节中描述的链上启用过程来设置。

10 性能

如第 7 节所述，Aptos 区块链能够通过其并行、批量优化和模块化的事务处理管道实现最佳吞吐量和硬件效率。其他性能举措，例如共识升级、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增加吞吐量并提高效率。

今天，区块链吞吐量通常以每秒交易量来衡量。然而，考虑到跨交易和基础设施的成本和复杂性范围广泛，这是比较系统的一种不精确的方法。交易延迟也同样存在缺陷，因为提交到最终性的起点和终点在实验中有所不同。

此外，一些系统需要交易输入和输出的先验知识，并强制将逻辑交易拆分为更小、更简单的交易。拆分事务会导致糟糕的用户体验并人为地影响延迟和吞吐量，而不考虑开发人员试图完成的任务。相比之下，Aptos 方法使开发人员能够自由地无限制地构建，并根据实际用例而不是合成事务来衡量吞吐量和延迟。

Aptos 区块链将继续优化单个验证器的性能，并尝试扩展技术以将更多验证器添加到网络中。两个方向都有不同的权衡。任何具有并行执行能力的区块链都可以通过需要更强大的硬件甚至将每个验证器构建为单个机器的集群来支持额外的并发性。但是，全局验证器的数量存在实际限制，这与验证器运营商的成本和复杂性相称。云服务中无服务器数据库的兴起和流行说明了很少有实体可以有效地部署和维护这些类型的复杂分布式系统。

10.1 同构状态分片

最初，Aptos 区块链将以单一分类帐状态启动。随着时间的推移，Aptos 网络将采用独特的方法来实现水平可扩展性，同时仍保持去中心化。这将通过多个分片账本状态发生，每个状态都提供同质的 API 和分片作为一流的概念。 Aptos 代币将用于所有分片的交易费用、质押和治理。

数据可以通过同构桥在分片之间传输。用户和开发者可以根据自己的需要选择自己的分片方案。例如，开发人员可以提出新的分片或在现有分片内集群用户以实现高分片内连接。此外，分片可能具有不同的系统特征。一个分片可以使用 SSD 进行计算优化，而另一个分片可以针对具有低计算特性的大型硬盘驱动器进行优化。通过在不同分片之间提供硬件灵活性，开发人员可以为他们的应用程序利用适当的系统特性。

总之，同构状态分片提供了横向吞吐量可扩展性的潜力，允许开发人员使用跨分片的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分片数据。这提供了显着的性能优势以及单个统一 Move 智能合约平台的简单性。

References

[1] "Aptos-core," 2022. [Online]. Available: https://github.com/aptos-labs/aptos-core

[2] "Move," 2022. [Online]. Available: https://github.com/move-language/move

[3] D. Matsuoka, C. Dixon, E. Lazzarin, and R. Hackett. (2022) Introducing the 2022 state of crypto report. [Online]. Available: 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. Chalkias, E. Cheng, A. Ching, A. Chursin, G. Danezis, G. D. Giacomo, D. L. Dill, H. Ding, N. Doudchenko, V. Gao, Z. Gao, F. Garillot, 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. Pott, B. Proctor, S. Qadeer, Rain, D. Russi, B. Schwab, S. Sezer, A. Sonnino, H. Venter, L. Wei, N. Wernerfelt, B. Williams, Q. Wu, X. Yan, T. Zakian, and R. Zhou, "The libra blockchain," 2019. [Online]. Available: https://developers.diem.com/papers/the-diem-blockchain/2020-05-26.pdf

[5] S. Blackshear, E. Cheng, D. L. Dill, V. Gao, B. Maurer, T. Nowacki, A. Pott, S. Qadeer, D. R. Rain, S. Sezer, T. Zakian, and R. Zhou, "Move: A language with programmable resources," 2019. [Online]. Available: https://developers.diem.com/papers/diem-move-a-language-with-programmable- resources/2019-06-18.pdf

[6] D. Dill, W. Grieskamp, J. Park, S. Qadeer, M. Xu, and E. Zhong, "Fast and reliable formal verification of smart contracts with the move prover," in 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) Lost passwords lock millionaires out of their bitcoin fortunes. [Online]. Available:
https://www.nytimes.com/2021/01/12/technology/bitcoin- passwords- wallets-fortunes.html

[8] The Diem Team, "State synchronization and verification of committed information in a system with reconfigurations," 2020. [Online]. Available: 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, and A. Spiegelman, "Narwhal and tusk: A dag-based mempool and efficient bft consensus," in Proceedings of the Seventeenth European Conference on Computer Systems, ser. EuroSys '22. New York, NY, USA: Association for Computing Machinery, 2022, p. 34–50. [Online]. Available: https://doi.org/10.1145/3492321.3519594

[10] The Diem Team, "Diembft v4: State machine replication in the diem blockchain," 2021. [Online]. Available: https://developers.diem.com/papers/diem-consensus-state-machine-replication-in-the-diem- blockchain/2021-08-17.pdf

[11] S. Cohen, R. Gelashvili, L. Kokoris-Kogias, Z. Li, D. Malkhi, A. Sonnino, and A. Spiegelman, "Be aware of your leaders," CoRR, vol. abs/2110.00960, 2021. [Online]. Available: https://arxiv.org/abs/2110.00960

[12] A. Spiegelman, N. Giridharan, A. Sonnino, and L. Kokoris-Kogias, "Bullshark: Dag bft protocols made practical," in Proceedings of the 20th Conference on Computer and Communications Security (CCS), ser. CCS '22. Los Angeles, CA, USA: Association for Computing Machinery, 2022.

[13] R. Gelashvili, A. Spiegelman, Z. Xiang, G. Danezis, Z. Li, Y. Xia, R. Zhou, and D. Malkhi, "Block-stm: Scaling blockchain execution by turning ordering curse to a performance blessing," 2022. [Online]. Available: https://arxiv.org/abs/2203.06871

[14] J. Lind, "The evolution of state sync: The path to 100k+ transactions per second with sub-second latency at aptos," 2022. [Online]. Available: https://medium.com/aptoslabs/52e25a2c6f10