**SlimChain: Scaling Blockchain Transactions through Off-ChainStorage and Parallel Processing**

**Abstract**

区块链技术已成为许多分散应用程序的基石，这些应用程序在其他不受信任的对等方之间运行。然而，众所周知，现有的区块链系统不能很好地扩展。事务通常是按顺序执行和提交的，以保持对总订单的相同视图。此外，为了保证完整性，有必要在区块链网络的每个节点中复制交易数据及其执行情况。这种存储和计算需求给区块链系统带来了巨大的负担，不仅限制了系统的可扩展性，还通过使网络更加集中，破坏了系统的安全性和健壮性。

为了解决这些问题，在本文中，我们提出SlimChain，这是一种新型的区块链系统，通过**链外存储和并行处理扩展交易**。SlimChain提倡无状态设计，**只维护链上分类账状态的短期承诺**，同时将事务执行和数据存储专用于链外节点。为了实现SlimChain，我们提出了链外智能合约执行、链内交易验证和状态承诺的新方案。我们还提出了减少网络传输的优化方案，以及进一步提高系统可扩展性的新分片技术。通过大量实验验证了所提出的SlimChain系统的性能。与现有系统相比，SlimChain减少了97%~99%的链上存储需求，同时也将峰值吞吐量提高了1.4X~15.6X。

**Introduction**

区块链是一种新兴技术，被认为有可能彻底改变数据库系统。它一直是金融、医疗和供应链等广泛领域中许多分布式应用的基石。在底层，区块链是一种安全的仅加(append-only)数据结构，它建立在P2P 网络中一组不受信任的节点共识的外来交易(incoming transactions)之上。它利用密码签名、散列链和共识协议在互不信任的对等体的基础上创建可信的分布式系统。比特币等第一代区块链系统是专门为支持加密货币而构建的，而以太坊等第二代区块链则扩展了它们的功能，以支持所谓的智能合约中的通用交易。智能合约是用户定义的、可信任的程序，允许用户处理区块链中的数据。它们可以部署在区块链中，并由未来的交易触发执行。

为了保持相同的交易总顺序、确保执行完整性并支持数据追踪(data provenance)，现有的区块链系统通常要求网络中的每个节点都保持对交易历史和账本状态的完整复制。然而，这些不断增长的数据结构在一段时间后变得太大了。例如，截至2021年7月，比特币的整个区块链账本约为350GB，以太坊已超过7TB。为了缓解这个问题，大多数区块链节点通常**只维护一个称为“验证状态”(validation states)的紧凑索引**，该索引大大小于整个分类账本，足以确定交易的有效性。但是，验证状态(validation states)仍然以GB为单位（例如，以太坊的验证状态约为870GB大小）。此外，区块链节点需要根据复制的状态在本地重放所有事务。这种繁琐的有状态数据带来了巨大的存储和计算成本，限制了系统的可扩展性。此外，由于能够处理如此大量数据的节点越来越少，网络变得更加集中，从而破坏了系统的安全性和鲁棒性。

在尝试解决这些问题时，很明显，要求每个区块链节点保留相同的数据副本并重复相同的事务执行是对存储和计算资源的巨大浪费。一种解决方案是**分片(sharding)**。分片将区块链水平划分为多个平行链，**每个平行链仅由节点的一个子集管理**。这是减少不同分片之间存储和计算重复的有效手段。然而，显而易见的是，这个操作只能通过一个常数因子（即分片数量）来缓解问题。在每个分片内，区块链节点仍然需要复制存储和计算。此外，现有的分片解决方案经常会引入新的问题，例如跨分片事务处理(cross-shard transaction processing，*跨分片的事务一致性难以保证*)和慢适应拜占庭对手的攻击(attacks by slowly-adaptive Byzantine adversaries)。另一种解决方案是使用**轻节点(light nodes)**。与存储完整状态的完整节点不同，**轻节点仅保留块头**。然而，虽然轻节点能够遵循共识协议，但它们本身无法验证交易的有效性。因此，使用轻节点并不能解决由于状态维护负担而导致的中心化问题。

最近，有人提出了**无状态区块链**的概念。这是一种**链下(off-chain,** *较不重要的交易活动可以在链下 (在单独的私有通道中)处理，并在稍后的时间最终在链上解决***)**扩展方法，将账本状态和交易执行从链下转移到节点的子集，从而减少链上负载。然而，现有的无状态区块链系统是专门为加密货币设计的，尝试开发一个支持智能合约的通用无状态区块链带来了几个新的挑战：

首先，一个基本问题是，在一般应用中，智能合约支持的交易可能包含任意逻辑。这就需要新的证明技术来证明链下执行的完整性。

其次，智能合约交易会引入任意大小的读写集。这比简单的加密货币交易要复杂得多，并且需要额外的设计来支持链上承诺更新。

第三，现有研究中提出的加密货币交换方法对并行交易执行的支持非常有限。为了提高系统吞吐量，尽管采用了无状态设计，但仍需要新的事务处理方法来支持验证和提交来自异步网络的并发事务。

为了应对这些挑战，在本文中，我们介绍了SlimChain，这是一种无状态区块链系统，可通过**链下存储**和**并行处理**来扩展交易。主要思想是利用链下存储节点来存储账本状态并模拟智能合约的执行，让区块链只维护账本状态的短期承诺。

我们首先为存储节点设计一个可验证的事务执行算法。它并行执行链下交易并计算一些辅助信息以证明执行的完整性以及促进链上共识节点的后续交易承诺。

然后，我们开发了一种新的链上临时状态，它提供了最少但足够的信息来支持无状态链上事务验证、并发控制和承诺。

特别地，提出了一种新颖的部分Merkle-trie结构，使无状态共识节点能够在不使用完整Merkle-trie的情况下保持链上状态的根(root)。

由此，我们的系统可以处理从异步网络提交的并行交易，即使它们以任意顺序到达；并确保链上数据可以在区块链网络中的所有节点之间共同维护和完美同步。

由于网络层通常是区块链系统中的瓶颈，我们还为SlimChain提出了一些优化，以减少节点同步期间的网络传输。为了进一步提高SlimChain的可扩展性，我们提出了一种**全新的方式来支持无状态下的分片**，解决了现有分片解决方案的许多缺点。我们的方法独立于协商一致的协议，可以在无许可和许可环境下使用。综上所述，本文的贡献如下：

介绍了SlimChain，一种新颖的无状态区块链系统，用于具有智能合约功能的可扩展交易处理。据我们所知，这在各类文献中尚属首次出现。

提出了新的链下智能合约执行、无状态链上交易验证和新的状态承诺方案，以在无许可和按任务设置中实现 SlimChain。

进一步提出了在节点同步期间减少网络传输的优化技术和一种新颖的存储分片技术，以进一步提高系统的可扩展性。

构建了一个端到端的原型并进行了广泛的实验来验证SlimChain 系统的性能。与现有系统相比，SlimChain 将链上存储需求降低了97% ~ 99%，同时还将峰值吞吐量提高了1.4 ~ 15.6 倍。

本文的其余部分安排如下。在第2节中介绍了背景和相关工作。第3节给出了SlimChain 的系统概述。第4节介绍了我们设计中的事务处理和分片技术。第5节和第6节分别介绍了实现细节和实验结果。最后，在第7节结束我们的论文。

**BACKGROUND AND RELATED WORK**

在本节中，我们将介绍一些介绍 SlimChain 的必要背景，包括密码学预备知识、区块链基础知识和并发控制方法。我们还提供了对相关区块链系统的回顾，并强调了 SlimChain 的新颖性。

**2.1 Cryptographic Preliminaries**

**加密哈希函数**。加密散列函数ℎ(·) 接受任意长度的字符串作为其输入，并返回一个固定长度的位字符串。它是抗碰撞的，很难找到两个不同的消息m1和m2使得h(m1) = h(m2)。经典的加密散列函数包括 SHA-2 和 SHA-3 系列。

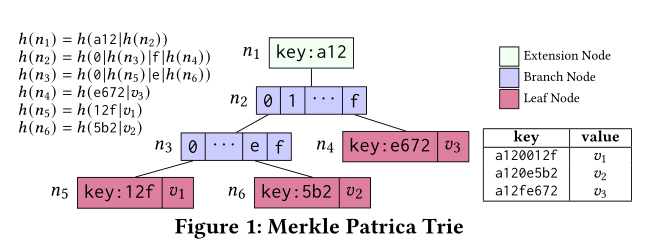
**Merkle Hash Tree（MHT）**是一种已认证的数据结构(*Merkle树是自底向上构建的*，*非叶子节点的value是根据它下面所有的叶子节点值，然后按照一定的算法计算而得出的*)，用于存储键值对(k1,v1), 它支持对数复杂度的成员资格检查。MHT 将数据存储在索引树中，其中每个节点都根据其散列数据值及其子节点分配一个摘要。MHT的root摘要可用于验证所有底层数据。在验证过程中，可以重构root摘要并将其与发布的值进行比较。（*在比特币网络中， Merkle树被用来归纳一个区块中的所有交易， 同时生成整个交易集合的数字指纹， 且提供了一种校验区块是否存在某交易的有效效途径*。）

在区块链系统中，经常使用 trie 作为索引树结构，以降低存储成本。如图1所示，trie中存在三种节点：

(1)扩展节点(extension node)，存储搜索键的切片和子节点；

(2)分支节点(branch node)，分支出固定数量的子节点；

(3)叶节点(leaf node)，存储剩余的搜索键及其对应的值。



如果在 trie 中不存在搜索键，则将相应的值定义为默认值，例如 0。这使我们能够以紧凑的大小存储具有巨大键空间（例如，地址空间为 2^256）的值。 Merkle Patricia trie 是一个专门的 MHT，它使用前面提到的 trie 作为索引结构。为简洁起见，以下我们将 Merkle Patricia trie 称为 Merkle trie。