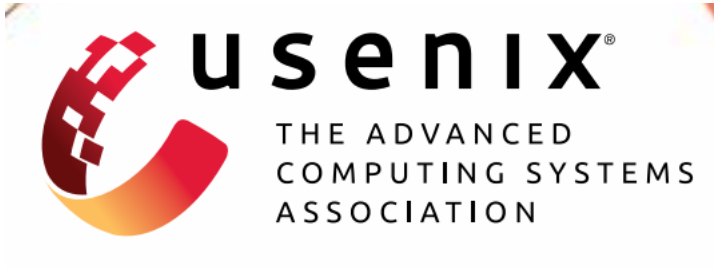


An Off-The-Chain Execution Environment for Scalable Testing and Profiling of Smart Contracts

Yeonsoo Kim and Seongho Jeong, *Yonsei University;* Kamil Jezek, *The University of Sydney;*
Bernd Burgstaller, *Yonsei University;* Bernhard Scholz, *The University of Sydney*

2021 USENIX Annual Technical Conference.

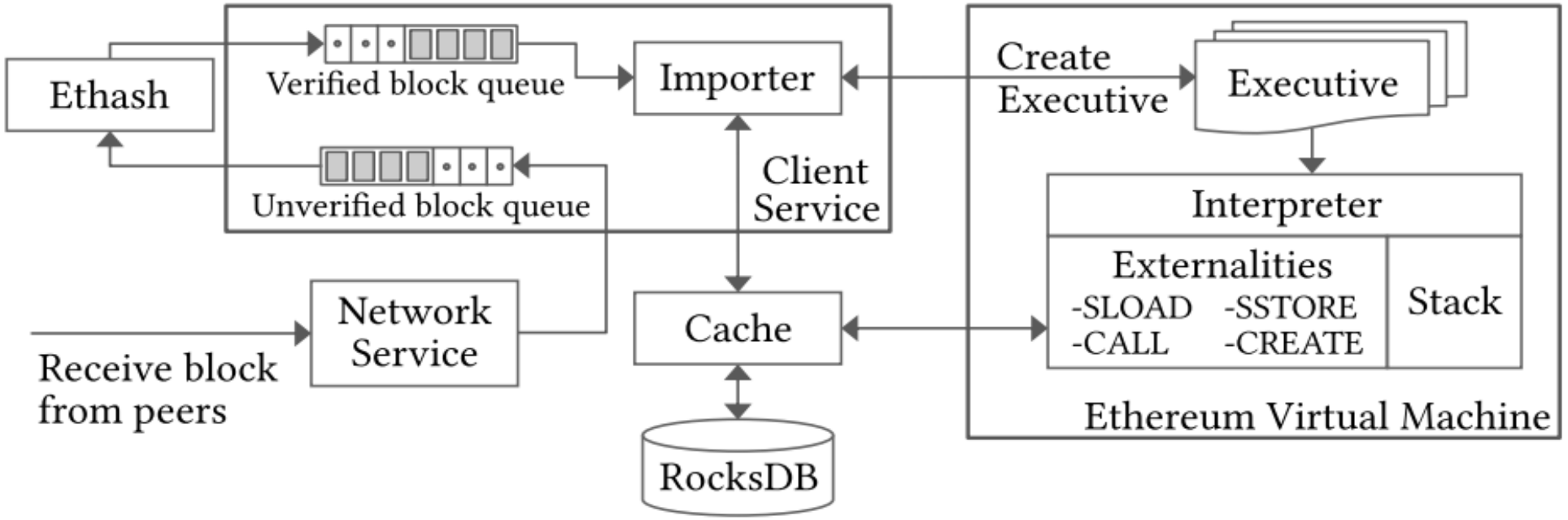
背景

- 区块链
  - 去中心化的P2P网络和数据库
  - 以太坊
    - 运行智能合约的最大的区块链平台
  - 智能合约
    - 运行在以太坊区块链上的程序
  - 账户
    - EOA
    - 合约账户

以太坊客户端

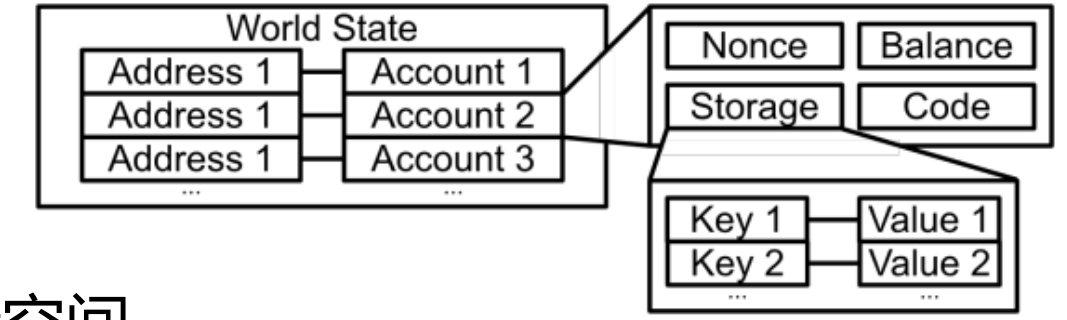
- **常见的客户端实现**
  - Geth（golang实现）
  - Parity（rust实现）

- **客户端处理区块的流程**
  - 从区块链公网上获取区块数据
  - 验证区块和
  - 导入区块进数据库
  - 执行合约

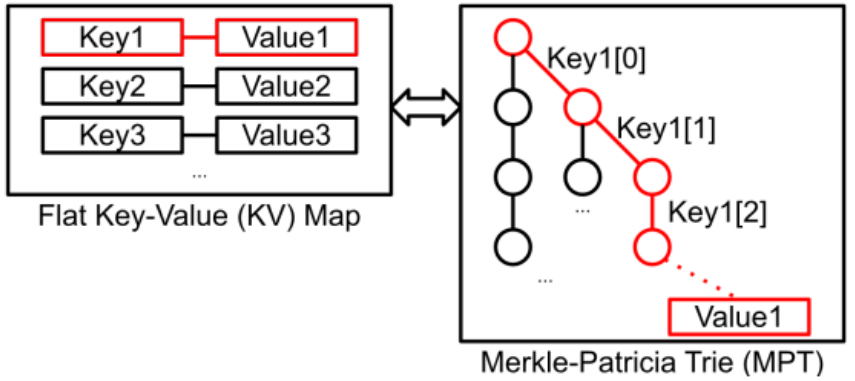


世界状态和默克尔帕米尔树

- 以太坊世界状态
  - 以太坊区块链的全局状态
  - 以太坊通过交易让世界状态转换
  - 合约属于世界状态的同时还有自身的存储空间



- 默克尔帕米尔树（Merkle-Patricia trie，简称MPT）：
  - 检索
    - 前缀树
  - 验证
    - 如果两个MPT相同，他们的根哈希相同

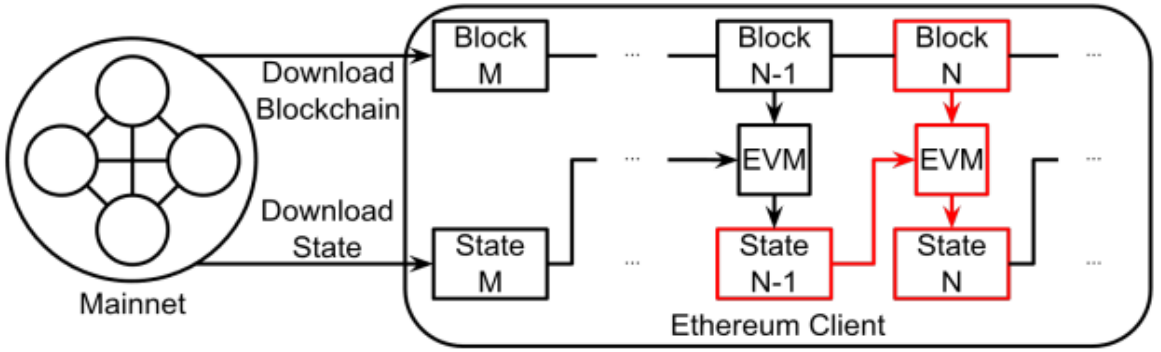


动机

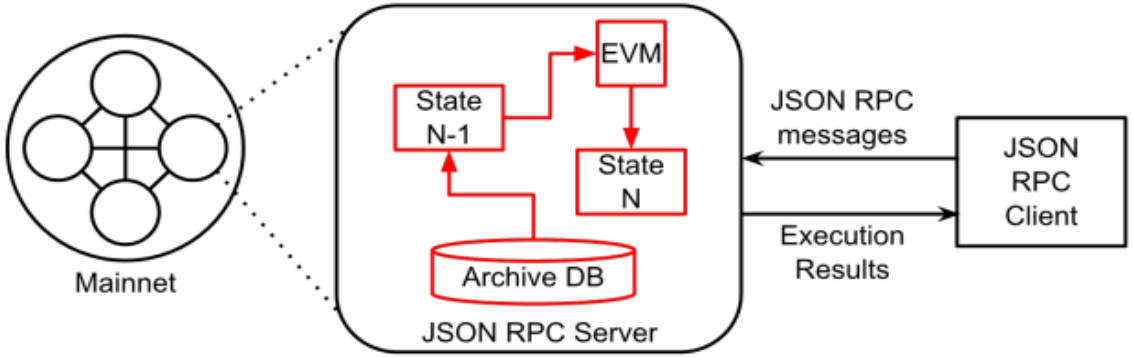
- 大规模的合约测试需求强烈
  - 智能合约控制着大量的金钱
  - 影响整个面向区块链的软件(BOS)生态系统的发展
- **动态分析**是测试方法中的一类最基本的测试方法
  - 取证分析
  - 合约审计
  - 调试合约
- 动态分析通常将**区块链客户端**用作测试环境
  - 客户端不适合用于动态分析

基于客户端的测试

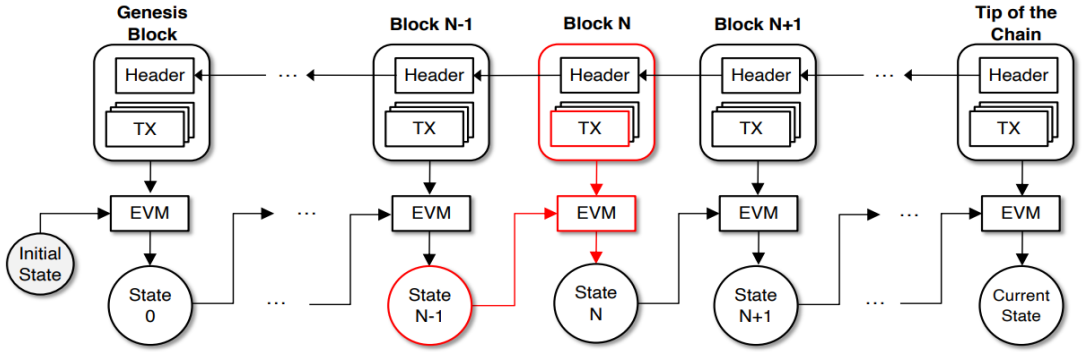
- 全结点
  - 全节点执行所有交易以恢复区块的全部历史
  - 只存储最新的区块状态



- 档案结点
  - 存档节点保存世界状态的整个历史（所有区块的状态）
  - 通过JSON RPC API进行交易的执行

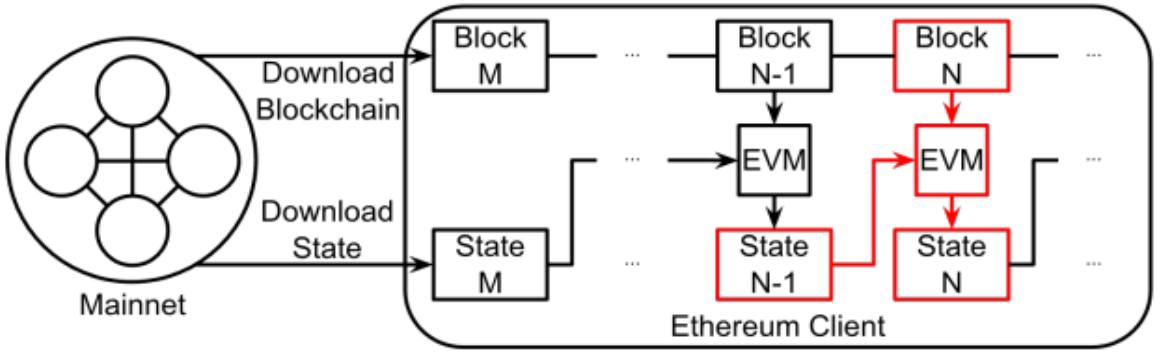


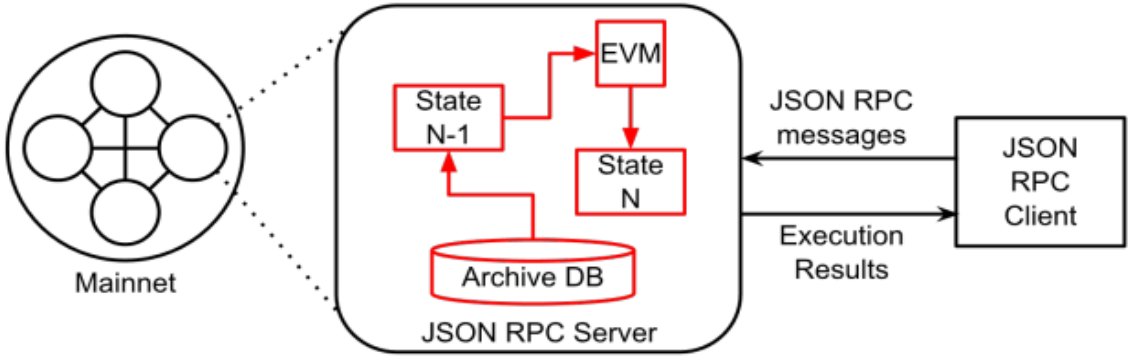
- 交易重放
  - 在交易对应的状态执行交易

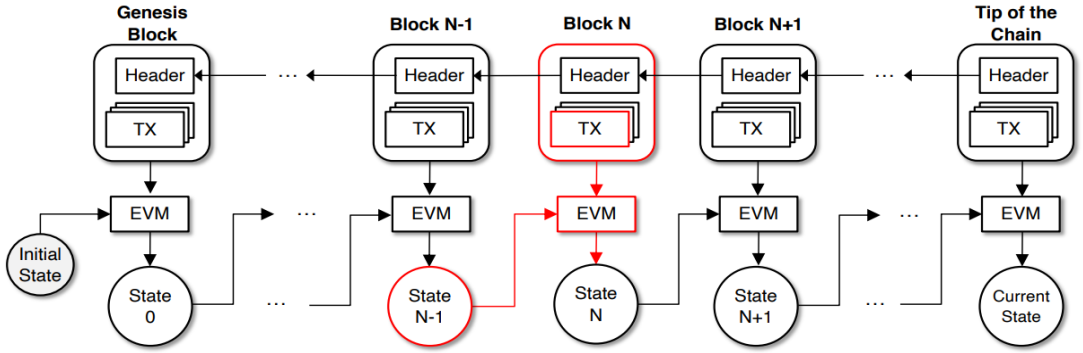


问题

- **全节点的不足之处**
  - 耗时长
  - 需要维护最新高度的区块链的状态消耗磁盘
  - 不友好
- **档案节点的限制**
  - 消耗时间和磁盘空间
    - 存储11.5M区块数据需要6TB空间
    - 使用ssd同步也需要数月时间
  - 非常低的事务重放吞吐量
    - 19.4 tx/s在9M块
- **交易重放**
  - 对智能合约的审计、测试、分析和调试重要
  - 挑战:获取事务最初执行的历史状态







高效的重放的交易一个未解决的问题

``EVM is a single-threaded machine that cannot run transactions in parallel. ... This maybe a big problem for people who have a higher requirement on the timely reaction and verification of their transactions.''

[50] Weiqin Zou, et al. ``Smart Contract Development: Challenges and Opportunities.'' IEEE Transactions on Software Engineering, 2019.

Developers suffering from *``Reliability of and the lack of good development tools. Like testing frameworks, and the difficulty of debugging.''*

Developers need *``Easy way of forking mainnets for testing purposes, ...''*

[5] Amiangshu Bosu, et al. ``Understanding the motivations, challenges and needs of Blockchain software developers: a survey.'' Empirical Software Engineering, 24(4):2636–2673, 2019.

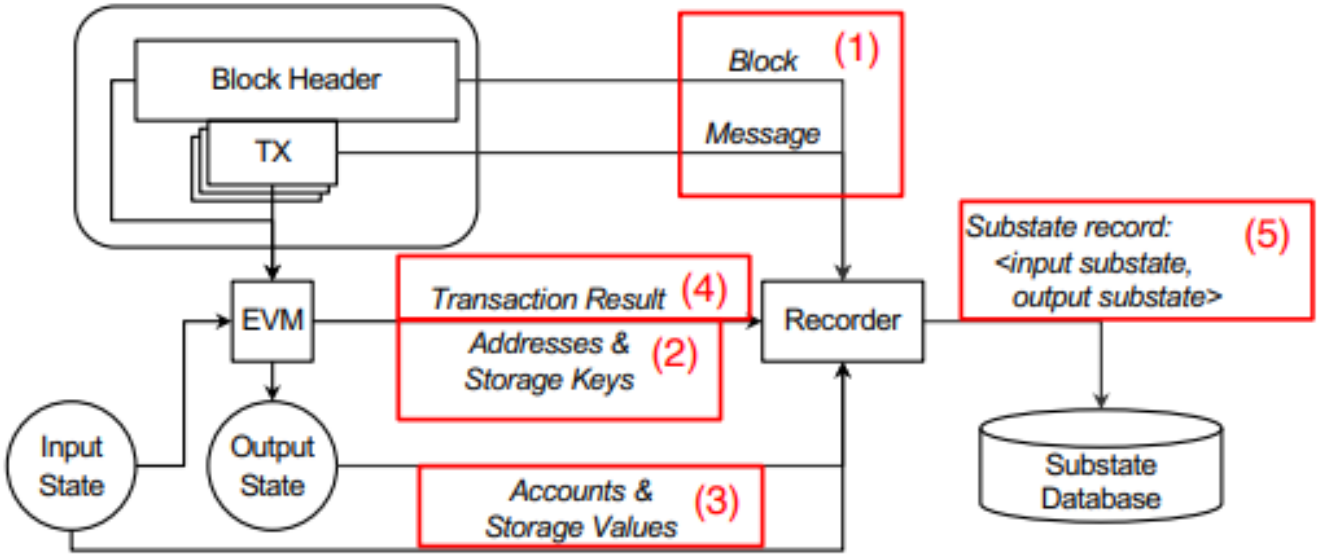
贡献

- **提供了一个基于以太坊的测试环境**
  - **目的：快速执行区块链历史交易**
- **主要贡献**
  - **交易记录和重放机制**
    - 交易细粒度的重放
    - 并发
    - 不依赖于链上环境
  - **将以太坊世界状态重组为与交易相关的子状态**
    - 节约空间且高保真
  - **Evaluation of the proposed approach through three use cases**
    - 用于硬分叉的测试
    - 死代码分析
    - 智能合约的模糊测试

主要方法

- 记录交易
  - 通过导入从传世区块到链的最新块的所有块。
- 捕获交易所需状态
  - 只需收集重放交易所需的世界状态的子集
  - 不依赖链上的其他交易
  - 子状态存储在子状态数据库中
- 交易重放利用子状态数据库从链外执行
  - 模仿以太坊实现，在没有分布式系统开销的情况下执行交易
  - 没有依赖关系，可以实时、大规模、并发回放历史交易

记录器



记录器捕获交易执行期时所需的子状态：

(1)环境参数(block、message)

(2)访问索引(address、storage keys)

(3)从帐户存储中访问的键值对(input/output tuples)

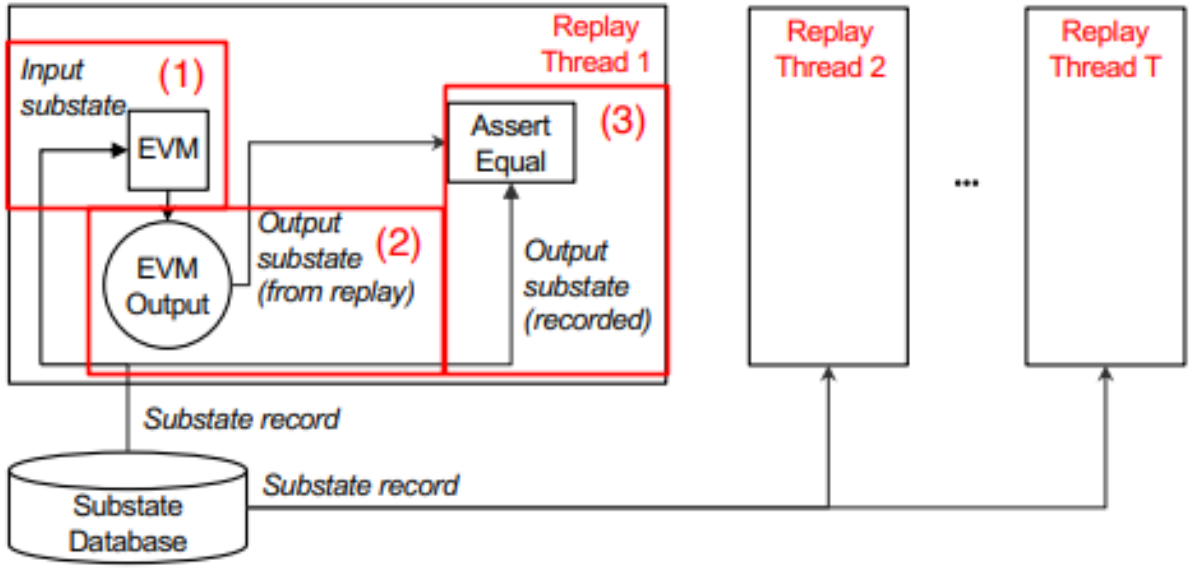
(4)交易结果((status code, logs, gas usage)

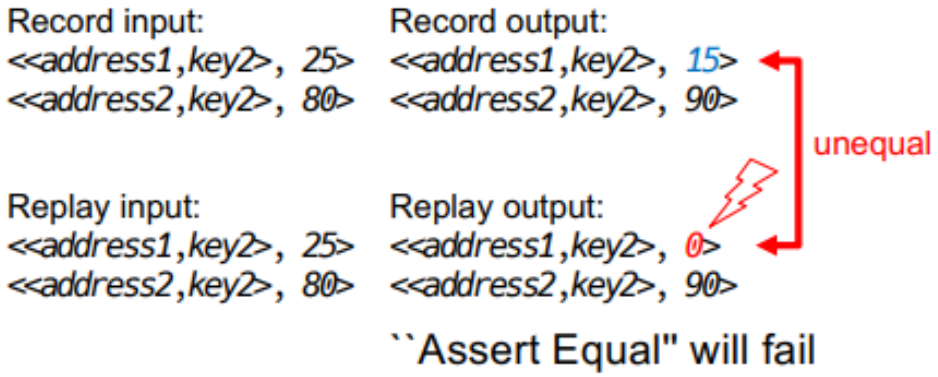
例:在帐户1的 (<address1,key1>)上将10个单位的代币传输到帐户2 (<address2,key2>)

Record input:
<address1,key1>, 25>
<address2,key2>, 80>

Record output:
<address1,key1>, 15>
<address2,key2>, 90>

重放器





重放器：加载子状态并在链外并发大规模的重放交易

(1)将输入子状态复制到内存中的EVM上下文

(2)孤立地执行交易，并像记录器那样捕获其输出子状态

(3)检查交易重放的正确性

- 如果访问的索引或重放的输出与记录的输出不一致，则抛出异常。

实验

Server	Geth full node	Substate replayer	Geth archive node
CPU	Intel Xeon E5-2699 v4, 2.2 GHz to 3.6 GHz, 22 cores × 2 sockets		
RAM	512 GB DDR4 RAM @ 2,400 MHz		
SSDs (PCIe)	Intel Optane 900P 480 GB Intel Optane DC P3700 800 GB		Samsung PM1725b 6.4 TB
OS	CentOS Linux release 7.9.2009 (core), kernel version 4.11.3-1.el7.elrepo.x86_64		
Filesystem	ZFS pool (1.2 TB total size)		XFS

- 实验范围
  - 以太坊区块链的0-9M区块，包含590M个交易
- 实验结果
  - 没有引发异常，即100%重放准确率

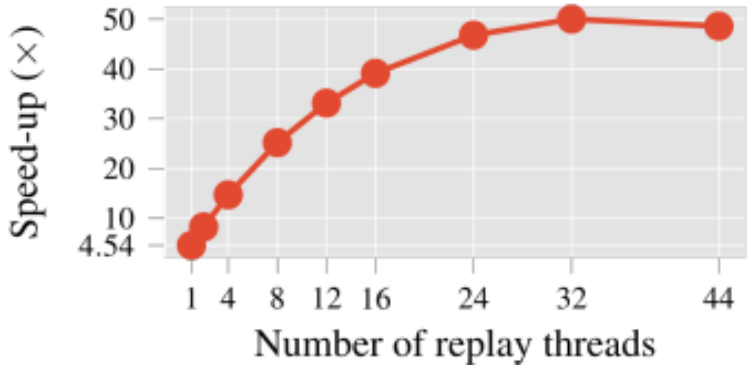
实验

➢ 重放的空间性能

- 与Geth全点相比，子状态数据库节省了59.24%的磁盘空间。

➢ 重放的时间性能

- 比带有单个线程的Geth满节点快4.54倍
- 当扩展到44核时，590M交易的重放在6小时内完成



Blocks (M)	Geth full node (GB)	Substate replayer (GB)	Space savings (%)
0–1	0.96	0.68	29.17
1–2	3.00	1.83	39.00
2–3	29.30	16.91	42.29
3–4	37.76	6.58	82.57
4–5	124.57	39.55	68.25
5–6	272.06	51.58	81.04
6–7	407.48	50.30	87.66
7–8	551.04	55.96	89.84
8–9	700.11	62.00	91.14
0–9	700.11	285.39	59.24

Blocks (M)	Geth full node		Substate replayer		Speed-up (×)
	Time (s)	tx/s	Time (s)	tx/s	
0–1	1184	1414.07	526	3183.01	2.25
1–2	2879	2217.13	1517	4207.73	1.90
2–3	28906	252.73	24125	302.82	1.20
3–4	10775	1946.33	5222	4016.03	2.06
4–5	94868	1196.67	28873	3931.90	3.29
5–6	165673	748.71	35390	3504.97	4.68
6–7	173503	552.82	33672	2848.55	5.15
7–8	224426	485.51	38060	2862.87	5.90
8–9	248519	447.70	41999	2649.17	5.92
0–9	950733	620.62	209384	2817.98	4.54

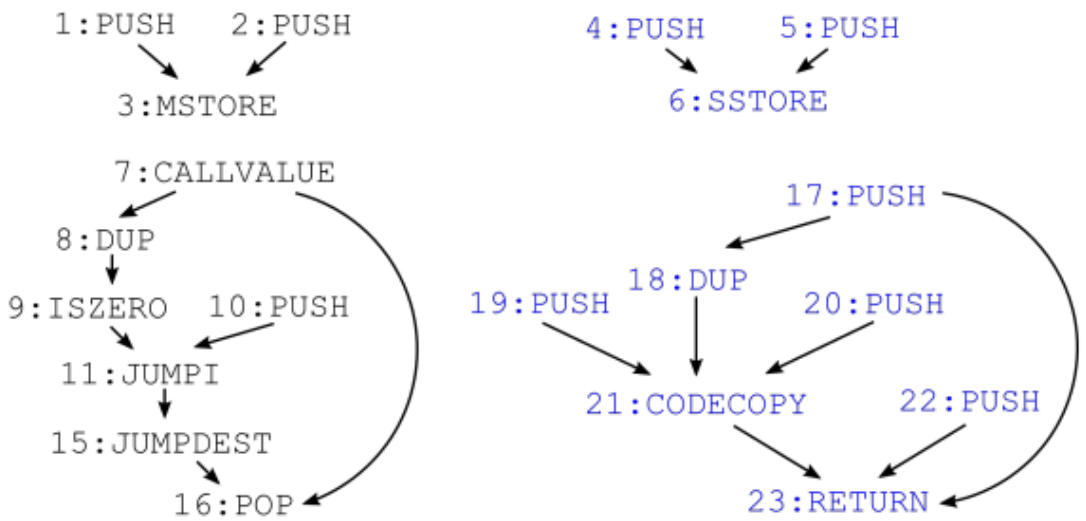
使用案例

- 1) 无用代码评估
- ~~2) 硬分叉测试~~
- ~~3) 模糊测试用例~~

案例1：无用指令评估

- 浪费的指令对区块链的状态没有持久化作用
- 指令必要性在控制流图中的后向传播
  - 6:SSTORE 和 23:RETURN
  - 以及这些指令所有依赖的指令（通过数据流判断）

Nr.	Opcode	Nr.	Opcode
1	PUSH 0x80	13	DUP
2	PUSH 0x40	14	REVERT
3	MSTORE	15	JUMPDEST
4	PUSH 0x00	16	POP
5	PUSH 0x01	17	PUSH 0x3797
6	SSTORE	18	DUP
7	CALLVALUE	19	PUSH 0x25
8	DUP	20	PUSH 0x00
9	ISZERO	21	CODECOPY
10	PUSH 0x15	22	PUSH 0x00
11	JUMPI	23	RETURN
12	PUSH 0x00		



案例1：无用指令评估

- 分析9M块花了75个小时
- 发现区块高度越高，浪费指令率和浪费气体越多.
- 在8-9M区块范围内，50%的指令和25k/block的汽油是浪费的

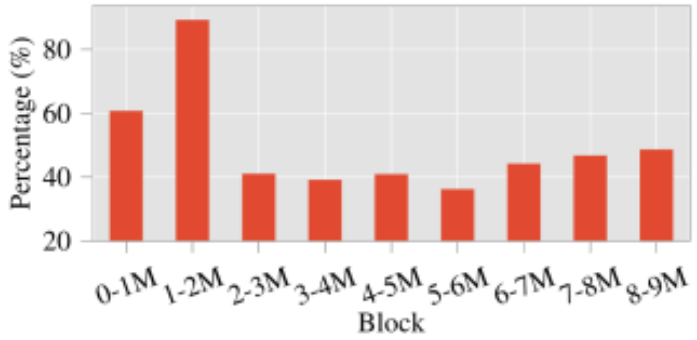


Figure 9: Average ratios of wasteful instructions for ranges of 1 M blocks.

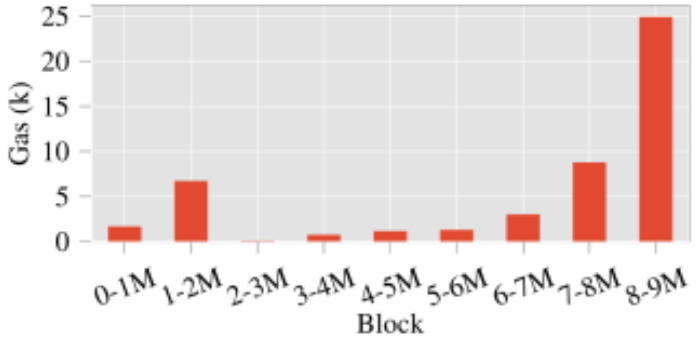


Figure 10: The average of wasted gas per transaction for ranges of 1 M blocks.

结论

- 本工做工作为独立和大规模的智能合约轻量级执行提出了一种新的基础设施。
  - 单线程执行所有可用的智能合约，比Geth全节点快4.54倍。
  - 在多核架构上提供更有效伸缩性，得到50.03倍的性能提升(在44核的情况下)。
- 本工作提出的执行环境非常适合需要快速和重复执行智能合约的场景。
  - 死代码分析评估
  - 智能合约的程序模糊器
  - 硬分叉的评估
- 本工作的基础设施可用于区块链整体，这是以前的方法无法做到的