天津市网络与数据安全技术重点实验室 | 南开大学智能计算系统研究室

高性能智能合约体系架构

——以合约更新与执行之视角

ATOM: Architectural Support and Optimization Mechanism for Smart Contract Fast Update and Execution in Blockchain-based IoT

Tao Li, **Yaozheng Fang**, Zhaolong Jian, Xueshuo Xie, *Ye Lu, Guiling (Grace) Wang

房耀政 2021级博士研究生 | 导师：李涛 教授 | 报告约30分钟 | 2021年11月21日

ATOM: Architectural Support and Optimization Mechanism for Smart Contract Fast Update and Execution in Blockchain-based IoT ǂ

李涛[1,2]
教授

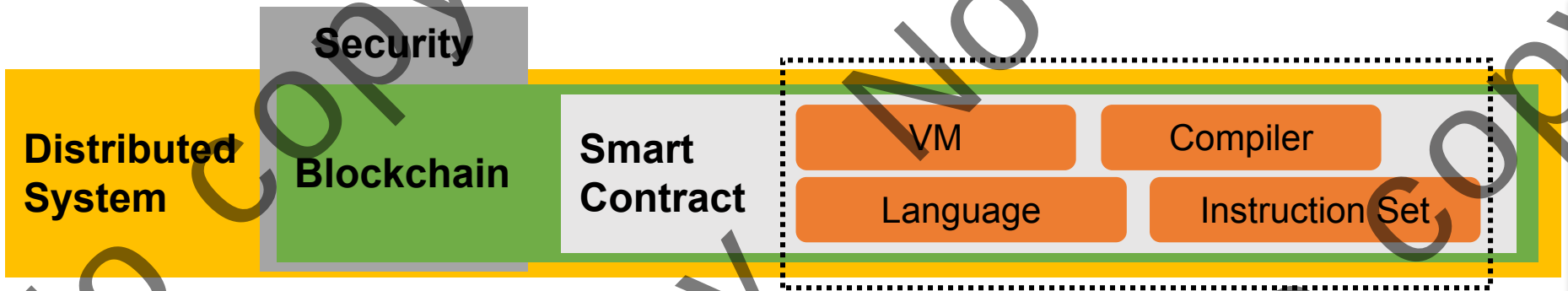
房耀政[1,2]
博士研究生

简兆龙[1,2]
博士研究生

谢学说[1,2]
博士后

卢冶[1,2,*]
副教授

Grace Wang[1,2]
Professor



- 为什么要做ATOM(WHY)
- ATOM是如何做的(HOW)
- ATOM带来什么(WHAT)
- 未来何去何从(WHERE)

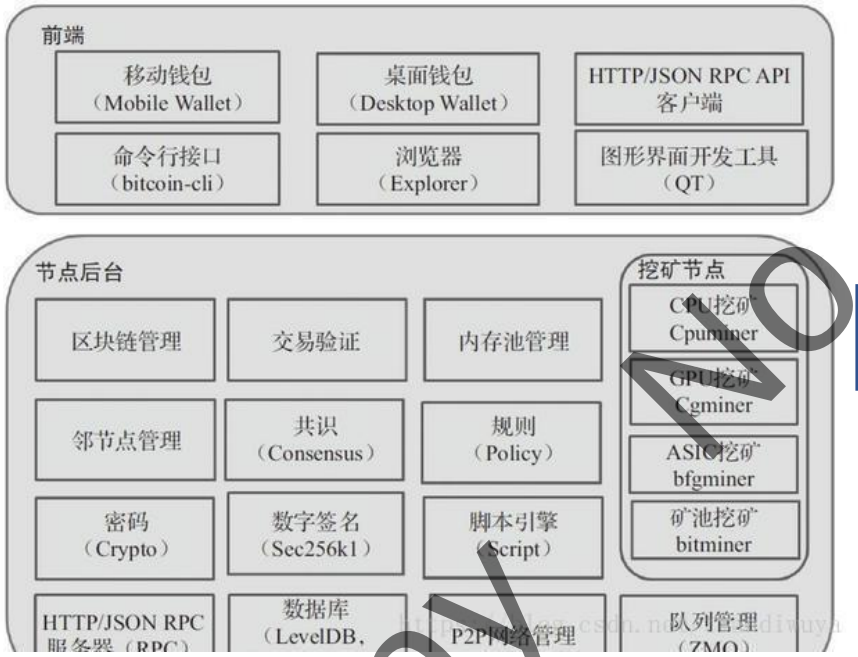
ǂThis paper is published by IEEE Internet of Things Journal (SCI-I, IF: 9.471) on 2021.08.23. DOI: 10.1109/JIOT.2021.3106942

[1]南开大学计算机学院 [2]天津市网络与数据安全技术重点实验室 [3] New Jersey Institute of Technology (USA) *通讯作者（Corresponding Author）

目录

目录

▶ 聚焦在何处？区块链发展脉络（区块链=分布式数据库+状态复制机）



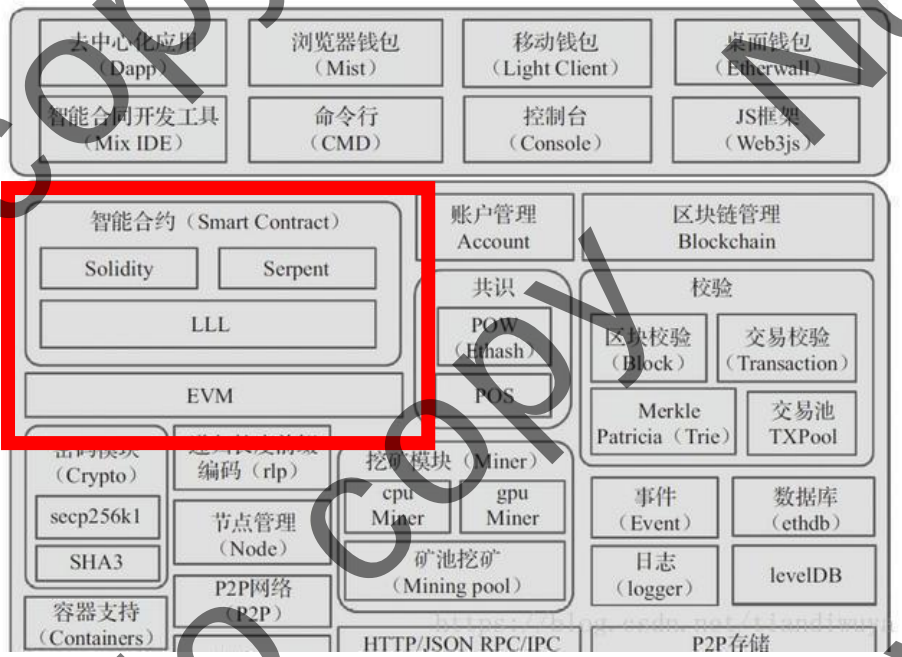
区块链1.0——数字货币
(2008-2014)

区块链2.0——可编程区块链
(2014年起)

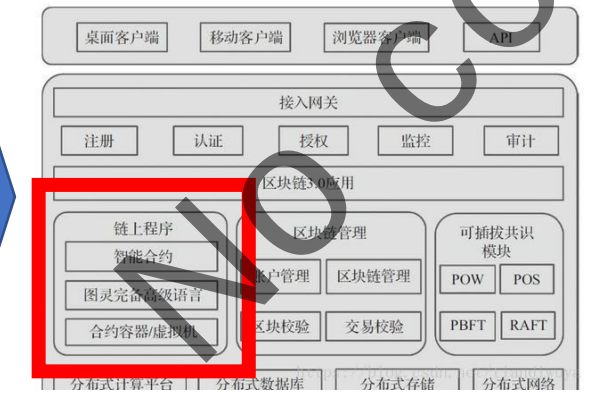
区块链3.0——生态&应用

▶ 聚焦在何处？区块链发展脉络（区块链=分布式数据库+状态复制机）

区块链1.0——数字货币
(2008-2014)



区块链2.0——可编程区块链
(2014年起)



区块链3.0——生态&应用

发展的主线：
存储潜力到计算潜力，
业务+计算+数据=应用雏形。

▶ 链上计算的承载者——智能合约(Smart Contract)

智能合约是存储在区块链上的一段可执行的代码。

▶ 链上计算的承载者——智能合约(Smart Contract)

智能合约是存储在区块链上的一段可执行的代码。

▶ 代码运行需要环境支持，如何满足？

容器

沙箱/沙盒

▶ 链上计算的承载者——智能合约(Smart Contract)

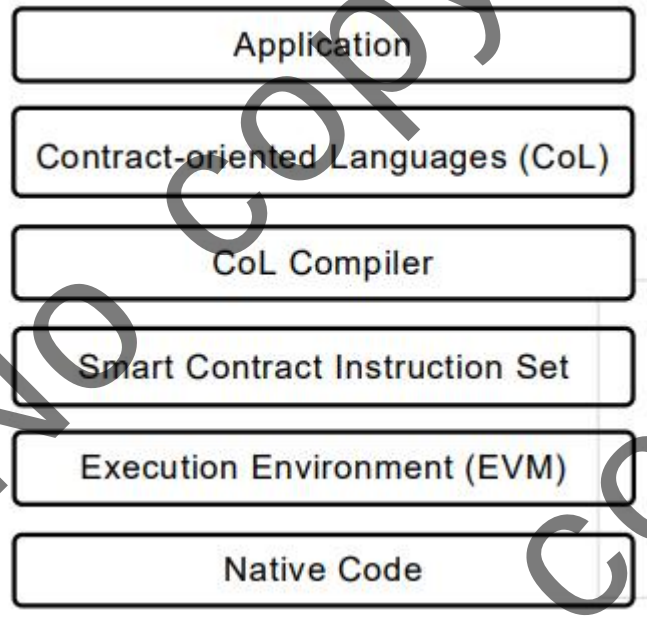
智能合约是存储在区块链上的一段可执行的代码。

▶ 代码运行需要环境支持，如何满足？

容器

沙箱/沙盒

3200+ Dapps on EVM

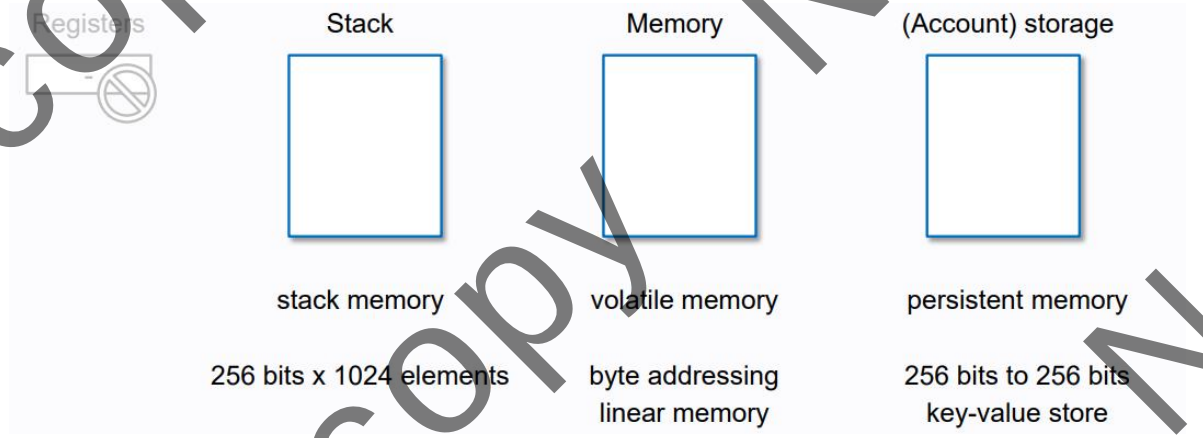


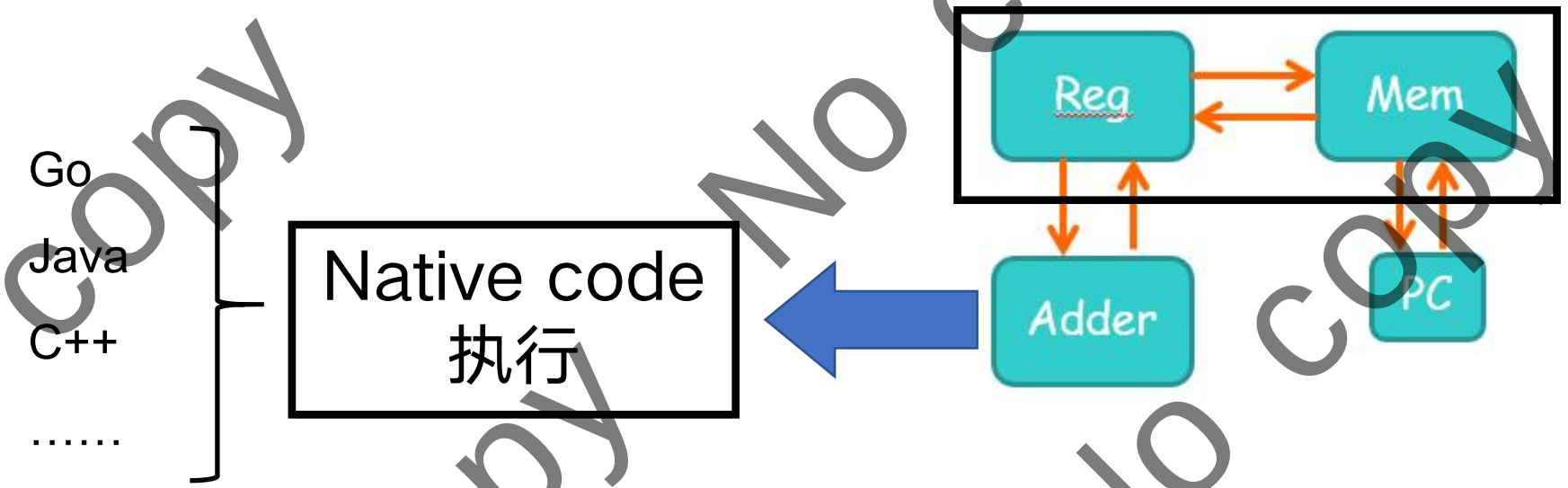
以太坊虚拟机
(Ethereum Virtual Machine)

```
pragma solidity ^0.4.11;

// THIS CONTRACT CONTAINS A BUG - DO NOT USE
contract TxUserWallet {
    address owner;

    function TxUserWallet() public {
        owner = msg.sender;
    }

    function transferTo(address dest, uint amount) public {
        require(tx.origin == owner);
        dest.transfer(amount);
    }
}
```

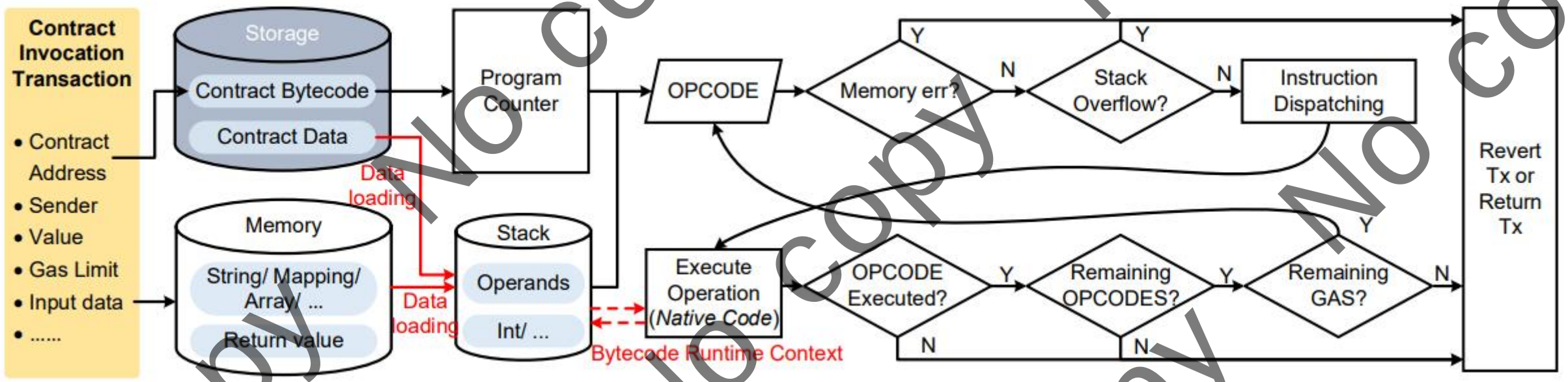
```
005 CALLVALUE
006 DUP1
007 ISZERO
008 PUSH2 0010
011 JUMPI
012 PUSH1 00
014 DUP1
015 REVERT
```

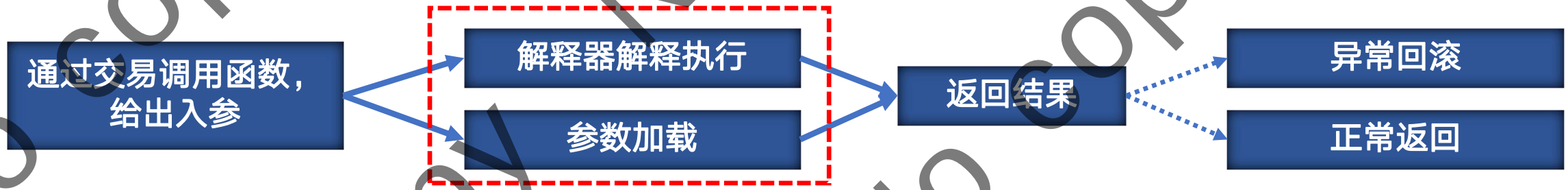
▶ 代码是如何在EVM支持下运行的？——EVM是什么样子？





▶ 代码是如何在EVM支持下运行的？——动态视角

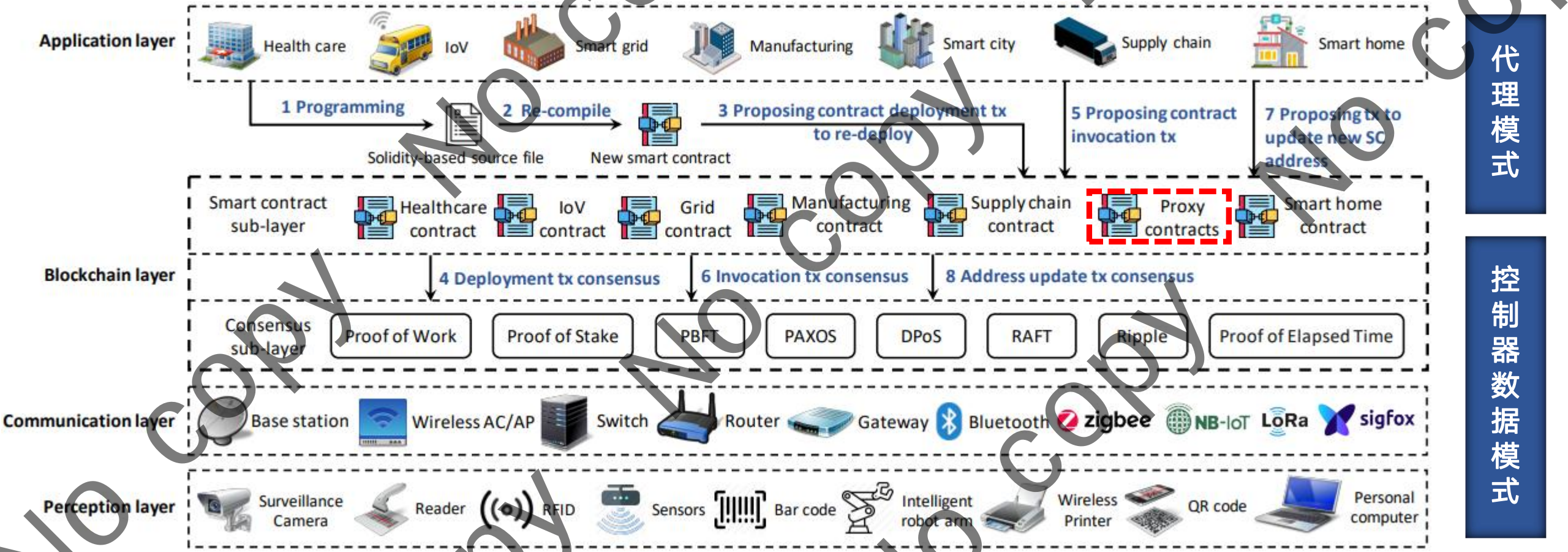




目录

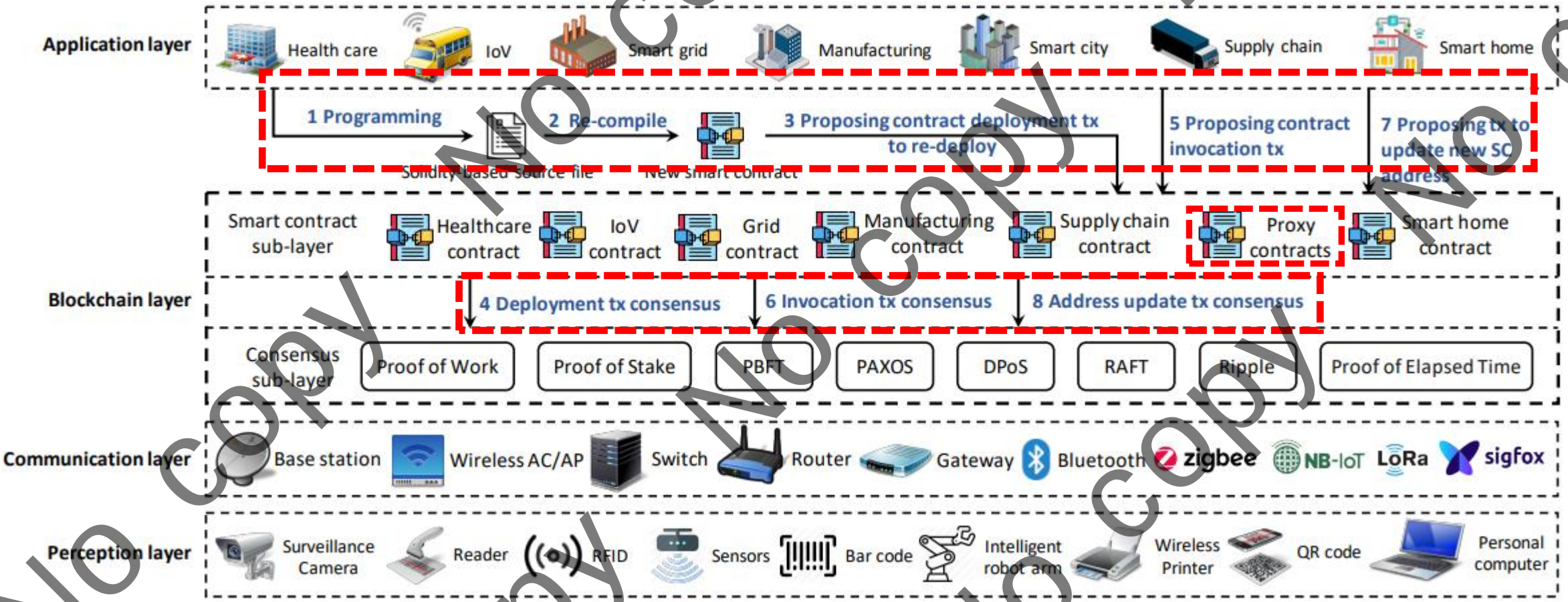
► 智能合约存在的主要问题（e.g., Blockchain based IoT）

代码漏洞、BUG、应用需求导致智能合约不断频繁更新，现有间接更新方法效率低。



► 智能合约存在的主要问题（e.g., Blockchain based IoT）

代码漏洞、BUG、应用需求导致智能合约不断频繁更新，现有间接更新方法效率低。

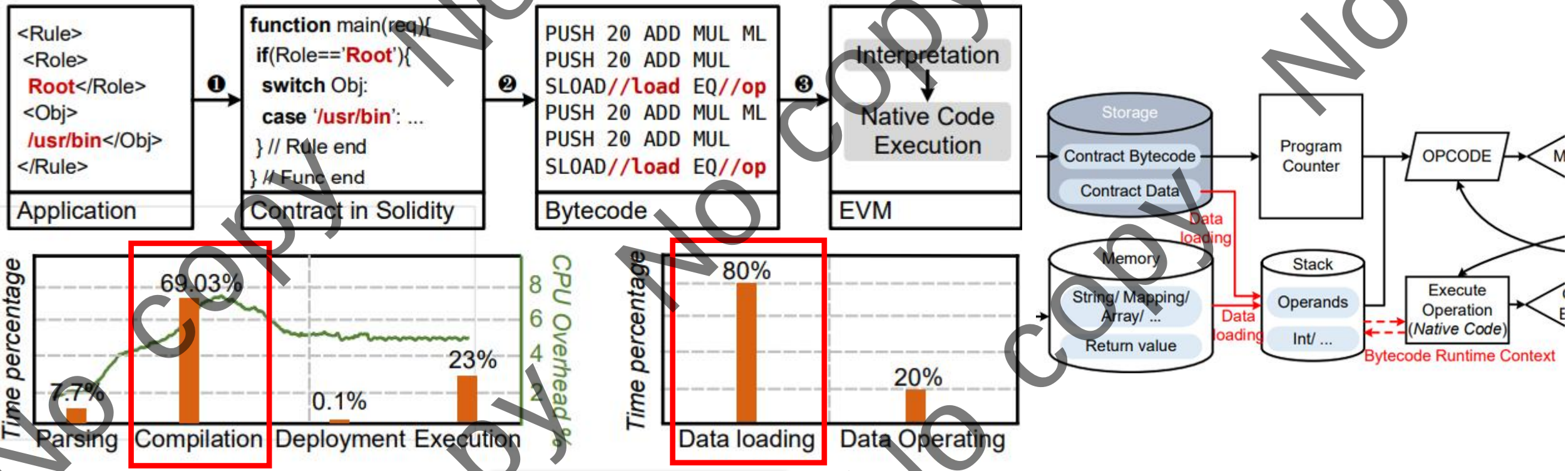


代理模式

控制器数据模式

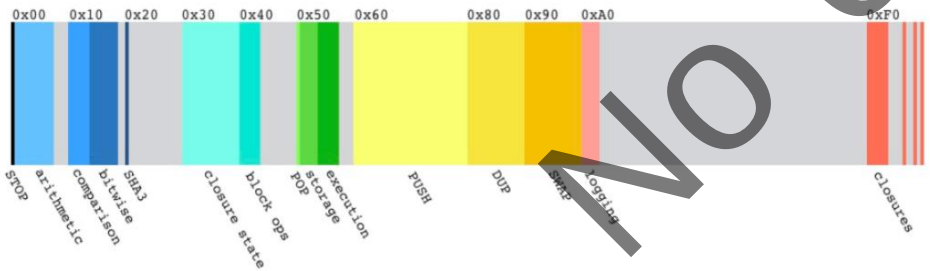
▶ 智能合约存在的主要问题

代码漏洞、BUG、应用需求导致智能合约不断频繁更新，现有更新方法效率低。
智能合约执行时效率低。

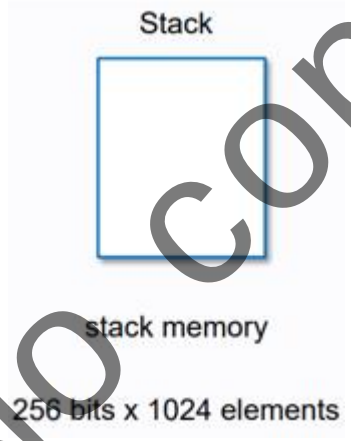


▶ 问题与难点归纳

问题1 合约无法**直接、高效**更新



EVM指令不支持

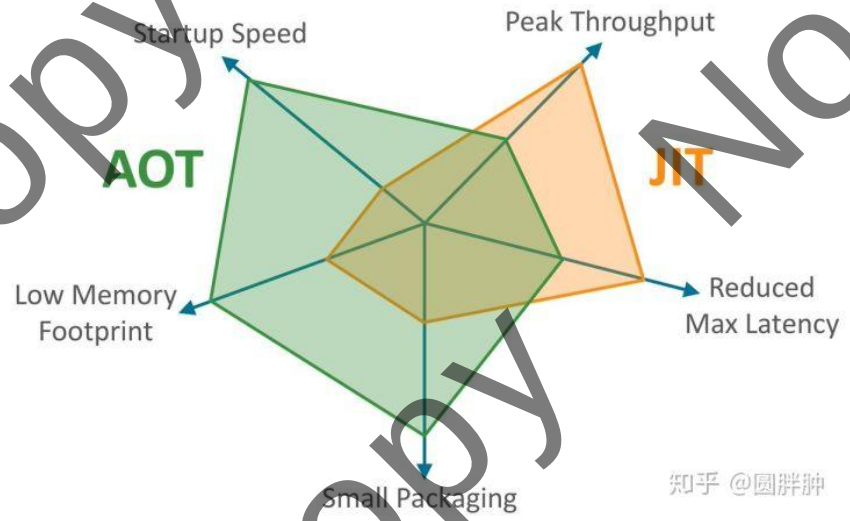


难点： 控制流和操作数都被存储在栈内，随意更改指令会导致控制流错误

(*e.g., 地址跳转错误*)

问题2 合约执行/运行效率低

传统的优化方法不适合智能合约



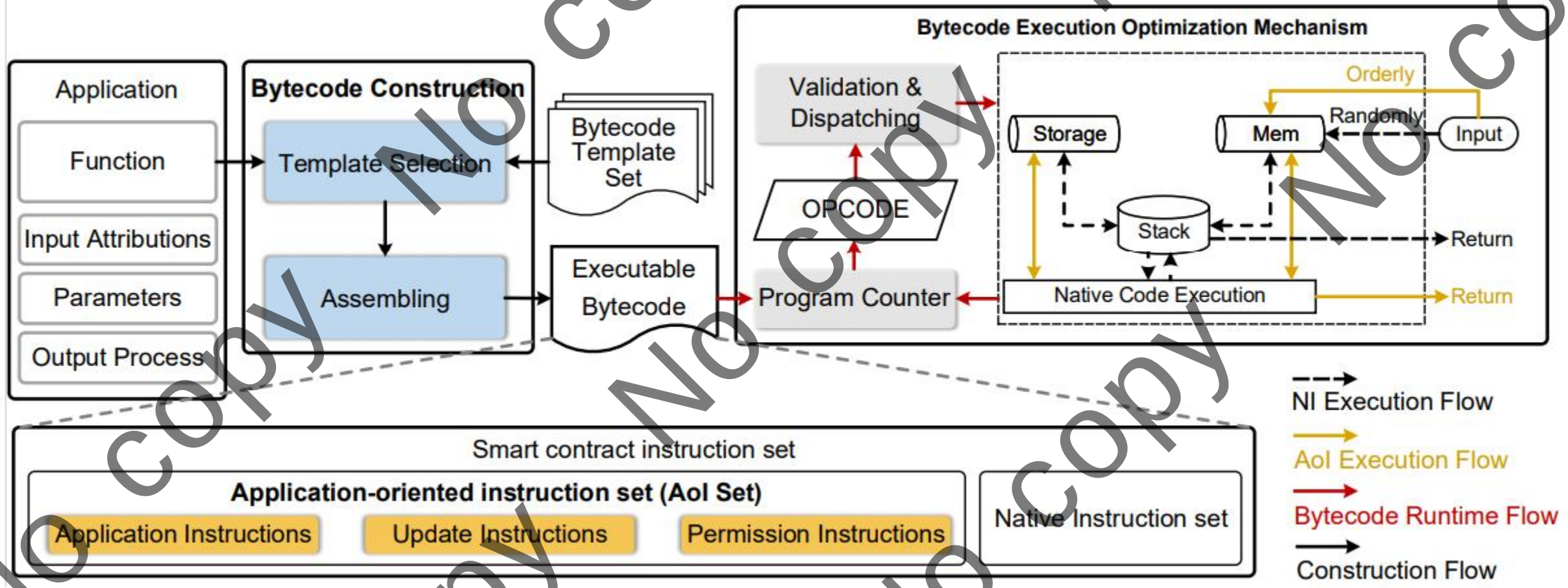
预先编译： 二进制不够灵活；

即时编译： 针对于动态语言，且可能破坏数据一致性

目录

▶ 我们的工作

ATOM: Architectural Support and Optimization Mechanism for Smart Contract Fast Update and Execution in Blockchain-based IoT



▶ 我们的工作 支持智能合约快速更新&执行的Application-oriented Compact Instruction Set

Type	Opcode	Name	Operand	Description
	0x0d	RBAC	Role, Source	Perform role-based checking with operands
	0x0e	ABAC	Attr, Source	Perform attribution-based checking with operands
Application	0x0f	SAC	Threshold	Perform simple request checking with operands
	0x1f	DENY	Source	Deny the requester
	0x21	ALLOW	Token	Return a fine token
	0x22	DU	Loc, Value	Modify the value under specified location to realize data update
Update	0x23	PU	Addr, Value	Modify the byte under specified address to realize function update
	0x24	AU	Addr, Value	Modify the byte under specified address to realize action update
Permission	0x25	UR	Account	Specify the updatable account
	0x26	UT	Type Number	Specify the update type (22, 23, 24 for DU, PU, AU accordingly)
	0x27	NOUPT0	Contract Address	Unenabled contract code update
	0x27	NOUPT1	Contract Address	Unenabled contract data update
Others	0x28	DMOV0	Loc, Value	Move the data from instruction to storage directly
	0x29	DMOV1	Loc, Value	Move the data from storage to instruction directly

TABLE I: **Examples of application-oriented instruction in access control.**

- 面向应用的领域特定指令，降低指令数量，提升执行速度
- 更新指令，降低更新负担，提升更新速度，底层支持让合约更新更加彻底和安全
- 数据加载指令，针对复杂类型变量，优化数据加载方式，提升数据加载速度

▶ 我们的工作 基于模板的智能合约字节码的快速构建

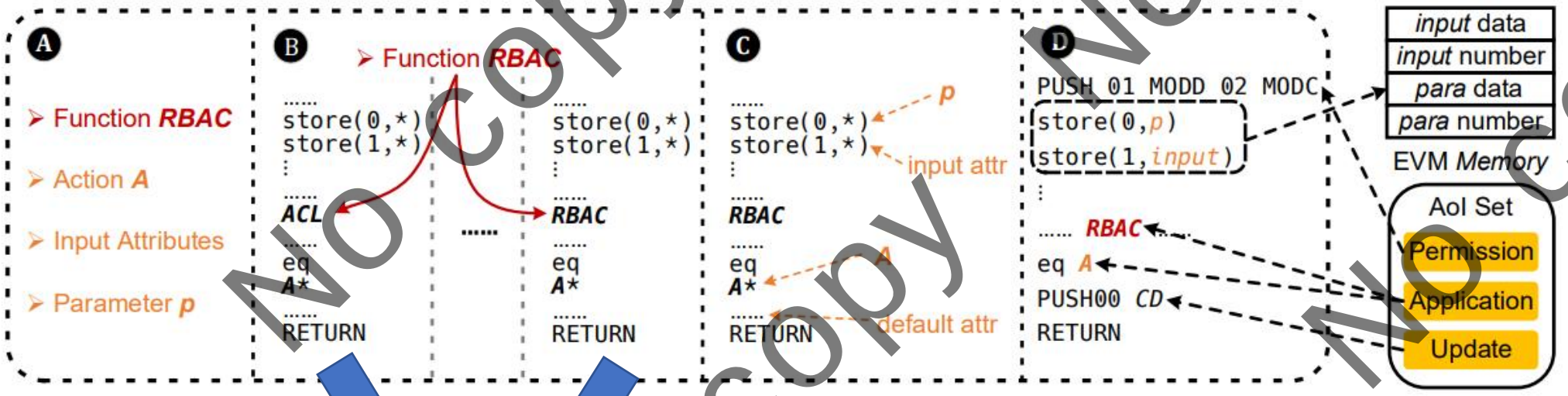
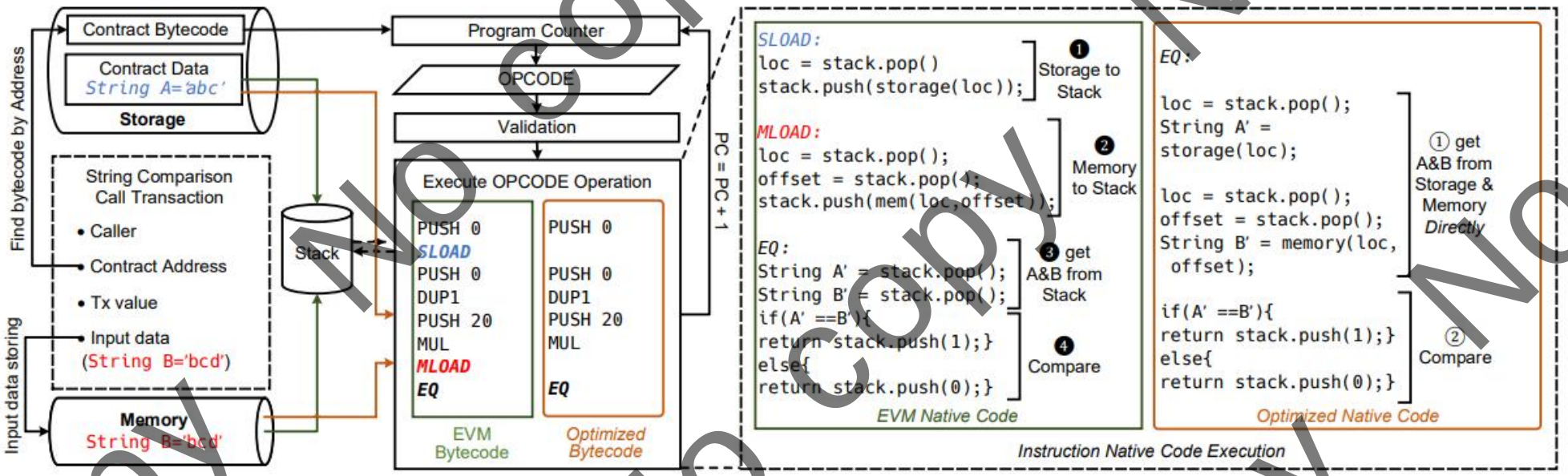


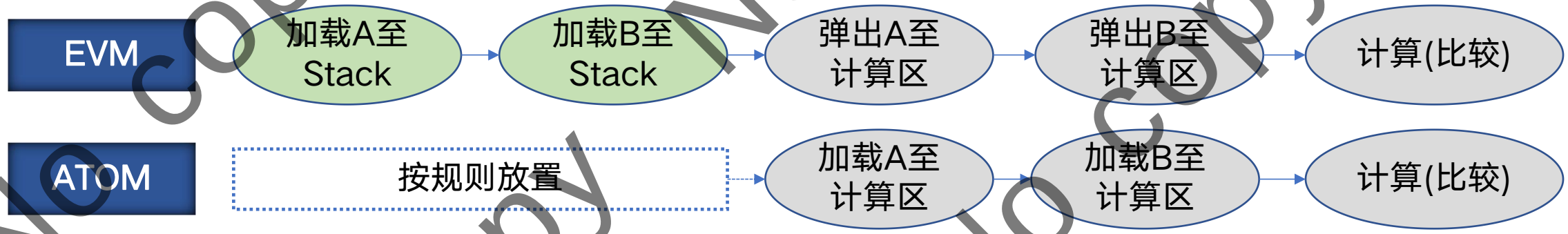
Fig. 5: **The two-step bytec construction** om left to right: ❶ Application ❷ Template selection from template set ❸ Assembling ❹ Executable by de.

应用操作模式较为固定 → 应用操作或计算模板库 → 模板字节码细节信息填充 → 可更新、可执行、高效的字节码文件（含指令）

▶ 我们的工作

面向频繁字符串操作的智能合约执行流程优化（以字符串比较为例）





目录

▶ 系统部署

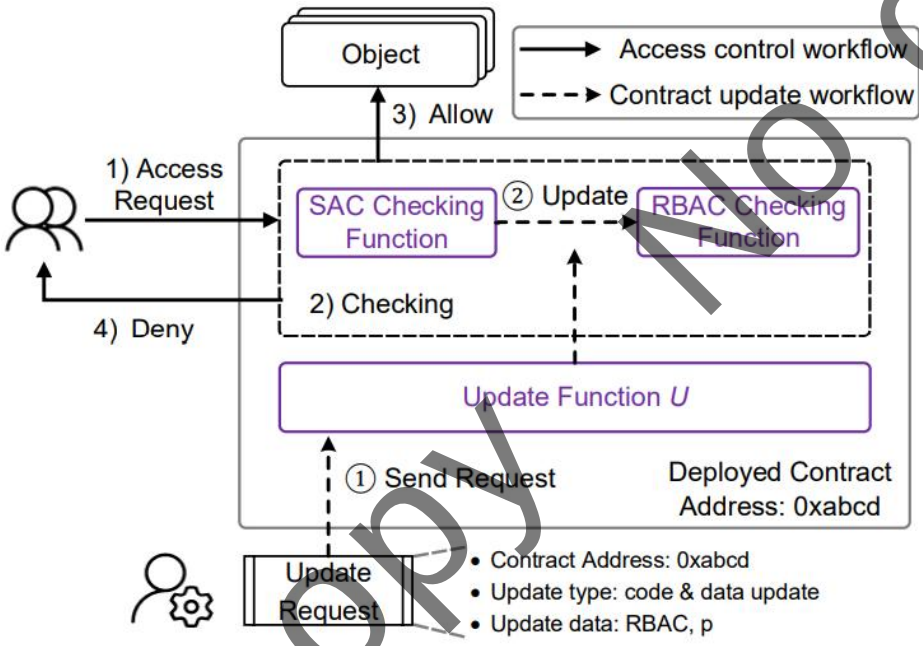
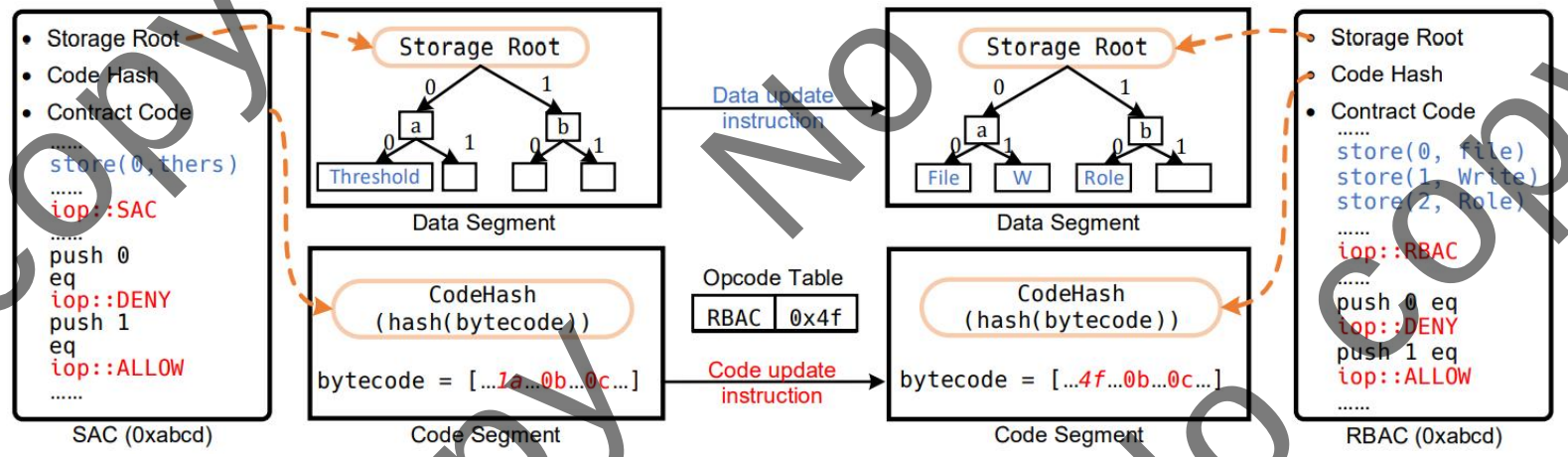


Fig. 7: **Smart Contract-based Access Control Overview.**

部署在基于物联网的访问控制应用中



基于物联网的访问控制应用中的合约更新实例：

- 含带AoI指令的访问控制应用合约
- 在合约代码段中定位指令位置
- 执行更新指令，编辑对应指令
- 指令替换，代码段更新完成，随后更新验证树
- 验证树更新完成，合约更新完成

仅需一次共识（资源消耗降低，GAS降低）

系统部署

主要指标（按环节划分）：

- **Overall performance**
  - 各阶段时延
  - 各阶段CPU负载
- **Bytecode construction**
  - 构建时延
  - 构建CPU负载
- **Contract update**
  - 更新时延
  - 账本大小
  - GAS用量
- **Contract execution**
  - 解释时延
  - 数据加载时延
  - CPU负载
  - EVM Memory访问

以太坊区块链(pro)

Solc Compiler

Ethereum Geth

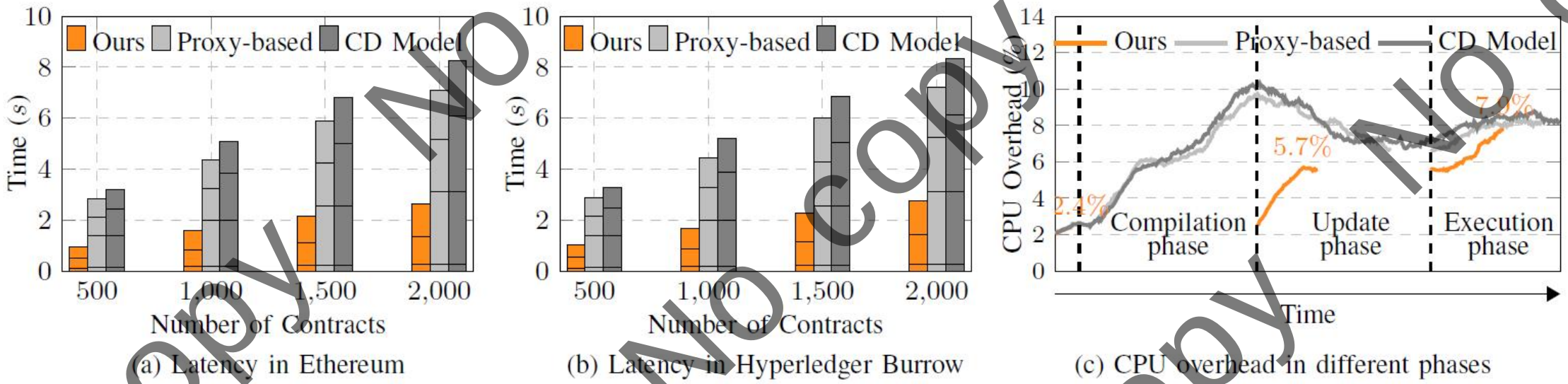
Ethereum VM

Baselines:

- Proxy-based model
- Controller-data model

Overall performance

- 各阶段时延
- 各阶段CPU负载



(a) Latency in Ethereum (b) Latency in Hyperledger Burrow (c) CPU overhead in different phases

Fig. 9: **Overall performance.**

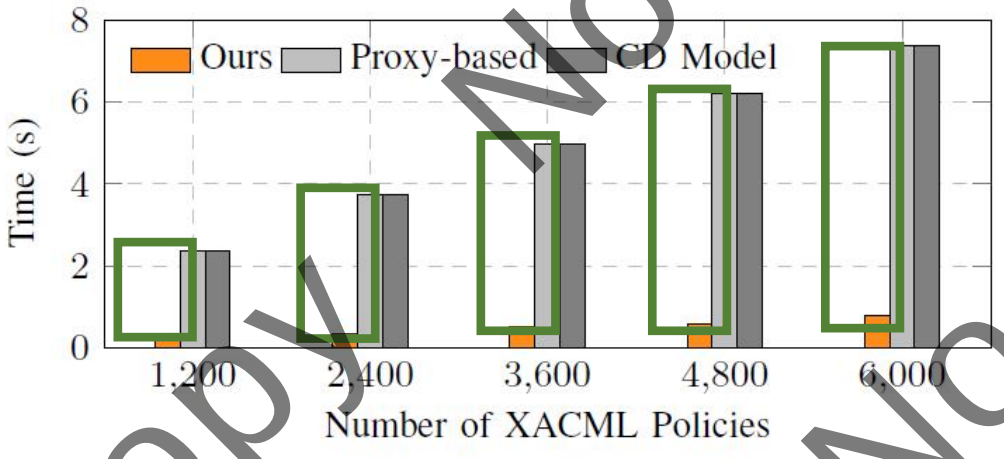
↓63.9% ↓68.7% (on average)

↓62.7% ↓67.6% (on average)

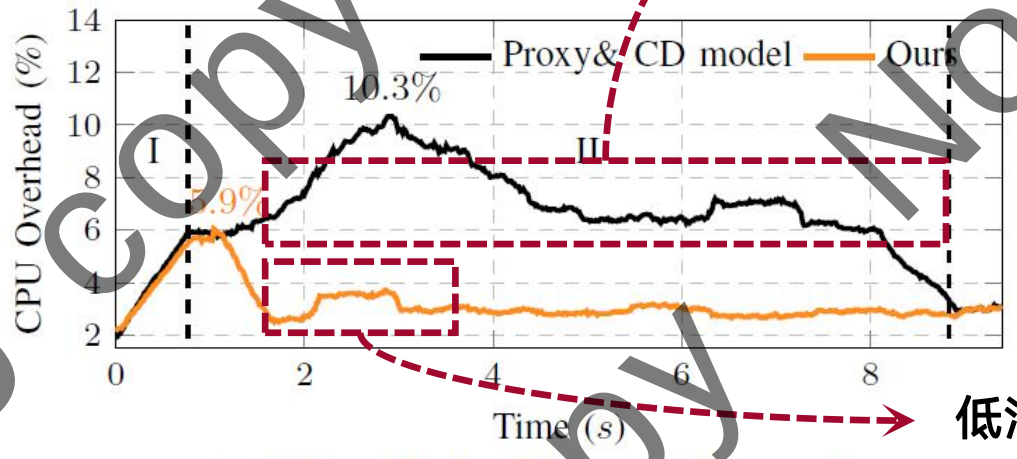
↓1.73% ↓1.97% (on average)

Bytecode construction

- 构建时延
- 构建CPU负载



(a) Bytecode construction latency.



(b) CPU overhead (I: Parsing, II: Compilation)

Fig. 10: **Bytecode construction performance.**

编译窗口期

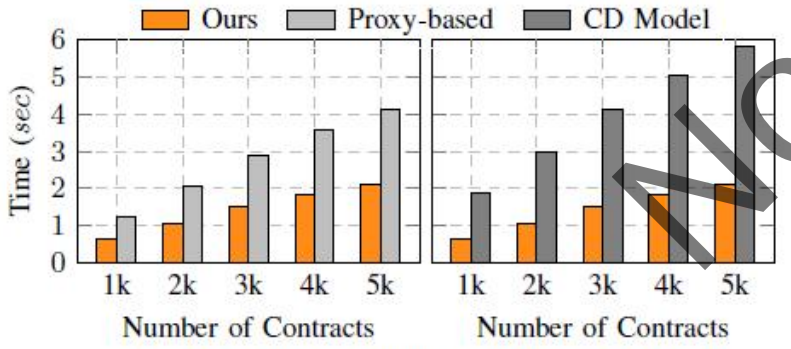
低消耗填充构建

构建速度提升90%

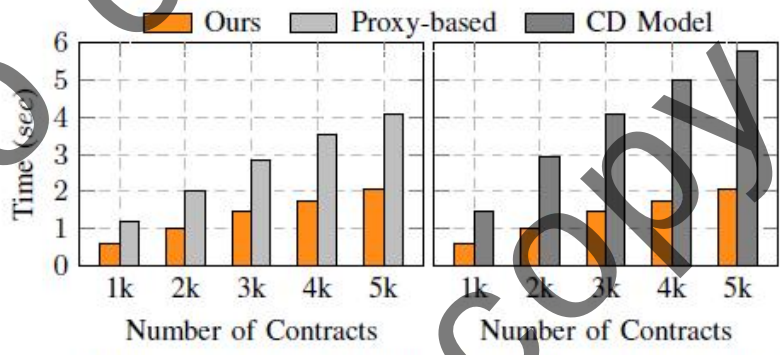
↓**4%** (on average)

Contract update

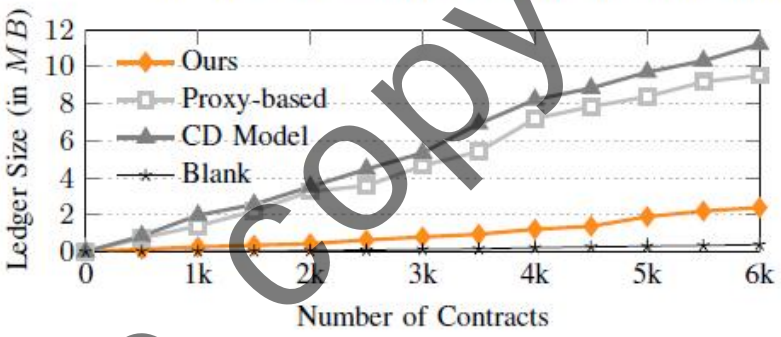
- 更新时延
- 账本大小
- GAS用量



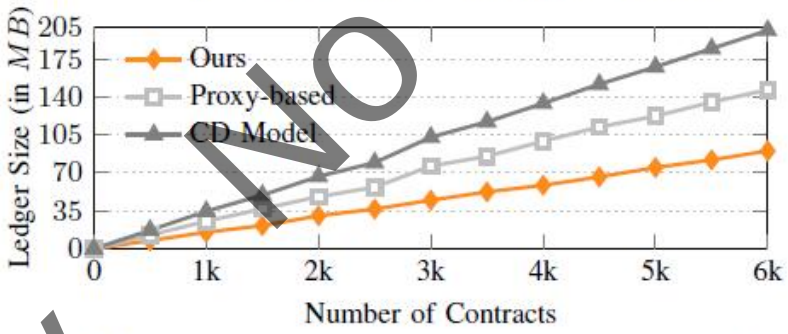
(a) Contract update latency comparison in Ethereum.



(b) Contract update latency comparison in Hyperledger.



(c) Ledger size comparison in contract update in Ethereum.



(d) Ledger size comparison in contract update in Hyperledger.

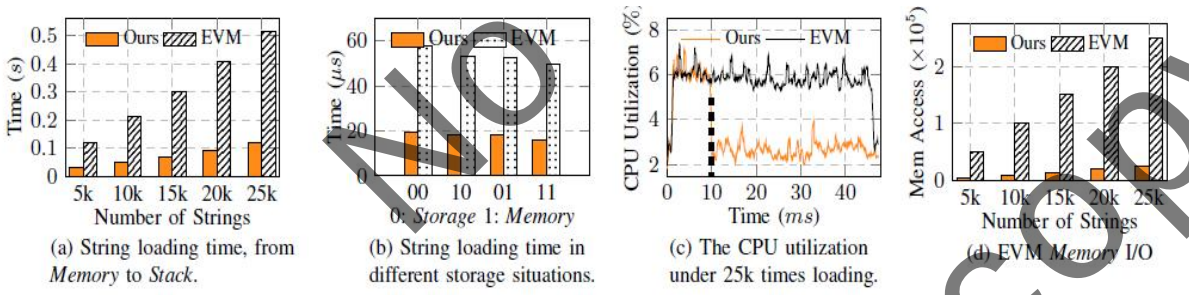
Fig. 11: **The contract update consumption.**

Method	Type	Gas Used ($\times 10^4$)	Reduced
Proxy-based	Deployment	26.2	Saved ✓
	Index Update	2.8	5.23 %
Controller-Data	Deployment	39.0	Saved ✓
	Index Update	2.8	5.46 %
ATOM	Instruction-layer Update	2.65	-

注：传统的以太坊虚拟机中Self-destruction指令仅会删掉索引，不会去除真正数据，安全问题依然存在。ATOM可以提供一定解决。

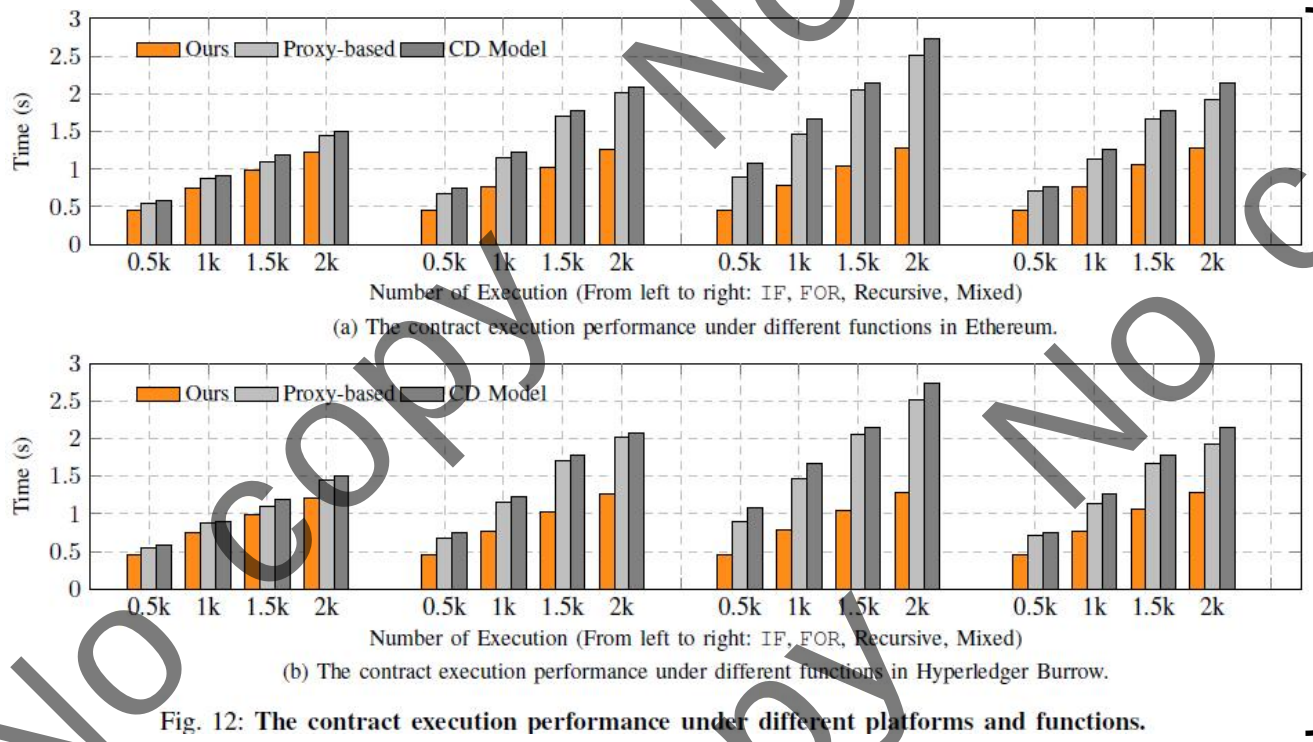
Contract execution

- 解释时延
- 数据加载时延
- CPU负载
- EVM Memory访问



(a) String loading time, from *Memory* to *Stack*. (b) String loading time in different storage situations. (c) The CPU utilization under 25k times loading. (d) EVM *Memory* I/O

虚拟机细节评测，EVM Memory访存、复杂类型数据加载与CPU负载



(a) The contract execution performance under different functions in Ethereum.

(b) The contract execution performance under different functions in Hyperledger Burrow.

Fig. 12: **The contract execution performance under different platforms and functions.**

Baseline	Contract execution latency reduction (%)							
	Ethereum platform				Hyperledger Burrow platform			
	IF	FOR	Recursive function	Mixed execution	IF	For	Recursive function	Mixed execution
Proxy-based method	13.5	35.8	48.5	34.4	13.5	34.1	**47.5**	31.7
Controller-Data model	19.4	39.4	**53.8**	39.7	19.4	37.3	52.7	37.3

TABLE III: Contract execution latency reduction in different platforms and function modes.

ATOM指令所带来的函数级评测与优化，时延可降低约50%

目录

区块链的兴起和智能合约的逐年发展（3200+ Dapps）[1]

	1	2	3
为什么要做ATOM？	通用系统	合约更新效率低 应用迭代不便	合约执行效率低 应用执行慢
ATOM是如何做的？	领域特定指令集设计	字节码快速构建与更新指令描述	提供复杂类型数据的新型加载机制
ATOM带来什么？	为智能合约优化带来系统性的新思路；助力整体效率提升	合约更新安全高效；拓展应用领域	基于栈的VM与解释器优化，使能时间敏感型应用的落地
未来何去何从？	虚拟机与硬件的协同设计与优化	字节码的多维度分析（安全、性能、关联）；编译器与编译技术	应用特定；与其它性能瓶颈的联合优化；编译器与编译技术

[1] L. Su, X. Shen, X. Du, X. Liao, X. Wang, L. Xing, and B. Liu, “Evil under the sun: Understanding and discovering attacks on ethereum decentralized applications,” in 30th {USENIX} Security Symposium ({USENIX} Security 21), 2021.

南開大學 Nankai University

报告内容到此结束，谢谢！

Q & A

- 欢迎访问：https://ics.nankai.edu.cn
- 研究室邮箱：aics@nankai.edu.cn
- 个人邮箱：fyz@mail.nankai.edu.cn
- 官方仓库：https://github.com/nkicsl

This work is partially supported by the National Key Research and Development Program of China (2018YFB2100300), Zhejiang Lab (2021KF0AB04), the Natural Science Foundation of Tianjin (20JCZDJC00610, 19JCQNJC00600), the Open Project Fund of State Key Laboratory of Computer Architecture, Institute of Computing Technology, Chinese Academy of Sciences (CARCHB202016, CARCH201905), and the National Natural Science Foundation (62002175).

计算机体系结构国家重点实验室 State Key Laboratory of Computer Architecture, ICT, CAS

之江实验室 ZHEJIANG LAB