高性能智能合约体系架构

个人系列工作介绍&交流

姓名：房耀政

南开大学计算机学院 智能计算系统研究室 2021级博士研究生

2023年10月

个人情况介绍

研究生期间科研成果一览表				
科技论文			专利	软件著作权
总计	一作	参与发表		
13	3+2在投（区块链虚拟机方向）	8	3（智能合约虚拟机）	3（区块链系统）

论文列表&PDF：

https://www.fangyaozheng.com

Email: fyz@mail.nankai.edu.cn

Wechat: nkufyz

朝阳起又落

中国大陆

扫一扫上面的二维码图案，加我为朋友。

承担的课题（已提前结项）		
类别	名称	属性
2021年天津市研究生科研创新项目	基于区块链的分布式可信计算技术研究及实现	区块链智能合约性能与安全性提升

参与的课题		
类别	名称	属性
计算机体系结构国家重点实验室	基于TEE的智能合约可信执行引擎设计与优化研究	区块链+安全+引擎性能
天津市自然科学重点基金	面向金融应用的高性能区块链关键技术研究	区块链+金融应用性能提升
国家重点研发计划	面向深度融合的异质物联网应用服务协同技术	物联网广域系统
CCF-蚂蚁联合基金、之江实验室开放课题、物流项目等		

➢ 科研成果特色与贡献

系统性跨层优化

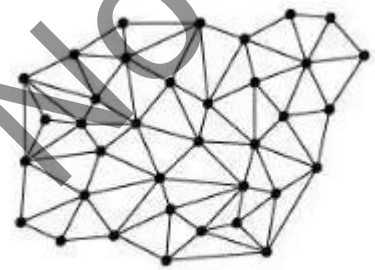
2021年 ATOM: Architectural Support and Optimization Mechanism for Smart Contract Fast Update and Execution in Blockchain-Based IoT (IOTJ, SCI-I，一作) 通信作者：卢冶

- ◆ 首次对区块链进行底层指令优化，突破合约更新难题及性能瓶颈；
- ◆ 2021年中国电子学会物联网专委会十大年度优秀期刊论文奖
- ◆ 2021年天津市网络与数据安全技术重点实验室优秀论文奖

2022年 SmartVM: A Smart Contract Virtual Machine for Fast On-Chain DNN Computations (TPDS,CCF A, 一作) 通信作者：卢冶

- ◆ 首次在区块链上进行高性能人工智能计算
- ◆ 提出区块链领域高级语言、编译器、执行环境的系统性联合优化方法
- ◆ **解决美国SKALE实验室在社区提出的Uber AI+BC问题**

区块链智能合约系统



distributed

- 分布式数据库+状态复制机
- 存储（数据）防篡改、操作（计算）可共识
- 公开透明
- 智能合约提供链上计算能力+虚拟机隔离环境

Hello World

pragma specifies the compiler version of Solidity.

```
// SPDX-License-Identifier: MIT
// compiler version must be greater than or equ
pragma solidity ^0.8.20;

contract HelloWorld {
    string public greet = "Hello World!";
}
```

docker

支持多种语言、环境
复杂需要配置

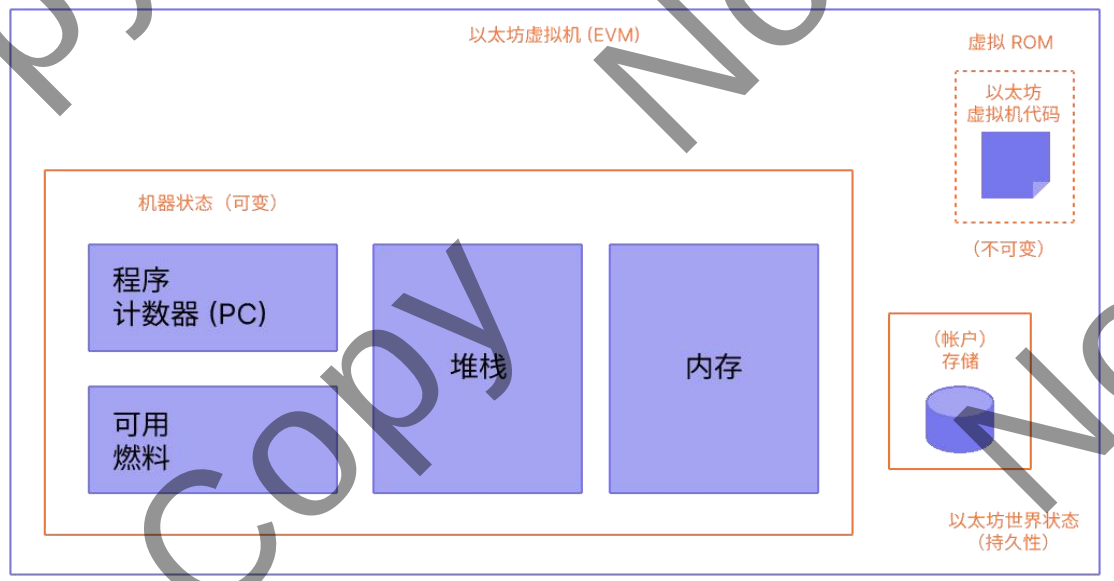
可移植性、实现简单

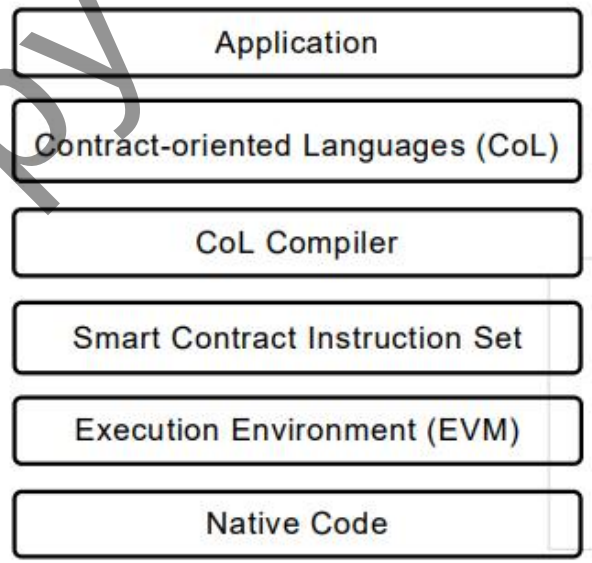
➢ 区块链智能合约系统

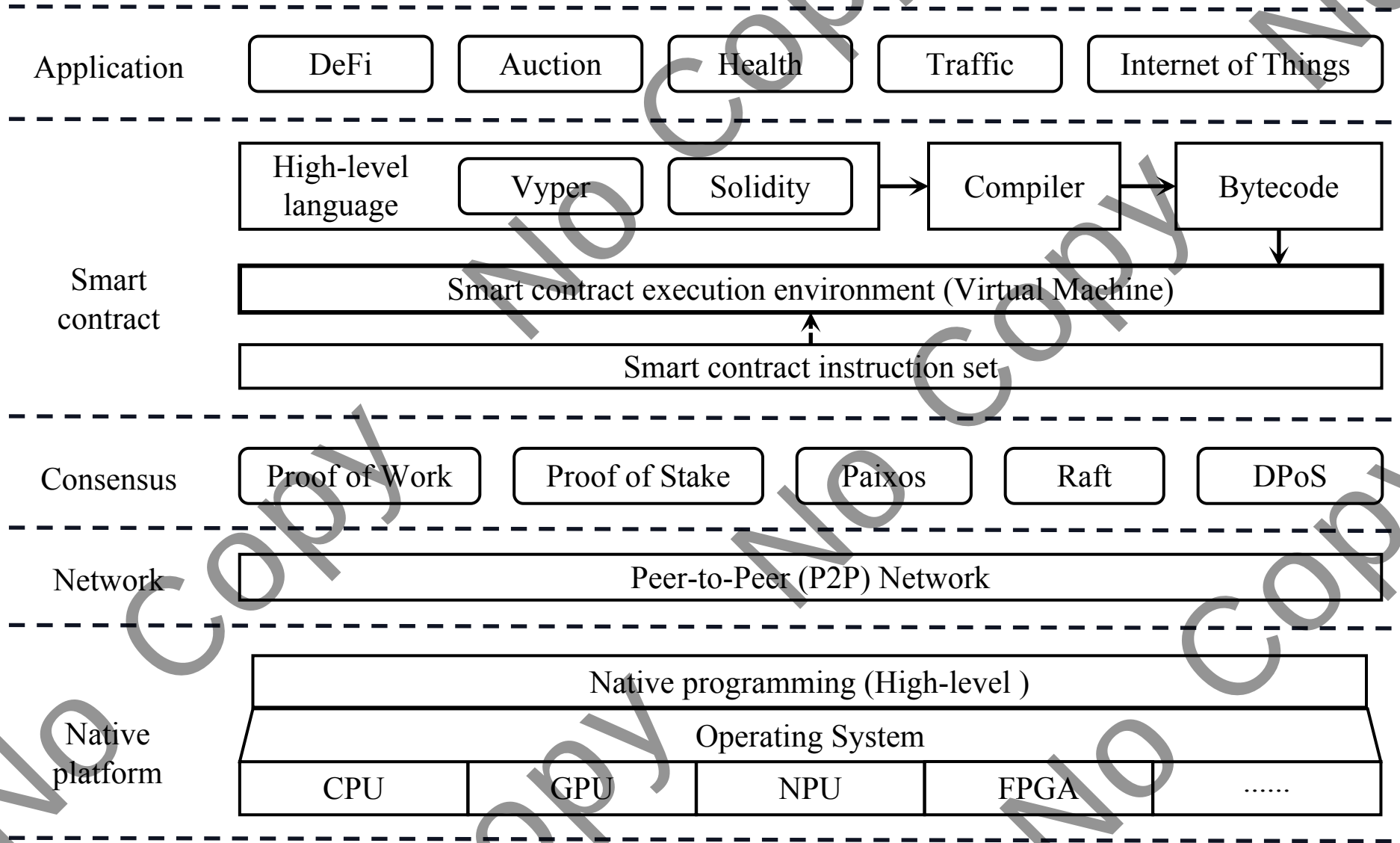
Hello World

pragma specifies the compiler version of Solidity.

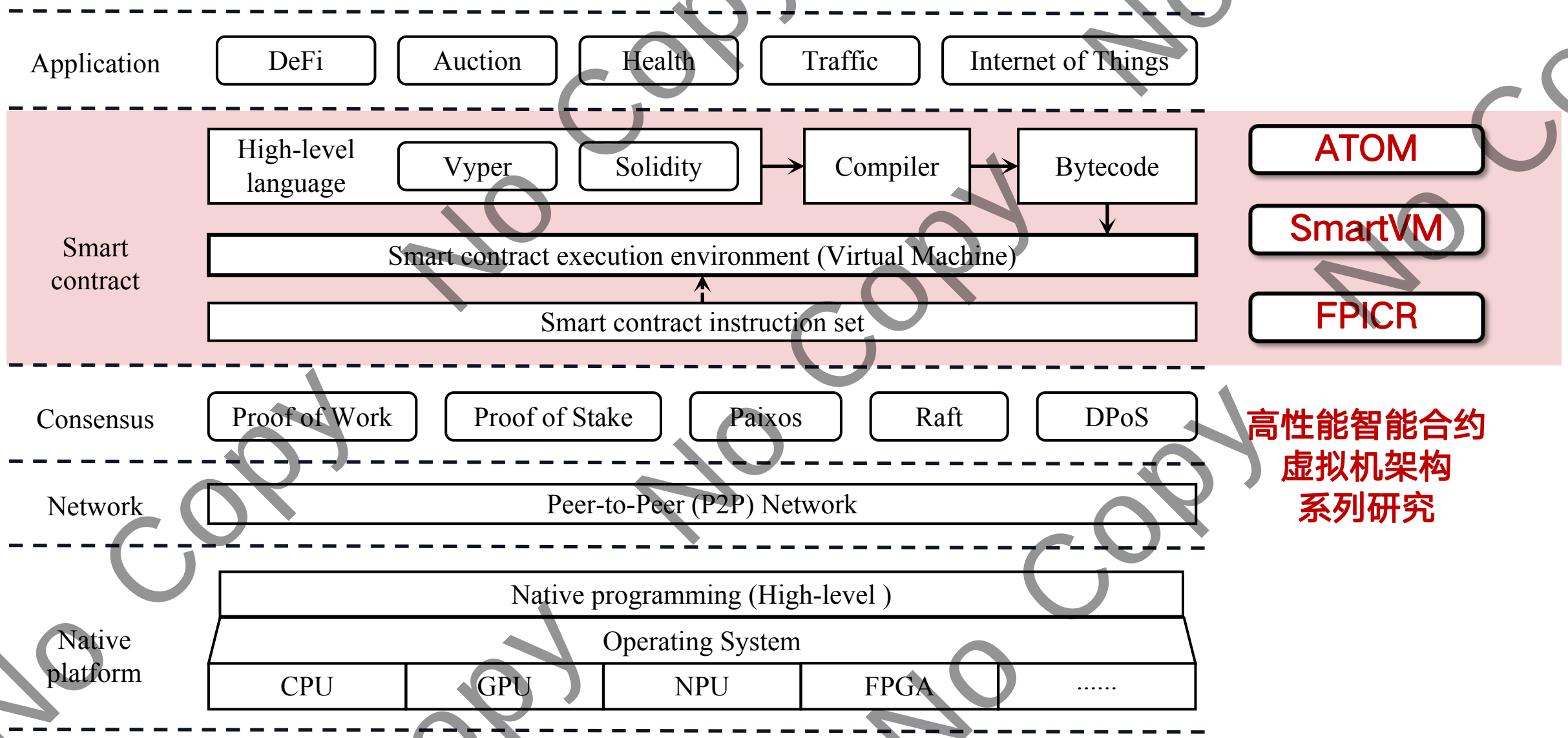
```
// SPDX-License-Identifier: MIT
// compiler version must be greater than or equ
pragma solidity ^0.8.20;

contract HelloWorld {
    string public greet = "Hello World!";
}
```



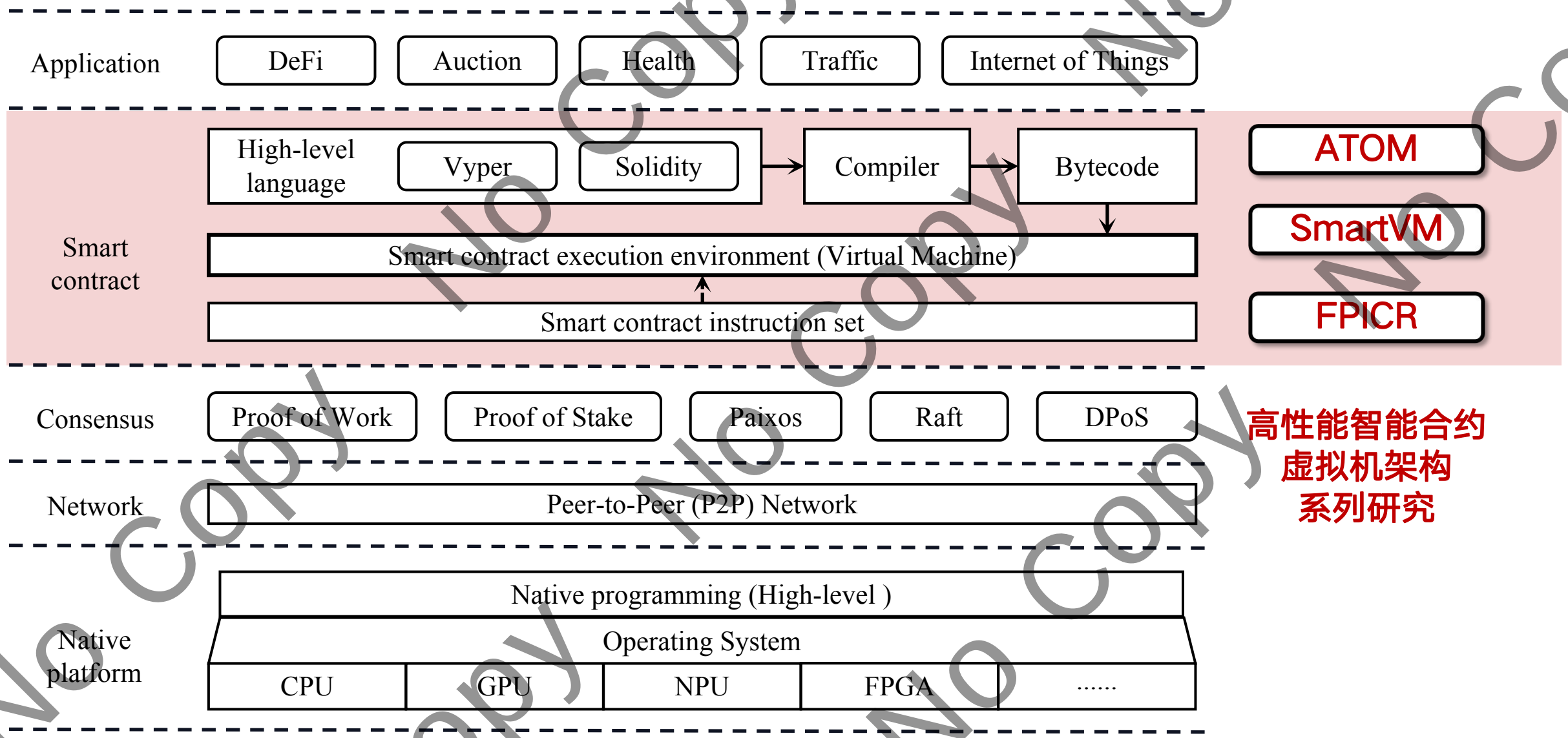




区块链的计算核心，
赋能应用落地



高性能智能合约
虚拟机架构
系列研究



高性能智能合约
虚拟机架构
系列研究

2021

- ATOM: Architectural Support and Optimization Mechanism for Smart Contract Fast Update and Execution in Blockchain-Based IoT
  Yaozheng Fang , Tao Li , Zhaolong Jian , Xueshuo Xie , Ye Lu* , Guiling Wang
  In IEEE Internet of Things Journal
  [Cite] [BibTeX] [Abstract] [Online] [Slides (zh-cn)]

➢ Full paper & slides are available at:
➢ https://www.fangyaozheng.com/

IEEE INTERNET OF THINGS JOURNAL, VOL. 9, NO. 11, JUNE 1, 2022 7959

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, and Guiling Wang

***Abstract*—Blockchain-based Internet of Things (BC-IoT) brings the advantages of blockchain into traditional IoT systems. In BC-IoT, the smart contract has been widely used for automatic, trusted, and decentralized applications. Smart contracts require frequent adjust and fast update due to various reasons, such as inevitable code bugs, changes of applications, or security requirements. However, previous smart contract architecture and updating mechanism are low speed and cause high overhead, because they are based on recompilation and redeployment in BC-IoT. Meanwhile, smart contract execution is so time consuming due to contract instruction dispatching and operand loading in the stack-based Ethereum virtual machine (EVM). To address these issues, we propose a new smart contract architecture and optimization mechanism for BC-IoTs, ATOM, which provides architectural supports to update contract economically and fast executing in instructionwise for the first time, to the best of our knowledge. We design a compact Application-oriented Instruction (AoI) set to describe application operations. We can construct the bytecode of smart contract from application by directly assembling templates prebuilt upon the AoIs rather than by compilation. We also present an optimized mechanism for AoI execution to enable access addressable storage place rather than the indirect access through stack. We perform ATOM on a BC-IoT testbed based on private Ethereum and Hyperledger Burrow. The experimental results highlight that ATOM is more efficient than state-of-the-art approaches. ATOM can reduce update latency by 62.7%, ledger size by 70%, and gas usage by 90% on average, respectively. Compared with the traditional smart contract architecture, ATOM can improve EVM Memory access efficiency significantly by up to 10× and achieve improvement of execution efficiency with up to 1.6×.**

***Index Terms*—Ethereum virtual machine (EVM), smart contract.**

Manuscript received March 8, 2021; revised May 20, 2021 and July 18, 2021; accepted August 19, 2021. Date of publication August 23, 2021; date of current version May 23, 2022. This work was supported by the National Key Research and Development Program of China under Grant 2018YFB2100300; in part by Zhejiang Lab under Grant 2021KF0AB04; in part by the Natural Science Foundation of Tianjin under Grant 20JCZDJC00610 and Grant 19JCQNJC00600; in part by the State Key Laboratory of Computer Architecture (ICT, CAS) under Grant CARCHB202016 and Grant CARCH201905; and in part by the National Natural Science Foundation under Grant 62002175. *(Corresponding author: Ye Lu.)*

Tao Li and Ye Lu are with the College of Computer Science, Nankai University, Tianjin 300071, China, also with the State Key Laboratory of Computer Architecture, Institute of Computing Technology, Chinese Academy of Sciences, Beijing 100190, China, and also with the Tianjin Key Laboratory of Network and Data Science Technology, Tianjin 300350, China (e-mail: luye@nankai.edu.cn).

Yaozheng Fang, Zhaolong Jian, and Xueshuo Xie are with the College of Computer Science, Nankai University, Tianjin 300071, China, and also with the Tianjin Key Laboratory of Network and Data Science Technology, Tianjin 300350, China.

Guiling Wang is with the Department of Computer Science, New Jersey Institute of Technology, Newark, NJ 07102 USA.

Digital Object Identifier 10.1109/JIOT.2021.3106942

I. Introduction

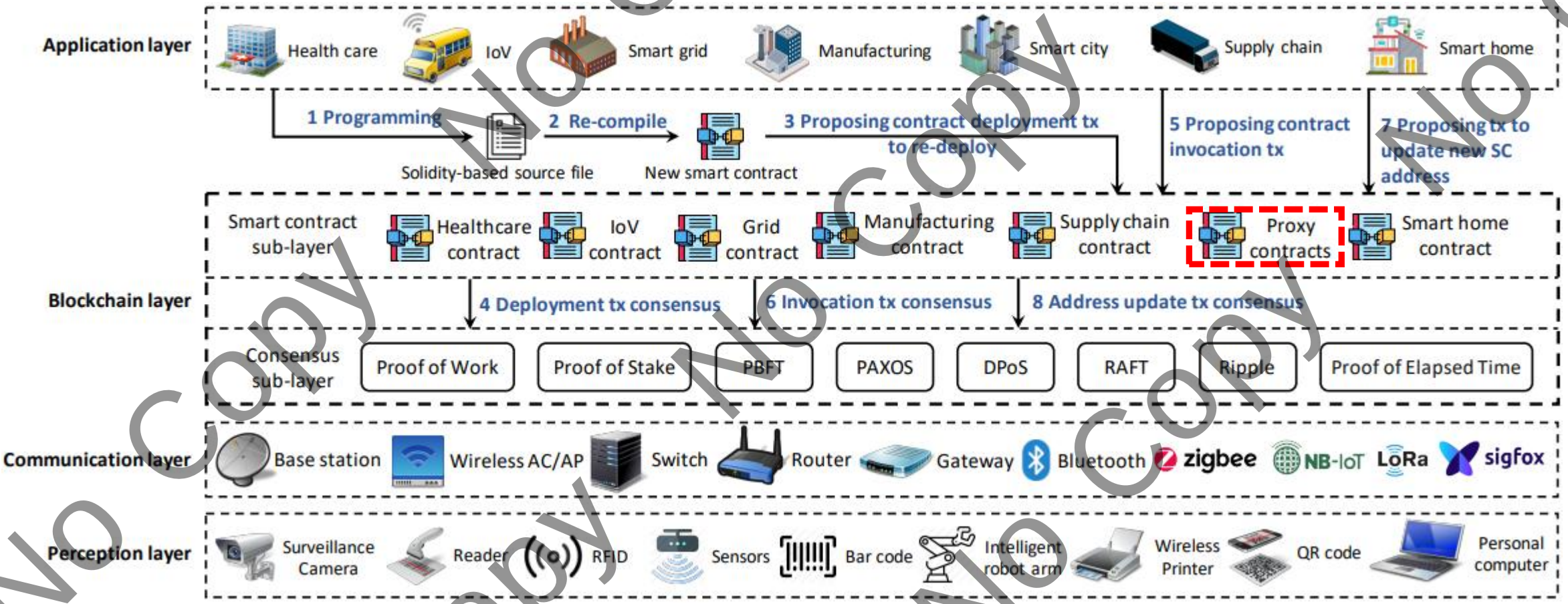
BLOCKCHAIN-BASED IoT (BC-IoT) is a new paradigm that uses blockchain to build distributed Internet of Things [1]–[6]. The paradigm has the advantages both of blockchain and IoT, e.g., trusted, decentralized, and tamper-proofing [7]–[9]. The BC-IoT has attracted extensive attention from both academia and industry [10], [11]. One of the most important parts of BC-IoT is smart contract, a computer program that can be automatically executed on blockchain such as Ethereum [12], and frees people from manual monitoring [13]–[15]. Owing to the attributes of autoexecution and consistent running result, the smart contract has been explored to enable many applications in BC-IoT [16]–[19], such as security management [20], [21].

Smart contracts in blockchain-based IoT system need to be updated frequently, because applications in such systems should be adjusted continuously for various reasons, such as inevitable code bugs and changes of application requirements [22]. For example, Luu *et al.* [23] and Huang *et al.* [24] pointed out that 8833 out of 19 366 existing Ethereum contracts are not bug free and vulnerable to attack. Thousands of smart contracts are potentially vulnerable, which should be corrected or patched up immediately through update [25]. However, the existing smart contract architecture in the original blockchain is not designed to support efficient contract update and execution. The most representative contract architecture and the most common-used contract execution environment are the Ethereum virtual machine (EVM). This work focuses on how to improve the contract update and execution performance on EVM.

The traditional contract update is based on recompilation and redeployment [26] in the application and the blockchain layer under the typical blockchain-based IoT [27], [28] as shown in Fig. 1. Updating a contract usually needs to take several steps as follows. First, smart contracts programmed by a high-level language (e.g., Solidity) are recompiled into the bytecode, which is composed of a series of contract instructions. In general, this recompilation requires long time and high memory footprint. Second, the bytecode needs to

► 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。

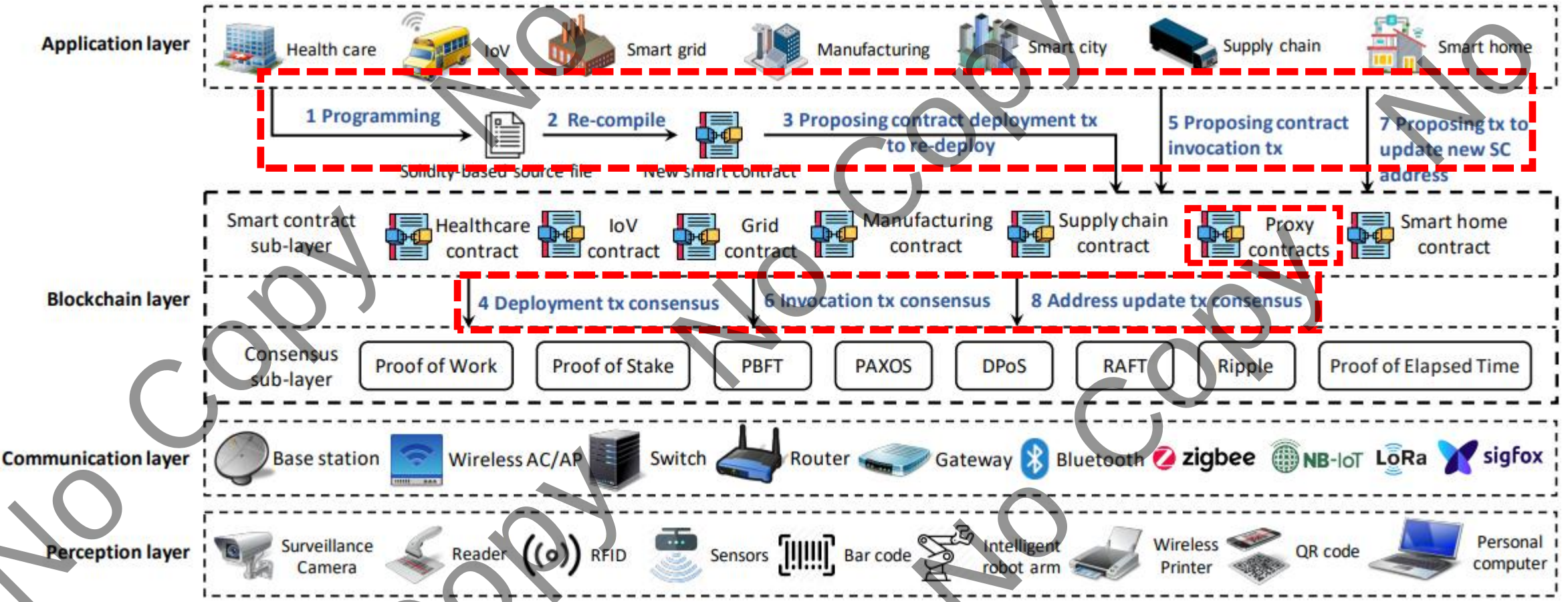


代理模式

控制器数据模式

► 智能合约存在的主要问题（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。

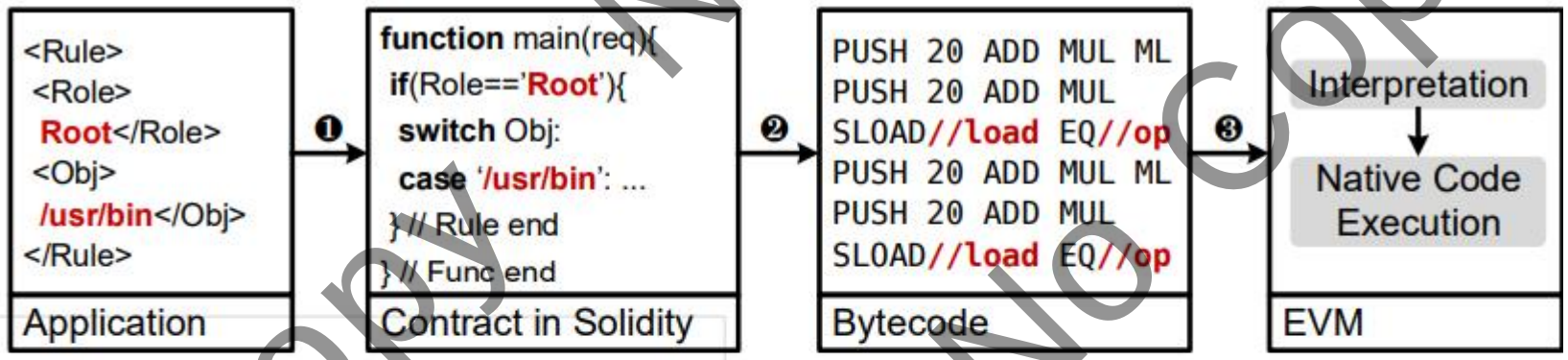


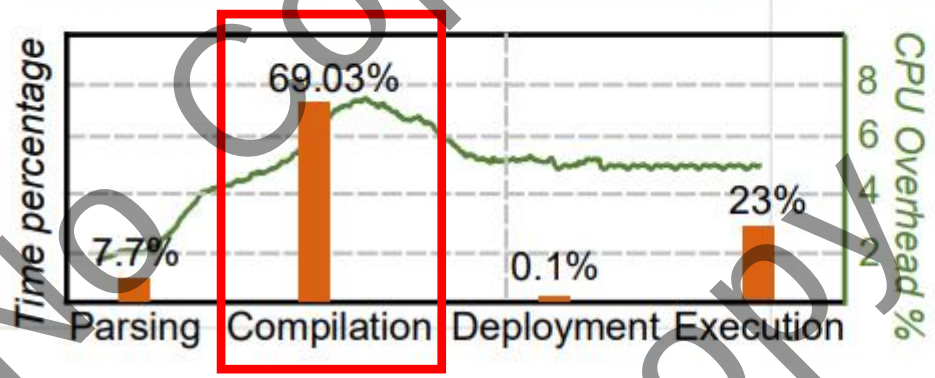
代理模式

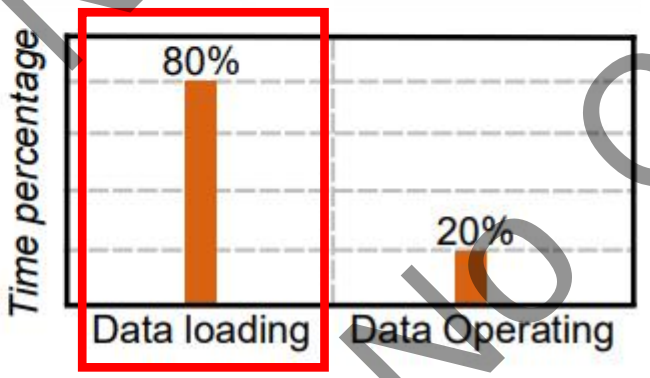
控制器数据模式

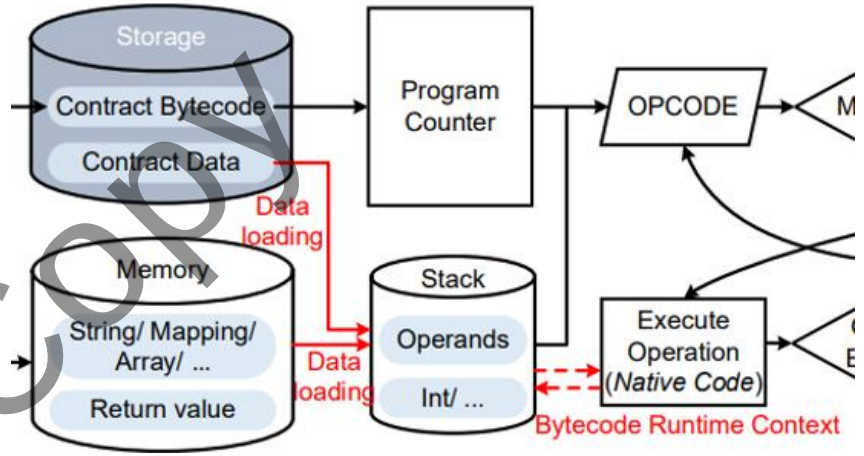
► 智能合约存在的主要问题

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。



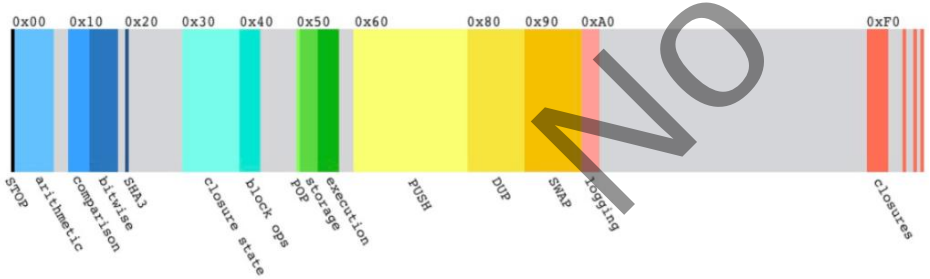




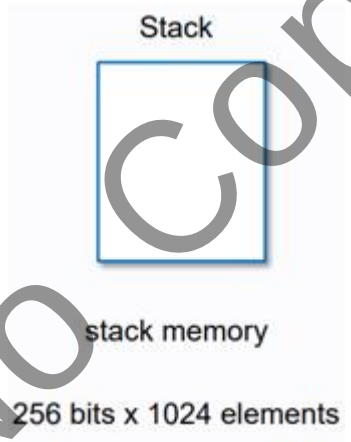


▶ 问题与难点归纳

问题1 合约无法**直接、高效**更新



EVM指令不支持

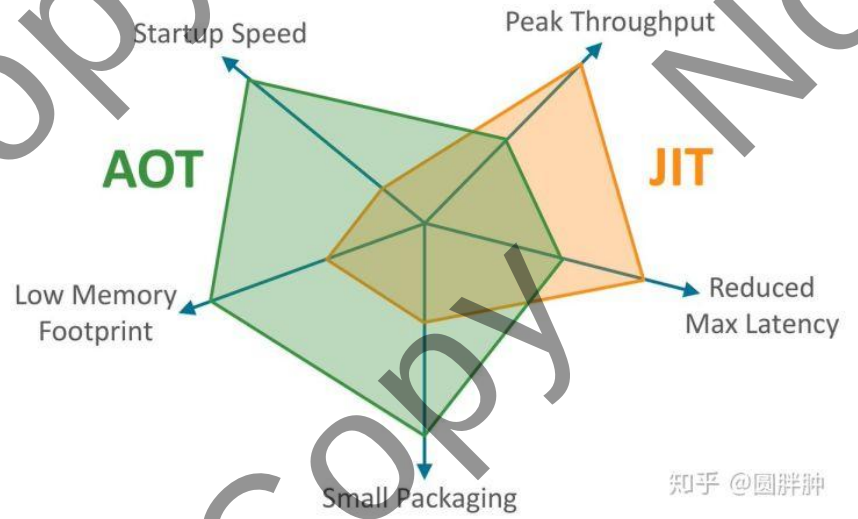


难点： 控制流和操作数都被存储在栈内，随意更改指令会导致控制流错误

(*e.g., 地址跳转错误*)

问题2 合约执行/运行效率低

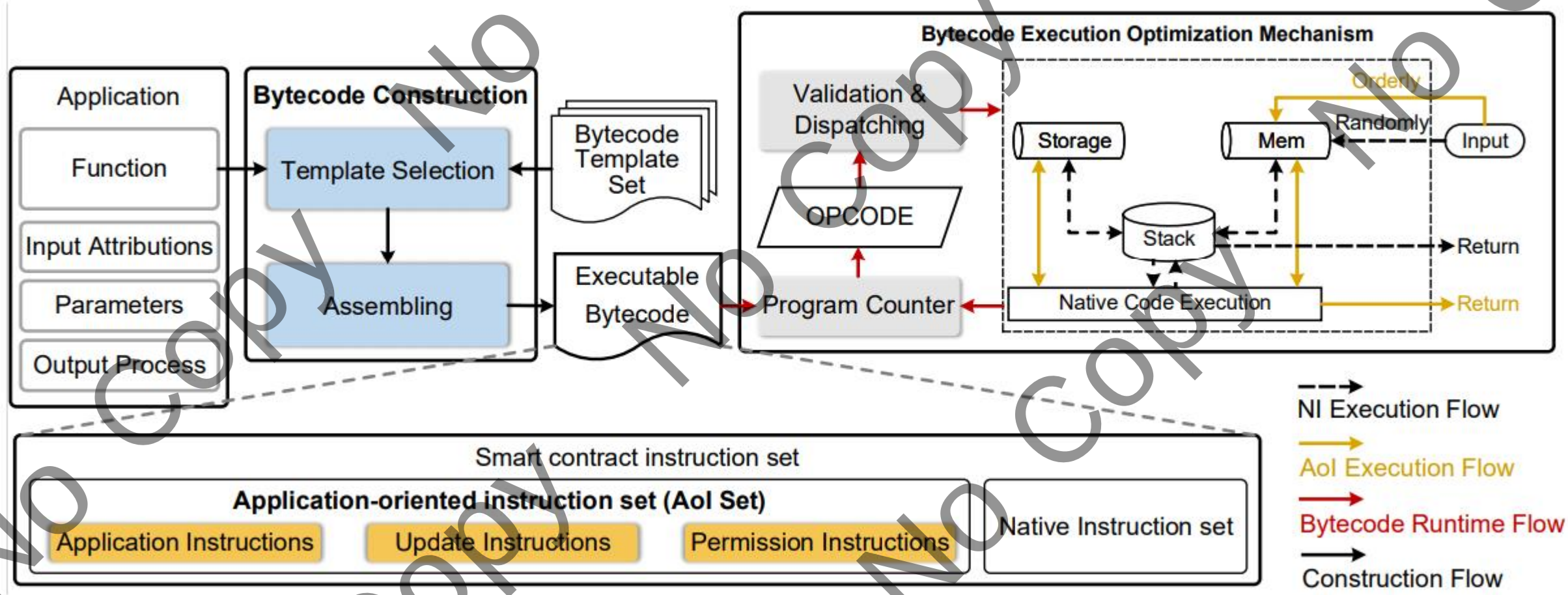
传统的优化方法不适合智能合约



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

► 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 file 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
	0x28	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

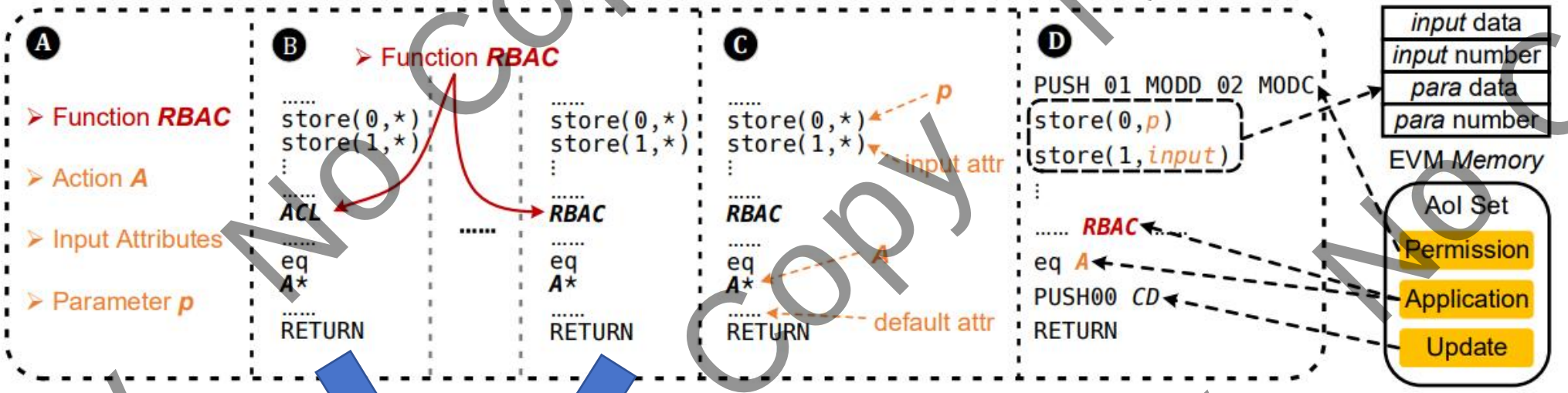
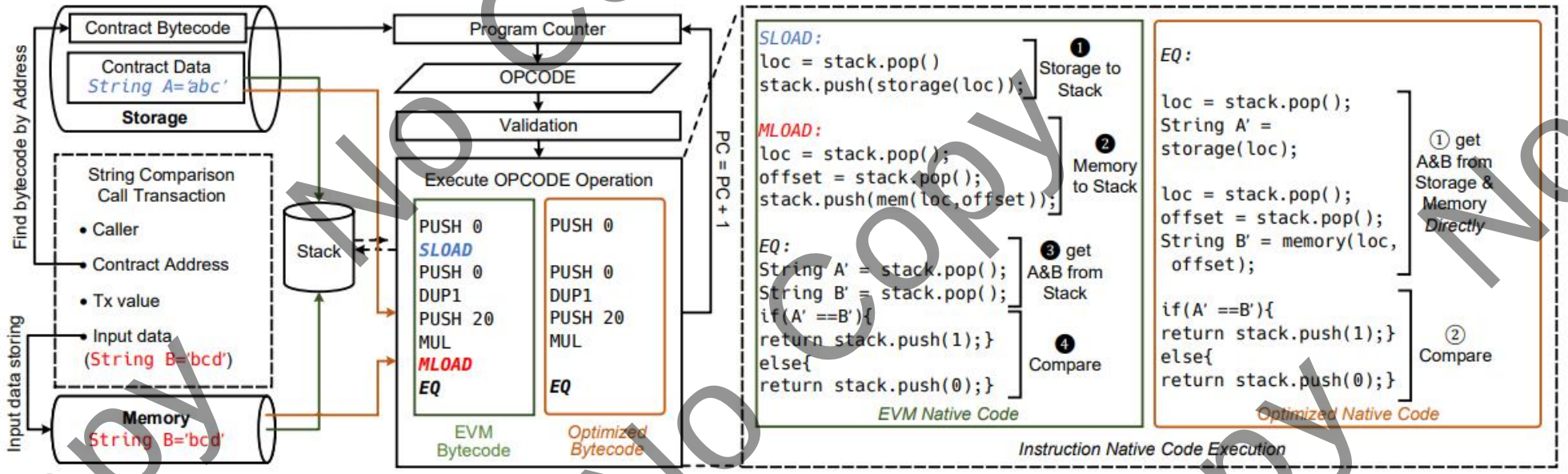


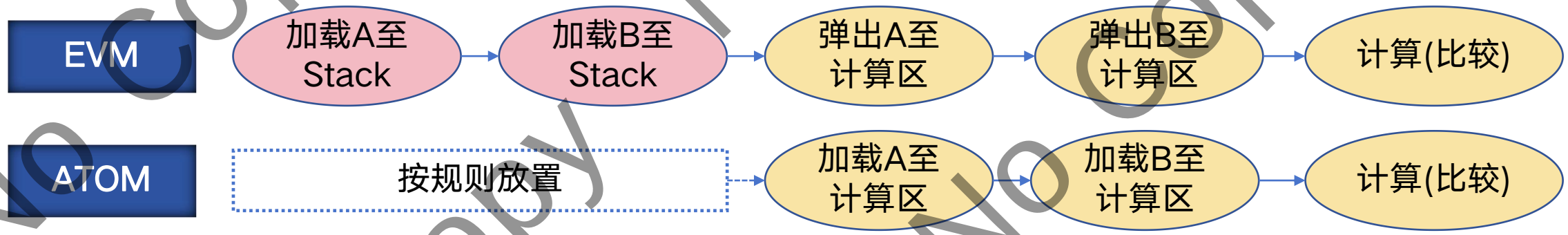
Fig. 5: **The two-step bytec... construction...** om left to right: ❶ Application ❷ Template selection from template set ❸ Assembling ❹ Executable by...de.

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）





科研工作（2/2）SmartVM

2022

- SmartVM: a smart contract virtual machine for fast on-chain DNN computations
  Yaozheng Fang , Tao Li , Ye Lu* , Jinni Yang , Zhaolong Jian , Zhiguo Wan , Yusen Li
  In IEEE Transactions on Parallel and Distributed Systems
  [Cite] [BibTeX] [Abstract] [Online] [Slides (zh-cn)]

➢ Full paper & slides are available at:
➢ https://www.fangyaozheng.com/

SmartVM: A Smart Contract Virtual Machine for Fast On-Chain DNN Computations

Tao Li, Yaozheng Fang, Ye Lu, Jinni Yang, Zhaolong Jian, Zhiguo Wan, and Yusen Li

Abstract—Blockchain-based artificial intelligence (BC-AI) has been applied for protecting deep neural network (DNN) data from being tampered with, which is expected to further boost trusted distributed AI applications in many fields. However, due to smart contract execution environment architectural defects, it is challenging for previous BC-AI systems to support computing-intensive tasks on-chain performing such as DNN convolution operations. They have to offload computations and a large amount of data from blockchain to off-chain platforms to execute smart contracts as native code. This failure to take advantage of data locality has become one of the major critical performance bottlenecks in BC-AI system. To this end, in this article, we propose SmartVM with optimization methods to support on-chain DNN inference for BC-AI system. The key idea is to design and optimize the computing mechanism and storage structure of smart contract execution environment according to the characteristics of DNN such as high computational parallelism and large data volume. We decompose SmartVM into three components: 1) a compact DNN-oriented instruction set to describe computations in a short number of instructions to reduce interpretation time. 2) a memory management mechanism to make SmartVM memory dynamic free/allocated according to the size of DNN feature maps. 3) a block-based weight prefetching and parallel computing method to organize each layer's computing and weights prefetching in a pipelined manner. We perform the typical image classification in a private Ethereum blockchain testbed to evaluate SmartVM performance. Experimental results highlight that SmartVM can support DNN inference on-chain with roughly the same efficiency against the native code execution. Compared with the traditional off-chain computing, SmartVM can speed up the overall execution by **70**×, **16**×, **11**×, and **12**× over LeNet5, AlexNet, ResNet18, and MobileNet, respectively. The memory footprint can be reduced by **84%**, **90.8%**, **94.3%**, and **93.7%** over the above four models, while offering the same level model accuracy. This article sheds light on the design space of the smart contract virtual machine for DNN computation and is promising to further boost BC-AI applications.

Index Terms—Deep neural network, smart contract, virtual machine, architectural support technology'

- *Tao Li is with the College of Computer Science, Nankai University, Tianjin 300071, China, with the College of Cyber Science, Nankai University, Tianjin 300071, China, with the State Key Laboratory of Computer Architecture, Institute of Computing Technology, Chinese Academy of Sciences, Beijing 100045, China, and also with the Tianjin Key Laboratory of Network and Data Science Technology, Tianjin 300071, China. E-mail: litao@nankai.edu.cn.*
- *Yaozheng Fang and Zhaolong Jian are with the College of Computer Science, Nankai University, Tianjin 300071, China, and also with the Tianjin Key Laboratory of Network and Data Science Technology, Tianjin 300071, China. E-mail: {fyz, jianzhaolong}@mail.nankai.edu.cn.*
- *Ye Lu is with the College of Computer Science, Nankai University, Tianjin 300071, China, with the College of Cyber Science, Nankai University, Tianjin 300071, China, with the Information Security Evaluation Center of Civil Aviation, Civil Aviation University of China, Tianjin 300300, China, with the State Key Laboratory of Computer Architecture, Institute of Computing Technology, Chinese Academy of Sciences, Beijing 100045, China, and also with the Tianjin Key Laboratory of Network and Data Science Technology, Tianjin 300071, China. E-mail: luye@nankai.edu.cn.*
- *Jinni Yang is with the College of Cyber Science, Nankai University, Tianjin 300071, China, and also with the Tianjin Key Laboratory of Network and Data Science Technology, Tianjin 300071, China. E-mail: tol2020_nk@foxmail.com.*
- *Zhiguo Wan is with Zhejiang Lab, Hangzhou, Zhejiang 311121, China. E-mail: zhiguo_wan@163.com.*
- *Yusen Li is with the College of Computer Science, Nankai University, Tianjin 300071, China. E-mail: liyusen@nbjl.nankai.edu.cn.*

Manuscript received 1 December 2021; revised 19 May 2022; accepted 20 May 2022. Date of publication 24 May 2022; date of current version 22 September 2022.
(Corresponding author: Ye Lu.)
Recommended for acceptance by J. Zhai.
Digital Object Identifier no. 10.1109/TPDS.2022.3177405

1 Introduction

Blockchain-based artificial intelligence (BC-AI) has been a new researching hotspot [1], [2], [3], expected to boost trusted distributed AI training and inference [4], [5], [6], such as protecting deep neural network (DNN) data from being tampered [7], [8]. Smart contract is a piece of code which can be deployed on blockchain for executing application logic [9], [10]. Various blockchains have provided execution environment or virtual machine, such as Ethereum Virtual Machine (EVM) [11], [12], for interpreting and executing smart contract. The execution on virtual machine of the smart contract deployed on the blockchain is called on-chain computing and conducting the smart contract out of the virtual machine is correspondingly called off-chain computing [13], [14], [15].

The existing main stream smart contract virtual machines have limited BC-AI application scope and further development, since previous they cannot process complex tasks. For example, although the smart contract virtual machines such as EVM sustain more than 3,200 kinds of Dapps [15], there is no DNN application that can run on the blockchain [16]. DNN inference as yet cannot be directly and efficiently performed on blockchain by smart contract [17], [18]. The primary reason is that the smart contract execution environment in previous BC-AI system lacks operators, instructions and corresponding mechanism to support redundant complex DNN operations with high computational and memory complexity.

These issues lead to the existing BC-AI applications on blockchain that can only simply store a large amount of

▶ Blockchain-based Artificial Intelligence (BC-AI)

区块链可以和分布式AI天然结合，构建安全、可信的分布式AI系统 (Blockchain-based AI Systems)

▶ Problems of off-chain DNN inference

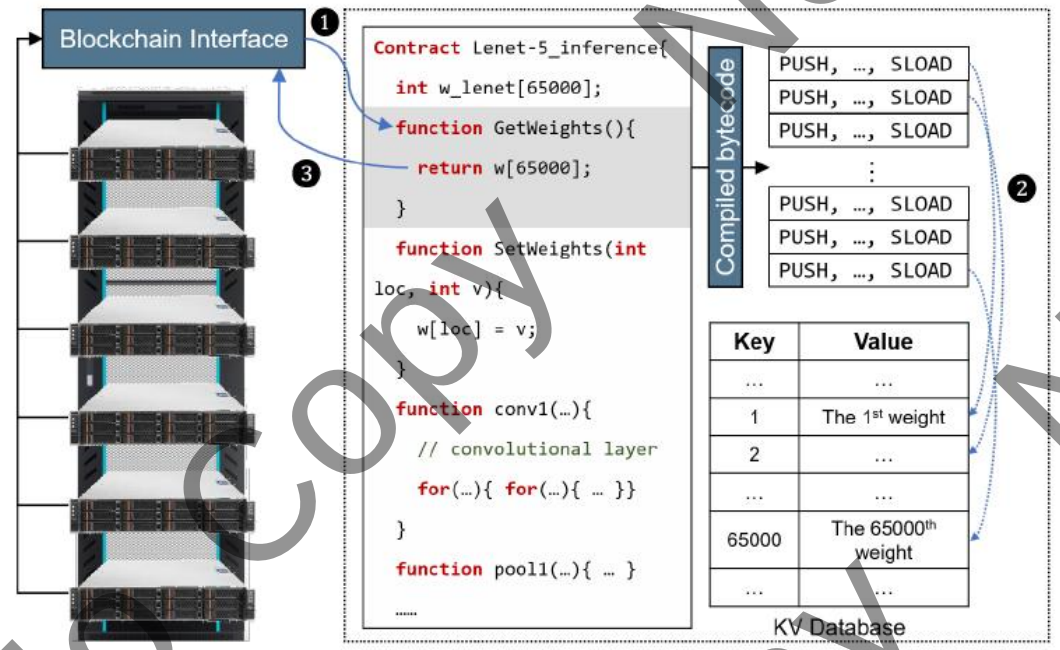


Fig. 2: CNN Computing Process in Typical BC-AI.

① 合约方法调用，**请求**获取权重

② 字节码执行，**按顺序**从数据库中**取权重**

③ 权重数据**返回**传输

问题：

① 合约方法执行时字节码数量庞大，达千万条

② 权重分散存储，数万条记录，且CNN权重越来越多

③ 数据转移代价

EVM+Solidity组合：

LeNet网络数分钟，AlexNet网络近1小时；数据搬运代价高

科研工作（2/2）SmartVM

Running Deep Learning on EVM

EVM

kladkogex 1 Jan '18

At my company we are starting an experimental project to extend EVM with basic deep learning capabilities.

This is not to train a neural network, but to use a pre-trained neural network inside a smart contract. Computation-wise using a pre-trained neural network is actually not so much more expensive than doing, say, RSA.

I understand that this may be a bit too heavy for the official Ethereum blockchain, so in our case we will run the EVM on a separate permissioned cluster with BFT-like consensus.

The current plan is that:

1. A pre-trained network is saved on the blockchain. We can use some of existing neural network serialization standards such as the ones used in Keras framework 36
2. The EVM will need to pull the neural network from the blockchain.
3. In the simplest case there we add a single **predict** instruction similar to predict from Keras framework 22 . This instruction will take a fully qualified name of the neural network and an input data array, run the neural network and produce output data.

As an example input data could be an English-language string, and output will be a German translation of this string.

One problem that we will need to solve in the process is introducing deterministic floating point numbers such as IEEE 754-2008 into the EVM in some way.

If there are other people interested to run AI on EVM, we would be willing to cooperate on this to establish a standard that everyone uses …

- **社区内的真实需求**
- Distributed Uber
- Trusted analysis

Well - I think many people have expressed different ideas about running a neural network as a trusted application so that all parties agree to the outcome.

As a toy example, you can consider an example of a smart contract that is an decentralized Uber which pays to drivers based on their behavior. The smart contract needs to differentiate bad drivers from good drivers by running a neural network on driver historical behavior. Good drivers are get paid and bad drivers do not get paid.
What you do, you feed driver's trajectory into a neural network, and then the driver gets either paid in-full or penalized based on her behavior …

Another example is when a smart contract buys apples from a supplier and it needs to find out whether an apple is a good apple or a bad apple based on chemical analysis data. Essentially you feed into the neural network 100 data points of chemical analysis and the network tells you whether the apple is good or bad.

In general, I think to answer your question, anytime a smart contract runs on data, there is no need to use neural networks. On the other hand if a smart contract runs on Big Data, then, arguably, you need a neural network to extract important info from the data, as the data itself is too complex.

In the examples above the neural network does not need to be confidential or encrypted in anyway, it could be a pre-trained network which is trusted by all participants.

I am not saying EVM is a perfect place to run neural networks, on the other hand making it some kind of an simple extension to EVM/Solifity would draw many developers. Another possibility is to run a totally different thing and then feed the results into Ethereum somehow …

- **Moving computations rather than moving data!**
- **Convolutional Neural Network**

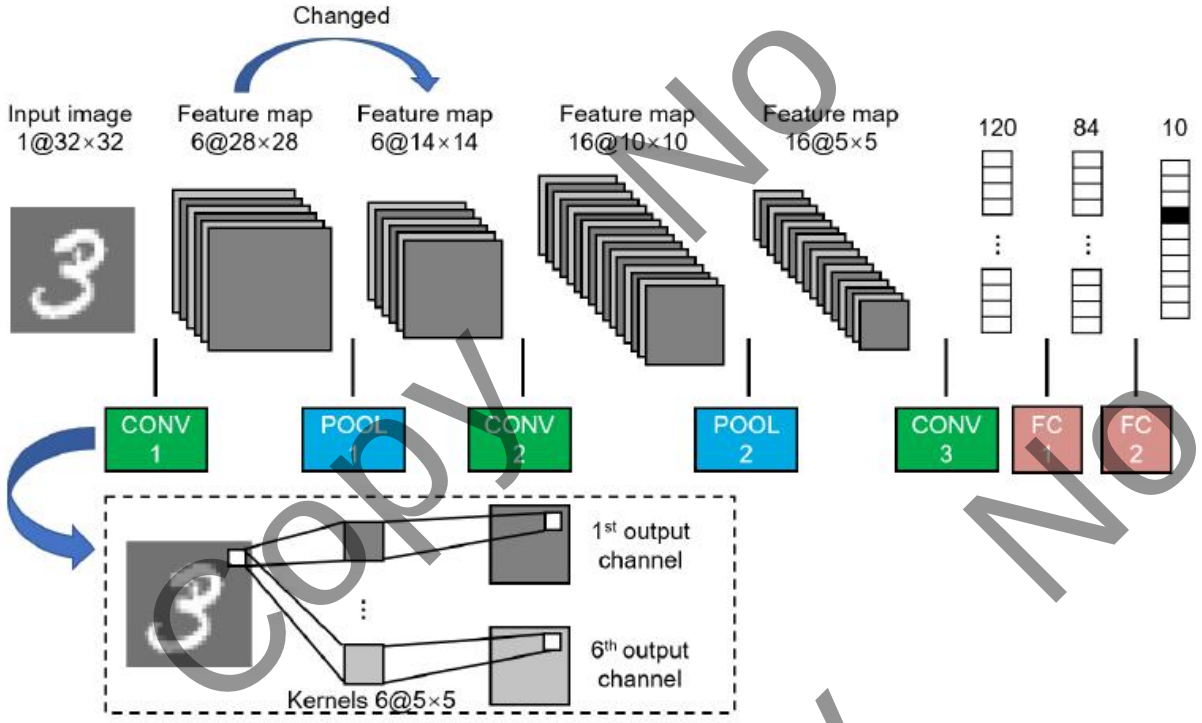
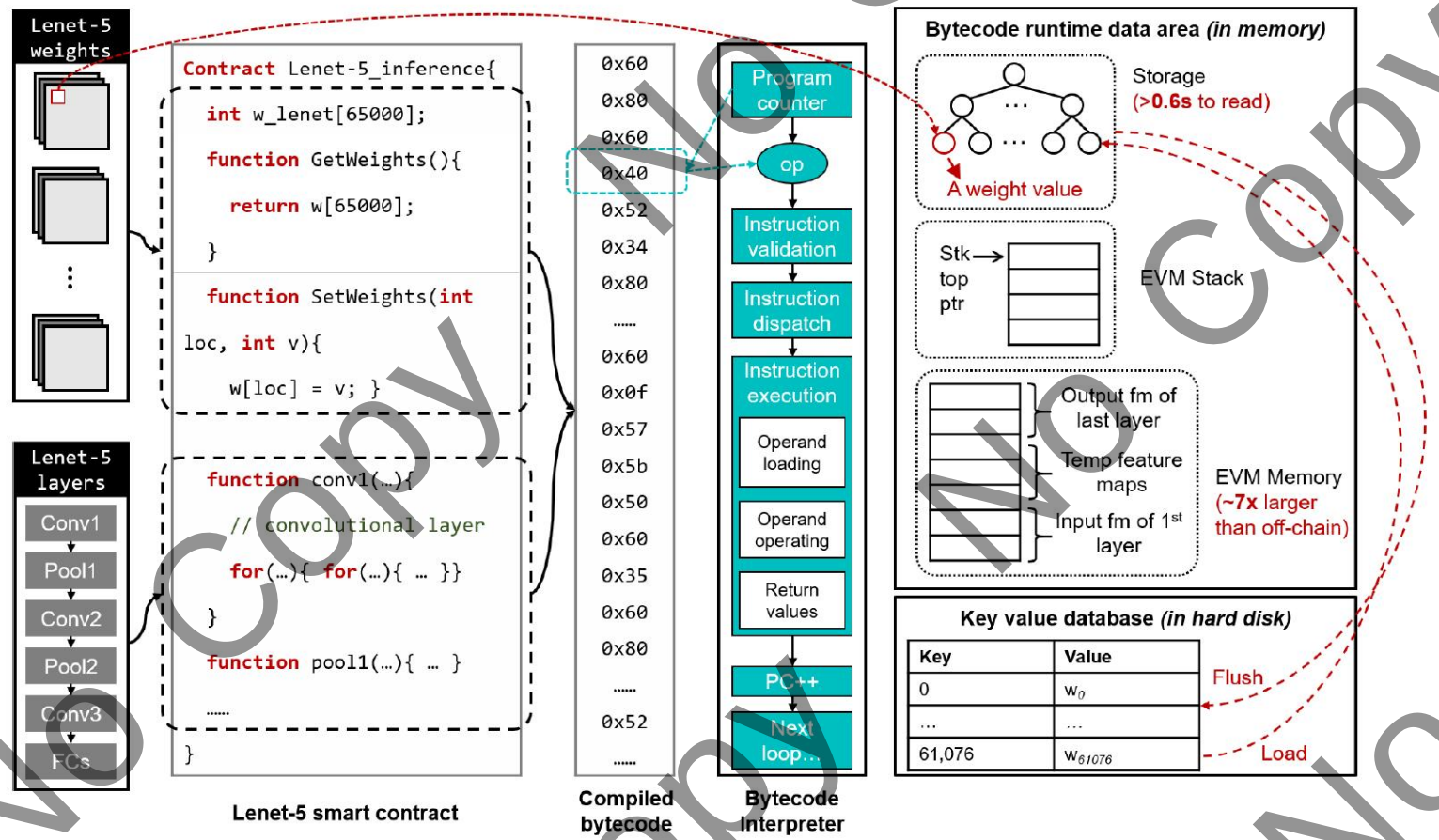


Fig. 1: LeNet-5 architecture of inference.

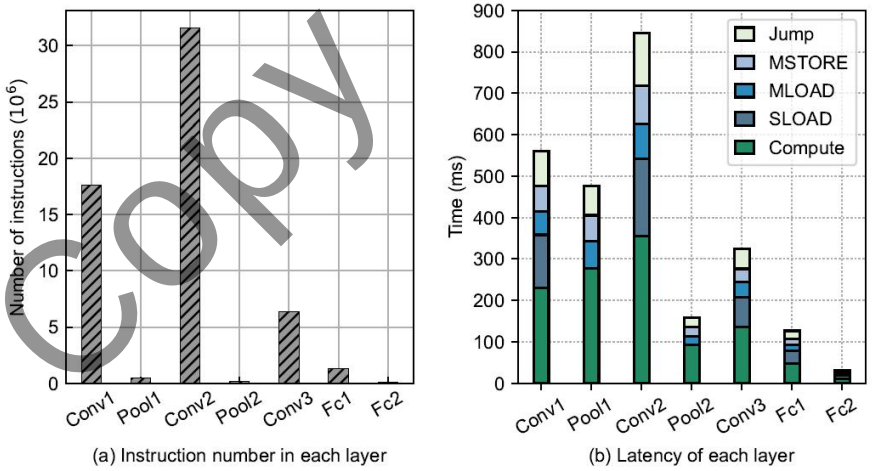
- **经过某层计算后，计算前后的特征图大小一般会不同**
  - 例如，卷积层增大，池化层减小
- **各同类型层计算逻辑基本相同**
  - 例如，Conv1和Conv2
- **权重获取和计算有先后顺序，且各通道相互独立**
  - 例如，第1通道和第6通道

科研工作（2/2）SmartVM

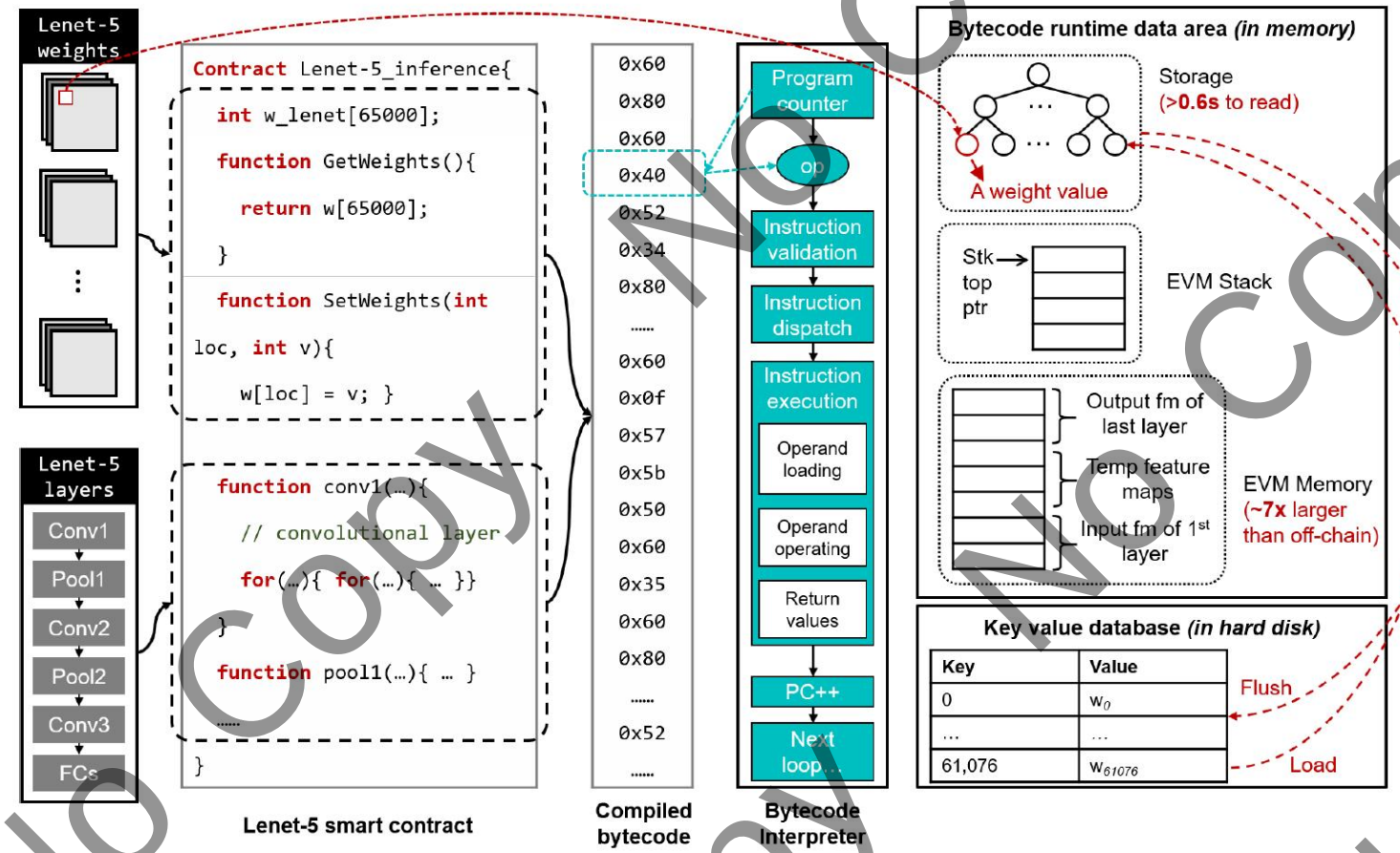
- **Moving computations rather than moving data!**
- **Problems of on-chain CNN inference**



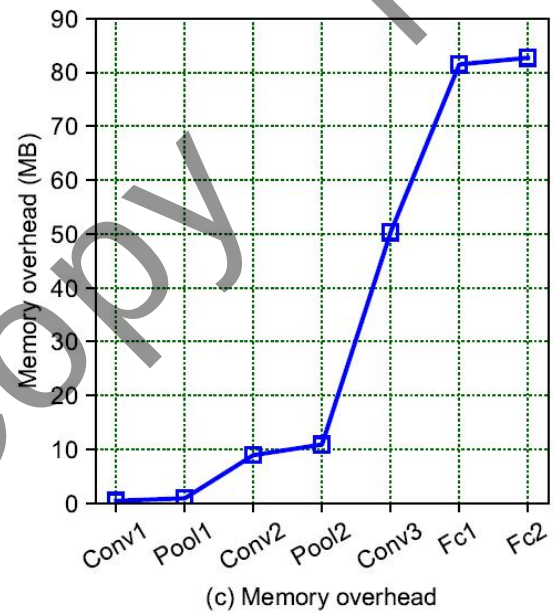
挑战一：通用指令集及特定指令缺失，基于栈的虚拟机需要频繁处理栈数据，导致CNN推理需执行上千万条指令



▶ Problems of on-chain CNN inference

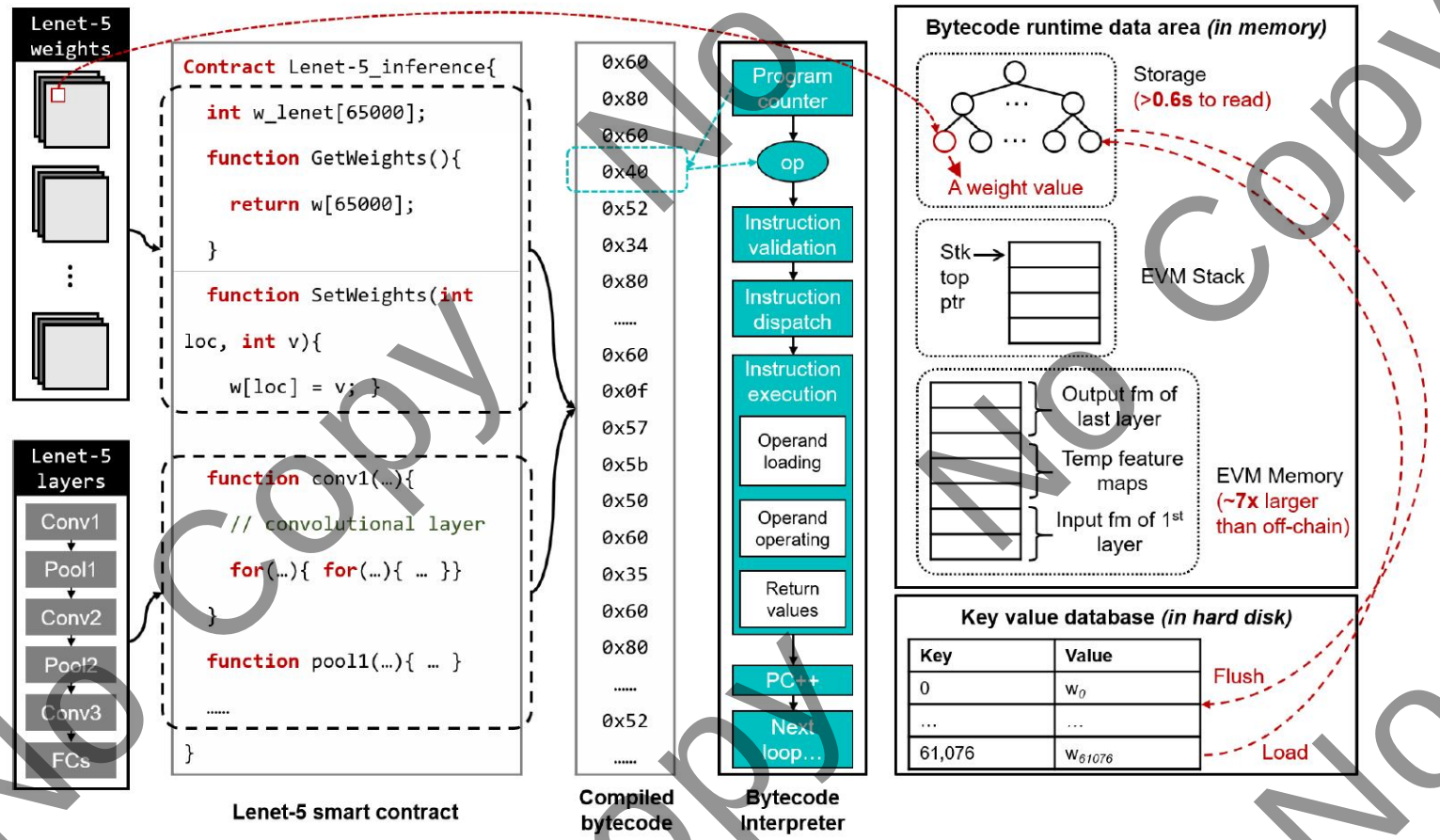


挑战二：内存管理欠缺，仅包括一个空闲指针管理(0x40)，导致CNN推理运行时大数据无法被有效管理

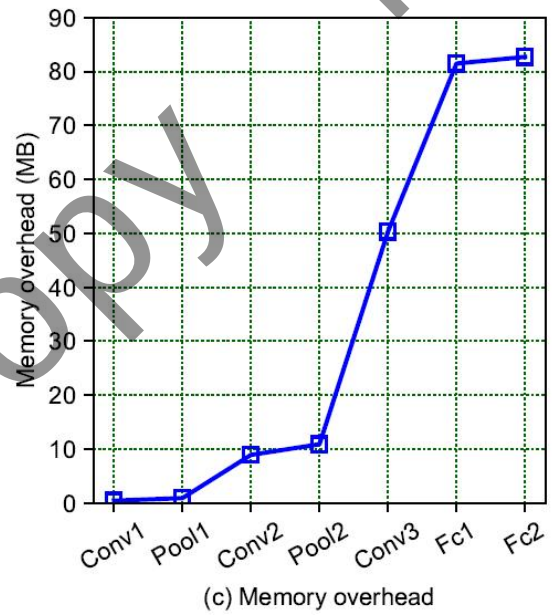


科研工作（2/2）SmartVM

- Moving computations rather than moving data!
- Problems of on-chain CNN inference

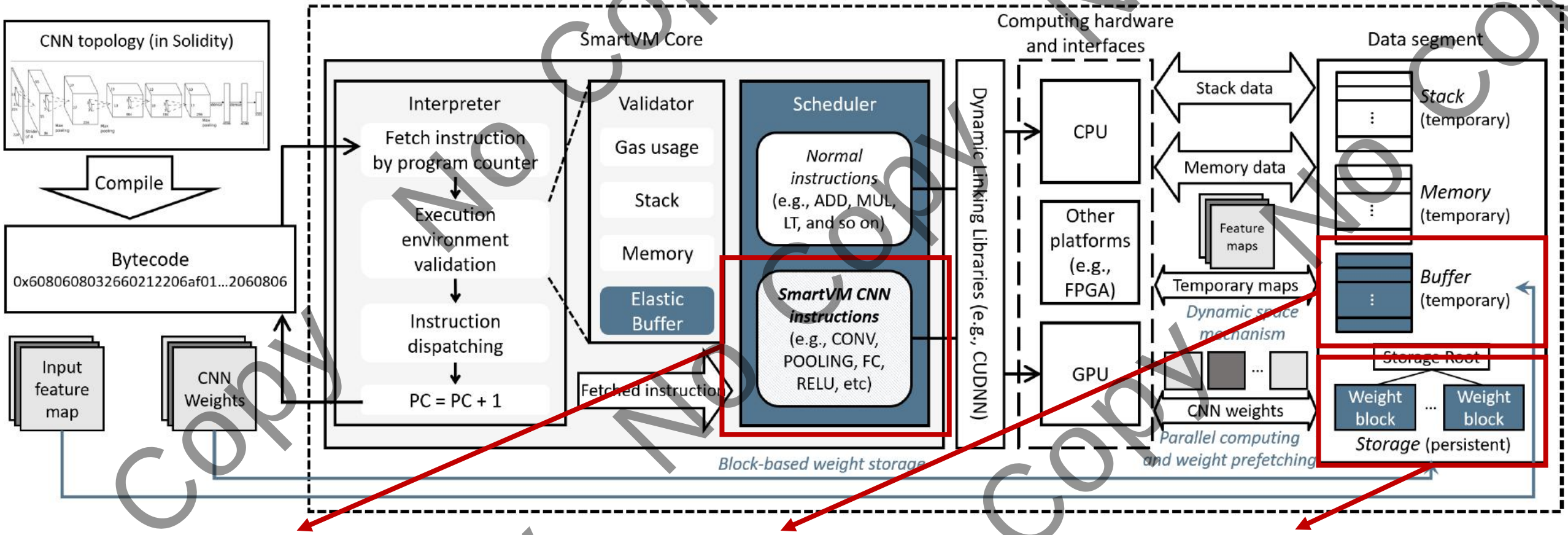


挑战三：串行(for)单平台计算(CPU)无法匹配CNN并行度高的特点和需求



科研工作（2/2）SmartVM

▶ SmartVM Overview



- ◆ CNN-oriented Instructions
- ◆ Dynamic Memory Management Method
- ◆ CNN Weight Prefetching and Parallel Computation

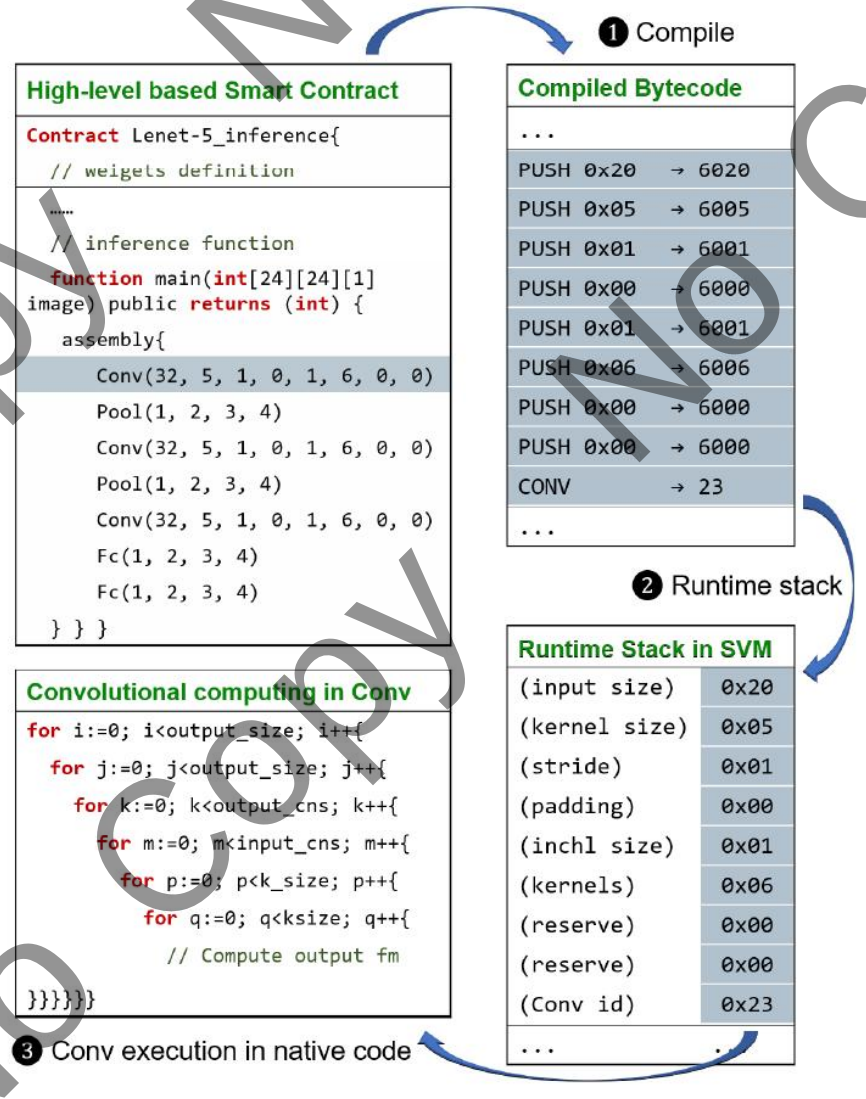
▶ CNN-oriented Instructions

Type	Name	Opcode	Description	Stack required (Key arguments)
Computation (Convolution)	CONV_SING	0x21	Implement single channel convolution	8 (Kernel, Output channel, Stride)
	CONV_MUL	0x22	Implement multi-channel convolution	8 (Kernel, Output channel, Stride)
	CONV_3D	0x23	Implement 3D convolution	8 (Kernel, Output channel, Stride)
	CONV_TPD	0x24	Implement transposed convolution	8 (Kernel, Output channel, Stride)
(Pooling)	POOL_MAX	0x25	Implement max pooling	5 (Stride, Input channel)
	POOL_AVG	0x26	Implement average pooling	5 (Stride, Input channel)
	POOL_OL	0x27	Implement overlapping pooling	5 (Stride, Input channel)
(Full connected)	FULL_CON	0x28	Implement full connected layer	5 (Input channel, Output channel)
	MAT_MUL	0x29	Implement matmul	2 (Addresses of two matrix)
(Active)	ACT_SM0	0x2a	Implement softmax function	1 (Value)
	ACT_SM1	0x2b	Implement Sigmoid function	1 (Value)
	ACT_RL	0x2c	Implement ReLU function	1 (Value)
	ACT_TANH	0x2d	Implement Tanh function	1 (Value)
(Buffer)	BUF_SCL0	0x2e	Increase Buffer's data with specific times	1 (Specific times)
	BUF_SCL1	0x2f	Reduce Buffer's data with specific times	1 (Specific times)
	BUF_BIAS	0x30	Add Buffer's data and bias	1 (Base address of bias)
Data transfer	MTOB	0x31	Transfer data from Memory to Buffer	2 (Data offset)
	BTOM	0x32	Transfer data from Buffer to Memory	2 (Data offset)
	BTOS0	0x33	Transfer data from Buffer to Stack	2 (Data offset, Size)
	BTOS1	0x34	Transfer data from Buffer to Storage	2 (Data offset, Size)
(Buffer set)	BUF_CLS	0x35	Clean Buffer's data	1 (Clean number)
	BUF_FIL	0x36	Fill Buffer's data with specific data	1 (Specific filled data)
	BUF_INIT	0x37	Initial Buffer with specific size	1 (Specific size)
	BUF_ALLO	0x38	Allocate specific size to Buffer	1 (Specific size)
	BUF_FREE	0x39	Free specific size from Buffer	1 (Specific size)
	BUF_COPY	0x3a	Copy a same Buffer	2 (Start and end pointers)

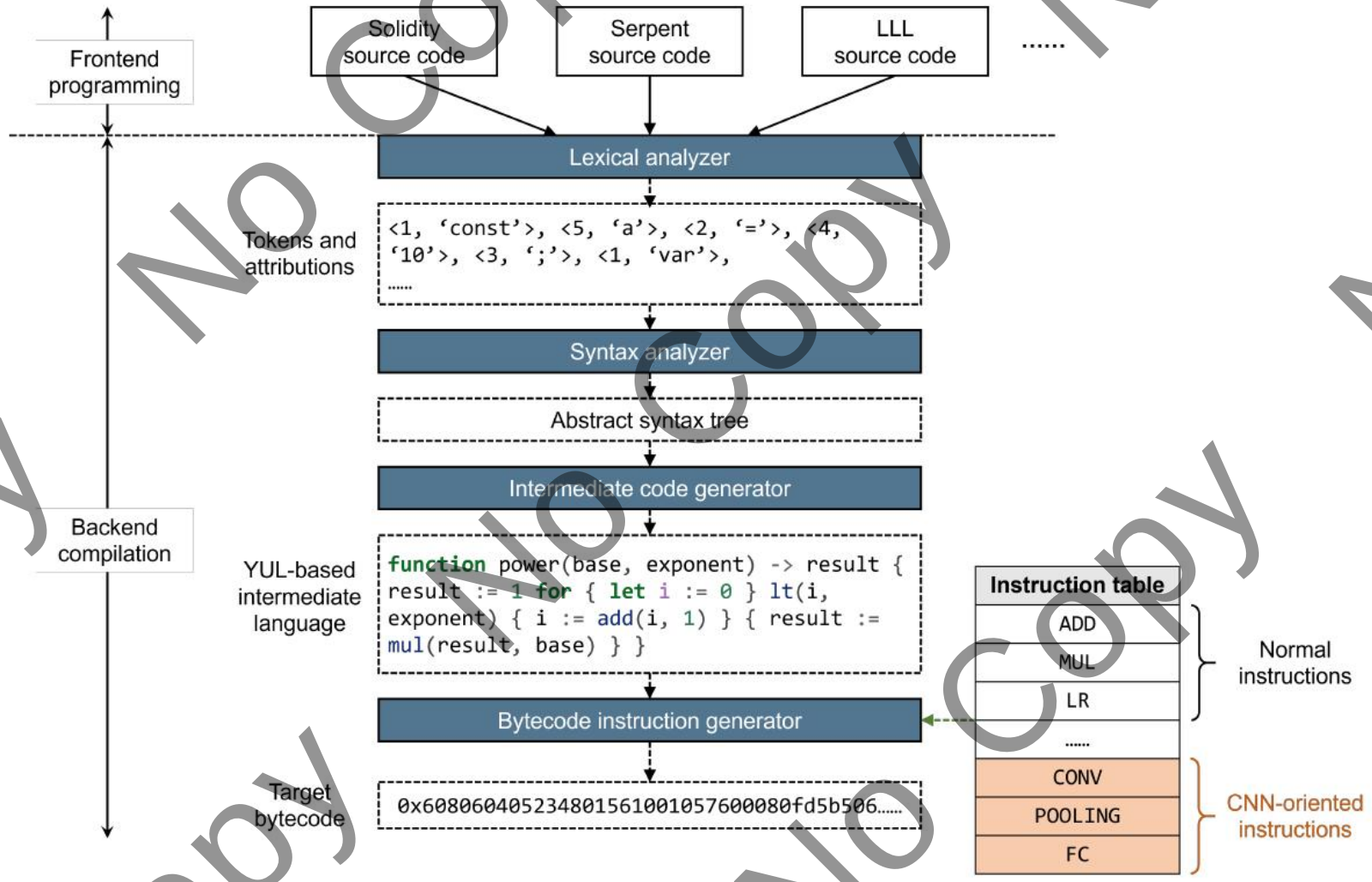
- 计算指令
  - Conv
  - Pool
  - FC
  - …
- 数据转移指令
  - Stg.
  - Stk.
  - Mem.
  - Buf.

► CNN-oriented Instructions Workflow

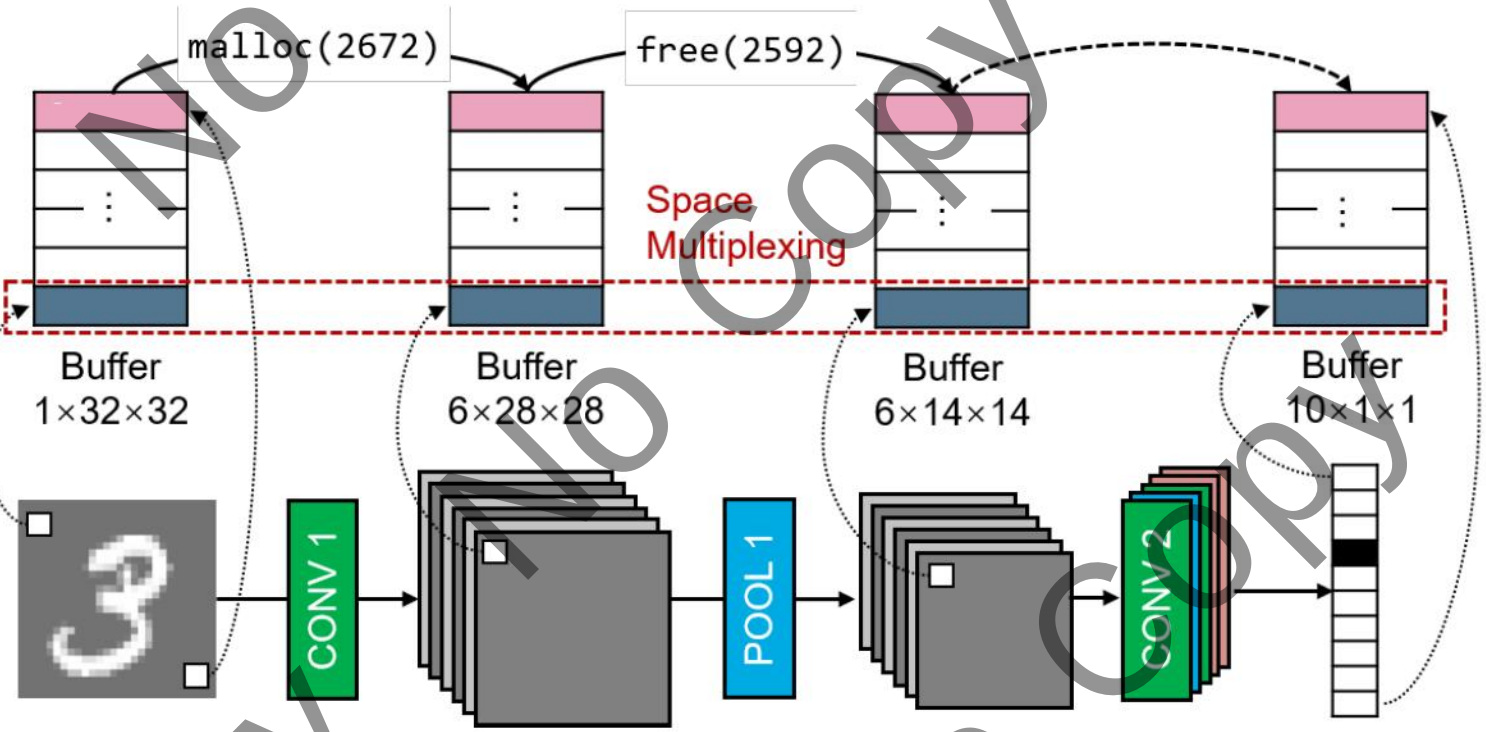
- ◆ 编译Implementation
  - ◆ 采取行内汇编的形式
  - ◆ 参数记录在栈中
  - ◆ N*Push + 1*CNNI
- ◆ 运行时参数入栈
- ◆ Native Code执行循环



► Compiler for CNN-oriented Instructions（编译层次）



► Dynamic Memory Management Method



▶ CNN Weight Prefetching and Parallel Computation

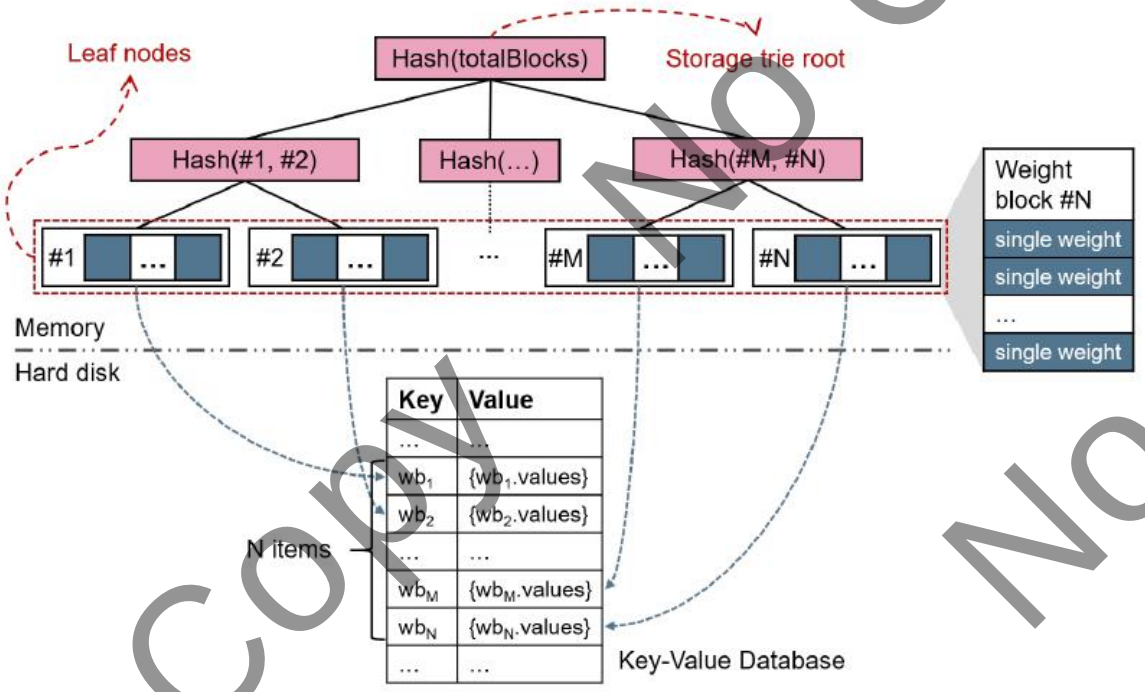


Fig. 9: The block-based weight storage.

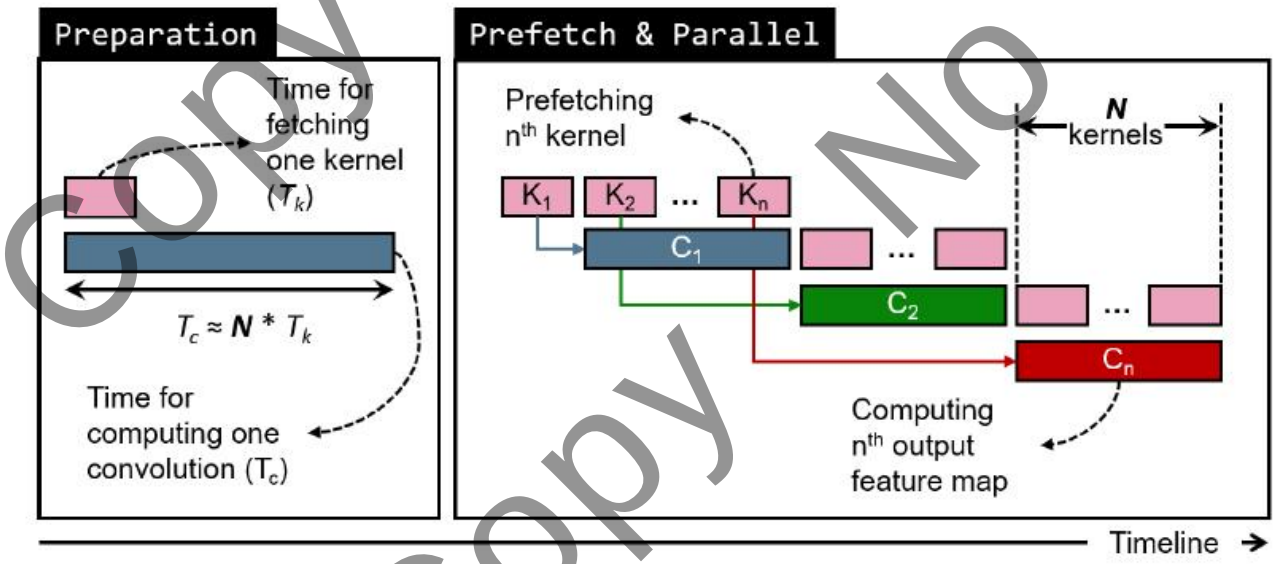


Fig. 10: The weights prefetching and parallel computation model in Conv instruction.

总结

- **确定研究层次和边界** -> 智能合约层，不涉及共识/网络/硬件
- **问题描述** -> 定性转变为定量，确切描述
- **解决方案** -> 对每个问题进行针对性地解答
  - 突出挑战
- **实验部分** -> 覆盖Motivation、Problems & Solutions

➢ 房耀政 2021级博士研究生
➢ 研究室邮箱：aics@nankai.edu.cn
➢ 个人邮箱：fyz@mail.nankai.edu.cn
➢ https://www.fangyaozheng.com/