Journal of Industrial Information Integration 41 (2024) 100674

Contents lists available at ScienceDirect

Journal of Industrial Information Integration

journal homepage: www.elsevier.com/locate/jii

Review article

An overview of blockchain smart contract execution mechanism

Yang Liu *, Jinlong He, Xiangyang Li, Jingwen Chen, Xinlei Liu, Song Peng, Haohao Cao, Yaoqi Wang

Henan University of Technology, Zhengzhou, China

ARTICLE INFO

Keywords:
Blockchain
Smart contracts
Contract execution performance
Concurrent execution
Parallel execution
Industry 4.0

ABSTRACT

Lacking a secure and reliable execution platform has hindered smart contracts from being applied in depth and width. Blockchain, as a decentralized, tamper-proof, and non-repudiation distributed computing platform, ensures that smart contracts are strictly enforced according to their established logic and not manipulated in the execution process. Programmability on the blockchain platforms is achieved through smart contracts to expand the application areas of blockchain extensively. However, current public blockchain and consortium blockchain platforms differ greatly in the execution mode of smart contracts, both of which have problems of how to execute smart contract efficiently and flexibly. This article provides a comparative analysis of mainstream blockchain platforms from the perspective of smart contracts execution mechanism, and summarizes the research status quo, challenges, and development trends from the perspective of smart contracts deployment and execution. We also discussed four perspectives on how smart contracts empower Industry 4.0 and explored the potential transition from digital industry to trusted industry through case studies. This article provides a distinct angle of view to survey smart contracts, which may count for the follow-up work of performance improvement for the blockchain platforms.

Contents

* Corresponding author.
E-mail address: liu_yang@haut.edu.cn (Y. Liu).

https://doi.org/10.1016/j.jii.2024.100674
Received 22 December 2022; Received in revised form 8 August 2024; Accepted 13 August 2024
Available online 24 August 2024

1. Introduction

The concept of smart contracts was initially introduced by Nick Szabo, who defined smart contracts as "A smart contract is a set of promises, specified in digital form, including protocols within which the parties perform on these promises" [1]. These contracts are intended to self-execute when predefined conditions are met. They were conceived as a way to bring automation to legal and contractual agreements [2]. At that time, due to the lack of a secure and trustworthy execution environment, the execution of smart contracts could not be guaranteed to be secure and reliable, causing the development of smart contracts to remain stagnant. The emergence of blockchain technology perfectly solved this problem.

Blockchain technology originates from digital crypto-currency systems [3] and is known as the distributed ledger technology [4]. As a technology of trust, its emergence provided feasibility support for the development of smart contracts. The trustworthy execution of smart contracts can be achieved through the underlying blockchain infrastructure [5]. With the help of smart contracts, blockchain has been endowed with programmability, thus meeting the complex requirements of various real-world application scenarios. A landmark event was the introduction of Ethereum [6]. Since then, blockchain and smart contracts have become inseparable.

As an abstraction of user applications on blockchain platforms, smart contracts possess unique execution characteristics distinct from other user task paradigms, such as automatic execution, multi-party verification, and network-wide propagation. The contract content can range from complex transaction logic based on business contracts to simple state data modifications or transaction compliance verifications. Smart contracts serve as the hub for interactions between various applications and the blockchain ledger, responsible for translating user business logic into concrete operations [7]. They expand the application scenarios of blockchain from cryptocurrencies to various fields requiring multi-party collaboration or trust assurance.

Smart contracts in blockchain systems can now execute complex logic and achieve a wide range of functionalities [8], far beyond the scope initially defined by Szabo. In blockchain systems, smart contracts can be written in various programming languages and deployed as scripts or code [9]. The current generation of smart contracts can be viewed as distributed ledger programs with predefined rules, states, and conditional responses, enabling information exchange, value transfer, and asset management. Additionally, smart contracts can extend to serve as alternatives to legal contracts, supplements to law, or intelligent software with self-verification and automatic execution capabilities [10].

However, the current development of blockchain and smart contracts faces several bottlenecks, particularly in the area of blockchain performance. Thus, achieving efficient execution of smart contracts is one of the research hotspots in the field of blockchain.

We have reviewed the existing surveys on smart contracts and observed that their focus is primarily on smart contract platforms, applications, languages, and vulnerability detection (Table 1). However, these articles lack an analysis of the complete lifecycle and execution patterns of smart contracts. Using the lifecycle of smart contracts as a framework, we employed an exploratory research approach utilizing qualitative research tools in a structured manner. This enabled us to investigate potential issues and solutions in the execution process of smart contracts. Our study particularly examines recent research efforts aimed at optimizing the execution performance of smart contracts.

Table 1
Taxonomy of existing smart contract-related surveys.

Year	Ref	Topic
2018	ref [11]	Smart contract languages
2018	ref [12]	Architecture, applications, and future trends
2019	ref [13]	Challenges and opportunities
2020	ref [14]	Challenges, advances and platforms
2021	ref [15]	Paradigms, tools, and systems
2021	ref [16]	Framework and its application
2021	ref [17]	Smart contract languages
2021	ref [18]	Formal specification and verification
2021	ref [19]	Access control
2022	ref [20]	Smart contract vulnerabilities
2022	ref [21]	Application
2023	ref [22]	Platforms, applications
2023	ref [23]	Smart contract vulnerabilities
2023	ref [24]	Security threat of smart contracts

We compared the current mainstream public blockchains and consortium blockchains, analyzing the reasons behind the low execution efficiency of smart contracts. While consensus optimization can improve the execution efficiency of smart contracts to some extent, its impact is limited. For example, even though Ethereum, a well-known blockchain platform, has transitioned its consensus mechanism from PoW (Proof of Work) to PoS(Proof of Stake) [6], it still fails to meet performance demands. Therefore, fundamentally enhancing the performance of contract execution requires addressing the issue of the execution model of smart contracts. The sharding strategy that Ethereum is about to adopt, as well as Hyperledger Fabric [25], which performs better in terms of performance, can well illustrate this issue.

Although existing research has explored this issue through concurrent transaction execution, it remains in its early stages and introduces new challenges, such as transaction concurrency conflicts. To address these issues, we conducted a comprehensive analysis of the latest high-quality research outputs that have not been included in other existing survey articles.

With the advancements in sensors, big data, and artificial intelligence, the digital development of Industry 4.0 is becoming increasingly sophisticated [26]. In industrial production, data is generated alongside goods, and this data needs to be trusted by enterprise managers to ensure production quality. Simultaneously, the data must be trusted by consumers to ensure that the products they purchase meet quality standards. Furthermore, regulatory bodies must trust the data to ensure compliance with production processes. However, the data generated by digital manufacturing currently lacks the crucial element of data

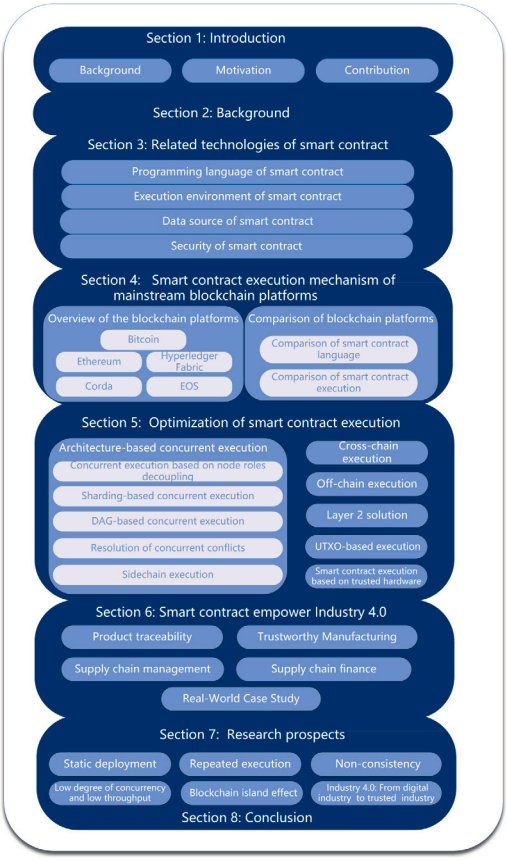

Fig. 1. The structure of this article.

trustworthiness. In this article, we illustrated how smart contracts and blockchain technology can ensure data trustworthiness.

We analyzed the relevant technologies involved in smart contracts and summarized various aspects of performance optimization for smart contract execution. Additionally, we briefly introduced the application scenarios of smart contracts and blockchain technology for building a trusted industrial environment. These application scenarios encompass not only academic achievements but also real-world cases.

We believe that this review can provide a more comprehensive perspective for the research on optimizing the performance of smart contracts, to promote its further development and serve real-world application scenarios. Additionally, we also hope that this technology can better serve industrial production and promote the development of Industry 4.0 from digital chain to trust chain.

The total number of articles considered in this study is 201. As of August 2024, the surveyed literature databases included Scopus, Web of Science, and IEEE, arXiv with supplementary literature retrieved via Google Scholar. The structure of the article is shown in Fig. 1. The relevant terms that appear in this article are listed in Table 2.

The main contributions of this article are as follows:

- We analyze the relevant technologies and challenges involved in each stage of the smart contract lifecycle from the perspective of smart contract execution.
- We compare the execution modes of smart contracts in current mainstream public blockchains and consortium blockchains, identifying the deficiencies in the existing execution models.
- We conduct a comprehensive classification and analysis of research related to the optimization of smart contract execution models.

Table 2
Acronyms and their meanings.

Acronym	Explanation
2PL	Two-Phase Locking
API	Application Programming Interface
BFT	Byzantine Fault Tolerance
DAG	Directed Acyclic Graph
Dapps	Decentralized Applications
DoS	Denial of Service
EVM	Ethereum Virtual Machine
FIFO	FIFO First In, First Out
HMM	Hidden Markov Model
IoE	Internet of Everything
IIoT	Industrial Internet of Things
IoT	Internet of Things
JVM	Java Virtual Machine
MVCC	Multi-Version Concurrency Control
OCC	Optimistic Concurrency Control
P2P	Peer-to-Peer
PBFT	Practical Byzantine Fault Tolerance
PoS	Proof of Stake
PoW	Proof of Work
RFID	Radio Frequency Identification
SCF	Supply Chain Finance
SCU	Smart Contract Unit
SGX	Secure Guard Extensions
SMEs	Small and Medium-sized Enterprises
TPS	Transactions Per Second
UTXO	Unspent Transaction Output

- We analyzed the empowerment provided by data trustworthiness to Industry 4.0 from four perspectives: product traceability, trustworthy manufacturing, supply chain management and supply chain finance. These application scenarios encompass not only academic achievements but also real-world cases.
- Based on the discussion and analysis of existing work, we further elaborate on the current research issues and propose future research directions.

The main contents of this article are organized as follows:

Section 1 introduces the concept of smart contracts and their connections with blockchain. Section 2 provides a brief introduction to the fundamentals of blockchain. Section 3 offers an overview of the relevant technologies associated with smart contracts. Section 4 delves into the execution mechanisms of smart contracts on mainstream blockchain platforms. Section 5 presents the current research landscape regarding the optimization of smart contract execution is summarized. Section 6 outlines four aspects of how smart contracts can empower Industry 4.0 and presents three real-world case studies. Section 7 provides a future research prospects Section 8 concludes this article.

2. Background

Blockchain originated from Bitcoin [27]. It integrates cryptography, distributed consensus, P2P networks, and smart contracts. It employs a block-chain data structure to verify and store data, uses distributed consensus algorithms to generate and update data, ensures the security of data transmission and access through cryptographic methods, and utilizes smart contracts for automated data programming and operations, creating a novel distributed infrastructure and computing paradigm.

A simple block data structure, as shown in Fig. 2, consists of two parts: the block header and the block body. It involves technical elements such as chain structures, hash algorithms, Merkle trees, and timestamps.

Blockchain can be seen as a new paradigm [28] of distributed computing with the characteristics as decentralization, tamper-proof, traceability, and high trustworthiness.

Traditional databases are centrally deployed within the same cluster and managed by a single entity. In contrast, blockchain is decentralized,

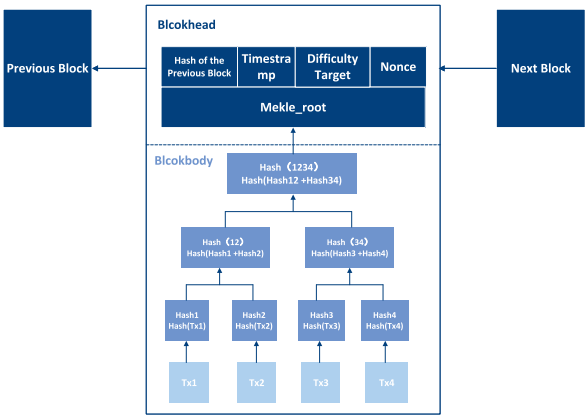

Fig. 2. Schematic Diagram of Block Data Structure.

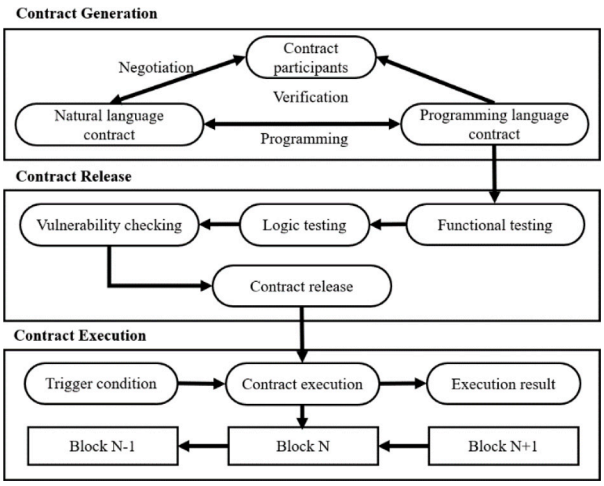

Fig. 3. A life cycle of Smart Contract.

with no central node, and is jointly managed and maintained by multiple participants. Each participant can provide a node and store data on the chain, achieving fully distributed multi-party information sharing. In a decentralized blockchain network, attacking a single node cannot control or disrupt the entire network. Blockchain applications do not rely on centralized institutions, enabling distributed data recording, storage, and updating. Decentralization is the most prominent and essential feature of blockchain.

(2) Tamper-proof.

Blockchain uses cryptographic principles to record data on the chain, with each block containing the timestamp of the previous block and arranged in chronological order. This endows blockchain with the characteristic of being tamper-proof or making tampering exceedingly costly. Tamper-proof means that once data is written onto the blockchain, no one can easily alter it without authorization. The high cost of tampering arises because one would need to control more than 51% [29] of the system's nodes to alter blockchain information. Given the numerous nodes in the blockchain system, achieving such widespread collusion is extremely expensive. This ensures the integrity, authenticity, and security of the data on the blockchain.

(3) Traceability.

The blockchain stores all transaction data since the system's inception, and based on these tamper-proof log-type data, it is easy to restore and trace all historical operations, facilitating auditing and supervision by regulatory agencies. The data on the blockchain includes the dimension of time, and the real-time recording of block data onto the chain is a crucial guarantee for the traceability of blockchain.

(4) High Trustworthiness.

Blockchain is a highly trustworthy database where participants can conduct peer-to-peer transactions without needing mutual trust or a trusted intermediary. Each transaction on the blockchain requires the sender's signature and must reach consensus across the network before being recorded on the blockchain. Once a transaction is written, it is immutable and undeniable by anyone.

Based on these characteristics, blockchain technology is suitable for various scenarios that require decentralized trust and stable system applications. It offers unique advantages in enhancing data credibility, establishing multi-party trust, supporting transparent regulation, and enabling cross-platform flow of digital assets. The functions of notarization, arbitration, and collaboration will no longer rely on the center or people to be carried out but automatically be executed by trusted machines according to an agreement recognized by all parties. It has become a new computing paradigm and collaboration model for establishing trust in an untrustworthy environment at low cost [30].

Blockchain is not only applicable to financial application scenarios, but also to many vertical fields including supply chain management, deposit certificate approval, product traceability, data sharing, livelihood services, and the Internet of Things [31] and etc. Blockchain can meet the business needs of unanimous bookkeeping among multiple distrust institutions. And it is considered as one of the next-generation basic protocols for global credit authentication and value transfer, which has a great impact on changing the operation rules of application scenarios in various industries. It is also consider as one of the indispensable core technologies for future digital economy development and modern trust system construction.

3. Related technologies of smart contract

A smart contract is a set of code consisting of executable functions that automatically determine whether the trigger conditions for a smart contract are met based on external trusted data sources and the state of world. And a new transaction is generated when a function is executed, and the states of the smart contract are updated [32]. Any node in the blockchain network can initiate a transaction. The execution of smart contract on each node should ensure the certainty of the result, in this way ensuring the correctness of user output.

The whole life cycle of smart contracts consists of three consecutive phases of contract generation, contract deployment and contract execution, as shown in Fig. 3.

(1) Generation of smart contracts

Contract participants need to negotiate business logic and translate it into natural language contracts. Next, the signed contracts in natural language must be programmed into smart contracts represented with program language that can be executed on distributed nodes. The natural language contracts and the program language contracts must then be verified by contract participants before moving on to the next deployment phase.

(2) Deployment of smart contracts

Contract testing and contract release make up the contract deployment phase. Contract testing is crucial because smart contracts cannot be changed once they are deployed on the chain. Contract function testing, logic testing, and vulnerability detection make up the bulk of the testing process. Only smart contracts that completely pass the above tests will be released to the blockchain platform. Contracts release means there could have specified policies to place smart contracts on nodes for following execution.

(3) Execution of smart contracts

Smart contracts can be invoked by nodes for execution. Once the transaction meets the trigger condition of the smart contract, the contract will be executed on distributed nodes. The execution results

will be packaged into transactions to generate a block, which will be attached to the blockchain ledger after consensus is reached by the consensus components.

3.1. Programming language of smart contract

Smart contracts extend the application of blockchain from cryptographic digital currency to any other domains that require multiparty collaboration and trust transfer as well.

As the pioneering blockchain application system, Bitcoin employs a stack-based scripting language that is non-Turing complete, resembling Reverse Polish Notation, and lacks the capability to execute intricate transactional logic [33]. In contrast to versatile general-purpose scripting languages, Bitcoin's scripting language is confined to accessing a single stack as its sole memory space. This scripting language plays an essential role in verifying Bitcoin transactions, executed by Bitcoin clients. The set of opcodes within Bitcoin's scripting language exhibits limited complex flow control capabilities. Transactions involve a combination of locking and unlocking scripts for execution and verification, with transaction termination achievable within a specified step count [34].

Upon validation of a Bitcoin transaction, the unlock script within each input value is executed alongside its corresponding locking script to assess whether the transaction meets payment conditions. However, the introduction of Turing-complete scripting introduces the potential for issues stemming from loops. For instance, allowing loops could lead to a script program with an infinite loop, causing validation nodes to be ensnared in an endless cycle, thereby exposing the system to Denial-of-Service (DoS) attacks and presenting a substantial vulnerability.

In comparison to more sophisticated programming languages, Bitcoin's scripting language possesses limited expressive power [34], lacks intuitive qualities, and the composition and comprehension of scripts may prove intricate. Bitcoin's scripting language is Turing incomplete, signifying its inability to manage all types of computational tasks. Consequently, unlike Turing-complete programming languages, Bitcoin's scripting language cannot facilitate complex computations like loops and recursion. Moreover, due to the constrained nature of Bitcoin's scripting language and its stack-based execution, transaction verification may require additional time. Complex scripts necessitate supplementary computational and validation steps, resulting in reduced transaction processing speeds.

Presently, Ethereum [6] encompasses three primary programming languages for smart contracts: Solidity [35], Vyper [36], and Yul [37]. Among these, Solidity prevails as the most favored and extensively supported language within the Ethereum ecosystem, boasting a substantial developer base and robust community backing. Serving as a contract-oriented high-level programming language, Solidity is specifically tailored for writing smart contracts, facilitating the construction and deployment of decentralized applications (Dapps) on diverse blockchains. Built upon C++ and JavaScript influences, Solidity exhibits a syntax akin to JavaScript, thereby facilitating ease of learning and mastery, while also possessing its unique syntax and applications [4].

Equipped with its own data types and structures, Solidity operates as a statically typed, strongly typed language, conducting type checks prior to executing source code. This necessitates programmers to declare the data type of each variable before compiling the code, with variables' data types immutable and incapable of conversion to alternative types. Influenced by C++ programming paradigms, Solidity supports inheritance, libraries, and intricate user-defined types. Concurrently, Solidity embraces an object-oriented programming paradigm, enabling modularity and reusability within smart contracts. The Ethereum ecosystem for smart contract development tools and libraries is extensive, providing Solidity developers with a wealth of resources and support.

Despite the manifold merits of Solidity, certain limitations persist. For instance, a notable deficiency pertains to its error-handling mechanism. Solidity exhibits relatively limited error-handling capabilities, mandating developers to autonomously implement error handling and exception mechanisms. Furthermore, owing to the immutable nature of Solidity smart contracts on the blockchain, vulnerabilities or errors can potentially yield severe consequences, underscoring the paramount importance of code security. To address the potential issue of infinite loops intrinsic to Turing-complete languages, Ethereum adopts a "gas" mechanism [6], compelling smart contracts to pay execution fees to mitigate concerns such as infinite loops and safeguard system stability.

Vyper [36], akin to Python, deliberately offers fewer features than Solidity to enhance contract security and auditability, characterized by concise and lucid syntax. Vyper is non-Turing complete and eschews support for modifiers, inheritance, inline assembly, function overloading, operator overloading, recursive calls, unlimited-length loops, and fixed-size binary floats. While the absence of these features augments contract security and auditability, developers are tasked with additional efforts to address these constraints. As a relatively novel language, Vyper is still evolving, implying that its feature set is relatively narrower compared to Solidity.

Yul [37], an intermediate language, is compilable into various backends and serves as an intermediate representation of code compiled from higher-level languages like Solidity. This design enables Yul's deployment across compilers for different backends, endowing it with heightened flexibility and adaptability to diverse target platforms. One of Yul's design objectives is to uphold readability. Even if code is generated by compilers for Solidity or other high-level languages, Yul-written programs should remain relatively comprehensible and amenable to human inspection, formal verification, and optimization.

Within Hyperledger Fabric [25], smart contracts are referred to as chaincode, specifically chaincode. Chaincodes can be written using programming languages such as Go, Node.js, or Java, enabling the implementation of various intricate business logic [38]. Go language stands as one of the earliest supported chaincode writing languages within Hyperledger Fabric, distinguished by its efficient execution performance and concurrent support. Chaincodes authored in Go language can achieve heightened smart contract execution performance within Hyperledger Fabric. Moreover, Go is a statically typed programming language, with its static type checks and compilation process aiding developers in detecting errors during the development phase.

Node.js [38], an event-driven and non-blocking I/O model-based JavaScript runtime environment, finds utility in the development of high-performance network applications. The dynamic nature and flexibility of JavaScript accelerate the development and iteration of chaincodes, rendering the process faster and more convenient. Writing chaincodes using Node.js can attract a broader participation of JavaScript developers in blockchain application development. However, compared to Go language, Node.js may exhibit lower performance when handling a significant volume of concurrent transactions.

Java finds widespread application in enterprise-level and large-scale system development. Employing Java for chaincode writing streamlines the integration of Hyperledger Fabric applications with existing Java projects. For teams or developers already familiar with Java, writing chaincode in Java is a more natural choice. Nonetheless, compared to Go and Node.js, Java chaincode might be more verbose, necessitating a balance between development convenience and code conciseness.

Furthermore, several other blockchain platforms, including EOS [39] and Corda [40], have introduced their own implementations of smart contracts, sharing functional similarities with the aforementioned three blockchain platforms. EOS, for instance, employs Go language as its primary smart contract development language, while Corda predominantly supports Java or Kotlin [41]. The Kadena platform [42] features smart contracts written in the verifiable language Pact, which possesses a restricted instruction set and is non-Turing complete.

Pact supports modular design and importability. The code structure of Pact enables rapid analysis and detection of any loops or recursion that could potentially trigger immediate faults and terminate all running code, thereby mitigating potential vulnerabilities in smart contracts to a certain extent. It finds relevance in high-security-demanding commercial transactions. Other smart contract programming languages like Rust and Move have also been adopted by various blockchain platforms, each capitalizing on its distinct attributes.

3.2. Execution environment of smart contract

The execution environment of smart contracts refers to a specialized computational context employed for the enactment of smart contracts within a blockchain network. Categorized according to their modes of implementation, the execution environments for smart contracts can be delineated into three distinct classifications: the stack-based execution environment, the sandboxed execution environment, and the execution environment based on trusted hardware and enclaves.

The stack engine possesses the characteristics of lightweightness, rapid execution, and high security, yet it provides only rudimentary instructions for smart support. Bitcoin employs a stack engine for transaction execution. In a transaction, the payer employs a locking script within the transaction script to establish the lock on the Unspent Transaction Output (UTXO). An unlocking script is then utilized to demonstrate its fulfillment of the locking condition, essentially proving ownership over a specific transaction output. Transaction scripts are executed and verified for validity within a stack local to all miner nodes.

Sandbox execution environments are categorized into two types: virtual machines (such as EVM, JVM) and containers (such as Docker). The overarching goal of both types is to ensure the execution of contract code within a sandbox, thereby isolating and constraining the resources employed by the contract.

The EVM is a stack-based virtual machine that furnishes a versatile environment for executing smart contracts, enabling the handling of intricate logic and conditions, thereby positioning Ethereum as one of the most feature-rich platforms for smart contracts. The EVM has garnered substantial developer interest, fostering an abundant ecosystem of smart contracts and DApps, with a multitude of readily available contracts for reuse. The EVM supports multiple programming languages, with Solidity being the most prevalent, alongside others like Vyper. This affords developers greater choice and flexibility, enabling them to engage in smart contract development using familiar languages.

Once a smart contract is deployed on the EVM, its code becomes immutable. Consequently, addressing vulnerabilities or errors in the contract may prove challenging and potentially lead to significant consequences. Additionally, the EVM faces limitations in directly accessing off-chain data, such as external APIs or data from other blockchains. This curtails the ability of smart contracts to interact directly with the external world.

Moreover, during Ethereum's Ethereum 1.0 phase based on the Proof-of-Work (PoW) consensus mechanism, the EVM encountered scalability issues when processing a high volume of transactions, resulting in transaction congestion and delays. This may impede Ethereum's ability to cater to the demands of large-scale applications under high load and require substantial gas fees. However, with the successful transition of the consensus mechanism from PoW to PoS, as well as the introduction of sharding [43] in the future, these issues are expected to witness significant improvements.

Fabric adopts Docker as the container for executing smart contract code, foregoing virtualization techniques, and enabling code execution directly on the underlying operating system. As a result, it achieves higher efficiency compared to the EVM. Docker containers furnish isolated execution environments for smart contracts, ensuring each contract operates within a distinct container. This isolation guarantees independent resources for each smart contract during execution, preventing mutual interference and enhancing security and stability. While Docker containers provide a certain degree of isolation, it remains imperative to address security vulnerabilities between containers, preventing potential vulnerabilities or attacks between them.

zmVM (Zilliqa Modified Virtual Machine) is a virtual machine deployed within the Zilliqa blockchain platform [44], tasked with executing smart contracts and processing computational tasks on the blockchain. Serving as a fundamental component of the Zilliqa blockchain, its primary objective is to establish a high-performance and highly secure environment for smart contract execution. Inspired by the EVM, zmVM has been tailored and optimized to accommodate the specific needs and characteristics of the Zilliqa blockchain.

Notably, zmVM introduces distributed smart contract execution, allowing multiple nodes to participate concurrently in computation and verification processes. This collaborative approach contributes to enhanced execution speed and throughput of smart contracts. The implementation of resource isolation ensures that distinct smart contracts remain insulated during execution, preventing mutual interference. Through parallel processing and sharding techniques, zmVM achieves high-performance smart contract execution, empowering the Zilliqa blockchain to manage substantial transaction volumes and computational tasks effectively.

In addition, zmVM features refined data storage and retrieval mechanisms, facilitating efficient access and manipulation of stored data within smart contracts. This optimization enhances the overall efficiency and effectiveness of data operations within the zmVM environment.

The execution environment of smart contracts based on trusted hardware and secure enclaves makes effective use of hardware-level security features, such as Intel's SGX(Secure Guard Extensions) [45] or ARM's TrustZone [46], to establish a protected execution environment. Within these environments, the code and data of smart contracts can be isolated and safeguarded, preventing unauthorized access, tampering, or observation.

In the realm of trusted hardware research, Fang et al. [47] introduced a concurrent execution framework utilizing SGX, achieving internal and inter-node concurrency for the first time. Each replica concurrently executes tasks assigned by the master node, obtaining trustworthy results through SGX. Subsequently, nodes acquire the execution results of other nodes through state replication to achieve consensus. To ensure the integrity and correctness of all data transmitted to the new exchange, they devised a novel method to efficiently generate Merkle proofs and parallelly verify data. Theoretical analysis and experimental results indicate that the proposed approach outperforms state-of-the-art solutions significantly.

3.3. Data source of smart contract

Blockchain oracles [48] serve as a bridge between the blockchain and real-world data, facilitating the secure and reliable integration of external data. All decentralized applications (DApps) that require interaction with external data sources necessitate the use of oracles [22]. As off-chain components, blockchain oracles utilize off-chain data as triggering conditions for smart contracts. They transmit this data to the blockchain via specific communication channels, such as platforms or sensors, thereby providing data services to smart contracts and enabling their responsiveness to the uncertainties of the real world.

Based on their architectural design, oracles can be categorized into two types: centralized single oracle mechanisms like Oraclize [49], and decentralized multi-protocol mechanisms like Chainlink [50] and DoS network [51]. For instance, in the Chainlink framework, user contracts initiate external data requests to the Oraclize contract, which in turn processes the external data and returns it to the user contract. However, research in the field of blockchain oracles is limited, and a definitive authoritative solution has yet to emerge. Pasdar et al. [52] conducted

research and analysis on how blockchain oracles provide feedback to blockchains and smart contracts. They categorized blockchain oracles into two types: reputation-based and voting-based oracles, elucidating how each design furnishes answers to smart contracts.

Muhlberger et al. [53] classified oracles into four fundamental modes based on the direction of data flow and the initiator of data exchange: pull-based inbound oracles, push-based inbound oracles, pull-based outbound oracles, and push-based outbound oracles. They delineated distinct data exchange scenarios for each type and analyzed the advantages and drawbacks of the four oracle modes. Quantitative analysis of the four oracle modes was conducted using two use cases on the Ethereum platform, revealing distinct performance and cost characteristics.

Blockchain oracles are employed to mitigate the risks of data tampering during transmission, creating a trusted conduit between providers and consumers of data, enabling untrusted parties to participate in value transfer. However, they cannot guarantee the authenticity of the data source itself. If the data source is flawed, the feedback provided to consumers can also be compromised. Given the potential manipulation, alteration, and malicious intent of externally imported data, oracle contracts should incorporate witnessing mechanisms, such as reputation-based algorithms or data attestation strategies, to ensure the credibility of external data sources.

3.4. Security of smart contract

Despite providing a naturally trusted and tamper-proof execution platform for smart contracts, blockchains are still susceptible to attacks [54,55]. The well-known case of "The DAO"[56] incident serves as an illustrative example. In June 2016, an attacker exploited the reentrancy vulnerability of a smart contract to steal approximately $60 million worth of Ether. As a result, Ethereum was compelled to execute a hard fork to mitigate the losses, while the attacker remained at large due to the anonymity of blockchain users [57]. This incident stands as one of the most prominent security events in Ethereum's history, leading to further forks within the Ethereum ecosystem. The occurrence of "The DAO" incident has heightened the scrutiny surrounding smart contract security. However, to this day, smart contract security concerns persist. According to the annual research report of SharkTeam, a leading blockchain security service team, in 2022, there were 116 security incidents caused solely by contract vulnerabilities globally, resulting in over 1.7 billion USD in financial losses [58]. Among these incidents, the majority, 38%, stemmed from logical vulnerabilities within contracts.

In this section, we approach the study of smart contract security from the perspective of the smart contract lifecycle, examining the security challenges that smart contracts will encounter throughout their entire lifecycle.

(1) The phase of smart contract generation

During the phase of smart contract generation, factors such as design flaws, non-standard contract code, or immature contract implementation languages can result in contract vulnerabilities. Once defective contracts are deployed or executed on the blockchain, they can lead to unpredictable consequences.

In the stages of smart contract design and coding, human-related factors such as incomplete understanding of smart contract functional requirements, inadequate coding skills, adverse incentive-driven behaviors, as well as inherent deficiencies in the smart contract programming languages themselves, can contribute to issues like abnormal, ambiguous, or redundant contract functionalities. For common and complex smart contracts, the definition of design patterns can facilitate automated smart contract code implementation, mitigating the risk of coding errors. However, the current development of smart contract design patterns remains immature, compounded by the diversity of smart contract platforms, resulting in limited applicability to specific scenarios.

Destefanis et al. [59] have proposed Ethereum smart contract design patterns, and Wohrer et al. [60] have introduced that design patterns can offer design guidelines for implementers, advocating the establishment of the discipline of blockchain software engineering to provide more specialized education for professionals in the field.

(2) The phase of smart contract deployment

Once smart contracts are deployed on the blockchain, they become immutable, making it imperative to conduct security checks on finalized smart contract code before deployment to enhance contract safety. Currently, a variety of computational programs and automated tools exist for auditing smart contract code, preemptively identifying potential vulnerabilities, errors, and security issues. These tools serve to assist developers, auditors, and security experts in assessing the security of smart contracts through different approaches, thereby mitigating risks that contracts may face.

From a code-level perspective, smart contract vulnerabilities primarily encompass issues such as Reentrancy vulnerabilities, Integer Overflow/Underflow, Arbitrary Address Write vulnerabilities, and Denial of Service vulnerabilities. Detection schemes for smart contract vulnerabilities can be classified into static analysis tools, dynamic analysis tools, and hybrid analysis tools based on their analytical models. Additionally, based on the analytical methods employed by the tools, categorization can be achieved through formal verification, symbolic execution, fuzz testing, deep learning, and other analytical methods.

The static analysis detection approach for smart contracts is a method based on the static structure of the code, aimed at identifying potential vulnerabilities and security issues within smart contracts.

Oyente [55] stands as one of the earliest static analysis tools for detecting vulnerabilities in smart contracts. It employs a method based on symbolic execution, utilizing the contract's control flow graph. Oyente takes smart contract bytecode and Ethereum state as inputs, simulating the EVM and traversing various execution paths of the contract. It supports the detection of vulnerabilities including reentrancy, exception handling, and transaction order dependency flaws. Similarly, ZEUS [54] also functions as a static analysis tool. It employs a formal verification-based approach to accurately and equitably analyze smart contracts. ZEUS employs abstract interpretation, symbolic model checking, and constraint statements to swiftly verify contract security, enabling the detection of various smart contract vulnerabilities, such as reentrancy, integer overflow, and exception handling flaws.

The static analysis tool Mythril [61], on the other hand, employs conceptual analysis, taint analysis, and control flow verification to detect common vulnerabilities in Ethereum smart contracts, including reentrancy, integer overflow, and exception handling flaws. The static analysis framework Slither [62] transforms Solidity source code of smart contracts into the intermediate representation SlithIR. Utilizing a static single assignment form and a streamlined instruction set, SlithIR simplifies the contract analysis process while retaining semantic information lost during the conversion from Solidity source code to EVM bytecode. Additionally, Shah et al. [63] have introduced a smart contract static code analysis tool named UBF-ChaincodeScan. This tool aids in identifying security risks in smart contract code developed using Node.js and Go languages.

Furthermore, the formal verification of smart contracts constitutes another facet of the static analysis detection strategy. It involves the utilization of mathematical techniques and intelligent analysis tools to model, deduce, and prove contract properties, thereby verifying essential features such as consistency, uncertainty, observability, verifiability, and access control [64]. For instance, Kevm [65] harnesses the K framework to construct an executable formal specification based on the EVM bytecode stack. This framework provides EVM specifications, reference interpreters, and tools for program analysis and verification.

Fuzz testing [66], a dynamic detection technique, primarily aims to unveil vulnerabilities, crashes, and unforeseen behaviors within software applications by subjecting them to diverse input combinations. In

comparison with alternative methodologies, fuzz testing demonstrates commendable scalability and applicability, capable of execution even in the absence of source code. Currently, three distinct methodologies of fuzz testing are applied in the context of detecting vulnerabilities within smart contracts: ContractFuzzer [67], Regurad [68], and ILF [69].

Furthermore, Ding et al. [70] have devised an innovative fuzzing testing approach named HFContractFuzzer, specifically tailored for identifying contract vulnerabilities within the Hyperledger Fabric framework. This method amalgamates fuzzing techniques with the go-fuzz tool, engineered for conducting fuzz testing on the go programming language. Through the assessment of five prototypical contracts, the study unveiled security vulnerabilities present in four of the examined contracts.

Hu et al. [71] introduced a knowledge graph-based defect detection scheme for Solidity smart contracts and presented the SoliDetector tool, which facilitates defect detection for 20 distinct types of defects and auto-generates queries. Initially, they established the ontology layer of the knowledge graph and an instance layer designed to capture syntactic and logical relationships. By incorporating defect patterns and employing inferential rules, intricate relationships were deduced to ascertain the presence of defects. However, it is noteworthy that SoliDetector does not encompass the capacity to detect defects stemming from syntactic variations, nor does it support altered address encoding recognition and length computation. Additionally, an ongoing process of knowledge graph construction is requisite to adapt to updates in the Solidity language.

Xu et al. [72] proposed a knowledge graph-based methodology to detect vulnerabilities within chaincode written in Go. This approach involves the generation of an abstract syntax tree from the chaincode, followed by the construction of a knowledge graph. Exploiting pattern matching within the knowledge graph, including pattern analysis and vulnerability localization, their detection tool CCDetector proficiently identifies 21 distinct categories of vulnerabilities.

With the advancement of artificial intelligence, an increasing number of researchers are directing their efforts towards integrating artificial intelligence techniques into the domain of smart contract vulnerability detection. SaferSC [73] emerges as the inaugural deep learning-based model for detecting vulnerabilities in smart contracts. It distinguishes itself by employing a categorization of contract vulnerabilities inspired by Maian and achieves heightened detection accuracy compared to the Maian approach. Moreover, SaferSC conducts its analysis at the level of Ethereum operation codes (opcodes), leveraging LSTM networks to construct an Ethereum opcode sequence model, thus achieving precise detection of smart contract vulnerabilities.

Zhang et al. [74] introduce an approach to detecting smart contract vulnerabilities using a hybrid deep learning model termed CBGRU. This model leverages FastText and Word2Vec embeddings to encode smart contract lexemes. By employing fifteen distinct deep learning models to extract salient features, the CBGRU model amalgamates these features to facilitate the detection of smart contract vulnerabilities. Evaluation of the CBGRU model on publicly available datasets reveals notable accuracy and classification performance, affirming its efficacy in vulnerability detection.

Mohanty et al. [75] pointed out that how to deploy smart contracts on the blockchain system concerns the enforcement of smart contracts, which should enable the binding of contract code with contract accounts, execution nodes/containers,incentive mechanism, and consensus mechanism, etc.

Brandenburger et al. [76] stated chaincode deployment is the most core and complex part of Fabric cloud, in which the deployment is a process of instantiation to dispatch chaincode on different nodes, and by introducing the idea of functional computing, the chaincode's container will no longer hold the resources after the chaincode instance finishes its operation.

Xu et al. [77] indicated that Hyperledger Fabric adopts a FIFO scheduling method which is inefficient and unfair, so proposed a fair scheduling method that can deploy transactions of different priorities with differentiated quality of service.

Manevich et al. [78] pointed out that currently in the consortium blockchain platform Hyperledger Fabric, clients need to statically configure the deployment address and endorsement policy of chaincodes, which is not adaptable to changes of nodes environment and policies. so a dynamic deployment method based on service discovery is proposed to help clients automatically discover executable chaincodes and their endorsement policies.

In a word, smart contract deployment is a task-to-node instantiation process, and an improper deployment strategy can bring security risks and trigger subsequent execution errors. While an optimized deployment strategy can ensure the reliability and security of smart contract execution and improve resources utilization. So far existing deployment strategies are relatively simple and limited in functionality, which tells the research on smart contract deployment is inits early stage, so further research is urgently needed.

(3) Security of the execution environment

The security of the smart contract execution environment is of paramount significance, as malevolent actors may potentially target the execution environment to compromise the integrity of the blockchain network. EVMfuzzer [79], an instrument tailored for the EVM, conducts fuzz testing on the smart contract execution environment. It autonomously generates a diverse array of smart contract inputs and vigilantly observes the outcome of execution, thereby detecting aberrant behaviors, system crashes, or latent vulnerabilities. The application of fuzz testing to the EVM facilitates the identification of potential vulnerabilities that may be exploited by malicious entities. Consequently, this tool aids smart contract developers in enhancing code quality and fortifying security measures.

In another regard, the issue of privacy concerning smart contracts assumes pronounced significance. The anonymization mechanisms intrinsic to blockchain fail to comprehensively redress the privacy quandaries inherent to smart contracts. In a theoretical context, given the universal accessibility of all transactions to all participants, the scrutiny of transactional structures potentially affords an avenue for the extraction of users' private information. Hence, it becomes incumbent to ensure the confidentiality of pivotal constituents to preclude inadvertent disclosures of sensitive data. In this sphere, Fabric's privacy preservation schema emerges as uniquely distinguished. Solely nodes within the same communication channel are endowed with the prerogative to access and oversee the ledger, thereby affording a measure of mitigation against concerns pertaining to information leakages.

In summation, the trajectory of smart contract-related technologies is advancing with celerity, facilitating the wide-ranging application of blockchain platforms across various domains. Presently, scholarly inquiry into the security of smart contracts predominantly converges upon the Ethereum domain, while research pertaining to vulnerability detection within smart contracts on alternative blockchain typologies is comparably limited. The resolution of smart contract security exigencies mandates a holistic vantage, one that harmonizes blockchain system architecture, deployment strategies, execution environments, and privacy-centric measures, in the pursuit of judicious solutions.

4. Smart contract execution mechanism of mainframe blockchain platforms

Numerous blockchain platforms have evolved as the developing of blockchain technologies, in which respective smart contract interfaces have been proposed to make the blockchain system better applied in various business scenarios. We compare five popular blockchain platforms—Bitcoin, Ethereum, Fabric, Corda, and EOS to examine the execution of smart contracts from seven different perspectives such as programming language, Turing completeness, number of execution nodes, whether the contract execution is coupled with the block production, data model used to create the smart contract, execution environment, and whether the concurrent execution of smart contract is supported. The detail is shown in Table 3.

Table 3
Smart contract execution for mainstream blockchain platforms.

Plantform / Smart contract	Bitcoin	Ethereum 1.0	Hyperldeger Fabric	Corda	EOS
Programming language	OP_RETURN	Solidity	Go	Java	C++
Turing completeness	Incomplete	Complete	Complete	Incomplete	Complete
Execution nodes	All	All	1 or N	1 or N	1 or N
Execution and block production	Couple	Couple	Decouple	Decouple	Decouple
Data model	UTXO	Account	Account	UTXO	Account
Execution environment	Stack Engine	EVM	Docker	JVM	WebAssembly
Concurrent execution	Not	Not	Partially	Partially	Partially

4.1. Overview of the blockchain platform

(1) Bitcoin

Bitcoin transactions constitute the quintessential constituent of the Bitcoin system, with all other elements within the system oriented towards ensuring the generation, propagation, validation, and eventual inclusion of Bitcoin transactions in the global ledger. Each transaction serves as a public record on the Bitcoin blockchain, embodying a structured dataset containing pertinent information concerning the transfer of value among transaction participants [80].

Within the framework of the Bitcoin system, the initiation of a transaction commences with its creation, necessitating the signature of the sender, which serves as authorization for the expenditure of Bitcoin funds. Notably, the concept of accounts or balances is absent in Bitcoin, as a user's Bitcoin funds are distributed and stored in the form of unspent transaction outputs across various blocks of the blockchain. Each transaction effectively expends a designated prior associated UTXO, akin to a model wherein each spending instance corresponds to the utilization of a predetermined UTXO. Full nodes locally maintain a comprehensive ledger of the current UTXO set across the network, facilitating the tracking of unspent transaction outputs. Each individual transaction encompasses inputs and outputs, accompanied by a hash value denoting the transaction itself. This hash value assumes a pivotal role in the construction of the Merkle tree.

Upon the completion of transaction construction, it is disseminated onto the Bitcoin network, whereupon recipient nodes engage in the verification of its validity and correctness. In the event of successful verification, miners incorporate the transaction into their respective transaction pools, subsequently proliferating its dissemination until all network nodes acknowledge receipt. Alternatively, if the transaction fails verification, it is refused acceptance and returns an error message [81]. Miner nodes undertake the packaging of transactions from their respective pools, integrating a random number into the associated computation.

The node first to compute a hash value that adheres to stipulated criteria attains the privilege of recording the encapsulated transactions within the packaged block, thus effectuating the creation of a new block. Subsequently, this node disseminates the calculated new block across the entire network. Upon receipt of the block, other nodes promptly engage in the validation of its legitimacy and accuracy. Successful validation prompts the incorporation of the received new block into the local chain, while concurrently expunging the originally packed block from the individual transaction pool.

Upon the linking of the next block to the initial block, transactions receive further confirmation. Irreversible confirmation is bestowed upon transactions within a block after a succession of six blocks are linked to it. The miner responsible for generating this confirming block attains a stipulated quantity of newly issued bitcoins, alongside relevant transaction fees.

(2) Ethereum

In comparison to Bitcoin, which primarily facilitates singular transactional functionality, Ethereum aspires to empower the blockchain with programmable capabilities, enabling the execution of intricate commercial logic operations. Thus, Ethereum employs the Turing-complete programming language for smart contracts. Users can leverage Ethereum to create smart contracts and develop decentralized applications (DApps), thereby addressing concerns of trust and security. This renders the blockchain not only an asset transaction medium but also a value provider for other services.

Ethereum 1.0 adopts the PoW consensus mechanism akin to Bitcoin, maintaining network-wide consensus. In terms of execution mechanics, Ethereum follows a transaction execution sequence that involves transaction construction, broadcasting valid transactions across the network, and miners' validation of the resulting block information. Subsequently, validated blocks are recorded on the blockchain, updating ledger data. Smart contracts are executed within the EVM, transforming Ethereum into a complete Turing system. To prevent transaction looping, Ethereum introduces a gas mechanism, where each operation requires a specific amount of gas, encompassing execution of smart contracts and transaction sending. This gas mechanism not only constrains computational effort but also serves as part of Ethereum's incentivization model.

Although PoW offers high security, its substantial computational and energy costs, along with throughput limitations, prompted the shift to PoS consensus mechanism in Ethereum 2.0. Additionally, Ethereum 2.0 [82] aims to enhance throughput by introducing a "sharding" mechanism.

In Ethereum 2.0, the role of block validation previously performed by miners is now assumed by validator nodes, which are required to stake 32 Ether for participation [83]. Validators must ensure their actions are non-malicious, as punitive measures can affect their staked assets. Validators run three independent software components: execution clients, consensus clients, and validators. This separation contributes to the decoupling of execution and consensus in Ethereum to a certain extent.

In Ethereum 2.0, block generation operates on a fixed rhythm. Proof of Stake Ethereum introduces time intervals known as slots (12 s) and epochs (32 slots) [84]. Each slot provides an opportunity to add a new block to the Ethereum blockchain. Within each slot, a validator is randomly selected as a block proposer, removing the need for competitive computational power in block creation. This validator constructs and broadcasts the next block to be added to the Ethereum blockchain and updates the global state. Additionally, a validator committee is randomly selected within each slot to vote on the validity of the proposed block.

When a validator node in the consensus layer receives a new beacon block over the network, it transmits it to the execution client. Transactions contained in the block are re-executed within the execution client, and block signatures are checked for validity. Valid transactions are added to the node's local database. Validators then broadcast votes (called attestations) across the network in support of the block. Within each epoch, the first slot's block forms a checkpoint. Initially, if at least 2/3 of the active validators' staked ETH balance can attest to the validity of the two most recent checkpoints, the blocks between these two checkpoints are "justified". Once a subsequent checkpoint is "justified" following a previous "justified" checkpoint, the preceding checkpoint is "finalized", indicating the immutability of transactions prior to that point.

(3) Hyperledger Fabric

Hyperledger Fabric [25] is an open-source blockchain framework maintained and developed by the Hyperledger project under the Linux

Foundation. It aims to provide a highly flexible, secure, and controllable blockchain solution for enterprise applications. Chaincode, also known as smart contracts in Hyperledger Fabric, serves as the implementation of smart contracts. It encompasses the code responsible for updating the ledger's state and constitutes the business logic component within the blockchain network. Chaincode defines the business rules and transaction behaviors executed on the blockchain, and they are invoked by client applications to process transactions on the blockchain.

Hyperledger Fabric achieves customizability according to business requirements and diverse application scenarios by offering a modular architecture and pluggable components [25]. The Hyperledger Fabric network comprises three primary types of nodes: client nodes, peer nodes, and orderer nodes [85].

During the execution phase, clients submit transaction proposals to endorser(peer) nodes. Endorser nodes satisfying endorsement policies first verify the legitimacy of received transaction proposals and independently simulate the execution of valid proposals. The simulation process among endorser nodes is independent and parallel, generating read sets and write sets based on the execution results without modifying ledger data. Subsequently, endorser nodes return the endorsement results, including signatures and read–write sets, to the clients. After collecting and validating a sufficient number of legitimate endorsement results, clients send the endorsement results to the orderer nodes.

Orderer nodes, known as the ordering service nodes, are responsible for packaging transactions into blocks and broadcasting these blocks to all peer nodes. Orderer nodes globally order transactions based on consensus protocols (such as solo, Kafka, Raft) to ensure that all peer nodes execute transactions in the same sequence, thus ensuring network consistency. The ordered transactions are packaged into blocks and sent to peer nodes for verification.

In the validation phase, newly generated blocks are disseminated to peer nodes for verification. Each peer node independently checks the validity of transactions, including endorsement and validation policy checks. Only blocks that pass verification are accepted and added to the ledger of peer nodes.

With its attributes of flexibility, scalability, and privacy, Hyperledger Fabric provides a robust blockchain platform for enterprise-level applications. Its modular architecture and extensive capabilities position it as an ideal choice for constructing diverse blockchain solutions.

(4) Corda

The Corda platform [40], introduced by the financial consortium blockchain R3, has been developed primarily for interbank asset clearing purposes. It finds significant application within the domains of finance and commerce, notably in areas such as cross-border payments, asset trading, financial settlements, and supply chain financing.

Diverging from other public blockchain platforms, Corda adopts the design of a private blockchain, wherein transaction data is exclusively shared among participating entities, ensuring transactional privacy. The Corda network comprises a peer-to-peer network of nodes, each running Corda software and Corda applications known as CorDapps. Each node employs a digital certificate to link real-world legal identities with network identities. Inter-node communication occurs via point-to-point encrypted channels, in contrast to global broadcasting. In Corda, there exists no singular centralized repository of data. Instead, each node independently maintains facts relevant to itself, with no node possessing knowledge of all content. When a fact is shared among multiple nodes in the network, the databases of all nodes aware of the fact are simultaneously updated.

Facts recorded on the ledger are represented using States. State updates are achieved by transforming the original state into a historical record and subsequently adding a new version of the state. States can encompass diverse data types, facilitating representation of various factual entities (e.g., stocks, loans, identity information, etc.). Corda employs the UTXO (Unspent Transaction Output) model, ensuring immutability of every state on the ledger [40].

A transaction must pass consensus checks and fulfill signature verification requirements before submission. Uniqueness consensus is enforced to prevent double-spending, and verification ensures that the transaction and all its dependencies are contractually valid. Additionally, the transaction necessitates signatures from all relevant parties. Corda supports pluggable consensus mechanisms, allowing deployment of high-speed, high-trust algorithms such as Raft or low-speed, low-trust algorithms like BFT based on specific requirements.

The Corda network employs point-to-point message transmission rather than global broadcasting [86]. Coordination of ledger updates necessitates explicit specification by network participants regarding what information needs to be sent, to whom, and in what sequence. This design enables Corda to handle more complex and confidential transactions, making it suitable for scenarios demanding high privacy and flexibility, particularly in the financial sector.

Moreover, Corda achieves transaction automation through Flows. A flow comprises a sequential series of steps instructing a node on how to implement a specified ledger update. The Flow framework enables nodes to simultaneously run multiple flows, enhancing system parallelism.

(5) EOS

EOS [87] aims to construct a blockchain operating system that facilitates the development of decentralized applications. Employing the DPoS consensus mechanism, EOS verifies transactions and generates new blocks through the election of a representative group of 21 individuals. This mechanism realizes high throughput and low latency transaction processing while minimizing energy and computational resource wastage.

Analogous to other smart contract platforms, EOS permits developers to write and deploy smart contracts, which autonomously execute specific rules and logic on the blockchain. EOS utilizes the WebAssembly [88] technology as the runtime environment for its smart contracts, affording elevated performance and versatility across multiple platforms. EOS supports horizontal scalability, enabling the system to augment processing capacity as needed, devoid of necessitating a hard fork. This characteristic empowers EOS to more effectively cater to the requirements of high traffic and large-scale applications.

The execution of EOS smart contracts is facilitated by the WebAssembly technology. The code of EOS smart contracts is compiled into an open standard binary format, WebAssembly, which operates within an isolated sandbox environment, ensuring exceptional performance along with robust security. Capitalizing on WebAssembly's high-performance attributes, the EOS blockchain expedites smart contract execution, an imperative trait for applications necessitating rapid responsiveness and high throughput.

Given that WebAssembly is language-agnostic, EOS smart contracts can be scripted using various programming languages such as C, C++, Rust, and others. Developers are bestowed with the liberty to opt for the language best suited to their requirements and proficiencies for composing smart contracts. The adoption of WebAssembly by EOS simplifies and enhances the deployment and upgrading of contracts, without compromising the stability of the entire blockchain network.

4.2. Comparison of platforms

(1) Comparison of contract language

The scripting language of Bitcoin is tailored for specific transactions and functionalities rather than general-purpose programming. Therefore, Bitcoin does not require intricate Turing completeness; its fundamental capabilities suffice for its intended applications. Moreover, over the past decade of stable operation, its core functionalities and security have been effectively validated. To accommodate more intricate smart contract logic and data processing, Ethereum employs Solidity as

its principal contract programming language. Hyperledger Fabric, while supporting intricate transaction logic, has adopted languages like Go to better meet performance requirements.

In its quest to facilitate developer convenience, it extends support to Node.js and Java as well. In Corda, contracts can be scripted using Java, Kotlin, or other JVM-based languages. Java, with its robust ecosystem and mature infrastructure, is particularly well-supported. Kotlin strikes a balance between language flexibility and robustness when compared to Java. EOS, in its commitment to efficiently host numerous applications on the blockchain, predominantly employs C++. Additionally, EOS accommodates languages compiled into WebAssembly, such as RUST and Python, to cater to diverse developer needs.

In essence, the choice of smart contract programming language hinges upon the design and requisites of the blockchain platform. As blockchain platforms evolve, so too will the array of available smart contract programming languages.

(2) Comparison of contract execution

From the perspective of smart contract execution mechanisms, substantial differences exist in the execution process due to varying platform architectures and consensus mechanisms [89]. Bitcoin and Ethereum 1.0 employ the Proof-of-Work (PoW) consensus mechanism. While this mechanism ensures security, it leads to relatively low overall efficiency and imposes significant resource consumption, which severely constrains Ethereum's developmental potential. In response, ETH 2.0 has adopted the more efficient Proof-of-Stake (PoS) consensus mechanism and intends to introduce a sharding mechanism in the future to enhance system throughput [90].

Ethereum 2.0 envisages partitioning the entire blockchain network into multiple parallel segments termed shards. Each shard encompasses a set of validator nodes along with associated transactions and states. Every shard employs its distinct consensus mechanism known as shard consensus. The relative independence of shards facilitates concurrent transaction execution between shards, potentially elevating Ethereum's throughput from the Ethereum 1.0 era's 15 transactions per second to 100,000 transactions per second.

Fabric and Corda, representing consortium blockchain platforms [91], adopt execution architectures distinct from public blockchain platforms. Their decoupled architecture segregates contract execution from block production, thereby enhancing system throughput through increased parallelism. However, Fabric's concurrent execution introduces novel challenges, particularly concurrent conflicts that could result in the termination of numerous transactions. Section 4.1.4 provides a more detailed exploration of concurrent conflicts.

In contrast to Bitcoin and Ethereum, chaincode in Fabric is not stored within blocks; instead, it continuously operates atop consortium nodes. Whereas the execution of smart contracts on Bitcoin or Ethereum can be perceived as instances of smart contract templates, Fabric lacks instances of smart contracts, mandating that all ledger reads and writes occur through chaincode execution. Furthermore, Corda, which also employs the UTXO model, enhances platform smart contract execution efficiency through independent transaction negotiation, parallel execution of distinct smart contracts, and sharding and subnetwork design [92].

5. Optimization of smart contract execution

The blockchain system is rapidly emerging and evolving, yet a certain gap persists between its development and the fulfillment of diverse user requirements. Presently, scalability has become a bottleneck issue for extending the application of blockchain technology, particularly in enhancing the system's throughput capacity. Notably, within the context of blockchain systems, it is insufficient to solely enhance the performance of individual nodes to boost overall throughput. Overemphasis on strategies such as increasing block size or accelerating block generation time to achieve limited throughput improvements may potentially result in the exclusion of less capable computers from joining the network, thereby deviating to some extent from the decentralized principle.

Within the realm of distributed computing theory, two approaches are available for enhancing scalability: vertical scaling and horizontal scaling. In comparison to vertical scaling, horizontal scaling involves the transformation of a serial system into a parallel system to expedite instruction processing, making it more widely favored. As a novel paradigm in distributed computing systems, horizontal scaling is also applicable to blockchain, where scalability can be achieved by disaggregating the system architecture and reconfiguring it into a parallel execution model for smart contracts. Furthermore, researchers have begun contemplating strategies for enhancing performance and scalability through Layer 2 protocols without altering the primary chain protocol, leading to the emergence of Layer 2 solutions [93] such as the Lightning Network [94].

In this section, we investigate the current research landscape from the perspective of smart contract execution, aiming to analyze the strategies for alleviating performance bottlenecks in blockchain systems.

5.1. Architecture-based concurrent execution

In this section, we delve deeper into the categorization of concurrent execution within the context of blockchain technology, and examine approaches aimed at resolving challenges arising from concurrent conflicts.

5.1.1. Concurrent execution based on node roles decoupling

Presently, within the domain of public blockchains, the execution of smart contracts and the generation of blocks exhibit a tight coupling, leading to the redundancy of transaction execution. To enhance system parallelism and throughput, numerous researchers and entities have endeavored to decouple node functionalities, thereby disentangling the execution of smart contracts from block generation. Furthermore, architectural support for the parallel execution of smart contracts has been fostered through the adoption of channels, groups, partitions, and analogous constructs. Additionally, various researchers have pursued optimizations from alternative vantage points to further augment the parallelism inherent to smart contract execution.Relevant studies are summarized in Table 4.

Hyperledger Fabric [25] categorizes nodes within the network into endorsement nodes, ordering nodes, and validation nodes. By employing an "execute-order-validate" execution architecture, Fabric decouples transaction execution from validation, thereby facilitating parallel execution within the system. Fabric's channel mechanism permits the creation of multiple distinct sub-networks within the network, each channel encompassing a specific set of members and smart contracts. Transactions and data between different channels remain isolated, affording participating channel members transaction privacy. Furthermore, inter-channel transactions can be processed in parallel, allowing each channel to independently undertake consensus and ordering operations.

By introducing a group-based architecture, Fisco Bcos [95] has achieved the capability to initiate multiple groups among blockchain nodes. This design ensures mutual isolation of transaction processing, data storage, and block consensus among these groups. This approach not only safeguards the privacy of the blockchain system but also reduces the complexity of system maintenance. Transactions between different groups can be executed in parallel, thereby enhancing the overall system performance. According to reports, The throughput of Fisco Bcos can reach up to 20,000 tps.

The Monoxide blockchain [96] employs a parallel partitioning scheme. Within Monoxide, the nodes of the entire blockchain system are divided into multiple partitions, each possessing its own blockchain ledger and state database. These partitions achieve inter-partition parallelism through independent consensus and transaction processing

Table 4
Concurrent execution based on node function decoupling.

Recognition	Methodology
Hyperledger Fabric [25]	Execute-order-validate;Channel;
Fisco Bcos [95]	Sort-execute-verify;Group
Wang [96]	Asynchronous Consensus Zones; Multiple independent and parallel instances of single-chain consensus systems
Hyperchain [97]	Hyperchain execute-order-validate execution architecture
Wei [98]	A transaction parallel execution model by multi-threading techniques
Gao [99]	A fair contract partitioning algorithm employing integer linear programming and a distribution protocol that randomly assigns these subsets to user subgroups
Jin [100]	A multi-threaded parallel execution model based on transaction categorization.
Zhang [101]	A parallel execution methodology that bifurcates the roles of mining nodes and validation nodes.
Liu [102]	A novel lock-free parallel execution paradigm for scalable smart contract execution;
Zhang [103]	BlockPilot: Proposer-Validator Parallel Execution;
Xian [104]	A pioneering concurrency control system for high contention workloads.
Qi [105]	SChain extends concurrent execution to multiple peers in each organization and achieves continuous parallel execution across blocks based on a pipelined workflow

processes. The introduction of partitions also entails a dilution of computational power. To address the security concerns arising from this computational dilution, Wang et al. [96]proposed a method called "Crossbow Mining". Experimental results indicate that the system demonstrates a throughput 1000 times greater and a capacity 2000 times higher than that of Bitcoin and Ethereum.

The HyperChain [97] has introduced a novel parallel Merkle tree data structure and devised a decentralized architecture that separates the smart contract execution module from the consensus module. In this architecture, transactions are dispatched to consensus nodes for block generation and are subsequently categorized for parallel execution. However, this framework does not detail the contract verification process.

Wei et al. [98] employed multi-threading techniques to realize their proposed model for transaction parallel execution. This approach utilizes a simple static analysis transaction partitioning algorithm to group transactions into categories of unrelated, directly related, and indirectly related transactions. Subsequently, transactions are distributed for multi-threaded execution on individual nodes using breadth-first traversal. Finally, experimental analysis validates that this parallel model indeed significantly enhances the performance of transaction processing.

Gao et al. [99] have introduced a scalable solution for executing smart contracts that enables the concurrent execution of multiple smart contracts to enhance the system's throughput. The solution relies on two pivotal technologies: a fair contract partitioning algorithm employing integer linear programming to divide the set of smart contracts into multiple subsets, and a distribution protocol that randomly assigns these subsets to user subgroups.

Jin et al. [100] have pointed out that the current efficiency of smart contract execution on blockchain platforms is inadequate to meet real-time business demands in certain scenarios. To address this issue, they have introduced a multi-threaded parallel execution model based on transaction categorization. However, it is important to note that this model only supports inter-group parallelism and does not effectively handle transactions within the same group.

Zhang et al. [101] have identified an observed inefficiency within existing public blockchain platforms such as Ethereum, wherein miner nodes sequentially execute smart contracts during the consensus phase and subsequently duplicate the execution during the validation phase, resulting in suboptimal efficiency. Consequently, they propose a parallel execution methodology that bifurcates the roles of mining nodes and validation nodes. Contract invocations are treated as atomic operations, and provisions for rollback are permitted in cases of shared conflicts.

Liu et al. [102] have introduced Saber, a novel paradigm for scalable smart contract execution, heralded as the first lock-free parallel execution paradigm. Saber achieves asynchronous execution of smart contracts by categorizing nodes into consensus nodes and execution nodes, allowing each node to independently execute ordered transactions. Within the Saber framework, consensus nodes are responsible for sorting complex transactions and allocating them to distinct subsets of nodes (execution groups) for parallel execution.

Zhang et al. [103] introduce BlockPilot, a novel framework engineered to address the throughput limitations inherent in traditional blockchain systems, especially those incorporating EVM compatibility. The framework's innovation lies in its ability to facilitate concurrent transaction execution by proposers and validators, a departure from the conventional serial execution model. At the heart of BlockPilot is an optimistic concurrency control (OCC) algorithm, which ensures proposers can execute transactions in parallel while maintaining a serializable schedule. Validators, on the other hand, are equipped with a pipeline workflow that enables the concurrent processing of multiple blocks, thereby enhancing efficiency.

Xian et al. [104] present SC-CHEF, a pioneering concurrency control system designed to significantly enhance the concurrent execution of smart contracts, particularly for high contention workloads. The system's core innovation is the strategic decomposition of transactions into multiple subtransactions, which can operate independently without compromising the original logic. Implemented and rigorously tested on a private Ethereum platform, SC-CHEF has been proven to surpass both OCC and 2PL methods by an average of 60% and 70% respectively in high contention scenarios, showcasing its superior efficiency through extensive experimental validation.

Qi et al. [105] introduce SChain, a novel blockchain system that addresses the challenges of scalability and flexibility faced by permissioned blockchains in enterprise collaborations. The system is designed to provide high performance by enabling scalable concurrent execution of transactions through a flexible architecture.SChain's innovative approach involves decoupling the functionalities of a traditional blockchain node into three distinct sub-functions – ordering, execution, and finalization – and assigning them to different peers within an organization. This separation allows each organization to scale each sub-function independently without the need for inter-organizational negotiation.

5.1.2. *Sharding-based concurrent execution*

The concept of sharding initially found its application in the field of databases, primarily aimed at allocating storage overhead. Sharding in the context of blockchain constitutes a technological strategy for augmenting the performance and capacity of the blockchain network. Within the confines of a traditional singular blockchain network, such as Ethereum 1.0, all transactions and data necessitate processing and storage by all participating nodes, thereby engendering potential network congestion and performance bottlenecks.

To address this predicament, blockchain sharding entails the segmentation of the entire network into multiple fragments, wherein each shard assumes responsibility for processing transactions and data within a specified range. Each shard maintains its distinct transaction history, state, and smart contracts, albeit inter-shard correlations of data and transactions are also conceivable. This orchestration empowers shard-based blockchains to achieve parallel execution of smart

Table 5
Sharding-based concurrent execution.

Recognition	Methodology
Tennakoon [106]	A dynamic blockchain sharding approach that uses smart contracts for real-time shard reconfiguration, enhancing security and transparency.
Zheng [107]	A sharded consortium blockchain system that enhances cross-shard efficiency.
Yang [108]	A cluster-based sharding blockchain strategy for IoT collaborative computation that addresses non-cooperative clustering issues and uses PBFT for inter-shard transactions asynchronously.
Zhang [109]	ShardCon enhances blockchain scalability and efficiency through an off-chain execution model and TEE for cross-shard smart contracts.
Jia [110]	A low cross-shard protocol with a multilevel state model and COPRAS algorithm for optimized state distribution and increased transaction throughput.
Mu [111]	A hierarchical state partitioning scheme with a state transfer protocol and transaction-aware allocation algorithm for scalable blockchain performance.
Bakhtiary [112]	A blockchain-based access control system integrating hierarchical ABAC, smart contracts, and sharding for enhanced flexibility and scalability.
Yu [113]	A full sharding approach with overlapping networks and virtual accounts for enhanced security and transaction throughput in blockchain systems.
Yang [114]	A sharding scheme for LIoT applications that enhances scalability and efficiency through regional sub-chains and a coordinating shard for cross-regional data interaction and multi-objective scheduling.
Li [115]	A dynamic blockchain sharding scheme using HMM for adaptive updating of shards to address transaction load imbalance and enhance scalability and efficiency in IoT environments.
Xi [116]	A dynamic blockchain sharding scheme using HMM for fine-grained, adaptive updating of shards to reduce CSTs and improve system throughput and latency in IoT environments.
Cui [117]	An optimized sharding scheme integrating EC with blockchain to improve performance in IoT environments, using a MaOEA-IE algorithm for efficient task offloading and enhanced scalability.
Kanwhen [118]	A blockchain-enabled demand response solution using sharding and LSTM neural networks for privacy-enhanced smart contracts and scalable energy trading optimization.
Dhulavvagol [119]	A scalable blockchain architecture using a hybrid sharding and data partitioning approach to increase transaction throughput and decrease network latency, with parallel transaction execution and optimized resource utilization.

contracts, thereby substantially amplifying the network's processing capabilities and throughput in aggregate.Relevant studies are summarized in Table 5.

Tennakoon et al. [106] addressed the issue of inflexible static sharding solutions by introducing a dynamic blockchain approach. This approach allows for the creation and closure of shards as needed and adjusts their sizes during runtime, all without requiring a hard fork. They achieved this through specialized smart contracts that reconfigure shards. Importantly, this solution not only enhances the security of shard adjustments but also ensures transparency similar to other blockchain data. Their testing revealed that the system's performance closely correlates with the number of shards. With just eight shards, the system achieved a remarkable throughput of nearly 14,000 transactions per second (TPS).

Zheng et al. [107] have contributed to the domain of consortium blockchains with their systematic inquiry, culminating in the proposal of a sharded consortium blockchain system named "Meepo". Meepo enhances cross-shard efficiency via the cross-epoch and cross-call. On a test-bed of 128 AliCloud servers, setting 32 shards and 4 consortium members, Meepo can achieve more than 140,000 cross-shard TPS under the workload of 100,000,000 asset transactions.

Yang et al. [108] introduced a cluster-based sharding blockchain strategy for collaborative computation in the Internet of Things (IoT), aiming to address the non-cooperative clustering issue posed by other blockchain sharding solutions in IoT collaborative computation. The system operates in parallel and employs the practical byzantine fault tolerance (PBFT) consensus mechanism asynchronously for inter-shard transactions.

Zhang et al. [109] introduce ShardCon, an innovative off-chain model designed to enhance the scalability and efficiency of blockchain sharding systems, particularly when executing complex cross-shard smart contracts. The system addresses the limitations of existing lock-based cross-shard protocols, which struggle with low-efficiency executions and deadlocks when handling multi-shot cross-shard transactions. ShardCon's key innovation lies in its off-chain execution model, which decouples contract execution from cross-shard consensus, thereby enabling efficient execution of complex contracts. The model utilizes Trusted Execution Environments to implement a cross-shard contract execution engine, reducing communication overheads and ensuring consistency between on-chain and off-chain states.

Jia et al. [110] introduce Estuary, a novel low cross-shard blockchain sharding protocol developed to notably augment blockchain transaction throughput. This protocol's innovative approach is grounded in a multilevel state model and a state splitting and aggregation mechanism, which effectively decouples state identity from quantity, facilitating intra-shard transaction completion among users. A key feature is the community overlap propagation algorithm for sharding, optimizing state distribution to align closely with transaction patterns between users.

Mu et al. [111] propose EfShard, an innovative blockchain system designed to efficiently address scalability issues through state sharding. The system introduces a hierarchical state partitioning scheme that allows for flexible and timely allocation of states across shards in response to recent transactions. EfShard incorporates a novel state transfer protocol ensuring consistency and liveness while efficiently migrating states between shards. Additionally, a transaction-aware state allocation algorithm is employed to reduce cross-shard transactions and balance workload, thereby improving overall system performance.

Bakhtiary et al. [112] introduce Combo-Chain, an innovative blockchain-based access control system tailored for IoT environments. This system integrates hierarchical attribute-based access control models, smart contracts, and sharding techniques to enhance flexibility, dynamicity, and scalability. The core innovation of Combo-Chain lies in its hierarchical attribute-based access control model, which introduces hierarchies for both subject and object attributes, simplifying policy and attribute management. The system manages access policies and attributes through deployed smart contracts and leverages sharding to distribute storage and computational overhead, addressing challenges like low scalability and performance issues associated with blockchain technology.

Yu et al. [113] introduce OverShard, an innovative blockchain scaling solution that employs full sharding with an overlapping network and virtual accounts. This approach enhances security and communication efficiency in a sharded system by enabling nodes to join multiple shards. OverShard's key innovation lies in its ability to process all transactions within a single shard, thereby addressing cross-shard transaction challenges and achieving near-linear scalability with an increasing number of shards.

Yang et al. [114] introduce co-sharding, a novel sharding scheme tailored to enhance the scalability and efficiency of blockchain technology in large-scale IoT applications. The co-sharding framework divides

the large-scale IoT ledger into shards managed by sub-chains overseeing distinct geographic regions, with elected nodes maintaining a coordinating shard to facilitate cross-regional communication and data interaction.

Li et al. [115] propose LB-Chain, a dynamic blockchain sharding scheme that addresses performance limitations in existing blockchain systems by focusing on transaction load imbalance. LB-Chain employs a novel approach based on the hidden Markov model to achieve adaptive dynamic incremental updating of blockchain shards, which significantly reduces cross-shard transactions. The system is designed to improve scalability and efficiency in managing large-scale IoT sensing tasks by leveraging fine-grained blockchain sharding and embracing the dynamic assemblage characteristic of IoT collaborative sensing.

Xi et al. [116] propose HMMDShard, a dynamic blockchain sharding scheme that utilizes the hidden Markov model to address the scalability and efficiency challenges in collaborative IoT environments. The innovative approach of HMMDShard lies in its fine-grained, adaptive dynamic incremental updating of blockchain shards, which significantly reduces the number of cross-shard transactions. By embracing the dynamic assemblage characteristic of IoT collaborative sensing and leveraging HMM, the scheme enhances system throughput and minimizes transaction confirmation latency.

Cui et al. [117] introduce an optimized sharding scheme for blockchain performance improvement in an end-edge-enabled IoT environment. This scheme integrates edge computing with blockchain technology, establishing an EC architecture based on blockchain sharding to address security issues during task offloading and to enhance the scalability of the blockchain system. The proposed model aims to reduce consensus latency, energy consumption, and sharding failure probability while improving sharding throughput.

Kanwhen et al. [118] propose an optimization and scalability solution for blockchain-enabled demand response smart contracts by employing sharding and neural networks. The system is designed to allow small residential energy consumers to participate in demand response mitigation and peer-to-peer energy trading, eliminating the need for a trusted third party. To address scalability, especially in terms of throughput due to the integration of numerous IoT devices and associated data, the blockchain is implemented with sharding technology. The system demonstrates improved scalability and an optimized bidding process for energy trading, leading to a more efficient and secure demand response mechanism.

Dhulavvagol et al. [119] present a scalable blockchain architecture that leverages a hybrid sharding generation and data partitioning approach to enhance transaction throughput and reduce network latency. The proposed system addresses the scalability challenge in blockchain networks by enabling parallel and distributed transaction execution through a simultaneous block-level transaction approach. A hybrid sharding generation algorithm is introduced, creating multiple shardings that function as independent blockchains with their own verified users and data, strategically distributing transaction load to optimize resource utilization.

Currently, research pertaining to sharding predominantly centers within the realm of public blockchain networks. Despite the manifold advantages offered by blockchain sharding technology, several challenges persist, encompassing inter-shard interactions, security provisions, and the facilitation of cross-shard transactions. Consequently, the exploration of diverse sharding schemes coupled with judicious mitigation strategies to address the predicaments posed by sharding constitutes the forthcoming research trajectory within the domain of sharding. This pursuit aims to engender more efficacious, secure, and scalable blockchain systems, constituting the imminent research direction within the sharding domain.

Table 6
DAG-based concurrent execution.

Recognition	Methodology
IOTA [120]	A DAG structure called the "Tangle"
Hashgraph [121]	A DAG structure and the ABFT algorithm for asynchronous consensus on transaction order.
NANO [122]	A block lattice structure with individual "Account Chains" for each user
Spectre [123]	The DAG structure uses serialization and cryptographic techniques for quick transaction confirmation and efficient consensus, enhancing throughput and security.
Byteball [124]	Using "units" for a blockless, scalable design that supports real-time transactions and ensures security and immutability through "stable units."
Piduguralla [125]	A framework for concurrent smart contract transaction execution using a DAG-based scheduler module to increase transaction speed and optimize throughput.
Zhang [126]	A blockchain architecture with workload-aware elasticity that dynamically adjusts storage concurrency for optimized throughput and security.
Yang [127]	An adaptive parallel scheduling scheme for smart contracts on blockchain, designed to enhance transaction throughput and reduce latency—key challenges in blockchain performance.
Wang [128]	A SoK on DAG-based blockchain systems that analyzes and abstracts a general model of core features and design patterns, evaluating them across structure, consensus, security, and performance dimensions.

5.1.3. DAG-based concurrent execution

A DAG-based blockchain represents a distinct form of distributed ledger technology that diverges from conventional blockchain structures such as those seen in Bitcoin and Ethereum. It employs a data structure characterized by a directed acyclic graph to store and manage transactions, eschewing the conventional notions of blocks and chains found in traditional blockchains. The DAG blockchain holds significant promise in transcending the performance limitations inherent in conventional blockchains, owing to its inherent parallel structure. Relevant studies are summarized in Table 6.

IOTA.[120] employs a DAG structure known as the "Tangle", where each transaction is treated as a node interconnected by linking directly to the preceding two transactions. This configuration facilitates concurrent processing of multiple transactions, enhancing transaction throughput. The Tangle's structural attributes confer a high degree of scalability upon IOTA when confronted with substantial transaction volumes. With an expanding network of participants, transaction confirmation speed and network security stand to be augmented. Empirical measurements indicate an average throughput of 1500 TPS [129], with transaction confirmation times ranging from 1 to 5 min. IOTA is primarily tailored to the domain of the IoT, aiming to provide solutions for minuscule transactions among IoT devices. These encompass applications such as data sharing between devices, micro-payments, and supply chain tracing.

In the DAG structure of Hashgraph [121], each node represents a transaction or event, while directed edges indicate the temporal relationships between events. Each event encompasses its creation time, transaction information, and references to other events. This network of references forms a complex directed graph with no cycles, thus earning its designation as a DAG. Hashgraph employs the "Asynchronous Byzantine Fault Tolerance" algorithm to maintain and update the DAG structure. This algorithm allows nodes to asynchronously exchange information within the network, achieving consensus and ultimately determining the order of transactions and events. The position of a node in the DAG and its relationships with other events dictate the chronological sequence of events.

NANO [122] employs a block lattice structure, where each user account possesses its dedicated blockchain known as an "account chain". Solely housing the transaction history corresponding to the account, each account chain operates independently. In this architecture, every transaction functions as an individual block, solely permissible for addition to the account chain by the account owner. Within the block lattice, transactions undergo asynchronous confirmation, bypassing the necessity for network-wide consensus waiting. Consequently, transactions achieve prompt confirmation, almost instantaneously reflecting in the account balance. NANO's decentralized ledger design guarantees autonomous storage and management of transaction records for each account, addressing the limitations present in conventional blockchain systems that consolidate all transactions into a single chain.

Spectre [123] structure features a novel approach to addressing blockchain scalability and confirmation latency. Unlike traditional blockchain systems, Spectre constructs its DAG by serializing transactions and incorporating cryptographic techniques to enable quick confirmation of transactions and efficient consensus. Spectre's DAG structure enhances throughput and maintains security through its unique method of transaction inclusion and voting, ensuring that the network remains resilient against attacks while providing a mechanism for efficient transaction processing and confirmation.

The DAG structure of Byteball [124] is constructed using "units" as its fundamental units. Each unit is built by directly linking the preceding unit and a new unit containing transactions and data, forming a continuous data structure without the need for blocks. This design eliminates traditional blockchain blocks and miners, achieving a high degree of parallelism and scalability while supporting real-time transaction confirmation. Byteball's DAG structure also incorporates a concept known as "stable units", ensuring that historical transaction records cannot be altered, thus ensuring the security and immutability of transactions.

Piduguralla et al. [125] introduce an efficient framework for the concurrent execution of smart contract transactions (SCTs) within blockchain networks. The framework's innovative approach lies in the utilization of a parallel DAG based scheduler module, which enables the concurrent and efficient execution of SCTs. This module can be easily integrated into the blockchain framework, thereby increasing the speed of transactions and optimizing throughput by addressing dependencies among a block's transactions through the DAG data structure.

Zhang et al. [126] introduce MorphDAG, a groundbreaking blockchain architecture that addresses the limitations of traditional and existing DAG-based systems by introducing workload-aware elasticity. MorphDAG is the first of its kind to dynamically adjust the degree of storage concurrency in response to varying workload sizes and skewed access patterns, thereby optimizing throughput without sacrificing security. They establish an elastic degree of storage concurrency theory to determine the optimal level of concurrency for high performance and security under different loads.

Yang et al. [127] propose an adaptive parallel scheduling scheme for smart contracts on blockchain, designed to enhance transaction throughput and reduce latency—key challenges in blockchain performance. The authors construct a dependency DAG to model the relationships between smart contracts, allowing them to identify and manage potential conflicts. A conflict model is introduced, derived from extensive experiments, to dynamically adjust the execution strategy of smart contracts based on the defined conflict factors. This approach moves beyond static analysis by incorporating dynamic scheduling, which is particularly effective in environments with high transaction conflicts.

Wang et al. [128] provide an in-depth Systematization of Knowledge on DAG-based blockchain systems, aiming to address the limitations of classical blockchain architectures in terms of latency and scalability. The authors systematically analyze existing DAG-based blockchain systems by abstracting a general model that encapsulates the core features of such systems and identifying six distinct design patterns. Their approach involves a comprehensive evaluation of these systems across multiple dimensions, including structure, consensus mechanisms, properties, security, and performance.

5.1.4. Resolution of concurrent conflicts

While concurrent execution offers a promising solution for enhancing blockchain performance, it also introduces the challenge of concurrent conflicts arising from multiple transactions concurrently reading and writing to the same account or key. As reported, hyperledger fabric has experienced a transaction abortion rate of up to 40% due to concurrent conflict issues [130]. Currently, researchers are exploring various approaches to address the challenge of concurrent conflicts.

One approach to address the issue of concurrent conflicts involves employing methods such as static and dynamic transaction analysis to preemptively predict conflict relationships among transactions within a block, thereby preemptively aborting conflicting transactions. Another avenue of resolution primarily leverages database concurrency control techniques such as caching, locking, multi-version concurrency control, and reordering, aiming to manage conflicting transactions during the execution process. Additionally, collaborative deployment of one or more of these methods constitutes further strategies to effectively mitigate encountered instances of concurrent conflict.

In summary, the core idea underlying these solutions is to judiciously assign a unique global total order to transactions, ensuring a rational handling of concurrency and thereby mitigating conflicts. Relevant studies are summarized in Table 7.

Fisco Bcos [95] addresses transaction conflict issues and achieves parallelism by introducing a transaction dependency graph. Within a batch of transactions, a method is employed to identify the mutually exclusive resources each transaction needs to access. Based on the transaction order within the block and the resource occupancy relationships, a transaction dependency DAG is constructed. Transactions without any dependency relationships are given priority for execution. Once executed, these transactions are removed from the transaction dependency DAG, simultaneously generating new independent transactions. This process continues until all transactions are completed.

Lu et al. [131] have proposed a rapid execution method for smart contract transactions based on fine-grained read–write analysis. The core idea revolves around extracting detailed read–write information from the smart contract compiler. Using the read–write sets of each transaction, all transactions are organized into a DAG, which is then used to partition these transactions into specific groups. This partitioning reduces the number of read–write items and subsequently minimizes the latency of transaction grouping. Upon the completion of all transactions within a group, the system checks for read–write conflicts. The transaction that initially modifies a given state variable is validated, and the remaining transactions are regrouped and redistributed to computing units for re-execution. Following re-execution, if read–write conflicts persist, the corresponding conflicting transactions are marked as failed executions. Experimental results demonstrate that this method can reduce overall transaction execution latency by 58.1%.

Xu et al. [132] employed a locking mechanism to identify conflicts at the onset of a transaction flow. This mechanism utilizes a straightforward locking scheme, where conflicts trigger waiting, thus preventing rollbacks. However, this approach may give rise to issues such as deadlocks and high latency, making it less suitable for latency-sensitive IoT applications. Moreover, the locking mechanism falls short in effectively addressing scenarios of high concurrency.

Xu et al. [133] have devised a novel LMLS (locking mechanism and ledger storage) approach tailored for enhancing write-write concurrency conflicts in hyperledger fabric. This method combines a locking mechanism with a classified ledger storage system. Furthermore, they introduce a cache-based approach to improve Read–Write concurrency conflicts. Employing caching accelerates data retrieval speed, complemented by the addition of a cache log in hyperledger fabric to ensure data consistency.

Zhang et al. [134] have introduced an approach involving a cache layer to mitigate the risks associated with non-deterministic transactions in the Hyperledger Fabric. This method integrates a cache layer at

the client side, enabling the preprocessing and sorting of transactions prior to submission. As a result, it effectively reduces the occurrence of non-deterministic transaction risks arising from read–write conflicts and transaction order dependencies. This enhancement contributes to improved system reliability and performance.

Debreczeni et al. [135] introduce a novel taxonomy for managing transaction conflicts in Hyperledger Fabric, a leading permissioned blockchain platform. The authors approach the issue from a system engineering perspective, emphasizing the need for a model-driven engineering process to prevent multi-version concurrency control (MVCC) conflicts at the design phase. They propose two storage-level entity attribute partitioning algorithms, total partitioning and attribute affinity-based partitioning, to minimize potential conflicts by refining the granularity of data access.

Xu et al. [136] address the challenge of concurrent transaction conflicts in Hyperledger Fabric, by proposing a solution that leverages distributed locks and message queues. They proposed method, LMQF, is designed to enhance the system's ability to handle high levels of concurrency without compromising performance in conflict-free scenarios. Through a series of experiments, the authors demonstrate that LMQF significantly improves transaction success rates and effective transaction throughput under various conflict levels, presenting a promising approach for enterprise-level blockchain applications requiring high concurrency and throughput.

Wu et al. [137] present FabricETP, an innovative high-throughput optimization solution for Hyperledger Fabric, aimed at resolving concurrent conflicting transactions that hinder performance. The authors identify two primary sources of conflict: intra-block and inter-block conflicts. To address intra-block conflicts, FabricETP introduces a transaction scheduling algorithm that employs graph theory to rearrange transaction execution orders, minimizing conflict occurrences. For inter-block conflicts, FabricETP devises a cache-based mechanism to detect and abort invalid transactions early, thus preventing them from entering the blockchain.

Silva et al. [138] introduce a novel approach for the parallel execution of blockchain transactions based on dynamic conflict analysis. This method operates by constructing a DAG that captures the dependencies between transactions, allowing for the parallel execution of independent transactions while managing conflicts. The authors propose a conflict function that dynamically determines if two transactions are dependent, thus enabling validators to self-verify conflicts using the transaction parameters. To reduce performance overhead, the system employs a parallelizer thread to build the DAG while a pool of worker threads executes transactions with resolved dependencies.

Another conflict resolution solution is derived from the reordering mechanism commonly used in database solutions. Sharma et al. [139], among others, first introduced this concept from database systems to the blockchain context through Fabric++. They introduced the notion of recording a transaction dependency graph in each block to detect conflicting transactions and then reordered these transactions to minimize unnecessary conflicts. Unlike locking mechanisms, these reordering methods do not require additional services or coordination time. Moreover, they can increase the number of effective transactions per block, reducing the need for unnecessary retransmissions and consequently lowering energy expenditure for IoT devices.

Ruan et al. [140] have introduced a novel fine-grained reordering mechanism named FabricSharp based on Hyperledger Fabric. This mechanism not only effectively addresses transaction conflicts intra-blocks but also considers conflicts that arise inter-blocks. Experimental results demonstrate that FabricSharp offers a throughput improvement of 25% compared to Fabric++.

Hong et al. [141] addressed the potential conflict issues arising from the independent and random cross-shard transaction scheduling in different shards. They introduced the concept of Byzantine fault-tolerant deterministic ordering and developed a conflict-free blockchain called Prophe, aimed at minimizing transaction aborts caused by non-deterministic contract invocations. Additionally, they proposed efficiency-enhancing designs such as fine-grained ordering, asynchronous error correction, and parallel pre-execution. In comparison to state-of-the-art sharding systems, Prophe achieved a 3.11× increase in throughput for 1 million Ethereum transactions.

Table 7
Resolution of concurrent conflicts.

Recognition	Methodology
Fisco-Bcos [95]	Transaction dependency Directed Acyclic Graph
Lu [131]	Fine-grained read–write analysis and Transaction dependency Directed Acyclic Graph
Xu [132]	A straightforward locking scheme
Xu [133]	Locking Mechanism and Ledger Storage (LMLS)
Zhang [134]	Mitigate conflict by a cache layer
Debreczeni [135]	Two storage-level entity attribute partitioning algorithm
Xu [136]	Distributed lock and message queue
Wu [137]	A transaction scheduling algorithm and a cache layer
Silva [138]	Transaction dependency Directed Acyclic Graph and a conflict function to find dependent
Sharma [139]	Reordering
Ruan [140]	A novel fine-grained reordering mechanism
Hong [141]	Byzantine fault-tolerant deterministic ordering

5.1.5. Sidechain execution

A sidechain refers to an auxiliary blockchain established in parallel to the main chain, possessing distinct characteristics, consensus mechanisms, and smart contracts [142]. This allows sidechains to focus on specific application scenarios or requirements without compromising the overall performance of the main chain. Sidechains can interact with the main chain, facilitating cross-chain asset transfers between the main chain and the sidechain. Despite offering numerous advantages, sidechain technology also encounters challenges such as cross-chain interoperability, security, and decentralization. Different sidechain solutions employ varied approaches to address these challenges, catering to diverse use cases and demands. In summary, sidechains provide blockchain technology with heightened flexibility and customization, enabling it to adapt to various application scenarios and developmental trajectories.

Rootstock [143] is an intelligent contract platform built upon the Bitcoin blockchain, achieved by introducing a sidechain compatible with the EVM onto the Bitcoin network. This enables developers to construct and deploy smart contracts on the UTXO-based Bitcoin network, thereby realizing functionalities akin to those of Ethereum.

Purnaadi et al. [144] explore various strategies to enhance blockchain scalability, focusing on the implementation of sidechain technologies. They propose several methods to alleviate the well-known scalability issues in public blockchain applications, such as Bitcoin and Ethereum. The core of their approach involves limiting the ledger network's extent by employing sidechains, which are secondary chains connected to the main chain but operate with an independent consensus mechanism. One proposed method is reducing the number of nodes within a sidechain to expedite transaction verification. Additionally, segregating active node transactions into sidechains is suggested to alleviate the load on the main chain.

While sidechains hold promising prospects, research into sidechain technology remains relatively nascent at present.

5.2. Cross-chain execution

Up to the present moment, most existing blockchain systems have adopted diverse architectures and execution methods, resulting in a

lack of interconnectivity among these blockchain systems, effectively forming isolated silos. This necessitates the development of cross-chain technology to facilitate interaction and communication between distinct blockchain networks, enabling them to collaboratively engage and execute cross-chain transactions or smart contracts.

In the realm of research on cross-chain execution of smart contracts, Hellen et al. [145] emphasized that smart contracts transform blockchains from merely append-only ledgers into programmable state machines. Consequently, merely transferring assets between chains is insufficient. To address this, they introduced the HyperService platform, which offers cross-chain interoperability and programmability. Within this framework, a unified programming model was proposed to construct cross-chain DApps, facilitating the state transition of smart contracts between chains.

Su et al. [146] proposed an asynchronous cross-chain exchange model which determines asset transfer based on transactions embedded with control conditions. This approach eliminates the need for pre-deployed smart contracts, enabling transactions to be sent and matched in parallel, effectively reducing transaction latency.

Zhang et al. [147] propose a novel cross-chain interoperability and collaboration model for keyword-based embedded smart contracts within the Internet of Things (IoT) ecosystem. The model leverages specific keywords within smart contracts to identify their status, enabling cross-chain operations and facilitating asset exchanges and smart contract collaborations across different blockchains. The model introduces an innovative method of contract execution where the deployment step is embedded into the invocation transaction, allowing for both deployment and execution of smart contracts within a single transaction.

Chen et al. [148] introduce ATOMCI, an innovative system that addresses the challenges of atomic cross-chain smart contract invocation across heterogeneous blockchains. The system is underpinned by a triad of novel components: smart contract design patterns for crafting contracts that support atomic operations; a cross-chain expression protocol that facilitates the precise definition of cross-chain logic; and a cross-chain service community that operates on a smart contract basis to provide trustworthy and verifiable services.

Sober et al. [149] introduce a framework for asynchronous cross-blockchain smart contract calls, addressing the interoperability challenge in the blockchain landscape. The framework is inspired by the remote procedure call protocol and employs blockchain relays to facilitate communication between smart contracts on different blockchains, specifically those utilizing the EVM. This approach minimizes trust in intermediaries and maintains a high degree of decentralization.

Han et al. [150] introduce VM-Studio, a universal crosschain smart contract verification and execution scheme that addresses the challenge of interoperability among heterogeneous blockchain systems. The core concept of VM-Studio is to migrate the virtual machines from the original blockchain to the target blockchain by encapsulating them in containers. The approach allows for the simulation of the original blockchain's VM execution environment within the target blockchain, thereby enabling crosschain transaction execution and smart contract verification.

López et al. [151] delve into the development of bridging protocols in DAG-based distributed ledger technologies, with a particular focus on enabling interaction between the IOTA network and other DLT platforms. The authors underscore the significance of interoperability in the fragmented DLT ecosystem and propose a model that establishes secure and efficient communication channels between different networks.

Presently, various heterogeneous blockchain platforms, including public blockchains and consortium blockchains, have gained widespread adoption. However, the issue of cross-chain interoperability was not initially considered in the design of these systems. To facilitate cross-chain interaction and transactions, smart contracts can play a pivotal role. Nonetheless, ensuring the determinism and termination of cross-chain smart contract execution to achieve atomic transactions and enhance scalability remains an unresolved challenge for blockchain systems.

5.3. Off-chain execution

Off-chain execution is an alternative technique for optimizing the execution of smart contracts. This approach allows computational tasks of smart contracts to be processed outside the blockchain network. Consequently, it reduces on-chain resource consumption, lowers transaction costs, improves transaction processing speed, and enhances data privacy protection. The adoption of off-chain execution offers numerous advantages, including cost-effectiveness, increased efficiency, enhanced privacy, improved scalability, greater flexibility, and rapid response capabilities. This method is particularly suitable for applications that require the handling of large volumes of data or the execution of complex computations.

Ali et al. [152] introduce SRP, an innovative runtime protection framework for blockchain-based smart contracts. The framework's primary innovation lies in its hybrid architecture that integrates off-chain mechanisms with on-chain contract execution, aiming to minimize the on-chain burden of runtime verification. SRP is designed to protect already-deployed smart contracts from attacks in real-time without compromising the underlying blockchain's throughput. They present a protocol customized for off-chain runtime-verification interoperability and validate their approach using a proof-of-concept implementation on a local Ethereum network instance. Empirical and experimental results demonstrate SRP's efficiency, outperforming on-chain-only mechanisms in terms of service time and throughput, especially under increasing workloads.

Xian et al. [153] introduce ICOE, an innovative off-chain execution model that addresses the scalability challenges associated with smart contract-based industrial applications on blockchain platforms. The core of ICOE lies in its lightweight group-consensus approach, which facilitates cross-group contract transactions without the need for cumbersome state synchronization or duplicate executions. ICOE operates by executing invoked smart contracts within their respective consensus groups and leveraging the two-phase commit protocol to ensure consistent commits across multiple groups within the same transaction. This approach significantly reduces computational overhead and communication costs. The authors' implementation and extensive experimental results demonstrate that ICOE achieves a 14× improvement in throughput compared to the state-of-the-art models, showcasing its potential for enhancing blockchain scalability and efficiency in industrial applications.

Frassetto et al. [154] introduce POSE, an innovative off-chain protocol for executing smart contracts that addresses the limitations of on-chain smart contract execution, such as high costs and scalability issues. The POSE framework leverages trusted execution environments (TEEs) to perform computations efficiently and ensure the swift recovery from failures, either accidental or malicious. The protocol operates by randomly selecting a subset of TEEs to execute each smart contract, with one enclave acting as the executor and the others as watchdogs. The approach guarantees liveness, as contract execution can proceed as long as one TEE is responsive. POSE provides strong security guarantees, supports contracts with private states, and allows dynamic participation and arbitrary lifetimes without the need for large collaterals. The authors evaluate POSE's efficiency and effectiveness through a proof-of-concept implementation and demonstrate its potential for complex applications, including federated machine learning.

Reno et al. [155] introduce an innovative approach to addressing the scalability and storage challenges inherent in blockchain technology, particularly focusing on public blockchain such as Bitcoin. They propose a decentralized peer-to-peer (P2P) file distribution system combined with a novel Twin-Ledger architecture to enhance transaction processing rates and reduce space requirements. The system leverages the InterPlanetary File System (IPFS) for off-chain storage of validated transactions, generating a Content Identifier of significantly reduced size compared to the actual transactions. This method allows for a

substantial increase in the number of transactions per block, thereby improving throughput without expanding block size.

Cai et al. [156] introduce a comprehensive two-layer scaling framework that synergistically harnesses the power of on-chain and off-chain technologies to facilitate fine-grained transaction support. This framework is designed to accommodate the high throughput and low latency demands of IoT-driven applications while ensuring data integrity and security.

To address the limitations of existing transaction execution models, which either overburden nodes with hardware requirements or introduce additional verification overheads, potentially undermining the decentralization and security of blockchains. Chen et al. [157] presents an innovative concurrent execution scheme that harmonizes on-chain and off-chain operations to significantly enhance the performance of blockchain systems. The core of this approach is a novel consistent information scheduling mechanism that segments transaction scheduling information based on execution-related data, thereby refining the efficiency of information transmission and execution among nodes.

5.4. Layer 2 solution

The concept of Layer 2 [93] was first introduced when major blockchain platforms like Bitcoin and Ethereum encountered scalability issues. People began to explore ways to enhance blockchain performance and scalability without modifying the main chain protocol. This led to the idea of executing smart contracts "off-chain" to alleviate congestion.

In other words, Layer 2 involves constructing an additional protocol layer above the blockchain's main chain to handle transactions and data, with the final outcomes being submitted to the main chain. This approach reduces the burden on the main chain while still ensuring security through the main chain.

Prominent Layer 2 solutions encompass the Lightning Network, State channels, Raiden Network, and others. These solutions are designed to establish more efficient communication and transaction mechanisms between on-chain and off-chain environments, facilitating swifter and more cost-effective transaction processing.

The Lightning Network [94] is one of the earliest Layer 2 technologies, initially designed for Bitcoin. It aims to significantly enhance the scalability and transaction efficiency of the Bitcoin blockchain through the use of bidirectional payment channels and smart contract technology. The Lightning Network leverages complex multi-signature contracts and mechanisms like Hash Time-Lock Contracts to enable rapid channel opening and closing while ensuring payment security and immutability. This technology transfers a large number of small-value transactions from the main chain to off-chain channels, effectively relieving the burden on the main chain. It reduces transaction congestion, lowers fees, and greatly increases the transaction throughput of the Bitcoin system.

The Raiden Network [158] is a Layer 2 scaling solution for the Ethereum blockchain, focused on addressing Ethereum's scalability challenges. Operating akin to Bitcoin's Lightning Network, the Raiden Network facilitates multiple off-chain transactions, with the final outcomes being committed to the Ethereum main chain only when necessary. This approach effectively alleviates the load on the Ethereum main chain, as a substantial portion of transactions takes place within off-chain channels, eliminating the need for every transaction to occur directly on the main chain.

State Channels [159] are an innovative blockchain scalability technology designed to enhance transaction throughput and performance of blockchain systems. By establishing a specific protocol layer off-chain, State Channels enable users to engage in multiple interactions and transactions without requiring each transaction to be executed on the main blockchain. Leveraging multi-signature and smart contract technology, State Channels ensure transaction security and reliability while maintaining the state and balance information of the involved parties off-chain. This approach only records the final outcomes on the main chain, significantly reducing the number of on-chain transactions and thereby substantially improving the throughput and responsiveness of the blockchain system.

State Channel technology holds potential in addressing blockchain congestion, lowering transaction fees, and enhancing user experience, providing robust support for the rapid development of practical business applications and financial transactions. Other common Layer 2 solutions include Rollups [160] and Plasma [161].

The execution of smart contracts generates a substantial amount of transactional state data with resource consumption implications. The decision of whether to execute data-storing smart contracts on-chain or off-chain constitutes two distinct computational models for smart contracts. As is commonly understood, the on-chain model is inherently non-scalable and becomes resource-constrained with increasing data accumulation, rendering it unsuitable for scenarios characterized by high transaction frequency and significant data access demands. Conversely, the off-chain model involves executing smart contracts and storing data either partially or entirely outside the main chain, potentially offering improved scalability. However, this approach also introduces challenges related to the interaction and consistency of data between on-chain and off-chain environments.

5.5. UTXO-based execution

In comparison to blockchain platforms such as Ethereum that support programmable smart contracts, the Bitcoin script and UTXO-based smart contract model are relatively rudimentary. However, through continuous research and development, innovative solutions have emerged that allow the Bitcoin network to play a role in the field of smart contracts as well.

Chakravarty et al. [162] introduced an extended UTXO model, which encompasses more expressive forms of validation scripts. This model includes the implementation of a universal state machine and enforces script invariants throughout the entire transaction chain.

Dai et al. [163] observed that with the proliferation of the UTXO collection, the speed of input verification experiences a substantial decline, leading to a deceleration in the overall verification process and ultimately posing a threat to system security. They deconstructed the input verification functionality into three distinct components: existence verification, unspent verification, and script verification. Additionally, they introduced a novel block validation mechanism termed as Efficient Block Verification. Through extensive experimental comparisons between EBV and Bitcoin, the EBV mechanism successfully achieved a remarkable reduction of 93.1% in memory requirements and a significant decrease of 93.5% in block verification time.

Based on the UTXO model, Li et al. [164] developed the Conflux blockchain platform, which integrates a UTXO (Unspent Transaction Output) model in its design, similar to the transaction model employed by Bitcoin. Each transaction input references previous transaction outputs, ensuring the validity of transactions. Conflux combines Proof of Work with DAG and draws inspiration from the GHOST algorithm for chain selection. Operating within a decentralized framework, Conflux ensures blockchain security while simultaneously enhancing scalability. The Conflux platform achieves a throughput of up to 6400 transactions per second (tps) and is capable of confirming transactions within a time frame of 4.5 to 7.4 min.

5.6. Optimization of smart contract execution based on trusted hardware

Trusted hardware can provide an isolated and secure execution environment for smart contracts, and it can also support the parallel processing of multiple smart contracts. Smart contract execution based on trusted hardware not only ensures the security of contract code and data but also significantly improves execution efficiency.

Li et al. [165] introduce FASTBLOCK, a novel framework that accelerates the block lifecycle on blockchain platforms, notably Ethereum, by harnessing fine-grained concurrency. The methodology of FASTBLOCK is threefold: it employs a symbolic execution-based analyzer to delineate minimal atomic sections within transactions, a concurrent execution step that leverages hardware transactional memory (HTM) to execute potentially conflicting transactions in parallel, and a concurrent validation step that utilizes a happen-before relation to deterministically re-execute transactions.

Lu et al. [166] propose the smart contract unit (SCU), a hardware-based accelerator designed to enhance the execution performance of smart contracts on the Ethereum platform. The authors conduct a thorough analysis of Ethereum's execution model, the EVM, and identify its limitations, including a lack of parallelism and a stack-based model that hinders performance. To address these, SCU introduces a novel RISC-style instruction set architecture that facilitates instruction-level parallelism through dynamic pipelining and register renaming. Additionally, SCU employs a heterogeneous multicore design that supports transaction-level parallelism, capitalizing on the observation that transactions within a block exhibit light data dependencies, allowing for parallel execution. The proposed SCU is implemented on a Xilinx FPGA platform, demonstrating significant speedups over software implementations on an Intel CPU and outperforming state-of-the-art designs.

Fang et al. [167] propose the smart contract unit (SCU), a hardware-based accelerator designed to enhance the execution performance of smart contracts on the Ethereum platform. The SCU is implemented on a Xilinx FPGA platform, demonstrating significant speedups over software implementations on an Intel CPU and outperforming state-of-the-art designs introduce an innovative execution framework that leverages the confidentiality and integrity guarantees of SGX to achieve both intra- and inter-node concurrency for smart contracts. The cornerstone of their approach involves a two-phase execution model that initially designates a primary node to parcel out transactions to various replica nodes for parallel processing. The method is optimized for traditional two-phase execution and is augmented with a mixed optimistic concurrency control protocol to enhance both execution and validation efficiency.

6. Smart contracts empower Industry 4.0

The integration and application of emerging technologies such as the Internet of Things and artificial intelligence in traditional industries have significantly increased production efficiency and further enhanced the digitization of industrial production. Folgado et al. [26] summarized the contributions of emerging technologies in the field of industrial informationization and proposed reference architectures for Industry 4.0. However, a new issue arises. Although data acquisition systems can now achieve real-time recording of production processes, It suffers from issues of centralization and tampering. This is one reason why, despite having relatively comprehensive production standards and regulatory frameworks, numerous quality issues still arise. Thus, a crucial element is still lacking in the current framework of Industry 4.0: trustworthiness of data.

Therefore, introducing a trust mechanism to ensure data reliability in digital production will help advance Industry 4.0 from digital chain to a trust chain. And blockchain can serve as an excellent candidate.

Blockchain is a trust technology [168] that establishes trust between people through data and mathematics. This new trust system sets rules for trust through smart contracts and ensures the secure and reliable execution of contracts via the blockchain system. It not only helps drive innovation in societal trust mechanisms but also offers new possibilities for creating a more open, transparent, secure, and trustworthy market economy.

The current trust system relies on centralized institutions, where the verification process can involve multiple steps and intermediaries, leading to decreased efficiency and increased costs. Additionally, establishing trust can be more challenging in the absence of long-term relationships and shared backgrounds, particularly in international trade and cross-cultural exchanges.

Through the immutability and transparency of blockchain, entities can establish a trust mechanism based on data and mathematics, allowing trust to be built between unfamiliar entities—something traditional centralized institutions cannot provide. This trust mechanism has the potential to transform industry processes and dynamics, offering significant advantages in promoting data sharing [169], optimizing business processes, reducing operational costs, enhancing collaborative efficiency, and building a trustworthy system. Thus, blockchain technology can drive innovation in societal trust mechanisms, prompting a reevaluation and redesign of how trust is established, leading to more efficient, secure, and equitable social interactions. In essence, blockchain represents a shift in production relationships.

Today, the internet has evolved beyond just a consumer platform to encompass the industrial internet, closely linked with sectors such as finance, manufacturing, logistics, and retail. The ability to fully harness and utilize the value of data accumulated in these industries has become crucial for intelligent decision-making, enhancing competitiveness, and creating competitive barriers.

Moreover, the industrial internet connects data from governments, businesses, organizations, and individuals, enabling data resource sharing while safeguarding data security and privacy. This facilitates the creation of a secure, transparent, trustworthy, and efficient data industry, further advancing Industry 4.0 from digital chain to trust chain. This necessitates a secure and trustworthy data sharing platform. Blockchain's characteristics perfectly align with these requirements.

This section analyzes how smart contracts can empower Industry 4.0 from four angles: product traceability, trustworthy manufacturing, supply chain management and supply chain finance.

6.1. Product traceability

The objective of a traceability system is to achieve trustworthiness, fairness, transparency, and traceability. Traditional product traceability information comes from centralized sources, and its authenticity cannot be verified by a trusted third party, leading to potential falsification. Additionally, traceability information may not cover the entire supply chain or only includes partial supply chain data, preventing comprehensive end-to-end traceability. Furthermore, the systems of different supply chain enterprises are not interconnected, making it difficult to trace production information and resulting in information silos that reduce transparency. Therefore, achieving full lifecycle traceability – from raw materials, production, and testing to transportation and sales – and ensuring data credibility is essential.

According to data from the World Health Organization (WHO), nearly 600 million people worldwide (approximately 10% of the population) fall ill each year due to the consumption of contaminated food, with 420,000 deaths resulting, leading to a loss of 33 million disability-adjusted life years [170]. Unsafe food consumption perpetuates a vicious cycle of disease and malnutrition, disproportionately affecting infants, the elderly, and the infirm. Even in countries with ample food production, food safety remains an urgent issue.

The food industry encompasses numerous processes including production, procurement, processing, storage, transportation, and sales. Given the elongated and intricate nature of its supply chain, consumers often struggle to access comprehensive quality and safety information behind the products they consume, primarily relying on regulatory oversight and the endorsement of corporate brands. This situation poses significant risks to food safety. To address this, some researchers have proposed blockchain-based traceability systems to achieve comprehensive tracking of the food supply chain and ensure food safety.

Tian et al. [171] established an agricultural food supply chain traceability system by leveraging RFID and blockchain technology. By systematically collecting, transmitting, and sharing authentic data pertaining to various stages of agricultural product production, processing, warehousing, distribution, and sales, this system achieves comprehensive traceability of trustworthy information throughout the agricultural product supply chain. This initiative effectively safeguards food safety.

Mao et al. [172] devised a blockchain-based credit evaluation system. This system leverages smart contracts on the blockchain to collect credit assessment texts from traders. Utilizing a deep learning network known as long short-term memory, the collected texts are directly analyzed. The credit outcomes serve as references for regulatory oversight and management. Through the application of blockchain technology, traders can take responsibility for their actions during transactions and credit evaluations. Regulatory bodies can gather more reliable, authentic, and comprehensive information about traders.

Baralla et al. [173] have introduced a blockchain-based universal agricultural food supply chain traceability system, which implements the "From Farm to Fork" (F2F) model currently employed in the European Union. This system integrates existing traceability rules and processes, allowing consumers to reconstruct a product's history all the way back to its origin. This enables verification of the product's health and quality through a simple QR code scan. All relevant stakeholders within the supply chain can identify any participant across the entire chain, thereby enhancing trust levels among organizations and individuals through self-managed chronological activity.

Casino et al. [174] have developed and tested a traceable, distributed, and trustless security architecture for food supply chains, which they applied to a case study of food traceability in a dairy company. Botcha et al. [175] proposes an advanced solution that combines Internet of Things (IoT) edge devices with blockchain systems to achieve greater transparency in the pharmaceutical supply chain. Xu et al. [176] uses the Internet of Things technology to obtain source data of raw materials to ensure that the data uploaded to the blockchain system is authentic, credible, transparent and traceable.

Gaudio et al. [177] have devised a multi-layered blockchain solution aimed at enhancing traceability within the agri-food supply-chain. This solution is underpinned by a smart contract developed on the Hyperledger Fabric, which facilitates the tracking and verification of product information across the supply-chain. The approach is initiated by a system engineering analysis to discern the informational requisites along the agri-food supply-chain, as dictated by consumer demands.

Leteane et al. [178] propose an innovative framework that integrates blockchain technology with a trust model to enhance the reliability of traceability data. The core concept of their approach is to employ a decentralized ledger that is tamper-proof, complemented by a trust model that computes trust scores based on multiple trust metrics. These trust metrics are crucial for accurately assessing the trustworthiness of data generated from various sources within the supply chain. The framework is designed to be adaptive and extensible, allowing for the incorporation of different 'trust packages' – sets of trust metrics and instructions for their use – which are developed by metrics developers to cater to the specific trust needs of different supply chain links.

To address the challenge of counterfeit products and enhance transparency is paramount, Mohammed et al. [179] introduced TrustChain, a decentralized system leveraging blockchain technology and decentralized storage systems to bolster product authenticity and traceability. The fundamental approach of TrustChain is to employ a blockchain-based framework that facilitates the secure recording and verification of product history within the supply chain. By utilizing the Ethereum test network and implementing Solidity smart contracts, the system facilitates services such as adding new products, transferring product ownership, and viewing the entire traceability process.

Blockchain-based traceability systems possess characteristics of security, trustworthiness, and immutability, ensuring data security and privacy protection. Integrating traceability data entirely onto the blockchain effectively guarantees data authenticity and integrity [180]. This approach also mitigates issues of information asymmetry and operational opacity found in traditional traceability systems, thereby reducing risks associated with inaccuracies.

Currently, due to the urgent demand for food safety, research on food traceability based on smart contracts and blockchain technology is more widespread. However, research on the traceability of other industrial products remains insufficient. The primary reason for this gap is the confidentiality of industrial production, which makes it difficult for researchers to access relevant production data and process details. This area of research requires more policy guidance or leadership from key enterprises.

6.2. Trustworthy manufacturing

Industrial production data is a core component of product traceability, especially in specialized fields such as pharmaceutical research and production. Companies must ensure the safety and compliance of their products at all stages, including research, production, storage, and transportation. Currently, this information relies on extensive paper records and manual verification, which is costly and inefficient and poses risks of tampering. Ensuring the authenticity and reliability of industrial data, while maintaining information security, is a critical issue that digital industry must address. By utilizing the programmability of smart contracts to create relevant agreements for various industrial processes and leveraging blockchain technology to ensure the authenticity and trustworthiness of data at each stage, it is possible to further achieve trustworthy manufacturing.

Addressing data silos and promoting industry collaboration are key challenges for manufacturing enterprises seeking to advance digital transformation and smart manufacturing [181]. Blockchain technology can effectively bridge the gaps between data silos across manufacturing stages, ensuring data immutability while facilitating information sharing on the blockchain [182]. Additionally, blockchain's access control mechanisms enable information sharing while safeguarding enterprise data privacy. These immutable digital production records can also effectively support internal decision-making and industry collaboration.

Lu et al. [183] comprehensively elucidated the application of blockchain technology in the oil and gas industry across four distinct dimensions: transactions, management and decision-making, regulatory compliance, and network security. They underscored that blockchain has the potential to substantially reduce transaction costs, enhance data security, transparency, and efficiency within the realms of oil and natural gas. Furthermore, they postulated a trajectory towards the evolution of a hybrid blockchain model within the oil and gas sector.

Mohamed et al. [184] discussed the role of blockchain in supporting intelligent manufacturing applications within the manufacturing industry. They introduced an intermediary middle ware approach that leverages blockchain services and functionalities to achieve enhanced security, trustworthiness, traceability, reliability, and autonomy in intelligent manufacturing applications. This approach serves to establish and ensure a robust level of trust among stakeholders within the manufacturing value chain, thereby providing a safeguard for the realization of Industry 4.0 initiatives.

In the realm of industrial Internet of Things for the petroleum industry, Umran et al. [185] introduce a novel framework that integrates blockchain technology to address the multifaceted challenges of cybersecurity, data integrity, and system scalability. They presents a comprehensive blockchain-based strategy that fortifies the petroleum industry's IIoT infrastructure with a focus on security, scalability, and efficiency.

Yang et al. [186] address the scalability and security challenges associated with sharing massive device data across heterogeneous network domains in IIoT. To mitigate these challenges, the authors propose

a blockchain-based data-sharing framework, EdgeShare, which integrates a two-layer overlay network topology with edge computing paradigms. This innovative approach significantly alleviates system pressure and reduces communication delays, thereby enhancing the efficiency of data sharing.

Zhang et al. [187] have devised an innovative mechanism underpinned by smart contracts and driven by data quality assessments. The crux of their approach revolves around establishing a secure data-sharing framework that ensures the confidentiality and integrity of data while simultaneously incentivizing participation and fostering high-quality data contributions. The research presents a comprehensive framework that harmonizes data security, quality assurance, and incentive alignment, paving the way for a more robust and equitable IoT ecosystem.

For enterprises, these immutable digital records can help confirm the compliance of their production processes. Regulatory agencies can directly access the data on the blockchain, facilitating compliance checks for the entire research and production process without extensive manual verification. This also aids in achieving digital regulation and significantly reduces labor and time costs. From this perspective, blockchain and smart contracts can effectively ensure transparency and immutability of industrial data while protecting privacy, thereby assisting regulatory bodies and enterprises in ensuring product quality and improving production efficiency and regulatory convenience.

In the burgeoning field of Unmanned Aerial Vehicle (UAV) manufacturing, ensuring the authenticity, origin, and traceability of components is paramount. Hawashin et al. [188] introduces a pioneering blockchain and Non-Fungible Token (NFT)-based solution that adeptly addresses these concerns. This system not only manages and certifies UAV parts but also traces their lineage and ownership with a high degree of transparency, decentralization, and trustworthiness. The authors propose a model that employs composable NFTs, which hierarchically capture UAV data, with parent NFTs representing the assembled UAV and child NFTs denoting individual components. This innovative approach facilitates comprehensive system-wide traceability and smooth ownership transfers.

In the realm of intelligent manufacturing, the intricacies of supply chain management are magnified by the exigencies for transparency, security, and efficiency. Addressing these challenges, Zhao et al. [189] introduce the Blockchain Consensus-based Traceability Method for Smart Factory Supply Chain (BCTMSSF), a pioneering model that leverages blockchain technology to ensure the veracity and traceability of supply chain transactions within a smart factory ecosystem. The BCTMSSF model is undergirded by a novel Verifiable Delegated Proof of Stake (VDPoS) scheme, which not only mitigates the centralization and security issues inherent in traditional supply chains but also introduces a dynamic probability mechanism and a fusion mechanism to enhance the robustness of the consensus process. The VDPoS consensus algorithm is particularly noteworthy for its ability to dynamically adjust the selection probabilities of nodes, thereby reducing the risk of centralization and ensuring a more equitable distribution of block production rights.

Industrial production generates the largest economic value globally, while the need for trust between enterprises is also pressing. As a technology for establishing trust in untrusted environments, blockchain offers significant application potential. Combined with the programmability of smart contracts, blockchain technology holds immense potential for application in the field of industrial informationization. Currently, various industry organizations have begun to recognize and invest in exploring its applications.

6.3. Supply chain management

The supply chain is a complex system engineering within industrial production, involving core enterprises, suppliers, logistics companies, customers, and other participants. Within the supply chain system, different enterprises collaborate, leveraging their respective strengths to form a large and competitive industry alliance.

Currently, supply chain management faces issues of information asymmetry and opacity, which can lead to erroneous demand forecasts and trust issues. As an interconnected network, establishing trust mechanisms among supply chain members is crucial for safeguarding the overall interests of the supply chain. Compared to trust between individual enterprises, trust among supply chain members is more complex and multifaceted.

The supply chain involves various industries and enterprises, and may span different cities, provinces, or even countries. Establishing an effective trust mechanism, which ensures privacy protection while enabling information sharing, is a key factor in determining whether supply chain members are willing to engage in information exchange. The decentralized and open nature of blockchain technology can effectively address traditional supply chain trust issues, thereby breaking down information barriers and facilitating industry collaboration and information sharing.

Dietrich et al. [190] introduce an innovative blockchain solution underpinned by smart contracts, aimed at mitigating supply chain risks and enhancing transparency. The crux of their approach lies in the deployment of a smart contract with a static network address that facilitates dynamic mapping of supply chain (SC) changes.

Okanlawon et al. [191] delve into the burgeoning intersection of blockchain technology and the construction supply chain, employing a structural equation modeling approach to quantitatively assess the impact of blockchain adoption. The study's findings underscore the transformative potential of blockchain in enhancing transparency, efficiency, and traceability within the construction industry, thereby contributing valuable insights to the discourse on Industry 4.0 technological innovations.

Ciotta et al. [192] introduce a proof-of-concept for the integration of blockchain technology and smart contracts into the construction information flows, specifically within the context of structural systems. The authors propose a novel approach to enhance the reliability and transparency of decision-making processes on construction sites. Central to their methodology is the utilization of smart contracts that operate on varying levels of complexity, ranging from basic to advanced. The multi-tiered smart contract framework is bulit on the tamper-proof and traceability inherent in blockchain technology, which serves to create a verifiable and trustworthy record of all significant events during the construction process.

Chen et al. [193] propose a pioneering approach that integrates blockchain technology with image-based deep learning to enhance smart contract automation in the construction sector. Central to this methodology is the development of a blockchain oracle capable of interfacing real-world construction progress, as captured through images, with smart contracts residing on the blockchain. This oracle is pivotal in translating visual data into actionable insights that the smart contracts can execute upon, thereby bridging the gap between physical construction activities and digital contract terms.

Zhang et al. [194] introduce a blockchain-enabled framework aimed at streamlining the reverse supply chain management of power batteries. The authors' approach leverages a four-layer blockchain platform, which incorporates a double-chain structure for robust data management, an information tracing module for lifecycle tracking of batteries, and smart contracts for secure and automated trading management. The study represents a significant advancement in the application of blockchain technology for sustainable and efficient management of power battery recycling ecosystems.

In the quest to enhance supply chain management within the offsite manufacturing sector, Roberto et al. [195] propose a Supply Chain Management Model that leverages the convergence of the Internet of Things, Building Information Modeling (BIM), and blockchain technology. This integration is encapsulated within a framework that addresses the complex dynamics of modern supply chains, emphasizing the need

for improved data traceability, security, and process efficiency. The framework is designed to streamline processes, mitigate the risks of information fragmentation, and bolster interoperability and reliability across the design, production, and implementation phases in offsite manufacturing. This research endeavor sets a foundation for subsequent investigations within the burgeoning field of supply chain management enhancement through technological integration.

A blockchain-based supply chain management platform offers the following advantages: First, multiple business participants can work on a unified collaboration platform, creating a reliable data flow across the entire supply chain. This enhances supply chain management and business collaboration efficiency, and guides supply-side enterprises in optimizing capacity. Second, the blockchain supply chain platform can simultaneously record the information flow, capital flow, and logistics within the supply chain system. This facilitates asset transfer and real-time settlement among multiple institutions, reducing transaction and reconciliation costs. Additionally, these data can serve as corporate assets and collateral for financing.

6.4. Supply chain finance

Supply chain finance [196] is a typical scenario in trade finance. Traditional supply chain finance heavily relies on manual cross-checking, with banks spending substantial time and effort verifying the authenticity and accuracy of various paper trade documents. Moreover, the trade ecosystem involves multiple participants, each of whom only has access to partial transaction, logistics, and capital flow information, resulting in low information transparency. Additionally, due to the lack of connectivity between banks' information systems, regulatory data retrieval is delayed, complicating financial management and oversight. Furthermore, small and medium-sized enterprises often face high costs when applying for financial financing.

Wang et al. [197] delve into this intersection through an evolutionary game-theoretic lens, examining the strategic interactions between financial institutions and small to medium-sized enterprises within the accounts receivable financing framework. Their research constructs a model that encapsulates the decision-making processes influenced by the maturity of blockchain technology and the risk preferences of financial entities. The study posits that blockchain, with its inherent features of decentralization, traceability, and tamper-proof nature, can significantly reduce information asymmetry and foster collaboration, thereby lowering the incidence of collusion and financial uncertainties in supply chains.

Natanelov et al. [198] investigates the innovative potential of blockchain and smart contracts in supply chain finance within the context of Australia-China beef supply chains. Utilizing a hybrid approach that combines business process redesign and the resources-events-assets accounting model, the study maps the current state of the supply chain, introduces blockchain and smart contracts to enhance processes and traditional supply chain finance models, and evaluates the impact of these technologies on supply chain finance innovation. The research identifies how financial risks can be mitigated through blockchain and smart contracts and proposes a group buying business model that integrates with supply chain finance, offering new areas for supply chain finance models to reduce financial costs and improve cash flow performance.

Supply chain finance involves numerous participants, including core enterprises, upstream and downstream companies, and third-party logistics providers. Core enterprises play a leading role in the supply chain, serving as central hubs for logistics, information flow, and capital flow. By leveraging blockchain systems to connect the supply chain, consensus algorithms can address trust issues, and smart contracts can mitigate performance risks, thereby enabling effective trust transmission along the supply chain and reducing collaboration costs.

Through multi-level blockchain integration, trustworthy data sharing can be achieved within each level of the blockchain, with data pushed to the next level based on authorization and need. The use of a shared ledger and smart contracts not only resolves data trust issues but also enhances transaction efficiency for all parties. Furthermore, this trusted data can serve as a core asset for corporate financing, helping to address the issue of high financing costs for enterprises.

Su et al. [199] presents a comprehensive feasibility analysis of integrating blockchain technology into supply chain finance to alleviate bottlenecks faced by small and medium-sized enterprises in China. The study explores blockchain's key features such as decentralization, immutability, smart contracts, and transparency, and demonstrates how these can address trust issues and reduce operational costs.

Battelli et al. [200] examines the integration of smart contracts in the financial sector, highlighting their potential to revolutionize transactions through automation and efficiency while also addressing the challenges of rigidity and regulatory opacity. Battelli emphasizes the importance of a transparent and unified legal framework to accommodate the technological advancements and ensure the compatibility and proper application of smart contracts in the evolving Fintech landscape.

Yu [201] introduces a blockchain-based supply chain financial system that integrates smart contracts to address the financing challenges faced by SMEs. By employing Ethereum-based smart contracts, the proposed system streamlines three main financing models – accounts receivable pledge financing, warehouse receipt pledge financing, and inventory pledge financing – enhancing transparency, security, and efficiency in the supply chain finance processes. The decentralized nature of blockchain technology mitigates information asymmetry and trust issues, offering a more accessible and cost-effective financing solution for SMEs compared to traditional approaches.

To ensure self-repayment of trade finance, banks often require enterprises to provide collateral, such as deposits, pledges, or guarantees, which raises the financing threshold for small and medium-sized enterprises and increases financing costs. By utilizing the immutable and traceable characteristics of blockchain to convert the credit of core enterprises (e.g., bills, credit limits, or payment rights) into digital certificates, trust can be effectively transmitted along the supply chain, reducing collaboration costs and enabling seamless credit integration.

Moreover, smart contracts can facilitate the multi-level splitting and circulation of digital certificates, significantly improving the utilization of funds and reducing the risk management complexity for financial institutions. This approach effectively addresses issues related to financing difficulties and high costs for small and medium-sized enterprises.

6.5. Real-world case study

6.5.1. An integrated platform for the industrial internet in the cable industry

In a bid to develop a cable industry cluster, optimize the industrial structure, and enhance the level of digitalization in production, a core cable manufacturing enterprise in the region partnered with a blockchain platform company to establish an integrated industrial Internet platform for the cable industry.

This blockchain platform is primarily designed to address the major data security issues inherent in industrial Internet platforms, the data credibility challenges in cable quality traceability, and the cross-border payment problems prevalent in current international commodity trading platforms.

It has achieved the integration of cable production, quality traceability, and cross-border transactions by dividing industrial production scenarios into multiple trust domains. Utilizing blockchain technology, it ensures the secure transmission, sharing, and collaboration of production, circulation, and transaction data across the three major support platforms. The platform architecture is shown in Fig. 4.

Compared to traditional industrial Internet platforms, this blockchain platform offers several advantages. Firstly, it enhances the credibility of cable product quality traceability by adding identification analysis at various stages such as product processing, manufacturing,

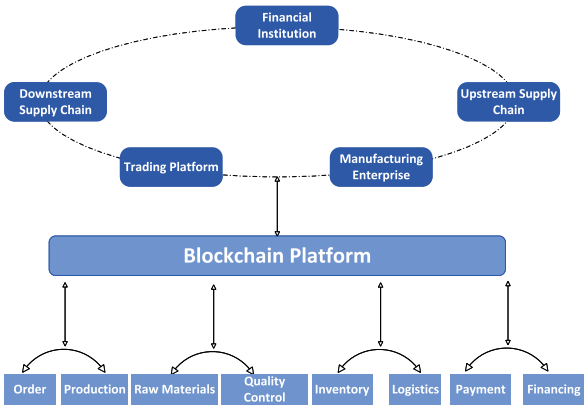

Fig. 4. The architecture of Integrated Platform for the Industrial Internet in the Cable Industry.

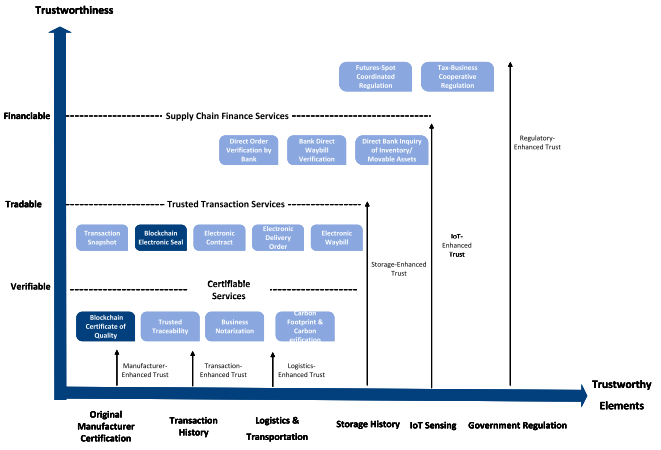

Fig. 5. The architecture of Trusted Steel Supply Chain System.

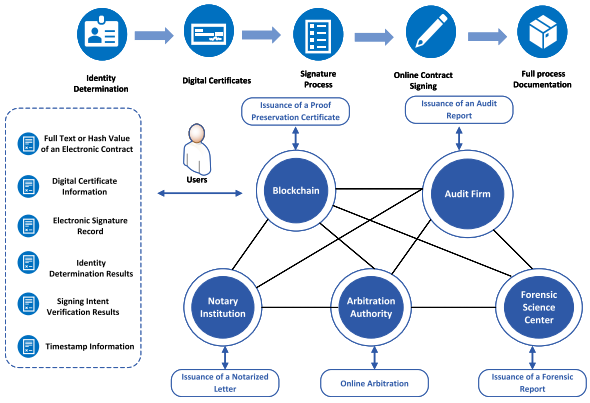

Fig. 6. The architecture of Trusted Electronic Signature System.

testing, storage, and transportation. Through trusted on-chain technologies like blockchain oracles, the platform ensures the secure and efficient on-chain storage of traceability data.

Secondly, it deploys the verification process on the blockchain as smart contracts, preventing privacy leaks of original data during traceability transmission, thereby ensuring that traceability data remains public and transparent while protecting user data privacy.

Additionally, the platform employs an internationally driven order system for comprehensive monitoring of the cable manufacturing process, enabling reverse data flow among the three major support platforms, thereby ensuring the integration of data flow, logistics, and financial flow in international cable trade.

This blockchain platform integrates regional industry resources and supply chain systems, effectively enhancing the intelligence and security levels of industrial manufacturing and the industrial Internet in the area. Additionally, its establishment has contributed to optimizing business processes and reducing operational costs in the region. Following the platform's completion, the area's industrial competitiveness has been significantly improved.

6.5.2. A blockchain-based platform for distribution and traceability in the steel industry

To address trust issues within the steel trade ecosystem and to reconstruct business oversight and governance, a renowned company in the steel sector has developed a blockchain service platform for the steel industry. This platform facilitates efficient deployment and operational services of blockchain technology, and integrates core scenarios of the steel industry to establish shared service capabilities. Typical scenarios include processing, transportation, warehousing, and trading. The blockchain platform offers comprehensive, multi-scenario business innovation applications for the steel industry and allows participation from various stakeholders such as steel manufacturers, users, warehousing, transportation, processing, and regulatory bodies within the steel ecosystem.

Currently, the platform has given rise to blockchain application services such as trusted resource verification, blockchain certificates of quality, blockchain electronic seals, blockchain delivery receipts, transaction snapshots, integrated supervision, and tax regulation. Based on these services, business systems have been established, including production service systems, intelligent logistics systems, transaction center systems, and production-supply-sales systems. As shown in Fig. 5.

These applications and services assist enterprises in establishing a trustworthy steel supply chain system from six perspectives: original equipment manufacturer certification, credible transaction history, trustworthy logistics and transportation, dependable storage records, credible IoT sensing, and trustworthy government regulation.

Application Example 1: Blockchain Certificate of Quality

In traditional methods, steel manufacturers issue certificates of quality only to direct customers and not to distributors or steel-using enterprises involved in the distribution process. This practice results in the widespread use of copied certificates, exposing them to risks of tampering and forgery. Moreover, paper-based certificates require careful storage, increasing both management workload and costs.

The steel plant utilizes the blockchain service platform to provide original, tamper-proof quality certificate data and to facilitate the online generation and splitting of these certificates. The trustworthy splitting of certificates allows downstream multi-level distributors and steel-using enterprises to obtain blockchain-based certificates, accommodating the multi-tier flow of steel transactions. As of February 2024, the blockchain service platform has stored over 5.1 million certificates and has served more than 3300 customers.

Application Example 2: Blockchain Electronic Seal

Traditional offline paper-based electronic seals suffer from prolonged signing cycles, low efficiency, risks of forgery, and high costs associated with document printing, mailing, and storage.

Deploying smart contracts on a blockchain platform with the same legal validity as paper contracts effectively addresses the aforementioned issues. As shown in Fig. 6. This solution creates a comprehensive evidence chain through services such as identity determination, digital certificates, confirmation of signing intent, online contract signing, full process documentation, and issuance of proof documents. The process is further validated by endorsements from multiple authoritative institutions, ensuring the immutability and undeniability of electronic evidence, thereby enhancing the credibility and applicability of electronic seals.

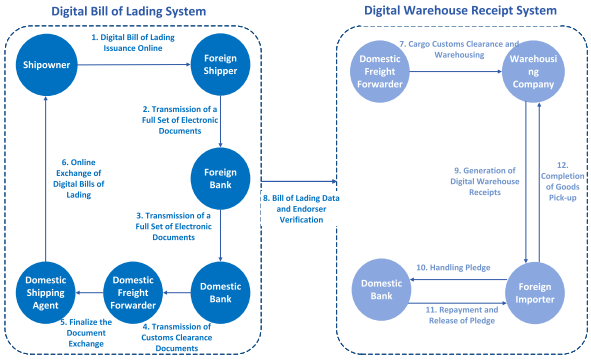

Fig. 7. The architecture of International Bulk Commodities Trade Platform.

Practical experience shows that using blockchain electronic seal services optimizes processes and enhances efficiency, reducing the entire cycle by 90% compared to traditional offline paper-based seals. Additionally, it strengthens risk management, lowers labor costs, and contributes to carbon reduction and environmental sustainability.

6.5.3. A blockchain-based digital service platform for international bulk commodities trade

The internet economy has sparked a new wave of economic development, yet the digitalization of international trade remains sluggish. As a new form of trade, digital trade spans a broad range of areas, and there is still a lack of consensus on cross-border data flow, personal privacy protection, and digital security. The global digital trade regulatory framework is also in urgent need of establishment. Blockchain technology, with its inherent alignment with the logic of international trade, has the potential to accelerate the digital transformation of international trade.

Currently, most bulk commodity trade across oceans is conducted via shipping, which is physically transparent worldwide. However, these transactions rely on centralized systems. For over a century, the core documents of international trade have been conveyed through paper, which, while known only to the holder, suffers from clear disadvantages: slow transmission, susceptibility to forgery, and risk of loss, with the added risk of virus transmission during pandemics. Existing electronic bill of lading systems are predominantly centralized, compromising the security and privacy of data.

With blockchain-based digital documents, clients can retain control over their own data. No market participant will disclose any transaction details or content on the blockchain platform, while benefiting from the significant conveniences brought by the digitization of core documents.

Additionally, establishing secure channels for assets and funds in bulk commodity trade, while addressing the TEA issues – Title, Existence, and Attornment – is a significant challenge that international trade must resolve. As shown in Fig. 7. The innovative integration of digital bills of lading and digital warehouse receipts can comprehensively address the TEA issues. Blockchain-based digital bills of lading effectively resolve the problems of title and transfer during international trade transportation. Meanwhile, digital warehouse receipts address the authenticity and transfer issues of goods while they are in domestic warehouses.

By combining these two solutions, it is possible to accurately and comprehensively record the entire chain of information – from export and arrival at port to warehousing and receipt transactions or financing – on the blockchain, creating a more secure on-chain digital ecosystem.

7. Research prospects

Blockchain has witnessed vigorous growth over the past decade, giving rise to a diverse array of distinctive blockchain platforms. The concurrent development of smart contract platforms has, however, engendered a concomitant set of novel challenges.

Currently the deployment of smart contracts by mainstream blockchain platforms is in a coarse manner, either propagated across the network or specified by the client. The execution of smart contracts is typically serial, which results in inefficient execution. In public blockchain, smart contract execution and blocks production are closely coupled which may lead to repeated execution of transactions and the inability to detect data conflicts in advance. In order to increase the throughput and scalability of existing blockchain platforms, more research on execution architecture and deployment of smart contract is quite necessary. We summarize the problems and difficulties that most blockchain platforms may have as following.

(1) Static deployment

The deployment of smart contracts on mainstream blockchain platforms is basically coarse and static, such as lacking consideration of the execution characteristics of different type of smart contracts, lacking reliability and availability evaluation of various nodes in the system, and lacking resources optimization of smart contract deployment. The execution of smart contract is short of flexibility and reliability. Optimizing the deployment methods of smart contracts to ensure a certain degree of flexibility is a worthwhile research angle. Additionally, ensuring the security of these contracts to avoid vulnerabilities is also a crucial research direction.

(2) Repeated execution

In the public blockchain system,such as Bitcoin and Ethereum, all the transactions are executed and verified homogeneously on each miner node, which results in low throughput with only a small number of transactions completed per second. Additionally, all the transactions are repeatedly executed many times for competing the right of bookkeeping, causing computing power wasted.

Although transactions can bypass network-wide validation in consortium blockchain systems, inconsistencies can still be detected during block verification and production. However, numerous transactions are repeatedly executed in vain, resulting in lowered throughput of valid transactions and wasted resources. The purpose of this repetitive execution is to ensure the trustworthiness of transactions. Investigating more optimal consensus algorithms that minimize the redundancy in transaction verification while ensuring transaction trustworthiness could significantly enhance the execution efficiency of smart contracts.

(3) Non-consistency

When multiple instances of a specific smart contract running concurrently, or when transactions within a block accessing the identical data or updating the identical state, system inconsistency may appear for the reason that the execution order of transactions on various nodes are uncertain. While some other blockchain systems introduce concurrent execution modes, this may cause node inconsistencies during the transaction ordering process, leading to concurrency conflict issues.

Currently, data conflicts arising during smart contract execution cannot be detected in advance. This is particularly true for Turing-complete blockchain systems. Whether transaction conflicts are resolved during the execution phase or the validation phase significantly impacts system performance and throughput. Effectively reducing transaction conflicts in concurrent execution processes is also a worthwhile research direction.

(4) Low degree of concurrency and low throughput

Given the current consensus mechanisms, solely optimizing consensus to enhance the execution efficiency of smart contracts is unlikely to bring about a qualitative improvement in throughput. The most feasible approach is to achieve large-scale concurrency in smart contract execution. Although current execution optimization strategies have explored concurrent execution to some extent, they are still in the early stages. Presently, research on concurrent execution remains at the end-to-end concurrency phase. Exploring larger-scale transaction concurrency execution models will effectively address the performance bottlenecks of blockchain systems.

(5) Blockchain island effect

Due to inconsistencies in technical standards and protocols, current blockchain platforms cannot achieve effective interoperability, leading to the phenomenon of blockchain islands. This significantly limits the potential of blockchain technology in cross-platform data sharing, value transfer, and the development of the blockchain ecosystem. Although research on cross-chain technology has made some progress, it is still in its infancy. Addressing the blockchain island effect requires a multifaceted strategy, including the establishment of unified technical standards to ensure compatibility between different platforms, the development of cross-chain technologies to enable data and asset exchange among various blockchains, the creation of open and interoperable platforms to foster innovation and collaboration, and the encouragement of industry cooperation and alliances to form a common technological development roadmap. These initiatives can further advance the blockchain ecosystem and expand its application scenarios.

(6) Industry 4.0: From digital industry to trusted industry

Leveraging blockchain platforms, smart contracts can significantly enhance the trustworthiness of digital production, effectively addressing the trust crisis caused by the lack of transparency in production and supply chains. Although this area is still under exploration, there have already been developments in sectors such as the food industry and supply chain traceability. Further advancing digital trust construction and establishing a "trusted industry" can effectively resolve the current trust crisis in societal production.

8. Conclusion

We have reviewed existing surveys on smart contracts and found that the focus of current research mainly revolves around smart contract platforms, applications, languages, and vulnerability detection (Table 1). However, these studies lack an in-depth analysis of the complete lifecycle and execution patterns of smart contracts. The performance issues associated with smart contract execution have emerged as a significant bottleneck hindering their widespread adoption. Enhancing smart contract execution performance has thus become a crucial research direction.

In this article, we have structured our study around the lifecycle of smart contracts to summarize and analyze potential issues and existing solutions throughout the execution process. Building upon this framework, our research primarily focuses on architectures, mechanisms and methods of how to optimize smart contract execution performance and of how to resolve conflicts during smart contracts execution. Additionally, we analyze the potential of smart contracts to empower Industry 4.0 from the perspectives of product traceability, trustworthy manufacturing, supply chain management and supply chain finance. We supplement our analysis with relevant case studies to provide practical insights.

By comprehending and applying these findings, both researchers and practitioners can make substantial advancements in blockchain technology and its applications within Industry 4.0. We envision that smart contracts and blockchain technology will facilitate the transformation of Industry 4.0 from digital industry to trusted industry. In conclusion, we identify current research gaps and propose future research directions. We believe that our study can serve as a guiding resource for researchers, engineers, educators, and readers interested in optimizing smart contract execution and its application within Industry 4.0.

CRediT authorship contribution statement

Yang Liu: Conceptualization, Supervision, Project administration, Funding acquisition, Writing – original draft, Methodology, Investigation, Validation, Writing – review & editing. **Jinlong He:** Conceptualization, Writing – original draft, Methodology, Investigation, Validation, Writing – review & editing. **Xiangyang Li:** Writing – original draft, Methodology, Investigation, Validation. **Jingwen Chen:** Writing – original draft, Investigation, Validation. **Xinlei Liu:** Investigation, Validation. **Song Peng:** Validation. **Haohao Cao:** Validation. **Yaoqi Wang:** Validation.

Declaration of competing interest

The authors declare the following financial interests/personal relationships which may be considered as potential competing interests: Yang Liu reports financial support was provided by The Fundamental Research Project of Key Scientific Research Project Plan of Colleges and Universities of Henan Province (No. 23ZX017). Yang Liu reports financial support was provided by The Key Scientific and Technological Research Project of Henan Province (No. 232102211082). Yang Liu reports financial support was provided by The Significant Scientific and Technological Project of Henan Province (No. 201300210200, 201300210100). Yang Liu reports financial support was provided by The Collaborative Innovation Key Project of Zhengzhou City (No. 21ZZXTCX07).

Data availability

No data was used for the research described in the article.

References

[1] S. Nick, Smart contracts: building blocks for digital markets, EXTROPY: J. Transhumanist Thought,(16) 18 (2) (1996) 28.
[2] W. Zou, D. Lo, P.S. Kochhar, X.-B.D. Le, X. Xia, Y. Feng, Z. Chen, B. Xu, Smart contract development: Challenges and opportunities, IEEE Trans. Softw. Eng. 47 (10) (2019) 2084–2106.
[3] Z. Zheng, S. Xie, H.-N. Dai, X. Chen, H. Wang, Blockchain challenges and opportunities: A survey, Int. J. Web Grid Serv. 14 (4) (2018) 352–375.
[4] A. Sunyaev, A. Sunyaev, Distributed ledger technology, in: Internet Computing: Principles of Distributed Systems and Emerging Internet-Based Technologies, Springer, 2020, pp. 265–299.
[5] N. Kannengießer, S. Lins, C. Sander, K. Winter, H. Frey, A. Sunyaev, Challenges and common solutions in smart contract development, IEEE Trans. Softw. Eng. 48 (11) (2021) 4291–4318.
[6] S. Tikhomirov, Ethereum: state of knowledge and research perspectives, in: Foundations and Practice of Security: 10th International Symposium, FPS 2017, Nancy, France, October 23-25, 2017, Revised Selected Papers 10, Springer, 2018, pp. 206–221.
[7] S. Wang, L. Ouyang, Y. Yuan, X. Ni, X. Han, F.-Y. Wang, Blockchain-enabled smart contracts: architecture, applications, and future trends, IEEE Trans. Syst. Man Cybern.: Syst. 49 (11) (2019) 2266–2277.
[8] M. Suvitha, R. Subha, A survey on smart contract platforms and features, in: 2021 7th International Conference on Advanced Computing and Communication Systems, ICACCS, Vol. 1, IEEE, 2021, pp. 1536–1539.
[9] S.-Y. Lin, L. Zhang, J. Li, L.-l. Ji, Y. Sun, A survey of application research based on blockchain smart contract, Wirel. Netw. 28 (2) (2022) 635–690.
[10] J. Gilcrest, A. Carvalho, Smart contracts: Legal considerations, in: 2018 IEEE International Conference on Big Data (Big Data), IEEE, 2018, pp. 3277–3281.
[11] S. Wang, Y. Yuan, X. Wang, J. Li, R. Qin, F.-Y. Wang, An overview of smart contract: architecture, applications, and future trends, in: 2018 IEEE Intelligent Vehicles Symposium, IV, IEEE, 2018, pp. 108–113.
[12] D. Harz, W. Knottenbelt, Towards safer smart contracts: A survey of languages and verification methods, 2018, arXiv preprint arXiv:1809.09805.
[13] W. Zou, D. Lo, P.S. Kochhar, X.-B.D. Le, X. Xia, Y. Feng, Z. Chen, B. Xu, Smart contract development: Challenges and opportunities, IEEE Trans. Softw. Eng. 47 (10) (2019) 2084–2106.
[14] Z. Zheng, S. Xie, H.-N. Dai, W. Chen, X. Chen, J. Weng, M. Imran, An overview on smart contracts: Challenges, advances and platforms, Future Gener. Comput. Syst. 105 (2020) 475–491.
[15] B. Hu, Z. Zhang, J. Liu, Y. Liu, J. Yin, R. Lu, X. Lin, A comprehensive survey on smart contract construction and execution: paradigms, tools, and systems, Patterns 2 (2) (2021).
[16] E.S. Negara, A.N. Hidayanto, R. Andryani, R. Syaputra, Survey of smart contract framework and its application, Information 12 (7) (2021) 257.
[17] Á.J. Varela-Vaca, A.M.R. Quintero, Smart contract languages: A multivocal mapping study, ACM Comput. Surv. 54 (1) (2021) 1–38.
[18] P. Tolmach, Y. Li, S.-W. Lin, Y. Liu, Z. Li, A survey of smart contract formal specification and verification, ACM Comput. Surv. 54 (7) (2021) 1–38.

[19] M. Sookhak, M.R. Jabbarpour, N.S. Safa, F.R. Yu, Blockchain and smart contract for access control in healthcare: A survey, issues and challenges, and open issues, J. Netw. Comput. Appl. 178 (2021) 102950.

[20] P. Qian, Z. Liu, Q. He, B. Huang, D. Tian, X. Wang, Smart contract vulnerability detection technique: A survey, 2022, arXiv preprint arXiv:2209.05872.

[21] S.-Y. Lin, L. Zhang, J. Li, L.-l. Ji, Y. Sun, A survey of application research based on blockchain smart contract, Wirel. Netw. 28 (2) (2022) 635–690.

[22] P. Sharma, R. Jindal, M.D. Borah, A review of smart contract-based platforms, applications, and challenges, Cluster Comput. 26 (1) (2023) 395–421.

[23] H. Chu, P. Zhang, H. Dong, Y. Xiao, S. Ji, W. Li, A survey on smart contract vulnerabilities: Data sources, detection and repair, Inf. Softw. Technol. (2023) 107221.

[24] N. Ivanov, C. Li, Q. Yan, Z. Sun, Z. Cao, X. Luo, Security threat mitigation for smart contracts: A comprehensive survey, ACM Comput. Surv. 55 (14s) (2023) 1–37.

[25] E. Androulaki, A. Barger, V. Bortnikov, C. Cachin, K. Christidis, A. De Caro, D. Enyeart, C. Ferris, G. Laventman, Y. Manevich, et al., Hyperledger fabric: a distributed operating system for permissioned blockchains, in: Proceedings of the Thirteenth EuroSys Conference, Association for Computing Machinery, 2018, pp. 1–15.

[26] F.J. Folgado, D. Calderón, I. González, A.J. Calderón, Review of industry 4.0 from the perspective of automation and supervision systems: Definitions, architectures and recent trends, Electronics 13 (4) (2024) 782.

[27] S. Nakamoto, Bitcoin: A peer-to-peer electronic cash system, Decentral. Bus. Rev. (2008) 21260.

[28] S. Maiyya, V. Zakhary, M.J. Amiri, D. Agrawal, A. El Abbadi, Database and distributed computing foundations of blockchains, in: Proceedings of the 2019 International Conference on Management of Data, 2019, pp. 2036–2041.

[29] C. Natoli, V. Gramoli, The blockchain anomaly, in: 2016 IEEE 15th International Symposium on Network Computing and Applications, NCA, IEEE, 2016, pp. 310–317.

[30] M. Herlihy, Blockchains from a distributed computing perspective, Commun. ACM 62 (2) (2019) 78–85.

[31] T. Hewa, M. Ylianttila, M. Liyanage, Survey on blockchain based smart contracts: Applications, opportunities and challenges, J. Netw. Comput. Appl. 177 (2021) 102857.

[32] W. Li, M. He, S. Haiquan, An overview of blockchain technology: applications, challenges and future trends, in: 2021 IEEE 11th International Conference on Electronics Information and Emergency Communication (ICEIEC) 2021 IEEE 11th International Conference on Electronics Information and Emergency Communication (ICEIEC), IEEE, 2021, pp. 31–39.

[33] H. Brakmić, H. Brakmić, Bitcoin script, in: Bitcoin and Lightning Network on Raspberry Pi: Running Nodes on Pi3, Pi4 and Pi Zero, Springer, 2019, pp. 201–224.

[34] C.S. Wright, A proof of turing completeness in bitcoin script, in: Proceedings of SAI Intelligent Systems Conference, Springer, 2019, pp. 299–313.

[35] P. Hegedűs, Towards analyzing the complexity landscape of solidity based ethereum smart contracts, in: Proceedings of the 1st International Workshop on Emerging Trends in Software Engineering for Blockchain, 2018, pp. 35–39.

[36] M. Kaleem, A. Mavridou, A. Laszka, Vyper: A security comparison with solidity based on common vulnerabilities, in: 2020 2nd Conference on Blockchain Research & Applications for Innovative Networks and Services, BRAINS, IEEE, 2020, pp. 107–111.

[37] X. Li, Z. Shi, Q. Zhang, G. Wang, Y. Guan, N. Han, Towards verifying ethereum smart contracts at intermediate language level, in: Formal Methods and Software Engineering: 21st International Conference on Formal Engineering Methods, ICFEM 2019, Shenzhen, China, November 5–9, 2019, Proceedings 21, Springer, 2019, pp. 121–137.

[38] L. Foschini, A. Gavagna, G. Martuscelli, R. Montanari, HyperLedger fabric blockchain: chaincode performance analysis, in: ICC 2020-2020 IEEE International Conference on Communications, ICC, IEEE, 2020, pp. 1–6.

[39] I. Grigg, Eos-an introduction, 2017, White paper. https://whitepaperdatabase.com/eos-whitepaper.

[40] R.G. Brown, J. Carlyle, I. Grigg, M. Hearn, Corda: an introduction, 1, (15) 2016, p. 14, R3 CEV, August.

[41] N. Dimitrijević, V. Milicevic, D. Cvijanovic, N. Zdravković, Learning the Kotlin programming language using an autograding system, 2021.

[42] W. Martino, S. Popejoy, The kadena public blockchain, 31, 2019, p. 2019, Online verfügbar unter https://d31d887a-c1e0-47c2-aa51-c69f9f998b07.filesusr.com/ugd/86a16f_1e25e5ac5db44fb7b7e4eb2fe845ce2d.pdf, zuletzt geprüft am.

[43] S.S. Chow, Z. Lai, C. Liu, E. Lo, Y. Zhao, Sharding blockchain, in: 2018 IEEE International Conference on Internet of Things (IThings) and IEEE Green Computing and Communications (GreenCom) and IEEE Cyber, Physical and Social Computing (CPSCom) and IEEE Smart Data (SmartData), IEEE, 2018, p. 1665.

[44] Z. Team, et al., The ZILLIQA whitepaper, 2017.

[45] W. Zheng, Y. Wu, X. Wu, C. Feng, Y. Sui, X. Luo, Y. Zhou, A survey of intel SGX and its applications, Front. Comput. Sci. 15 (2021) 1–15.

[46] Z. Hua, J. Gu, Y. Xia, H. Chen, B. Zang, H. Guan, vTZ: Virtualizing ARM TrustZone, in: 26th USENIX Security Symposium (USENIX Security 17), 2017, pp. 541–556.

[47] M. Fang, Z. Zhang, C. Jin, A. Zhou, An SGX-based execution framework for smart contracts upon permissioned blockchain, Distrib. Parallel Databases (2022) 1–36.

[48] S.K. Ezzat, Y.N. Saleh, A.A. Abdel-Hamid, Blockchain oracles: State-of-the-art and research directions, IEEE Access 10 (2022) 67551–67572.

[49] G. Caldarelli, From reality keys to oraclize. a deep dive into the history of bitcoin oracles, 2023, arXiv preprint arXiv:2302.07911.

[50] L. Breidenbach, C. Cachin, B. Chan, A. Coventry, S. Ellis, A. Juels, F. Koushanfar, A. Miller, B. Magauran, D. Moroz, et al., Chainlink 2.0: Next steps in the evolution of decentralized oracle networks, Chainlink Labs (2021).

[51] S. Ramesh, C. Yaashuwanth, K. Prathibanandhi, A.R. Basha, T. Jayasankar, An optimized deep neural network based DoS attack detection in wireless video sensor network, J. Ambient Intell. Humaniz. Comput. (2021) 1–14.

[52] A. Pasdar, Z. Dong, Y.C. Lee, Blockchain oracle design patterns, 2021, arXiv preprint arXiv:2106.09349.

[53] R. Mühlberger, S. Bachhofner, E. Castelló Ferrer, C. Di Ciccio, I. Weber, M. Wöhrer, U. Zdun, Foundational oracle patterns: Connecting blockchain to the off-chain world, in: Business Process Management: Blockchain and Robotic Process Automation Forum: BPM 2020 Blockchain and RPA Forum, Seville, Spain, September 13–18, 2020, Proceedings 18, Springer, 2020, pp. 35–51.

[54] S. Kalra, S. Goel, M. Dhawan, S. Sharma, ZEUS: Analyzing safety of smart contracts, in: NDSS, NDSS, 2018, pp. 1–12.

[55] L. Luu, D.-H. Chu, H. Olickel, P. Saxena, A. Hobor, Making smart contracts smarter, 2016, pp. 254–269.

[56] A. Montresor, Y. Velegrakis, The dao of wikipedia, 2019.

[57] X. Li, P. Jiang, T. Chen, X. Luo, Q. Wen, A survey on the security of blockchain systems, Future Gener. Comput. Syst. 107 (2020) 841–853.

[58] sharkteam, Annual Web3 Security Report 2022, https://sharkteam.org/report/analysis/20230116001A_en.pdf.

[59] G. Destefanis, Design patterns for smart contract in ethereum, in: 2021 IEEE 18th International Conference on Software Architecture Companion (ICSA-C), IEEE, 2021, pp. 121–122.

[60] C.R. Worley, A. Skjellum, Opportunities, challenges, and future extensions for smart-contract design patterns, in: Business Information Systems Workshops: BIS 2018 International Workshops, Berlin, Germany, 2018, Springer, 2019, pp. 264–276.

[61] N. Sharma, S. Sharma, A survey of Mythril, a smart contract security analysis tool for EVM bytecode, Indian J. Nat. Sci. 13 (2022) 75.

[62] J. Feist, G. Grieco, A. Groce, Slither: a static analysis framework for smart contracts, in: 2019 IEEE/ACM 2nd International Workshop on Emerging Trends in Software Engineering for Blockchain, WETSEB, IEEE, 2019, pp. 8–15.

[63] N.K. Shah, S. Nandurkar, M. Bilapate, M.A. Maalik, N. Harne, K. Shaik, A. Kumar, Smart contract vulnerability detection techniques for hyperledger fabric, in: 2023 IEEE 8th International Conference for Convergence in Technology (I2CT), IEEE, 2023, pp. 1–7.

[64] K. Bhargavan, A. Delignat-Lavaud, C. Fournet, A. Gollamudi, G. Gonthier, N. Kobeissi, N. Kulatova, A. Rastogi, T. Sibut-Pinote, N. Swamy, et al., Formal verification of smart contracts: Short paper, in: Proceedings of the 2016 ACM workshop on programming languages and analysis for security, ACM, 2016, pp. 91–96.

[65] E. Hildenbrandt, M. Saxena, N. Rodrigues, X. Zhu, P. Daian, D. Guth, B. Moore, D. Park, Y. Zhang, A. Stefanescu, et al., Kevm: A complete formal semantics of the ethereum virtual machine, in: 2018 IEEE 31st Computer Security Foundations Symposium, CSF, IEEE, 2018, pp. 204–217.

[66] M. Lahami, A.J. Maâlej, M. Krichen, M.A. Hammami, A comprehensive review of testing blockchain oriented software, ENASE 182 (2022) 355–362.

[67] B. Jiang, Y. Liu, W.K. Chan, Contractfuzzer: Fuzzing smart contracts for vulnerability detection, in: Proceedings of the 33rd ACM/IEEE International Conference on Automated Software Engineering, 2018, pp. 259–269.

[68] C. Liu, H. Liu, Z. Cao, Z. Chen, B. Chen, B. Roscoe, Reguard: finding reentrancy bugs in smart contracts, in: Proceedings of the 40th International Conference on Software Engineering: Companion Proceeedings, 2018, pp. 65–68.

[69] J. He, M. Balunović, N. Ambroladze, P. Tsankov, M. Vechev, Learning to fuzz from symbolic execution with application to smart contracts, in: Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security, 2019, pp. 531–548.

[70] M. Ding, P. Li, S. Li, H. Zhang, Hfcontractfuzzer: Fuzzing hyperledger fabric smart contracts for vulnerability detection, in: Evaluation and Assessment in Software Engineering, 2021, pp. 321–328.

[71] T. Hu, B. Li, Z. Pan, C. Qian, Detect defects of solidity smart contract based on the knowledge graph, IEEE Trans. Reliab. (2023).

[72] X. Xu, T. Hu, B. Li, L. Liao, Ccdetector: Detect chaincode vulnerabilities based on knowledge graph, in: 2023 IEEE 47th Annual Computers, Software, and Applications Conference, COMPSAC, IEEE, 2023, pp. 699–704.

[73] W.J.-W. Tann, X.J. Han, S.S. Gupta, Y.-S. Ong, Towards safer smart contracts: A sequence learning approach to detecting security threats, 2018, arXiv preprint arXiv:1811.06632.

[74] L. Zhang, W. Chen, W. Wang, Z. Jin, C. Zhao, Z. Cai, H. Chen, Cbgru: A detection method of smart contract vulnerability based on a hybrid model, Sensors 22 (9) (2022) 3577.

[75] Z. Li, B. Xiao, S. Guo, Y. Yang, Securing deployed smart contracts and DeFi with distributed TEE cluster, IEEE Trans. Parallel Distrib. Syst. 34 (3) (2022) 828–842.
[76] M. Brandenburger, C. Cachin, R. Kapitza, A. Sorniotti, Blockchain and trusted computing: Problems, pitfalls, and a solution for hyperledger fabric, 2018, arXiv preprint arXiv:1805.08541.
[77] X. Xu, G. Sun, L. Luo, H. Cao, H. Yu, A.V. Vasilakos, Latency performance modeling and analysis for hyperledger fabric blockchain network, Inf. Process. Manage. (2021) 102436.
[78] Y. Manevich, A. Barger, Y. Tock, Endorsement in hyperledger fabric via service discovery, 2019, 2–1.
[79] Y. Fu, M. Ren, F. Ma, H. Shi, X. Yang, Y. Jiang, H. Li, X. Shi, Evmfuzzer: detect evm vulnerabilities via fuzz testing, in: Proceedings of the 2019 27th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering, 2019, pp. 1110–1114.
[80] R. Saxena, D. Arora, V. Nagar, S. Mahapatra, Bitcoin: a digital cryptocurrency, in: Blockchain Technology: Applications and Challenges, Springer, 2021, pp. 13–28.
[81] A. Extance, Bitcoin and beyond, Nature 526 (7571) (2015) 21.
[82] J. Rosa-Bilbao, J. Boubeta-Puig, Ethereum blockchain platform, in: Distributed Computing to Blockchain, Elsevier, 2023, pp. 267–282.
[83] H. Arslanian, Ethereum, in: The Book of Crypto: The Complete Guide to Understanding Bitcoin, Cryptocurrencies and Digital Assets, Springer, 2022, pp. 91–98.
[84] S. Aggarwal, N. Kumar, Blockchain 2.0: smart contracts, 121, 2021, pp. 301–322.
[85] C. Cachin, et al., Architecture of the hyperledger blockchain fabric, in: Workshop on Distributed Cryptocurrencies and Consensus Ledgers, 310, (4) Chicago, IL, 2016, pp. 1–4.
[86] R.G. Brown, J. Carlyle, I. Grigg, M. Hearn, Corda: an introduction, 1, (15) 2016, p. 14, R3 CEV, August.
[87] B. Xu, D. Luthra, Z. Cole, N. Blakely, EOS: An architectural, performance, and economic analysis, 11, 2018, p. 2019, Retrieved June.
[88] W. Zou, D. Lo, P.S. Kochhar, X.-B.D. Le, X. Xia, Y. Feng, Z. Chen, B. Xu, Smart contract development: Challenges and opportunities, IEEE Trans. Softw. Eng. 47 (10) (2019) 2084–2106.
[89] T.-T. Kuo, H. Zavaleta Rojas, L. Ohno-Machado, Comparison of blockchain platforms: a systematic review and healthcare examples, J. Am. Med. Inform. Assoc. 26 (5) (2019) 462–478.
[90] V. Clincy, H. Shahriar, Blockchain development platform comparison, in: 2019 IEEE 43rd Annual Computer Software and Applications Conference, COMPSAC, Vol. 1, IEEE, 2019, pp. 922–923.
[91] I. Dernayka, A. Chehab, Blockchain development platforms: Performance comparison, in: 2021 11th IFIP International Conference on New Technologies, Mobility and Security, NTMS, IEEE, 2021, pp. 1–6.
[92] M.H. Tabatabaei, R. Vitenberg, N.R. Veeraragavan, Understanding blockchain: definitions, architecture, design, and system comparison, 2022, arXiv preprint arXiv:2207.02264.
[93] A. Gangwal, H.R. Gangavalli, A. Thirupathi, A survey of layer-two blockchain protocols, J. Netw. Comput. Appl. 209 (2023) 103539.
[94] J.-H. Lin, K. Primicerio, T. Squartini, C. Decker, C.J. Tessone, Lightning network: a second path towards centralisation of the bitcoin economy, New J. Phys. 22 (8) (2020) 083022.
[95] H. Li, Y. Chen, X. Shi, X. Bai, N. Mo, W. Li, R. Guo, Z. Wang, Y. Sun, FISCO-BCOS: An enterprise-grade permissioned blockchain system with high-performance, in: Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis, 2023, pp. 1–17.
[96] J. Wang, H. Wang, Monoxide: Scale out blockchains with asynchronous consensus zones, in: 16th USENIX Symposium on Networked Systems Design and Implementation (NSDI 19), 2019, pp. 95–112.
[97] L. Cai, Q. Li, X. Liang, Hyperchain application development fundamentals, in: Advanced Blockchain Technology: Frameworks and Enterprise-Level Practices, Springer, 2022, pp. 273–292.
[98] W. Yu, K. Luo, Y. Ding, G. You, K. Hu, A parallel smart contract model, in: Proceedings of the 2018 International Conference on Machine Learning and Machine Intelligence, 2018, pp. 72–77.
[99] Z. Gao, L. Xu, L. Chen, N. Shah, Y. Lu, W. Shi, Scalable blockchain based smart contract execution, in: 2017 IEEE 23Rd international conference on parallel and distributed systems (ICPADS), IEEE, 2017, pp. 352–359.
[100] C. Jin, S. Pang, X. Qi, Z. Zhang, A. Zhou, A high performance concurrency protocol for smart contracts of permissioned blockchain, IEEE Trans. Knowl. Data Eng. (2021).
[101] A. Zhang, K. Zhang, Enabling concurrency on smart contracts using multiversion ordering, in: Asia-Pacific Web (APWeb) and Web-Age Information Management (WAIM) Joint International Conference on Web and Big Data, Springer, 2018, pp. 425–439.
[102] J. Liu, P. Li, R. Cheng, N. Asokan, D. Song, Parallel and asynchronous smart contract execution, IEEE Trans. Parallel Distrib. Syst. 33 (5) (2021) 1097–1108.
[103] H. Zhang, J. Li, H. Zhao, T. Zhou, N. Sheng, H. Pan, BlockPilot: A proposer-validator parallel execution framework for blockchain, in: Proceedings of the 52nd International Conference on Parallel Processing, 2023, pp. 193–202.
[104] D. Xian, X. Wei, SC-chef: Turboboosting smart contract concurrent execution for high contention workloads via chopping transactions, IEEE Trans. Reliab. (2023).
[105] X. Qi, Z. Chen, H. Zhuo, Q. Xu, C. Zhu, Z. Zhang, C. Jin, A. Zhou, Y. Yan, H. Zhang, Schain: Scalable concurrency over flexible permissioned blockchain, in: 2023 IEEE 39th International Conference on Data Engineering, ICDE, IEEE, 2023, pp. 1901–1913.
[106] D. Tennakoon, V. Gramoli, Dynamic blockchain sharding, in: 5th International Symposium on Foundations and Applications of Blockchain 2022 (FAB 2022), Schloss Dagstuhl-Leibniz-Zentrum für Informatik, 2022.
[107] P. Zheng, Q. Xu, Z. Zheng, Z. Zhou, Y. Yan, H. Zhang, Meepo: Multiple execution environments per organization in sharded consortium blockchain, IEEE J. Sel. Areas Commun. 40 (12) (2022) 3562–3574.
[108] Z. Yang, R. Yang, F.R. Yu, M. Li, Y. Zhang, Y. Teng, Sharded blockchain for collaborative computing in the Internet of Things: Combined of dynamic clustering and deep reinforcement learning approach, IEEE Internet Things J. 9 (17) (2022) 16494–16509.
[109] J. Zhang, W. Chen, Z. Hong, G. Xiao, L. Du, Z. Zheng, Efficient execution of arbitrarily complex cross-shard contracts for blockchain sharding, IEEE Trans. Comput. (01) (2024) 1–14.
[110] L. Jia, Y. Liu, K. Wang, Y. Sun, Estuary: A low cross-shard blockchain sharding protocol based on state splitting, IEEE Trans. Parallel Distrib. Syst. (2024).
[111] K. Mu, X. Wei, Efshard: Towards efficient state sharding blockchain via flexible and timely state allocation, IEEE Trans. Netw. Serv. Manag. (2023).
[112] V. Bakhtiary, M. Mirabi, A. Salajegheh, S.H. Erfani, Combo-chain: Towards a hierarchical attribute-based access control system for IoT with smart contract and sharding technique, Int. Things 25 (2024) 101080.
[113] B. Yu, H. Zhao, T. Zhou, N. Sheng, X. Li, J. Xu, OverShard: Scaling blockchain by full sharding with overlapping network and virtual accounts, J. Netw. Comput. Appl. 220 (2023) 103748.
[114] H. Yang, X. Zhang, Z. Wu, L. Wang, X. Chen, L. Liu, Co-sharding: A sharding scheme for large-scale internet of things application, in: Distributed Ledger Technologies: Research and Practice, ACM New York, NY, 2024.
[115] M. Li, W. Wang, J. Zhang, LB-chain: Load-balanced and low-latency blockchain sharding via account migration, IEEE Trans. Parallel Distrib. Syst. (2023).
[116] J. Xi, G. Xu, S. Zou, Y. Lu, G. Li, J. Xu, R. Wang, A blockchain dynamic sharding scheme based on hidden Markov model in collaborative IoT, IEEE Internet Things J. (2023).
[117] Z. Cui, Z. Xue, Y. Ma, X. Cai, J. Chen, A many-objective optimized sharding scheme for blockchain performance improvement in end-edge enabled internet of things, IEEE Internet Things J. (2023).
[118] O. Kanwhen, J. Jain, A. Mohamed, Optimization and scalability of blockchain enabled demand response smart contracts using sharding and neural networks, in: 2023 IEEE PES Innovative Smart Grid Technologies Europe (ISGT EUROPE), IEEE, 2023, pp. 1–5.
[119] P.M. Dhulavvagol, M. Prasad, N.C. Kundur, N. Jagadisha, S. Totad, Scalable blockchain architecture: Leveraging hybrid shard generation and data partitioning, Int. J. Adv. Comput. Sci. Appl. 14 (8) (2023).
[120] W.F. Silvano, R. Marcelino, Iota Tangle: A cryptocurrency to communicate Internet-of-Things data, Fut. Gener. Comput. Syst. 112 (2020) 307–319.
[121] L. Baird, Hashgraph consensus: fair, fast, byzantine fault tolerance, Swirlds Tech Report, Tech. Rep., 2016.
[122] C. LeMahieu, Nano: A feeless distributed cryptocurrency network, Online Resource 16 (2018) 17.
[123] Y. Sompolinsky, Y. Lewenberg, A. Zohar, Spectre: A fast and scalable cryptocurrency protocol, Cryptology ePrint Archive (2016).
[124] C. Bai, State-of-the-art and future trends of blockchain based on dag structure, in: Structured Object-Oriented Formal Language and Method: 8th International Workshop, SOFL+ MSVL 2018, Gold Coast, QLD, Australia, November 16, 2018, Revised Selected Papers 8, Springer, 2019, pp. 183–196.
[125] M. Piduguralla, S. Chakraborty, P.S. Anjana, S. Peri, DAG-based efficient parallel scheduler for blockchains: Hyperledger sawtooth as a case study, in: European Conference on Parallel Processing, Springer, 2023, pp. 184–198.
[126] S. Zhang, J. Xiao, E. Wu, F. Cheng, B. Li, W. Wang, H. Jin, MorphDAG: A workload-aware elastic DAG-based blockchain, IEEE Trans. Knowl. Data Eng. (2024).
[127] W. Yang, M. Ao, J. Sun, G. Wang, Y. Li, C. Li, Z. Shao, Adaptive parallel scheduling scheme for smart contract, Mathematics 12 (9) (2024) 1347.
[128] Q. Wang, J. Yu, S. Chen, Y. Xiang, Sok: Dag-based blockchain systems, ACM Comput. Surv. 55 (12) (2023) 1–38.
[129] L. Baird, A. Luykx, The hashgraph protocol: Efficient asynchronous BFT for high-throughput distributed ledgers, in: 2020 International Conference on Omni-layer Intelligent Systems (COINS), IEEE, 2020, pp. 1–7.
[130] J.A. Chacko, R. Mayer, H.-A. Jacobsen, Why do my blockchain transactions fail? A study of hyperledger fabric, in: Proceedings of the 2021 International Conference on Management of Data, 2021, pp. 221–234.
[131] Y. Lu, C. Liu, M. Zhao, X. Duo, P. Xu, Z. Zhou, X. Feng, FSC: A fast smart contract transaction execution approach via read-write static analysis, Authorea Preprints (2023).

[132] L. Xu, W. Chen, Z. Li, J. Xu, A. Liu, L. Zhao, Locking mechanism for concurrency conflicts on hyperledger fabric, in: Web Information Systems Engineering–WISE 2019: 20th International Conference, Hong Kong, China, January 19–22, 2020, Proceedings 20, Springer, 2019, pp. 32–47.
[133] L. Xu, W. Chen, Z. Li, J. Xu, A. Liu, L. Zhao, Solutions for concurrency conflict problem on hyperledger fabric, World Wide Web 24 (2021) 463–482.
[134] S. Zhang, E. Zhou, B. Pi, J. Sun, K. Yamashita, Y. Nomura, A solution for the risk of non-deterministic transactions in hyperledger fabric, in: 2019 IEEE International Conference on Blockchain and Cryptocurrency (ICBC), IEEE, 2019, pp. 253–261.
[135] M. Debreczeni, A. Klenik, I. Kocsis, Transaction conflict control in hyperledger fabric: a taxonomy, gaps, and design for conflict prevention, IEEE Access (2024).
[136] Z. Xu, D. Liao, X. Dong, H. Han, Z. Yan, K. Ye, LMQF: Hyperledger fabric concurrent transaction conflict solution based on distributed lock and message queue, in: 2023 26th International Conference on Computer Supported Cooperative Work in Design, CSCWD, IEEE, 2023, pp. 1855–1860.
[137] Z. Xu, D. Liao, X. Dong, H. Han, Z. Yan, K. Ye, LMQF: Hyperledger fabric concurrent transaction conflict solution based on distributed lock and message queue, in: 2023 26th International Conference on Computer Supported Cooperative Work in Design, CSCWD, IEEE, 2023, pp. 1855–1860.
[138] J. Silva, E. Alchieri, F. Dotti, F. Pedone, Parallel execution of transactions based on dynamic and self-verifiable conflict analysis, in: Proceedings of the 12th Latin-American Symposium on Dependable and Secure Computing, 2023, pp. 110–119.
[139] A. Sharma, F.M. Schuhknecht, D. Agrawal, J. Dittrich, Blurring the lines between blockchains and database systems: the case of hyperledger fabric, in: Proceedings of the 2019 International Conference on Management of Data, 2019, pp. 105–122.
[140] P. Ruan, D. Loghin, Q.-T. Ta, M. Zhang, G. Chen, B.C. Ooi, A transactional perspective on execute-order-validate blockchains, in: Proceedings of the 2020 ACM SIGMOD International Conference on Management of Data, 2020, pp. 543–557.
[141] Z. Hong, S. Guo, E. Zhou, J. Zhang, W. Chen, J. Liang, J. Zhang, A. Zomaya, Prophet: Conflict-free sharding blockchain via Byzantine-tolerant deterministic ordering, 2023, arXiv preprint arXiv:2304.08595.
[142] A. Singh, K. Click, R.M. Parizi, Q. Zhang, A. Dehghantanha, K.-K.R. Choo, Sidechain technologies in blockchain networks: An examination and state-of-the-art review, J. Netw. Comput. Appl. 149 (2020) 102471.
[143] B. Podgorelec, M. Heričko, M. Turkanović, State channel as a service based on a distributed and decentralized web, IEEE Access 8 (2020) 64678–64691.
[144] C.W. Purnadi, S. Yazid, Sidechain implementation strategies to improve blockchain scalability, in: AIP Conference Proceedings, Vol. 2920, (1) AIP Publishing, 2024.
[145] Z. Liu, Y. Xiang, J. Shi, P. Gao, H. Wang, X. Xiao, B. Wen, Y.-C. Hu, Hyperservice: Interoperability and programmability across heterogeneous blockchains, in: Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security, ACM, 2019, pp. 549–566.
[146] H. Su, B. Guo, J.Y. Lu, X. Suo, Cross-chain exchange by transaction dependence with conditional transaction method, Soft Comput. 26 (3) (2022) 961–976.
[147] H. Zhang, H. Su, X. Wu, Y. Yang, Cross-chain interoperability and collaboration for keyword-based embedded smart contracts in the internet of things, IEEE Internet Things J. (2023).
[148] Y. Chen, A. Asheralieva, X. Wei, Atomci: A new system for the atomic cross-chain smart contract invocation spanning heterogeneous blockchains, IEEE Trans. Netw. Sci. Eng. (2024).
[149] M. Sober, G. Scaffino, M. Sigwart, P. Frauenthaler, M. Levonyak, S. Schulte, A framework for asynchronous cross-blockchain smart contract calls, in: 2023 Fifth International Conference on Blockchain Computing and Applications, BCCA, IEEE, 2023, pp. 294–301.
[150] T. Han, J. Mao, S. Xie, Q. Gao, Q. Wang, P. Zhang, Y. Fang, et al., VM-studio: A universal crosschain smart contract verification and execution scheme, Secur. Commun. Netw. 2023 (2023).
[151] C.Á. López, Y. Mezquita, D. Valdeolmillos, Bridging protocols in DAG-based DLTs: Facilitating interaction between IOTA and other DLT networks, in: International Congress on Blockchain and Applications, Springer, 2023, pp. 472–481.
[152] I.M. Ali, N. Lasla, M.M. Abdallah, A. Erbad, SRP: An efficient runtime protection framework for blockchain-based smart contracts, J. Netw. Comput. Appl. 216 (2023) 103658.
[153] D. Xian, X. Wei, Icoe: A lightweight group-consensus based off-chain execution model for smart contract based industrial applications, IEEE Trans. Ind. Inform. (2023).
[154] I.M. Ali, M.M. Abdallah, On off-chaining smart contract runtime protection: A queuing model approach, IEEE Trans. Parallel Distrib. Syst. (2024).
[155] S. Reno, S.H. Priya, G.A. Al-Kafi, S. Tasfia, M.K. Turna, A novel approach to optimizing transaction processing rate and space requirement of blockchain via off-chain architecture, Int. J. Inf. Technol. 16 (4) (2024) 2379–2394.
[156] T. Cai, W. Chen, K.E. Psannis, S.K. Goudos, Y. Yu, Z. Zheng, S. Wan, On-chain and off-chain scalability techniques, in: Blockchain Scalability, Springer, 2023, pp. 81–96.
[157] W. Chen, Z. Yang, J. Zhang, J. Liang, Q. Sun, F. Zhou, Enhancing blockchain performance via on-chain and off-chain collaboration, in: International Conference on Service-Oriented Computing, Springer, 2023, pp. 393–408.
[158] H. Hees, Raiden network: Off-chain state network for fast DApps, in: Devcon Two, Ethereum Foundation Bern, Switzerland, 2016.
[159] A. Miller, I. Bentov, S. Bakshi, R. Kumaresan, P. McCorry, Sprites and state channels: Payment networks that go faster than lightning, in: International Conference on Financial Cryptography and Data Security, Springer, 2019, pp. 508–526.
[160] L.T. Thibault, T. Sarry, A.S. Hafid, Blockchain scaling using rollups: A comprehensive survey, IEEE Access (2022).
[161] J. Poon, V. Buterin, Plasma: Scalable autonomous smart contracts, White paper, 2017, pp. 1–47.
[162] M.M. Chakravarty, J. Chapman, K. MacKenzie, O. Melkonian, M. Peyton Jones, P. Wadler, The extended UTXO model, in: Financial Cryptography and Data Security: FC 2020 International Workshops, AsiaUSEC, CoDeFi, VOTING, and WTSC, Kota Kinabalu, Malaysia, February 14, 2020, Revised Selected Papers 24, Springer, 2020, pp. 525–539.
[163] X. Dai, B. Xiao, J. Xiao, H. Jin, An efficient block validation mechanism for UTXO-based blockchains, in: 2022 IEEE International Parallel and Distributed Processing Symposium (IPDPS), IEEE, 2022, pp. 1250–1260.
[164] C. Li, P. Li, D. Zhou, W. Xu, F. Long, A. Yao, Scaling nakamoto consensus to thousands of transactions per second, 2018, arXiv preprint arXiv:1805.03870.
[165] Y. Li, H. Liu, J. Gao, J. Zhang, Z. Guan, Z. Chen, Accelerating block lifecycle on blockchain via hardware transactional memory, J. Parallel Distrib. Comput. 184 (2024) 104779.
[166] T. Lu, L. Peng, SCU: A hardware accelerator for smart contract execution, in: 2023 IEEE International Conference on Blockchain (Blockchain), IEEE, 2023, pp. 356–364.
[167] M. Fang, Z. Zhang, C. Jin, A. Zhou, An SGX-based execution framework for smart contracts upon permissioned blockchain, Distrib. Parallel Databases 42 (2) (2024) 143–178.
[168] K. Yavaprabhas, M. Pournader, S. Seuring, Blockchain as the "trust-building machine" for supply chain management, Ann. Oper. Res. 327 (1) (2023) 49–88.
[169] Z. Guo, G. Wang, Y. Li, J. Ni, G. Zhang, Attribute-based data sharing scheme using blockchain for 6g-enabled vanets, IEEE Trans. Mob. Comput. 23 (4) (2023) 3343–3360.
[170] WHO, Food safety, 2022(5), https://www.who.int/news-room/fact-sheets/detail/food-safety.
[171] F. Tian, An agri-food supply chain traceability system for China based on rfid & blockchain technology, in: 2016 13th International Conference on Service Systems and Service Management, ICSSSM, IEEE, 2016, pp. 1–6.
[172] D. Mao, F. Wang, Z. Hao, H. Li, Credit evaluation system based on blockchain for multiple stakeholders in the food supply chain, Int. J. Environ. Res. Public Health 15 (8) (2018) 1627.
[173] G. Baralla, A. Pinna, G. Corrias, Ensure traceability in European food supply chain by using a blockchain system, in: 2019 IEEE/ACM 2nd International Workshop on Emerging Trends in Software Engineering for Blockchain (WETSEB), IEEE, 2019, pp. 40–47.
[174] F. Casino, V. Kanakaris, T.K. Dasaklis, S. Moschuris, S. Stachtiaris, M. Pagoni, N.P. Rachaniotis, Blockchain-based food supply chain traceability: a case study in the dairy sector, Int. J. Prod. Res. 59 (19) (2021) 5758–5770.
[175] K.M. Botcha, V.V. Chakravarthy, et al., Enhancing traceability in pharmaceutical supply chain using internet of things (IoT) and blockchain, in: 2019 IEEE International Conference on Intelligent Systems and Green Technology, ICISGT, IEEE, 2019, pp. 45–453.
[176] J. Xu, J. Lou, W. Lu, L. Wu, C. Chen, Ensuring construction material provenance using Internet of Things and blockchain: Learning from the food industry, J. Ind. Inf. Integr. 33 (2023) 100455.
[177] M.T. Gaudio, S. Chakraborty, S. Curcio, Agri-food supply-chain traceability: a multi-layered solution, in: 2022 IEEE Intl Conf on Dependable, Autonomic and Secure Computing, Intl Conf on Pervasive Intelligence and Computing, Intl Conf on Cloud and Big Data Computing, Intl Conf on Cyber Science and Technology Congress (DASC/PiCom/CBDCom/CyberSciTech), IEEE, 2022, pp. 1–5.
[178] O. Leteane, Y. Ayalew, Improving the trustworthiness of traceability data in food supply chain using blockchain and trust model, J. Br. Blockchain Assoc. (2024).
[179] M. Mohammed, M. Alzahrani, A. Hejjou, M. Alharby, TrustChain: Trusted blockchain-based system for supply chain traceability, Arab. J. Sci. Eng. (2024) 1–19.
[180] M.M. Nuttah, P. Roma, G.L. Nigro, G. Perrone, Understanding blockchain applications in industry 4.0: From information technology to manufacturing and operations management, J. Ind. Inf. Integr. 33 (2023) 100456.
[181] R.Y. Zhong, X. Xu, E. Klotz, S.T. Newman, Intelligent manufacturing in the context of industry 4.0: a review, Engineering 3 (5) (2017) 616–630.
[182] W. Viriyasitavat, Z. Bi, D. Hoonsopon, Blockchain technologies for interoperation of business processes in smart supply chains, J. Ind. Inf. Integr. 26 (2022) 100326.

[183] H. Lu, K. Huang, M. Azimi, L. Guo, Blockchain technology in the oil and gas industry: A review of applications, opportunities, challenges, and risks, Ieee Access 7 (2019) 41426–41444.
[184] N. Mohamed, J. Al-Jaroodi, Applying blockchain in industry 4.0 applications, in: 2019 IEEE 9th Annual Computing and Communication Workshop and Conference (CCWC), 2019, pp. 0852–0858.
[185] S.M. Umran, S. Lu, Z.A. Abduljabbar, V.O. Nyangaresi, Multi-chain blockchain based secure data-sharing framework for industrial IoTs smart devices in petroleum industry, Int. Things 24 (2023) 100969.
[186] L. Yang, W. Zou, J. Wang, Z. Tang, EdgeShare: A blockchain-based edge data-sharing framework for Industrial Internet of Things, Neurocomputing 485 (2022) 219–232.
[187] C. Zhang, T. Shen, F. Bai, Toward secure data sharing for the iot devices with limited resources: A smart contract–based quality-driven incentive mechanism, IEEE Internet Things J. (2022).
[188] D. Hawashin, M. Nemer, K. Salah, R. Jayaraman, D. Svetinovic, E. Damiani, Blockchain and NFT-based traceability and certification for UAV parts in manufacturing, J. Ind. Inf. Integr. 39 (2024) 100597.
[189] H. Zhao, K. Hu, Z. Yuan, S. Yao, L. Feng, BCTMSSF: a blockchain consensus-based traceability method for supply chain in smart factory, J. Intell. Manuf. (2024) 1–17.
[190] F. Dietrich, A. Turgut, D. Palm, L. Louw, Smart contract-based blockchain solution to reduce supply chain risks, in: Advances in Production Management Systems. Towards Smart and Digital Manufacturing: IFIP WG 5.7 International Conference, APMS 2020, Novi Sad, Serbia, August 30–September 3, 2020, Proceedings, Part II, Springer, 2020, pp. 165–173.
[191] T.T. Okanlawon, L.O. Oyewobi, R.A. Jimoh, Effect of blockchain technology adoption on construction supply chain: a structural equation modelling (SEM) approach, J. Facilities Manag. (2024).
[192] V. Ciotta, G. Mariniello, D. Asprone, A. Botta, G. Manfredi, Integration of blockchains and smart contracts into construction information flows: Proof-of-concept, Autom. Constr. 132 (2021) 103925.
[193] G. Chen, M. Liu, Y. Zhang, Z. Wang, S.M. Hsiang, C. He, Using images to detect, plan, analyze, and coordinate a smart contract in construction, J. Manage. Eng. 39 (2) (2023) 04023002.
[194] X. Zhang, X. Feng, Z. Jiang, Q. Gong, Y. Wang, A blockchain-enabled framework for reverse supply chain management of power batteries, J. Clean. Prod. 415 (2023) 137823.
[195] R. Brandín, S. Abrishami, IoT-BIM and blockchain integration for enhanced data traceability in offsite manufacturing, Autom. Constr. 159 (2024) 105266.
[196] M. Javaid, A. Haleem, R.P. Singh, R. Suman, S. Khan, A review of blockchain technology applications for financial services, BenchCouncil Trans. Benchmarks Standards Eval. 2 (3) (2022) 100073.
[197] S. Wang, M. Zhou, S. Xiang, Blockchain-enabled utility optimization for supply chain finance: An evolutionary game and smart contract based approach, Mathematics 12 (8) (2024) 1243.
[198] V. Natanelov, S. Cao, M. Foth, U. Dulleck, Blockchain smart contracts for supply chain finance: Mapping the innovation potential in Australia-China beef supply chains, J. Ind. Inf. Integr. 30 (2022) 100389.
[199] M. Su, J. Yu, P. Du, W. Zan, H. Li, L. Han, Y. Bai, Feasibility analysis of blockchain technology in addressing supply chain finance bottlenecks, in: 3rd International Academic Conference on Blockchain, Information Technology and Smart Finance (ICBIS 2024), Atlantis Press, 2024, pp. 25–33.
[200] E. Battelli, Smart contracts in the financial sector: Fintech's prospects and risks, in: The Transformation of Private Law–Principles of Contract and Tort As European and International Law: A Liber Amicorum for Mads Andenas, Springer, 2024, pp. 995–1010.
[201] X. Yu, Blockchain-based supply chain financial services using smart contract, in: Proceedings of the 2021 4th International Conference on Blockchain Technology and Applications, 2021, pp. 63–69.

Yang Liu is a professor in School of information science and Engineering at Henan university of Technology, China. She received her Ph.D. degree from Northwestern Polytechnical University. Currently, she is a member of the System Software technical committee and Blockchain technical committee of China Computer Federation (CCF), deputy director of Academic Committee of Henan blockchain Technology Research Association. Her main research interests include blockchain, cloud computing, and distributed computing.