Survey | Received 14 April 2026; Revised 17 May 2026; Accepted 20 May 2026; Published 28 July 2026
https://doi.org/10.55092/blockchain20260005

A survey of blockchain transaction processing and scalability: toward a trust infrastructure for AI agents

Jiana Liao, Qinde Chen, Jian Zheng, Xiaoke Tang, Feihong Hu and Huawei Huang*

School of Software Engineering, Sun Yat-sen University, Zhuhai, China

* Correspondence author; E-mail: huanghw28@mail.sysu.edu.cn.

Highlights:

- We review the existing methods of blockchain transaction processing and scalability.
- We organize the survey into five layers, including data, network, consensus, execution, and application layers.
- We connect the blockchain scalability bottlenecks with the trust risks of AI agents. Specifically, the trust risks include the loss of the accountability, cross-domain inconsistency, delayed finality, and so on.
- We discuss open issues, which demonstrates what can blockchain technologies develop when combined with AI agent.

Abstract: With the increase of artificial intelligence (AI) agents being used in the operation of digital systems, they can perform complex tasks, manage resources and interact with distributed environments. However, they require additional support in checking the execution, assigning duties and guaranteeing reliability. Therefore, it is difficult to achieve autonomous decision-making along with stable system operation. Consequently, blockchain is regarded not only as a platform for decentralized applications but also as a dependable base for AI agents. Hence, the present blockchain architectures should not only enhance their efficiency but also ensure traceable state modifications, authentic operations and coordinated communications among various AI-related activities. Therefore, our paper investigates the transaction processing and expanding methods from different perspectives. We classify the previous research into five parts: state control and security measures, improvement of sharding, scalability of consensus and Layer-2 architectures, parallel transaction processing systems and application-oriented trading procedures. For each part, we introduce some typical systems, describe their features in design and evaluate the influences on constructing scalable and trustworthy AI agent systems. Besides, we point out the difficulties in establishing the blockchain framework for AI agents. Finally, this survey presents the existing technologies systematically and highlights their importance.

Keywords: blockchain scalability; transaction processing; AI agents; trust infrastructure; sharding; parallel execution; Layer-2 scaling; state management

1. Introduction

Blockchains regulate the interactions among untrustworthy entities by means of consensus, cryptographic authentication and smart contracts [1,2]. At present, they are applied in decentralized finance, digital identification and supply-chain administration. Nevertheless, the transaction processing and scalability problems still exist: the nodes have to keep larger states, deal with more transactions and ensure agreement under greater demands [3].

Prior studies have enhanced the scalability of the blockchain by using sharding, Layer-2 schemes, efficient consensus algorithms and parallel processing devices [2,3]. These systems are usually evaluated by the throughput, latency, storage expense, computational load, and modularity. However, these criteria are not sufficient to reflect the workload done by the software agents. An agent can perform actions repeatedly, handle tasks from different contracts and chains, use the most recent state,s and adjust to the incentives. In this situation, the scalability also influences the feasibility of verification, responsibility, and collaboration.

This shift is becoming particularly evident with the rapid development of Artificial Intelligence (AI) agents. The recent language-model-based and tool-using agents can accomplish tasks, interact with external systems and handle digital resources [4–8]. However, their behaviors are difficult to investigate and the decision-making processes may not be clear to the users, developers and external systems [8,9]. Moreover, there is another type of risk. The present system lacks a permanent and reliable record of what an agent has experienced, which tool has been utilized, which state changes have taken place and which person should bear the responsibility. Blockchain systems can overcome this issue by ensuring a credible execution, an auditable state alteration, a responsible arrangement and cooperation among the parties who do not fully trust each other [10].

However, the ability of blockchain to serve as such a trust infrastructure depends critically on its scalability. As the agents can carry out actions very quickly, the throughput and congestion control determine whether the commitments can be fulfilled before the environment changes. Moreover, since their work may involve contracts, networks and external services, the cooperation between different databases and different fields decide whether a joint action can maintain the integrity and be verified after failures. If the agent's decisions relate to assets, permissions or security operations, the scalable systems must also guarantee deterministic finality, verifiable state modifications and accountable sequences. Otherwise, delays, inconsistent states, unverified off-line computations or adjustable sequences may cause loss of trust though the fundamental protocol is secure.

Figure 1 presents the cross-layer perspective employed in this survey. It relates the agent-level requirements, including verifiable execution, coordination, auditability and economic reliability, to the blockchain layers which can satisfy these requirements. This perspective enables us to regard the existing scalability techniques as means of maintaining certain types of trust, rather than just improving throughput or latency.

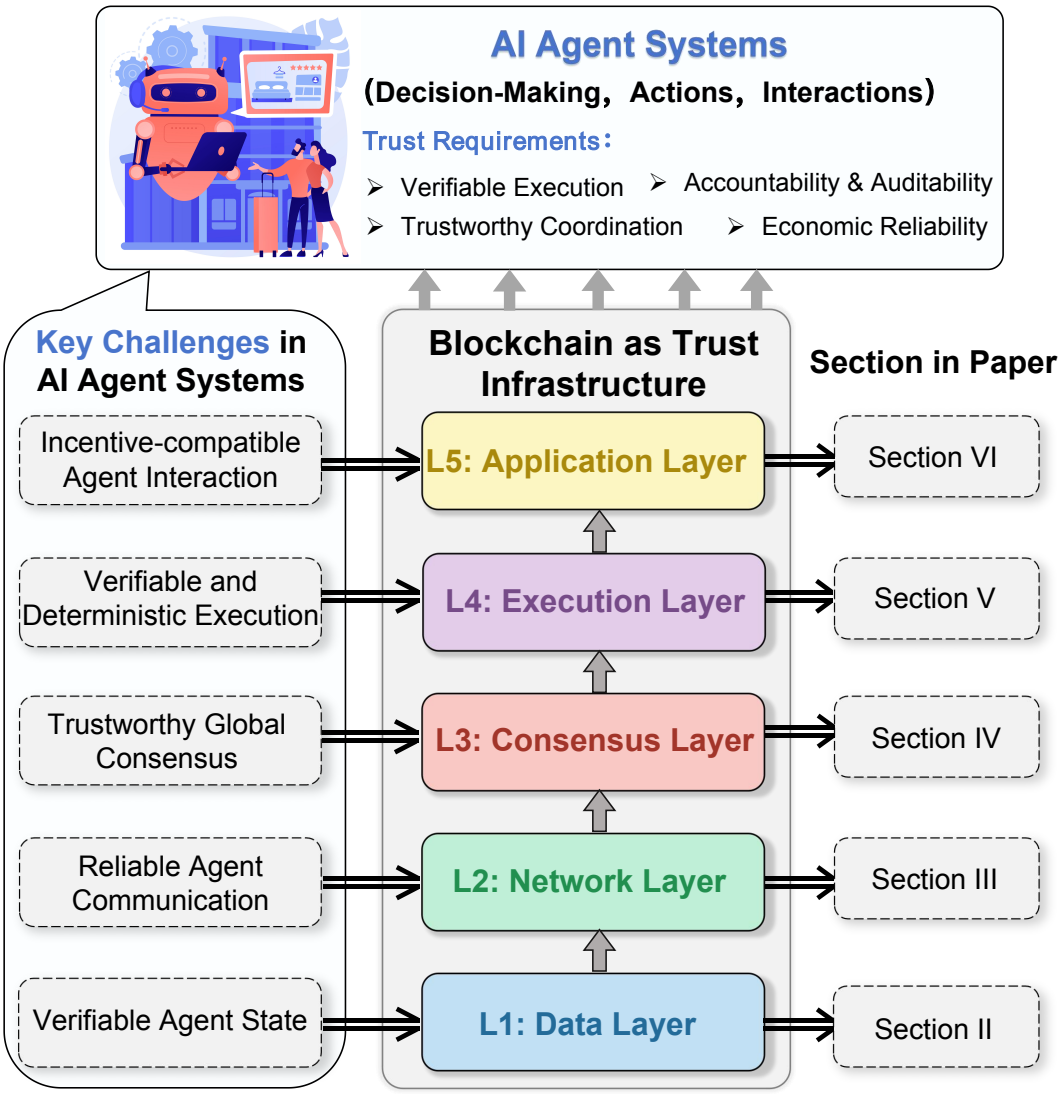

Figure 1. Blockchain as a trust infrastructure for AI-driven agents. Each layer (L1–L5) contributes a different type of support: evidence preservation, cross-domain coordination, finality, reproducible execution, or economic reliability. The figure maps agent-level trust risks to the corresponding blockchain layers reviewed in this survey.

Table 1 illustrates the mapping more clearly. An increased state growth increases the expense of past verification and reduces the continuous accountability. Delay across different divisions and partial completion may cause failure in multi-stage plans. Slowness in reaching consensus or congestion in settlement might result in finality occurring too late for an action. Sequential execution restricts concurrency, whereas unsafe concurrency affects the reliability of replaying. The fluctuations in fees, shortage of liquidity and controllable scheduling decrease the economic reliability. These instances indicate the importance of scalability techniques for accountability, coordination, finality, repeatability and stability under self-controlled loads.

Table 1. Mapping blockchain scalability techniques to AI-agent trust requirements.

Agent-level requirement	Blockchain scalability bottleneck	Relevant techniques	Trust risk if unresolved
Persistent accountability	Growing state and costly historical verification	State pruning, verifiable state transition, redactable governance	Agent actions become difficult to audit, attribute, or dispute over long time horizons.
Cross-domain coordination	Cross-shard latency and inconsistent partial execution	Cross-shard communication, dynamic sharding, cross-chain verification	Multi-step agent plans may fail without clear atomicity, rollback, or verifiable failure semantics.
Timely finality	Slow consensus and congested settlement layers	Directed Acyclic Graph (DAG) consensus, Layer-2 systems, succinct verification	Agent decisions may be executed on stale states, leading to unreliable commitments and strategic exploitation.
Concurrent trustworthy execution	Sequential smart-contract execution and ordering conflicts	Parallel execution, conflict detection, deterministic replay	Independent agent actions cannot scale, while unsafe concurrency may create inconsistent or non-reproducible states.
Economic reliability	Unstable fees, liquidity shortages, and manipulable scheduling	Resource scheduling, liquidity incentives, gas hedging	Autonomous agents face unpredictable execution costs and may be exposed to incentive manipulation or unfair ordering.

Several unsolved difficulties exist in the system [11–13]. The expansion of global state leads to higher storage and confirmation expenses. The division of shards causes delays in coordination. The consensus and settlement levels are still confronted with the dilemmas of latency, bandwidth and trust assumptions. Parallel processing enhances the throughput, but it may result in orderings, replays and conflict resolutions. At the application level, the charges, the liquidity and the policies for scheduling affect the predictable completion of the autonomous actions.

We classify the handling and scalability of blockchain transactions from this hierarchical viewpoint. The papers are categorized based on state management and verification, sharding, scalable consensus and Layer-2 structures, parallel execution and application-oriented trading schemes. At each phase, we examine the techniques used to improve scalability and ensure reliability for agent-based systems.

The survey judges each method by presenting three questions: whether it can accomplish an entire agent workflow from beginning to end, whether its assumptions are still reasonable in uncontrolled and hostile situations, and whether its restrictions are mainly due to the high cost of implementation or fundamental limitations concerning finality, data accessibility, semantic verification and economic influence. We also investigate whether the method maintains responsibility when agents communicate between different networks, agreements and external systems.

Table 2 shows the representative transaction-processing and scalability systems mentioned in the survey and their comparison across different architectural layers.

Table 2. Comprehensive comparison of representative blockchain transaction processing and scalability solutions.

Architectural Layer	Key Literature	Core Methodology	Optimization Goal
L1: Data Layer	MoltDB [14], Specular [15], Narrator-Pro [16]	Ancient state segregation; One-step fraud proofs; Two-level rewriting.	Storage ↓, Trust ↑
L2: Network Layer	DCchain [17], LightCross [18], Orbit [19], Sharon [20]	Dual-layer consensus; Distributed collectors; Active account migration.	Latency ↓, Throughput ↑
L3: Consensus Layer	Ladder [21], Porygon [22], MPC-Head [23]	DAG aggregation; 3D parallel statelessness; Succinct MPC proofs.	Finality ↑, Node Load ↓
L4: Execution Layer	Vegeta [24], Partial Order-Based Ledger (POBL) [25], Asynchronous Order Atomic Broadcast (AOAB) [26]	Speculate-order-replay; PGraph scheduling; Absolute timestamping.	Concurrency ↑, Fairness ↑
L5: Application Layer	Coral [27], Broker2Earn [28], LedgerHedger [29]	Adaptive Transaction Packing Algorithm (ADPA); Broker liquidity staking; Gas hedging contracts.	QoS ↑, Stability ↑

Table 3 evaluates these systems based on the tasks of AI-agents. It includes their assumptions, the scalability enhancements obtained, the guarantees of trust, the extra difficulties in deployment, and the unsatisfied oriented requirements of agents. The comparison indicates that the five levels deal with various aspects of the agent-trust issue. The data level keeps the evidence, the network level facilitates the combination of tasks in different domains, the consensus level offers ordering and finality, the execution level enhances concurrency, and the application level increases economic predictability. The guarantees are mainly defined for states, messages, transactions or payoff rules. They have not yet covered an agent's history, intention, delegated restrictions or multistep behaviour as a single auditable object. The main contributions of this paper are summarized as follows:

- We consider the problem of blockchain scalability as a trust issue in AI-based autonomous systems, wherein throughput, finality, auditability and accountability are involved.
- We divide the blockchain transaction processing and scalability study into five system aspects: state maintenance and credibility, partitioning, scalable consensus and Layer-2, concurrent execution, and application-level mechanisms.
- We examine typical systems and connect their assumptions and guarantees with the requirements at the agent level, such as persistent accountability, interdisciplinary cooperation, prompt completion, reproducible execution and economic stability.
- We detect the unresolved issues that occur when blockchain technologies are applied to adaptive agents instead of ordinary users or predetermined smart contracts.

The remaining part of the paper is arranged as follows. Section 2 investigates state management and verifiable security techniques. Section 3 surveys sharding-based scalability methods. Section 4 considers scalable consensus and Layer-2 schemes. Section 5 evaluates parallel execution architectures. Section 6 analyses application-level trading and incentive systems. Section 7 summarizes the existing problems and potential future research directions.

Table 3. Structured comparative assessment of blockchain scalability techniques for AI-agent environments.

Layer	Main Assumptions	Scalability Property	Security/Trust Guarantee	Deployment Burden	Suitability for AI-agent Environments
Data	State records, logs, or proofs are available and verifiable when needed.	Reduces storage and verification overhead.	Supports auditable state access and verifiable state transitions.	Requires additional state organization, proof management, or audit support.	Supports agent decision-making by providing verifiable state access and auditable on-chain records; however, current data-layer designs are not sufficient to represent, store, and retrieve long-term agent behavior, task context, and multi-step interaction histories.
Network	Cross-domain messages and coordination metadata can be delivered reliably and verified efficiently.	Improves cross-shard throughput and latency.	Enhances cross-domain consistency and communication reliability.	Requires additional communication, routing, or coordination support.	Enables agents to compose tasks across shards, contracts, and services with lower communication delay; however, it does not by itself explain which party is responsible when a multi-step workflow fails.
Consensus	A sufficient fraction of validators follow the protocol, and the network satisfies the required synchrony or availability conditions.	Improves finality and reduces validation load.	Provides ordering, finality, and settlement guarantees.	Requires additional ordering, validation, or proof infrastructure.	Provides reliable ordering and settlement for machine-speed agent commitments; however, it only proves protocol-level agreement, not whether the committed action is semantically correct.
Execution	Transactions have deterministic execution semantics, and their conflicts or dependencies can be correctly handled.	Increases concurrency and reduces execution bottlenecks.	Ensures conflict safety, deterministic replay, and execution fairness.	Requires additional scheduling, dependency handling, or replay support.	Supports high-frequency concurrent actions generated by many agents; however, it cannot check whether tool use, delegated constraints, or task-specific policies are respected.
Application	Participants are responsive to protocol-level incentives, service rules, and economic commitments [27–29].	Improves QoS, liquidity utilization, and fee predictability.	Provides incentive, service-level, or cost-related guarantees.	Requires additional incentive design, service coordination, or settlement support.	Helps agents plan and execute economically sensitive tasks under clearer cost, latency, and liquidity conditions; however, it still lacks mechanisms to ensure that adaptive agents remain aligned with user intent under changing market incentives.

2. State management and verifiable security

With the increase in the amount of information such as accounts, contract storage, logs and execution metadata, the storage, search, updating and later confirmation become more costly. For those systems which handle a large number of transactions, the data layer is not an inactive storage; it influences the transaction rate and the auditing expense.

It is important to keep state in autonomous AI systems to keep records of the effects of an agent's actions. The agent may perform transactions, deal with assets, carry out contracts and interact with other agents. Every action should have a state effect which is connected to the time when the decision was taken. Otherwise, it is hard to determine the responsibility and to settle the conflicts.

The problem is not only in the reduction of storage. In an agent system, a suitable data layer should enable the recovery of an agent's trajectory after delegation, contract calls and cross-domain execution. If it is difficult to obtain or confirm the previous states, it is hard to assign responsibility although the transactions are valid. The pruning, verifiable transitions and governing schemes should be assessed according to their ability to maintain the evidence for attribution, dispute settlement and policy examination.

Recent study concerns these problems by enhancing storage administration, guaranteeing verifiable execution and applying application-level control. Now we will explain these methods below.

2.1. State storage optimization

Storage is an important expense in efficient blockchain systems. Usually, these systems employ engines based on Log-Structured Merge-tree (LSM-tree), which are appropriate for handling write-intensive operations; however, I/O amplification happens when both the historical data and frequently used states are stored simultaneously. With the increase of the chain, continuous compaction and data transfer influence the storage access and transaction execution.

From the perspective of the trust infrastructure, this inefficiency also impedes the search and verification of past state modifications, which is necessary for examining the actions of self-controlled agents.

Liang *et al*. [14] recommended MoltDB for handling block chain states. MoltDB classifies old historical states from the current states by storing the ancient states in a separate layer and moving the inactive data to an archive using an extract–compact procedure. Therefore, by separating the cold data and reducing the unneeded compaction, MoltDB decreases the disk I/O load and improves the transaction speed [14].

Storage optimization should be carried out simultaneously with the consideration of accountability. Changing or deleting the old data may cause the audits of the future to rely on the archival nodes, data availability services or the proof generators. For self-controlled agents, the main problem is to make the evidence of their behaviors less expensive to obtain and more reliable, but at the same time, link it to the situation in which the action was performed.

2.2. State transition verifiability

With the increase of off-chain execution in scaling systems, the verification of state transitions has become important. For instance, optimistic rollups depend on fraud proofs to doubt the incorrect off-chain execution. These methods enhance scalability, but they might be based on certain client implementations and increase the trusted computing base.

For AI agents, the verifiable state changes are the basis of their accountability. The external parties should check that the state effects assigned to an agent are due to the stated execution path, rather than relying on a trusted intermediary.

Ye *et al*. [15] proposed Specular to reduce such trust dependencies. Specular provides an Ethereum Virtual Machine (EVM)-compatible one-step proof system on the main chain, enabling interactive dispute resolution for off-chain execution. When a dispute occurs, a participant submits a minimal proof for one computation step, allowing the base chain to verify that step and reducing dependence on a specific rollup client [15].

Lightweight clients have the same difficulty too. In the lazy blockchain model, blocks only store the transaction order instead of the real execution results. Tas *et al*. [30] suggested an interactive bisection approach which enables light clients to verify the execution without possessing the whole blocks. Through reducing the disputed portion step by step, the protocol attains a logarithmic verification complexity and thus makes secure validation more feasible for devices with limited resources [30].

Trusted Execution Environments (TEEs) introduce a new verification problem. Although TEEs possess hardware security, they can be defeated by rollback methods. Peng *et al*. [16] have designed a distributed auditing system named Narrator-Pro, which keeps the TEE operation records on the Internet. This audit trail helps to identify rollback operations and guarantees a decentralized origin of trust in secure

computing systems [16].

Existing verifiability mechanisms still rely on assumptions that may not hold in open agent environments. To prove fraud, there should be an honest person who discovers and opposes an error; the proof systems consist of the models which they have specified; and the TEE includes both hardware and authentication trust. Although these assumptions are satisfied, the guarantee is still almost at the machine level. It can confirm that a transaction has been carried out according to a virtual machine or protocol, but it cannot independently show that an agent has completed the entrusted task, has noticed the user's limitations or has prevented any interference.

2.3. State governance and application-layer verifiability

Furthermore, blockchains need to have methods to deal with states which might be altered, concealed or modified according to the policy regulations. In such a situation, compliance, error recovery and privacy may need some adjustments. The problem is to allow these changes without considering auditability as a less important issue.

For autonomous systems, these governance decisions influence the correction, review and regulation of agent actions after implementation. They determine the practical trustworthiness rather than the storage efficiency.

Wang *et al*. [31] suggested a stable and removable blockchain structure which enables restricted state changes while maintaining system integrity. Their two-level modification method and version identification strategy permit authorized modifications even if the trapdoor assumptions do not hold [31]. Zhao *et al*. [32] presented a credit system based scheme for controlling block chain modifications. Credits are allocated to those who perform state updates according to their reputation, forming a governance method that coordinates between the flexibility of editing and the responsibility of accountability [32].

Verifiable state management also supports decentralized services. Heiss *et al*. [33] constructed a carbon accounting system on a blockchain which uses zero-knowledge proofs for the verification of emission records at the same time protecting the confidential information. Zhang *et al*. [34] proposed a decentralized sealed-bid auction mechanism with delayed execution [34]. Chen *et al*. [35] also studied a hiding technique which embeds hidden data in the transaction fields, enabling its secret storage and transmission in the blockchain systems [35].

Government management should balance immutability, correction, privacy and accountability. Generally, redactable or updatable records suppose that the modification authorities, voting rules or policy triggers can be determined before implementation. However, it is more difficult in agent ecosystems as conflicts may occur in several regions, with ambiguous instructions and unverifiable off-line evidence. A good governance method must enable incorrect actions to be revised without making the past easily changeable strategically.

3. Sharding mechanism optimization

Sharding improves scalability by dividing the network into groups that process transactions and maintain local states in parallel. It increases throughput by letting different shards make progress at the same time, but it also creates coordination problems whenever a transaction depends on state outside its local shard.

For autonomous AI systems, sharding can also be employed as a coordination technique. It separates the workloads produced by several agents, although these agents might have interactions in different fields and submit state-dependent transaction sequences. An appropriate scheme should be integrated with both local execution ability and reliable coordination among shards.

Many trust failures happen at the shard boundaries. The agents can execute composite plans with some of their intermediate steps in different shards, contracts or external services. A protocol may increase the local throughput, but it is not appropriate for agent coordination if the cross-shard dependencies are slow, ambiguous or hard to trace. Efficient cross-shard communication needs fast message transmission, visible states and dependable error handling.

In fact, sharded blockchains have encountered various difficulties: cross-shard transactions need more communication and coordination; the fixed division of shards may result in an uneven distribution of workload; and the strict structures cannot adapt quickly to changes in workload or adverse conditions.

The following comment emphasizes cross-database communication, adaptive sharding and modifications in sharded structures.

3.1. Cross-shard communication optimization

Cross-shard transactions are generally the main obstacle in sharded systems as several validator groups need to reach an agreement on a common result. Multi-phase protocols guarantee safety, but they also increase the delay. In multi-agent environments, a similar difficulty exists, namely coordination among different execution domains: the system should not only make sure that the cooperative actions of agents are rapid, but also ensure their consistency and traceability [36].

To reduce coordination latency, Zhou *et al.* [17] proposed DCchain, which integrates a decentralized coordinator service with verifiable global states. Fine-grained preprocessing schedules transactions before execution, and compressed commitments reduce cross-shard coordination to a single round [17]. Qi *et al.* [18] introduced LightCross, a lightweight cross-shard execution framework that uses distributed collectors and a unified commit point to aggregate execution results, streamlining cross-shard contract calls [18].

Liang *et al.* [37] suggested SPARROW to solve the problem of strict inter-shard state isolation. They employed inter-shard caching, speculative execution and multi-branch exploration to improve the speed of cross-shard contract execution while ensuring correctness in Byzantine conditions [37]. Huang *et al.* [38] studied state consistency during shard transitions and introduced fine-grained locks for account migration to prevent double spending and inconsistent states [38].

There is still instability in the communication between different components of a system when agents participate in a complicated procedure. The current techniques can either reduce the number of communication steps or merge the commitments, but they often assume that the dependencies are known prior to execution, all participants in each part are available and faults can be easily detected. Agents violate these assumptions when they take part in various contracts, networks and services at a high rate. It is not enough to decrease the average latency if the protocol does not guarantee atomicity, offer a rollback mechanism or verify the responsibility for unfinished procedures.

3.2. Dynamic sharding mechanisms

Dynamic sharding rearranges the membership of shards, node allocation or account distribution according to the variation of workload. The aim is to decrease the imbalance and avoid excessive cross-shard communication without limiting the system to a certain partitioning structure.

From an AI-agent viewpoint, dynamic sharding is a method of resource management: computation and storage are adjusted according to the interactions created by the agents.

Wang *et al.* [19] suggested Orbit, a method to transfer accounts between different partitions based on the workload and dependency analysis. Certificate-based schemes and short-term locking were employed to ensure the safety of the transfers [19]. Wang *et al.* [39] developed Mitosis, where a single relay chain was substituted by several relay chains working dynamically coherently to enhance the inter-partition scalability [39]. Ren *et al.* [40] put forward AdaptiveShard, integrating adaptive coding with dynamic node administration to improve the reliability and exclude malicious nodes [40]. Jiang *et al.* [20] proposed Sharon, which adopted shard rotation and light-weight commit protocols to decrease the possibility of collusion and the cost of coordination [20].

Dynamic sharding improves flexibility, however it depends on the reconfiguration decisions. Moving accounts or validators may reduce the cross-shard calls in the future, but it also increases the synchronization costs, lock-overhead and the temporary unavailability. Some approaches predict the future requirements based on the recent dependencies; adaptive agents may influence this prediction by changing their strategies or adjusting the bursts. In the open agent market, the reconfiguration process should be explicit, avoid fraud and be reliable.

3.3. Sharding structure innovation

Besides improving communication and adjusting the placement, some studies modify the sharded structure itself. These modifications rearrange the organization of consensus, storage, account placement or evaluation infrastructure with a view to utilizing resources in a more effective manner.

Yang *et al.* [41] proposed EZchain, separating the consensus and storage into active and passive layers to reduce the consensus overhead and maintain the distributed data availability [41]. Tian *et al.* [42] suggested PartChain, which partitions the accounts based on the transaction relationships by using multi-objective optimization, balancing the workloads and reducing the cross-shard interactions [42]. Che *et al.* [43] developed Manifoldchain, a bandwidth-constrained sharding method that puts high-frequency accounts into high-capacity shards [43]. Huang *et al.* [44] created BlockEmulator, a whole system for evaluating sharding schemes under different workloads [44].

Division of structure can maintain the association of resources only under a stable workload pattern. The agent's workloads may change quickly and the strategic plans may affect the relationships. In addition, the off-line information may determine the necessity of further division. It is hard to apply these structures because the applications must take into account various execution regions, inactive storage layers or different capacities of heterogeneous divisions. These systems are advantageous for expansion, but they do not provide enough basis of confidence for the agents themselves.

Sharding supplies parallel processing capacity for large-scale agent activity, but its guarantees still depend on consensus and settlement layers that make distributed execution final and auditable. Section 4

examines the consensus and Layer-2 mechanisms that support these workloads.

4. Scalable consensus and Layer-2/multi-tier structures

AI-driven agents place new pressure on consensus and settlement. Consensus does more than order user transactions; it creates the record that later agent actions and disputes rely on. If ordering is delayed, unstable, or hard to audit, an agent may act on a commitment that is no longer valid when the full workflow finishes.

Scalable consensus and Layer-2 mechanisms are particularly necessary when the decisions of agents are time-sensitive and state-dependent. If the finality is delayed, an action may go out of its effective decision period, and weak settlement guarantees make it difficult to challenge the executed delegated actions. The problem is not only the number of transactions being finalized per second, but whether the protocol provides timely, verifiable and responsible ordering for actions which may involve different fields and have economic or security impacts.

Recent studies deal with this issue by using scalable consensus protocols and multi-level structures which distinguish between ordering, execution, storage and verification. We then describe structured DAG-based consensus, parallel and stateless execution systems and simple cross-chain and Layer-2 verification methods.

4.1. High-performance consensus via structured DAGs

DAG-based consensus protocols improve concurrency by allowing blocks to be generated at the same time. The blocks are depicted as vertices in a DAG, therefore the protocol can maintain a partial order before deciding the final order. This structure might increase the throughput in contrast to a sequential chain and also show the causal relations between the transactions.

The main problem is to transform a gradually increasing partial order into a definite global order. Linearizing the DAG may increase the computing and communicating burden, particularly when the graph is large or when the network delay results in numerous competing branches.

Hu *et al*. [21] presented Ladder, a structured DAG consensus algorithm with an independent layer for block ordering. By distinguishing between block generation and ordering, Ladder decreases the latency without losing the throughput. Cheng *et al*. [45] introduced SharDAG, which integrates DAG consensus with sharding and employs state pre-positioning to decrease the inter-shard communication [45].

DAG consensus has the advantage of quick block production, however, agent systems need stable finality and clear ordering as well. When there is network delay or malicious scheduling, an agent may answer to an intermediate commitment which is then rearranged or questioned. Thus, the efficiency of DAG protocols should be judged by the finality latency, the expense of verifying causal relations and the simplicity of order explanation, not solely by the overall throughput.

4.2. Parallel and stateless execution

Beyond agreement, the execution and state access may encounter problems when a large number of transactions are processed simultaneously. Present systems usually duplicate the entire state and carry out transactions one by one, thus increasing the computational and storage burden.

Recently, some studies have dealt with this issue by using parallel and stateless architectures. Chen *et al*. [22] introduced Porygon, a storage–consensus separation architecture which enables both inter-block and intra-block parallelism via pipelined execution. Porygon also employs stateless verification so that the nodes can check the transactions without keeping the whole global state. Such a structure enhances the resource efficiency and reduces the workload for the heterogeneous participants, such as the resource-limited nodes [22].

Parallel and stateless execution can avoid repeated computations and storage, but they need to assume the existence of proofs, witness generation and dependency management. It is hard to carry out stateless verification when the witnesses are large, alter frequently or are provided by those with poor incentives. Moreover, parallel processing at the consensus level cannot resolve semantic conflicts. Two transactions may have no common storage relationship but may conflict in terms of task, budget or policy. The parallelism at the execution level is not sufficient for trustworthy agent coordination.

4.3. Lightweight verifiable and cross-chain technologies

Interoperability is a necessary condition for multi-chain ecosystems. Cross-chain protocols transmit assets and information between different systems, but usually require additional communication, verification and trust management costs.

Recently, some research has dealt with this problem by using proof-based methods. Far *et al*. [23] presented a Layer-2 cross-chain structure based on short cryptographic proofs, transferring most of the computation to the off-chain side and decreasing the on-chain verification cost [23]. Yin *et al*. [46] developed a cross-chain protocol which employs threshold signatures to generate verification certificates of fixed size, thus reducing the communication and storage overhead [46].

Lightweight cross-chain and Layer-2 techniques decrease the verification expense, however they rely on the trust in the committees, data availability, proof generation or bridge security instead of direct verification. Such trusts are essential when an agent transfers a task between multiple chains with various finality regulations and trust models. The main problem is to decide the finality: the agent must be able to recognize when an operation in different fields becomes unalterable, find out the responsible person in case of an error, and be informed about the ways to solve conflicts in incompatible systems.

Scalable consensus is the foundation for maintaining consistent system performance. When consensus is enhanced, transaction processing will become the main restriction. In Section 5, the parallel execution frameworks for efficient transaction processing are introduced.

5. Transaction execution and parallel processing

The execution stage is a main problem affecting the scalability of the blockchain. Through the use of consensus algorithms and sharding or DAG structures, the transaction sequence can be enhanced, yet in general, the smart contracts are executed sequentially to ensure definite state changes. Sequential execution is reliable and beneficial for audit, but it cannot fully utilize concurrency when there are many independent transactions. This distinction is more noticeable in the artificial intelligence agent system, where several autonomous units may perform operations which can be processed simultaneously.

Moreover, execution involves a responsibility. In the agent system, the execution stage should yield results which are certain, verifiable and repeatable. The difficulty lies in achieving concurrency without sacrificing the capability to clarify how a certain state was attained.

Agent trust is supported in parallel execution only when the concurrency decisions can be repeated and clarified. Independent actions at the storage-level may be connected through a common task, deadline, budget or delegated policy. Therefore, the scalability of the execution layer must maintain deterministic replay, conflict assignment and fairness under high concurrency; otherwise, a high throughput may result in unaccountable states.

Recent studies solve these problems by means of speculative execution, dependency analysis and conflict detection. The economic significance of ordering has resulted in fairness mechanisms to prevent front-running and Maximal Extractable Value (MEV), whereas correctness and confidentiality are still challenging in parallel environments. We describe these three directions below.

5.1. Parallel execution frameworks

Many parallel execution frameworks attempt to eliminate serial processing through the use of speculation and dependency-based scheduling. They execute those transactions which do not have conflicting read-write sets concurrently, and then apply either replay or validation to maintain definite results.

Xu *et al*. [24] presented a parallel execution framework called Vegeta for leaderless consensus systems. Vegeta adopts a "speculate–order–replay" strategy: transactions are firstly executed speculatively to show the read–write dependencies, then arranged into partially ordered sequences, and subsequently replayed after consensus determines the final order. Through separating the discovery of dependencies from the final execution, Vegeta enhances the execution speed [24].

Yu *et al*. [47] proposed ParFabric, which uses conflict graphs and dynamic reordering to support parallel execution in Hyperledger Fabric. Partial-order block construction and concurrent validation improve throughput and latency in permissioned settings [47]. Zhao *et al*. [25] proposed the Partial-Order Block Ledger, where nodes agree on a consensus-level dependency graph to reduce conflicts and serialization overhead [25].

Partial-order and speculative execution are still restricted when they consider only read-write dependencies. Two agent actions may affect different storage keys but still conflict due to a common goal, budget, deadline or delegated policy. Speculation may also consume resources if adversaries produce a large number of conflicts or misleading ones, and it becomes expensive to replay deterministically as the dependency graphs increase. Another difficulty is to associate the storage-level conflict detection with the intent, policy constraints and recoverable rollback at the agent level.

5.2. Transaction ordering and fairness governance

Transaction sequence is also a governing problem, especially in decentralized finance. Validators may rearrange, insert or delay transactions, which give rise to front-running and MEV and consequently influence the fairness and incentives.

Recent work studies fair and verifiable ordering. Gramoli *et al*. [26] proposed the AOAB protocol, which gives globally checkable timestamps to transactions in order to ensure that the ordering is not

influenced by the validators' preferences. According to the timestamps, AOAB provides more fair guarantees for distributed systems [26]. From an economic-security viewpoint, Wadhwa *et al*. [48] presented rational binding transactions which coincide with the ordering policies. Ciampi *et al*. [49] hen formally discussed the ordering fairness in the Universal Composability framework and suggested fairness-preserving protocols using TEEs.

Fair ordering preserves the transactional neutrality, but the arrival-time or timestamp rules cannot express all the advantages that the agents can utilize. They might use latency games, private order flows, cross-domain sequencing or multi-step transaction methods which are not included in one rule. Moreover, the TEE-based protocols rely on certain hardware trust conditions. Therefore, the ordering fairness is still not sufficient for the agent economy until it considers cross-domain and multi-step strategies instead of single transactions.

5.3. Correctness guarantees and data security

Parallel execution brings about new errors in correctness, debugging and privacy. Concurrency faults may occur only under certain schedules and inconsistent state changes are hard to reproduce afterwards.

One line of work focuses on correctness testing. Zhou *et al*. [50] presented Chord, a system which describes conflict transactions and produces various concurrent workloads. By using a combination of dynamic testing methods and differential oracles, Chord identifies discrepancies between different implementations and has found some previously hidden errors in practical systems [50]. In addition, another research direction aims to protect data during its processing. Wang *et al*. [51] introduced the Transaction Commit-controlled Release (CORE) protocol, in which confidential information remains encrypted until the completion of a transaction. This "commit-controlled release" strategy prevents the leakage of information before the transaction is finalized, and at the same time ensures the efficiency of the execution [51].

Testing and privacy-preserving release enhance the reliability, but they cannot ensure the correctness in open high-concurrency environments. Testing can detect the implementation errors, however, it is unable to deal with all the adversarial schedules or application-level invariants. The late disclosure of data preserves the confidentiality, but it also hinders the observability and makes real-time inspection more difficult. For autonomous agents, correctness requires reproduction, confidentiality, and semantic conformity, which are hard to achieve simultaneously.

Parallel execution improves throughput, but its broader importance lies in making high-volume computation reviewable. That benefit holds only if schedules remain reproducible and auditable. As blockchain systems support more autonomous activity, the execution layer must provide correctness, fairness, and auditability together with performance. Section 6 examines application-layer mechanisms that complement these infrastructure-level advances.

6. Application-layer trading mechanisms and incentives

As blockchain systems apply to autonomous-agent applications, the application layer becomes important for maintaining economic reliability. Single increase in throughput is not enough if agents encounter varying fees, low liquidity, adjustable schedules or uncertain service agreements.

For agent ecosystems, the application-level scalability is important to maintain the economic usability of the infrastructure. A rapid ledger cannot ensure the stable operation of agents when the execution costs, resource supplies or scheduling priorities vary greatly during a task. The transaction scheduling, liquidity provision and gas-management mechanisms should be regarded as trust elements since they influence the predictable, fair and economic execution of an autonomous decision.

This part discusses the examination of transaction scheduling and resource allocation, the design of liquidity incentives and gas-fee control. These strategies link the basic execution capability with the cost, time and service assurance required by the agents under varying market circumstances.

6.1. Transaction scheduling and resource optimization

Decentralized auctions, supply-chain cooperation and immediate financial transactions require a better timing assurance than that given by common block confirmations. The gas-price order or block number rules may not be suitable when the network is busy or when demands vary rapidly.

Liu *et al*. [27] solved this problem by using Coral, a deadline-sensitive scheduling algorithm which incorporates explicit temporal restrictions in transaction processing. Coral applies a global time-consistency verification method and an Adaptive Deadline-based Transaction Packing Algorithm to achieve a balance between the urgency and the resource consumption. Through allocating more attention to deadline-dependent transactions without neglecting the system efficiency, Coral enhances the effective throughput in situations with time constraints [27].

Intelligent workload regulation has also solved this problem by using Coral, a deadline-sensitive scheduling algorithm that incorporates explicit temporal restrictions in transaction processing. Coral applies a global time-consistency verification method and an Adaptive Deadline-based Transaction Packing Algorithm to achieve a balance between urgency and resource consumption. Through allocating more attention to deadline-dependent transactions without neglecting the system efficiency, Coral enhances the effective throughput in situations with time constraints explored. Kim *et al.* [52] proposed an intelligent Transaction Generation Control (i-TGC) framework using reinforcement learning to adjust the transaction generation rates according to the network status. According to the congestion situations and the validator's ability, i-TGC can reduce the fluctuating workloads and improve the system stability [52].

Deadline-aware and learning-based scheduling enhance flexibility, but strategic agents might utilize them in an unexpected manner. They may alter the deadlines, divide the transactions, or modify the workload indications. The reinforcement-learning controllers also depend on the previous congestion conditions, which could be changed when there occur joint surges or artificial fluctuations in the demand. It is significant to investigate the scheduling for resistance to manipulation, preservation of service quality, and for throughput and latency.

6.2. Liquidity incentives mechanisms

Advanced blockchain structures, such as sharded systems, need the cooperation of participants in offering liquidity, routing capacity or other resources. Otherwise, rational individuals may not carry out these tasks and the quality of cross-domain services may still be affected even if the underlying protocol is technically right.

To solve this problem, Chen *et al*. [28] put forward the Broker2Earn (B2E) protocol, which establishes a market-oriented approach for cross-shard transactions. In B2E, broker nodes offer liquidity for intermediate cross-shard transfers, converting them into intra-shard operations and decreasing the coordination cost. The protocol also employs approximation algorithms to enhance the system throughput and broker's revenue, thus improving the service speed and aligning the incentives [28].

Beyond financial incentives, collaborative blockchain applications require flexible governance mechanisms for dynamic workflows. Brahem *et al*. [53] proposed a change management framework based on Dynamic Condition Response (DCR) graphs, where organizations update shared processes through hybrid on-chain/off-chain voting. This design improves adaptability while preserving coordination consistency across multiple parties [53].

Liquidity and governance mechanisms rely on certain stability in rational behavior, capital supply and cooperation. The presence of agent communities leads to a reduction in this stability. Adaptive agents may withdraw liquidity, utilize reward limitations or affect the governance participation. Besides, the broker-based systems experience a problem during the initial period because the quality of cross-blockchain services cannot be enhanced until a considerable amount of liquidity is provided. The main difficulty in achieving scalability in the agent environment is the incentive scheme.

6.3. Gas fee management strategies

Fluctuation of transaction fees makes it hard to plan for the execution of blockchain, particularly for Layer-2 protocols and applications with time sensitivity. A proof submission, liquidation, auction bidding or cross-chain settlement might lose its value if the necessary fee changes significantly before being included.

To solve this problem, Tsabary *et al*. [29] put forward LedgerHedger, a decentralized method for gas reservation that enables users to fix future transaction fees by bilateral agreements with validators. The scheme applies a subgame-perfect equilibrium model to coordinate the interests of validators and users and to prevent strategic manipulation. By transforming the fluctuating gas payments into forward commitments, LedgerHedger establishes a hedging market for the execution costs and enhances the stability of operations like the submission of rollup proofs [29].

Gas hedging and reservation can reduce the price uncertainty, but they cannot remove the shortage of space or the deliberate refusal. The effectiveness of these strategies depends on the liquid reservation market, reliable verifiers and correct demand forecasts which are not easily deceptive. Moreover, some independent agents may reserve a part of their capacity for defensive or speculative purposes, which may lead to new congestion and market effects. The constant fees only reflect a part of economic stability; the fair usage and distribution without any regulation have not been decided yet8lfg.

7. Open issues and future directions

Although the progress mentioned before, the present methods for transaction processing and scalability are still inadequate to establish a complete trust level for autonomous agents. The difficulty does not result from the absence of effective strategies. The majority of the guarantees are designed for transactions, validators, contracts or resource markets, but agents involve more complex histories, delegated choices, semantic goals and changing behaviors. These differences lead to the open issues described below.

The open issues discussed below are derived through a three-step analysis. First, prior studies on AI agents reveal challenges related to accountability, reliable tool use, semantic correctness, multi-agent coordination, and economic or strategic instability [8,54–56]. Second, the blockchain transaction-processing literature reviewed in Sections 2–6 provides mechanisms such as verifiable state transitions, cross-domain communication, finality, parallel execution, and incentive-aware resource allocation. Third, we map these agent-level challenges to blockchain architectural layers and identify where existing techniques need to be extended. Thus, the following directions should not be viewed as mature consensus conclusions or purely speculative claims, but rather as an author-synthesized research agenda grounded in the surveyed literature.

7.1. Tracking who the agent is and what it has done

Blockchains store the verified transaction records, but these records cannot fully describe the past of the responsible entities. In general situations, a public address and a certain state change are enough for examination. Nevertheless, intelligent agents complicate this situation. They can alter rules, employ external devices, get off-chain information, delegate subtasks, and deal with various aspects before a final transaction is saved on-chain. The transaction shows the final state, but it may not explain the chosen action or the observations that led to it.

Accountability should not be solely determined by assigning the action. A wrong action may reveal malice, a modeling mistake, incorrect data, a system fault, out-of-date knowledge or influence from other agents. The transaction hash is not adequate to differentiate these causes. It is more reasonable to set up a file that links the agent's name to the decision environment, the execution procedure, the entrusted sub-tasks, and the progressive modifications of the observable states over time.

As shown in Figure 2, one possible way is to integrate decentralized identifiers, verifiable credentials, cryptographic attestations and on-chain state records. This framework can enable an auditor to not only know which address has sent a transaction, but also which agent has made the decision, what context has been used, which tool chain has been followed and where the responsibility boundary should be set.

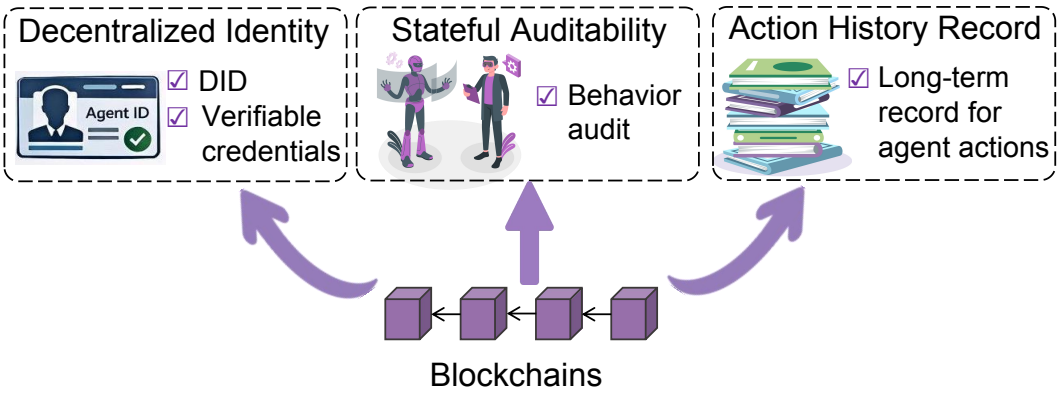

Figure 2. Blockchain-enabled accountability for autonomous agents. The framework links DIDs, attestations, stateful audit records, and long-term action histories so that auditors can connect an on-chain effect to an agent, a decision context, and a responsibility boundary.

Some design points are yet to be settled. The agent's identity must be able to withstand the changes caused by key rotation, model updates, change of ownership and delegation. The behavioral records should be comprehensive enough for inspection later, without including all the prompts, memories or traces on the network. Moreover, the accountability rules should distinguish between direct execution, delegated execution and the emergent multi-agent behaviour, because one result may be influenced by several agents with different degrees of control. These problems need technical means and governing regulations, rather than relying on the identity primitives alone.

7.2. Making cross-chain agent collaboration reliable

Blockchain ecosystems are already composed of multiple chains and the agent workflows will show greater differences. The current sharding and cross-chain mechanisms mainly deal with the transfer of assets, the transmission of messages or the execution of individual transactions. Though these functions are very important, they can not fully represent the collaboration among agents who deal with one domain, perform actions in another and depend on off-chain services. As shown in Figure 3, the problem is not only whether a message is successfully delivered. Now, the difficulty lies in whether the distributed tasks keep their correctness when they are executed according to various finality rules and trust foundations.

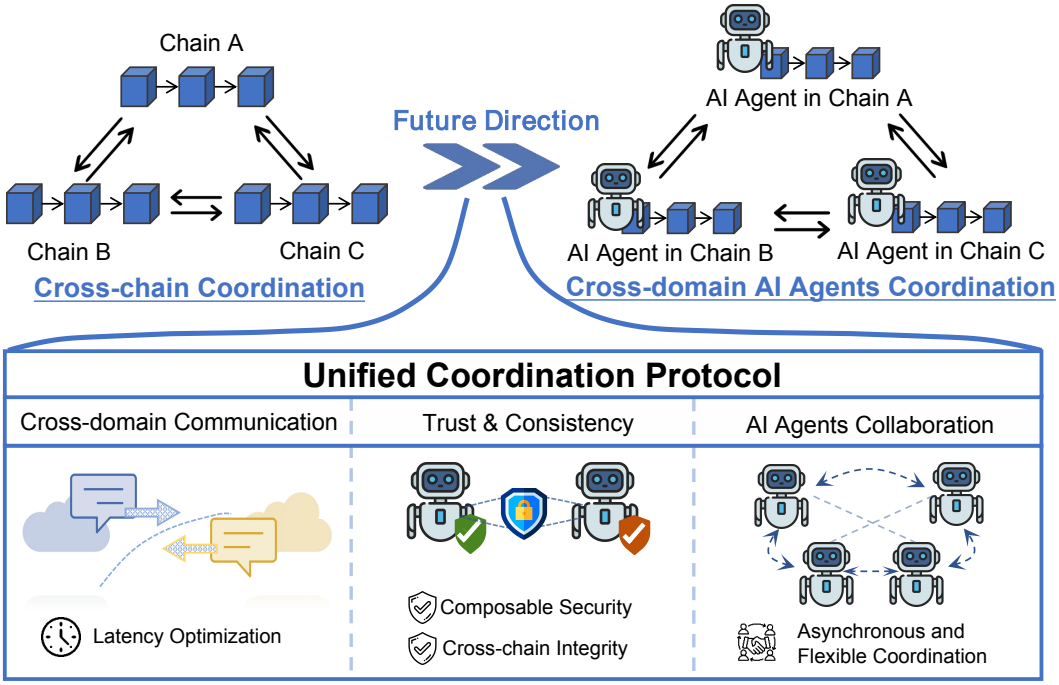

Figure 3. A possible architecture for cross-domain agent coordination. The coordination layer links heterogeneous chains and off-chain services through latency-aware communication, composable trust assumptions, and explicit failure-handling rules for asynchronous agent workflows.

Cross-chain agent workflows are not the same as simple atomic transactions. A job might include an observation on one chain, a contract operation on another chain and a response from an external source. The correctness of the workflow depends on whether these operations are performed in the same task instance and fulfill the requirements of sequence, time and finality. If any step results in a delay of a

message, alters the order of an event or impacts its finality, the agent may finish a locally valid transaction, but it would be wrong for the entire task.

An important unsolved question is to define the correctness at the workflow level. This definition should cover the atomicity between the dependent steps, the causal consistency of the messages, the execution windows aware of finality, the replay protection and the bounded failures when one chain cancels, delays or suppresses an action. Otherwise, it is difficult to differentiate an agent's error from an outdated cross-chain state or a bridge's failure

Explicitly stating the assumptions of the future coordination protocols is necessary instead of concealing them in a general bridge interface. The finality rules, validator models, data-availability guarantees, execution semantics and dispute periods may vary in different domains. The correctness cannot be derived from one consensus layer only. It must be analyzed compositionally in all the domains involved, with definite explanations of what each domain proves to the others.

7.3. Verifying what agents actually do

Blockchain verification is trustworthy when the transaction satisfies the execution environment's rules. However, there exists another type of difficulty. An agent's action may be syntactically proper but unsuitable for the user's needs. It might employ an improper tool, consult obsolete information, ignore a delegate restriction, or produce a valid result that is not satisfactory enough. Figure 4 indicates the distinction between transactional correctness and semantic validity according to this difference.

Present techniques are not adequate to check these properties. A smart contract can confirm whether an input satisfies a particular condition, but it usually cannot judge the correct understanding of a natural language instruction, the appropriateness of an external device, or if the concealed needs of the users have been taken into consideration. Most agent jobs do not have complete specifications either. Even recording the final action on the chain cannot guarantee that the decision-making process is semantically traceable.

The difficulty is mainly due to the reproducibility. The outputs of the agent may vary according to the sampling methods, the retrieved context, the states of the tools, and the external observations. Some conditions are also undefined and cannot be expressed clearly in the form of smart contract predicates or cryptographic assertions. The evidence may involve private prompts, confidential model parameters or off-line data, which cause a conflict between the auditability, the confidentiality and the data availability.

A practical verification stack should probably contain both cryptographic evidence and a level of evaluation. Some possible components are verifiable inference, proof-carrying data for model outputs and a hybrid validation method, in which independent judges examine off-line evidence according to on-line dispute rules. The main problem is that the semantic correctness may not have a unique true value. Two agents might carry out the same instruction by using different tools or ways. The specification language of the task may require the acceptable result sets instead of a definite output, and the dispute system should determine if the observed behaviour belongs to these sets. These systems should also prevent collusion among the evaluators, introduction of prompts, adversarial inputs and the agents learning to satisfy the verifier rather than accomplishing the task.

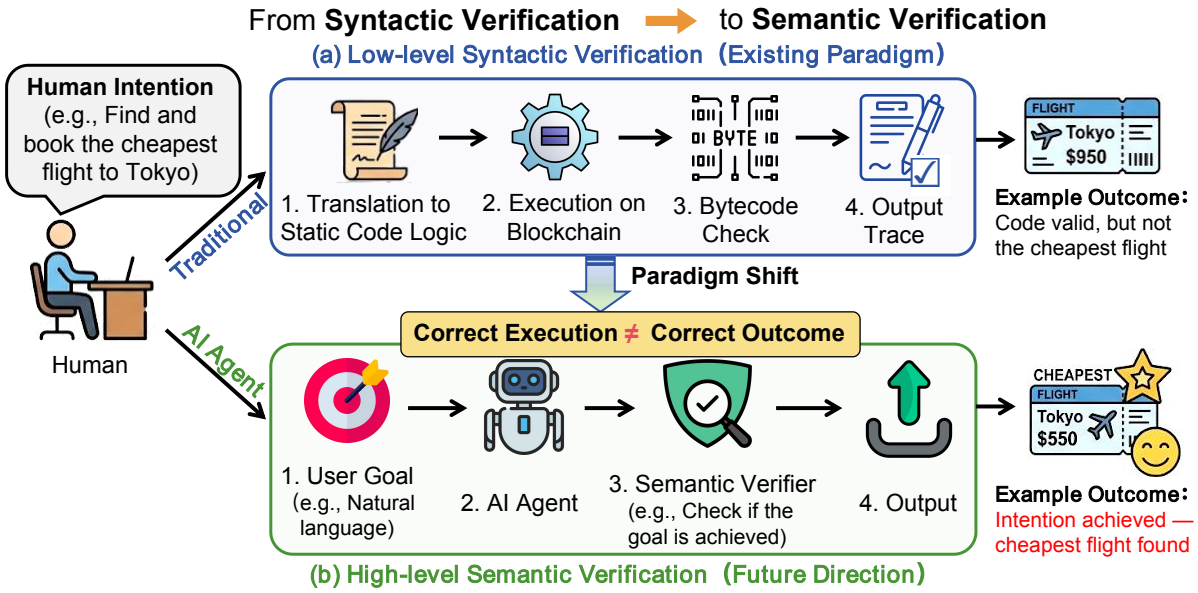

Figure 4. From syntactic verification to semantic verification. **(a)** Existing blockchain verification mainly checks deterministic execution traces such as EVM/WebAssembly (WASM) steps; **(b)** Agent-oriented verification must also judge whether a probabilistic action is consistent with user intent, task constraints, and acceptable outcomes.

7.4. Keeping agent-based blockchain markets stable

Economic design mainly focuses on the security of blockchain, but the combination of AI agents makes it more flexible [57,58]. The fee markets, liquidity rewards and sequencing policies are mainly considered for human users, automatic robots with fixed strategies or rational participants whose behaviors do not change greatly. However, this assumption is not suitable for an economic system including artificial intelligence. From Figure 5, the agents can interpret the protocol signals, modify their strategies promptly, distribute actions to various identities and achieve cooperation at a lesser expense [59]. The rules for general users may provide opportunities for boundary games, artificial traffic jams, time-delayed arbitrage or cooperation in reward collection.

The open problem is not only to enhance the incentives. The rule should be stable when the participants study it. A mechanism which is incentive-compatible in either a one-shot or repeated-game situation might be used by the agents to exploit its thresholds and settlement periods. For instance, the agents may shift the demand to the reward limits, divide the transactions to alter the fee distribution, or produce simulated activities to modify the signal recognized by the system.

The mechanism is included in the environment in which the agents learn. A fee rule, a reward rule or a liquidity policy restricts the behavior and at the same time produces signals that the agents can examine in subsequent rounds. The stability analysis should consider the changes with time, population-based feedback, agent identity separation and cooperation among agents operated by the same person. It must re-examine the incentive compatibility for the learners, the detection of collusion between different addresses, the prevention of reward hacking and the maintenance of price or allocation stability during high-speed demands.

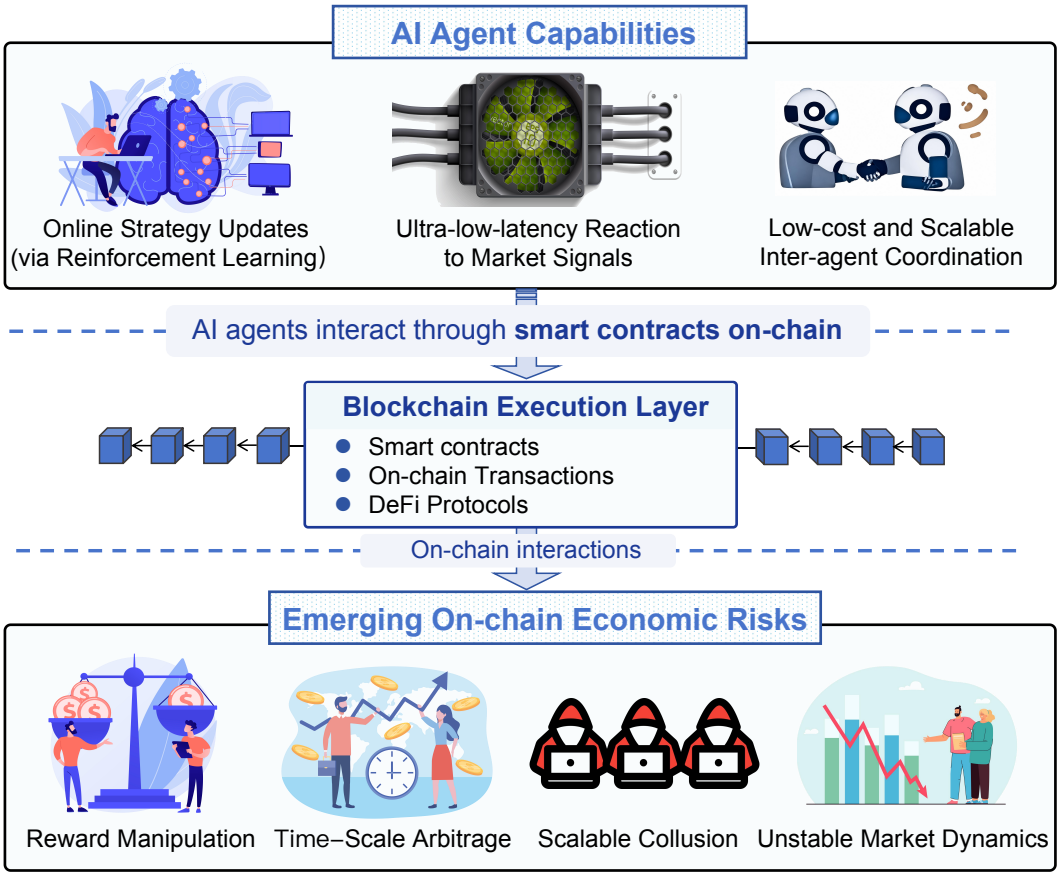

Figure 5. How autonomous agents change blockchain market dynamics. Low-latency strategy updates, identity splitting, and cheap coordination can turn fee, liquidity, or sequencing rules into targets for reward manipulation, time-scale arbitrage, congestion shaping, and collusion.

One possible method is to regard economic rules as adaptive control systems rather than fixed payoff tables. The aim is to decrease the benefit of many inquiries, preserve the evidence for detecting fraud and ensure the efficiency of automatic participation.

8. Conclusion

This survey investigates whether the present blockchain technologies for transaction processing and scalability can enable AI agents as persistent and responsible members. The literature has shown some achievements in state storage, verifiable execution, partitioning, DAG based consensus, stateless validation, parallel execution, fair ordering, liquidity rewards, and fee control. However, most assurances are only given for the states, messages, blocks, transactions or market regulations. The trust of an agent needs a larger scope of consideration. The system should keep records of the identity and history, ensure the correctness of the workflow across different chains and services, make the performance of tasks reviewable and maintain stability when the automatic participants adjust to the protocol. The scalability is related to the trust since the auditability, definiteness, repeatability and economic reliability should be kept under frequent autonomous activities. In the future, the blockchain structures will require more connections between the storage, coordination, consensus, execution and incentive layers so that the behaviour of the agents can be audited, questioned and controlled over a long period of time.

Declaration of generative AI and AI-assisted technologies

During manuscript preparation, the authors used ChatGPT and Gemini to check grammar, wording, and readability across a large portion of the manuscript. These tools were not used to generate the research ideas, select the surveyed literature, derive technical claims, analyze data, or formulate the conclusions. The authors reviewed and approved all technical content, comparisons, interpretations, and final wording, and take full responsibility for the accuracy and integrity of the manuscript.

Acknowledgments

This work was partially supported by the National Natural Science Foundation of China (NSFC) (No. 62272496) and Guangdong Basic and Applied Basic Research Foundation (No. 2025B1515020053).

Authors' contribution

Jiana Liao: conceptualization; methodology; investigation; writing—original draft preparation. Qinde Chen: methodology; writing—review and editing. Jian Zheng: methodology; writing—review and editing. Xiaoke Tang: investigation; writing—review and editing. Feihong Hu: investigation; writing—review and editing. Huawei Huang: conceptualization; writing—review and editing; supervision; project administration; funding acquisition. All authors have read and agreed to the published version of the manuscript.

Conflicts of interest

Huawei Huang holds the position of Associate Editor for *Blockchain* and has not peer reviewed or made any editorial decisions for this paper. The authors declare no conflicts of interest.

References

[1] Yang Q, Zhao Y, Huang H, Xiong Z, Kang J, *et al.* Fusing blockchain and AI with metaverse: a survey. *IEEE Open J. Comput. Soc.* 2022, 3:122–136.

[2] Dong S, Abbas K, Li M, Kamruzzaman J. Blockchain technology and application: an overview. *PeerJ Comput. Sci.* 2023, 9:e1705.

[3] Rao IS, Kiah MM, Hameed MM, Memon ZA. Scalability of blockchain: a comprehensive review and future research direction. *Cluster Comput.* 2024, 27(5):5547–5570.

[4] Yang Y, Chai H, Song Y, Qi S, Wen M, *et al.* A survey of AI agent protocols. *arXiv* 2025, arXiv:2504.16736.

[5] Xi Z, Chen W, Guo X, He W, Ding Y, *et al.* The rise and potential of large language model based agents: a survey. *Sci. China Inf. Sci.* 2025, 68(2):121101.

[6] Zhang R, Du H, Liu Y, Niyato D, Kang J, *et al.* Generative AI agents with large language model for satellite networks via a mixture of experts transmission. *IEEE J. Sel. Areas Commun.* 2024, 42(12):3581–3596.

[7] Huang X, Lian J, Lei Y, Yao J, Lian D, *et al.* Recommender AI agent: integrating large language models for interactive recommendations. *ACM Trans. Inf. Syst.* 2025, 43(4):1–33.

[8] Deng Z, Guo Y, Han C, Ma W, Xiong J, *et al.* AI agents under threat: a survey of key security challenges and future pathways. *ACM Comput. Surv.* 2025, 57(7):1–36.

[9] Zhang P, Ding S, Zhao Q. Exploiting blockchain to make AI trustworthy: a software development lifecycle view. *ACM Comput. Surv.* 2024, 56(7):1–31.

[10] Lui E, Sun R, Shah V, Xiong X, Sun J, *et al.* SoK: blockchain-based decentralized AI (DeAI). *arXiv* 2024, arXiv:2411.17461.

[11] Xu J, Wang C, Jia X. A survey of blockchain consensus protocols. *ACM Comput. Surv.* 2023, 55(13s):1–35.

[12] Heo JW, Ramachandran GS, Dorri A, Jurdak R. Blockchain data storage optimisations: a comprehensive survey. *ACM Comput. Surv.* 2024, 56(7):1–27.

[13] Ren K, Ho NM, Loghin D, Nguyen TT, Ooi BC, *et al.* Interoperability in blockchain: a survey. *IEEE Trans. Knowl. Data Eng.* 2023, 35(12):12750–12769.

[14] Liang J, Chen W, Hong Z, Zhu H, Qiu W, *et al.* MoltDB: accelerating blockchain via ancient state segregation. *IEEE Trans. Parallel Distrib. Syst.* 2024, 35(12):2545–2558.

[15] Ye Z, Misra U, Cheng J, Zhou W, Song D. Specular: towards secure, trust-minimized optimistic blockchain execution. In *Proceedings of 2024 IEEE Symposium on Security and Privacy (SP)*, San Francisco, USA, May 19–23, 2024, pp. 3943–3960.

[16] Peng W, Li X, Niu J, Zhang X, Zhang Y. Ensuring state continuity for confidential computing: a blockchain-based approach. *IEEE Trans. Dependable Secure Comput.* 2024, 21(6):5635–5649.

[17] Zhou K, Zhang X, Wang C, Cheng H. Accelerating cross-shard blockchain consensus via decentralized coordinators service with verifiable global states. *IEEE Trans. Serv. Comput.* 2024, 17(4):1340–1353.

[18] Qi X, Li Y. Lightcross: sharding with lightweight cross-shard execution for smart contracts. In *Proceedings of IEEE INFOCOM 2024-IEEE Conference on Computer Communications*, Vancouver, Canada, May 20–23, 2024, pp. 1681–1690.

[19] Wang X, Li B, Jia L, Sun Y. Orbit: a dynamic account allocation mechanism in sharding blockchain system. In *Proceedings of 2024 IEEE 44th International Conference on Distributed Computing Systems (ICDCS)*, Jersey City, USA, July 23–26, 2024, pp. 333–344.

[20] Jiang S, Cao J, Tung CL, Wang Y, Wang S. Sharon: secure and efficient cross-shard transaction processing via shard rotation. In *Proceedings of IEEE INFOCOM 2024-IEEE Conference on Computer Communications*, Vancouver, Canada, May 20–23, 2024, pp. 2418–2427.

[21] Hu D, Wang J, Liu X, Xu H, Wu X, *et al.* Ladder: a convergence-based structured DAG blockchain for high throughput and low latency. In *Proceedings of 22nd USENIX Symposium on Networked Systems Design and Implementation (NSDI 25)*, Philadelphia, USA, April 28–30, 2025, pp. 779–794.

[22] Chen W, Xia D, Cai Z, Dai H, Zhang J, *et al.* Porygon: scaling blockchain via 3D parallelism. In *Proceedings of 2024 IEEE 40th International Conference on Data Engineering (ICDE)*, Utrecht, The Netherlands, May 13–16, 2024, pp. 1944–1957.

[23] Far SB, Qazani MRC, Bamakan SMH, Rad AI, Zareravasan A. A novel Layer 2 framework for breaking the blockchain trilemma problem using MPC-in-the-Head. *Comput. Networks* 2025, 261:111148.

[24] Xu T, Zhong Y, Zhang Y, Xiong R, Zhang J, *et al.* Vegeta: enabling parallel smart contract execution in leaderless blockchains. In *Proceedings of 22nd USENIX Symposium on Networked Systems Design and Implementation (NSDI 25)*, Philadelphia, USA, April 28–30, 2025, pp. 795–811.

[25] Zhao S, Zhang Z, Wang J, Yuan Y, Zhang M, *et al.* Efficient partial order based transaction processing for permissioned blockchains. In *Proceedings of 2024 IEEE 40th International Conference on Data Engineering (ICDE)*, Utrecht, The Netherlands, May 13–16, 2024, pp. 1888–1901.

[26] Gramoli V, Lu Z, Tang Q, Zarbafian P. AOAB: optimal and fair ordering of financial transactions. In *Proceedings of 2024 54th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN)*, Brisbane, Australia, June 24–27, 2024, pp. 377–388.

[27] Liu Y, Jia L, Wang X, Huang H, Zhao Q, *et al.* Coral: a blockchain protocol for handling transactions with deadline constraints. *Comput. Networks* 2024, 251:110620.

[28] Chen Q, Huang H, Yin Z, Ye G, Yang Q. Broker2earn: towards maximizing broker revenue and system liquidity for sharded blockchains. In *Proceedings of IEEE INFOCOM 2024-IEEE Conference on Computer Communications*, Vancouver, Canada, May 20–23, 2024, pp. 251–260.

[29] Tsabary I, Manuskin A, Bar-Zur R, Eyal I. LedgerHedger: gas reservation for smart contract security. In *Proceedings of 28th International Conference on Financial Cryptography and Data Security (FC 2024)*, Willemstad, Curaçao, March 4–8, 2024, pp. 248–270.

[30] Tas EN, Tse D, Yang L, Zindros D. Light clients for lazy blockchains. In *Proceedings of 28th International Conference on Financial Cryptography and Data Security (FC 2024)*, Willemstad, Curaçao, March 4–8, 2024, pp. 3–21.

[31] Wang W, Peng H, Duan J, Wang L, Hu X, *et al.* Resilient and redactable blockchain with two-level rewriting and version detection. *IEEE Trans. Inf. Forensics Secur.* 2024, 20:1163–1175.

[32] Zhao L, Guo D, Luo L, Shen Y, Ren B, *et al.* Concordit: a credit-based incentive mechanism for permissioned redactable blockchain. *Comput. Networks* 2024, 255:110848.

[33] Heiss J, Oegel T, Shakeri M, Tai S. Verifiable carbon accounting in supply chains. *IEEE Trans. Serv. Comput.* 2023, 17(4):1861–1874.

[34] Zhang H, Yeo M, Estrada-Galinanes V, Ford B. Zeroauction: zero-deposit sealed-bid auction via delayed execution. In *Proceedings of International Conference on Financial Cryptography and Data Security*, Willemstad, Curaçao, March 4–8, 2024, pp. 170–188.

[35] Chen Z, Zhu L, Jiang P, He J, Zhang Z. A generic blockchain-based steganography framework with high capacity via reversible gan. In *Proceedings of IEEE INFOCOM 2024—IEEE Conference on Computer Communications*, Vancouver, Canada, May 20–23, 2024, pp. 241–250.

[36] Han X, Wang N, Che S, Yang H, Zhang K, *et al.* Enhancing investment analysis: optimizing AI-Agent collaboration in financial research. In *Proceedings of the 5th ACM International Conference on AI in Finance*, Brooklyn, USA, November 14–17, 2024, pp. 538–546.

[37] Liang J, Yao P, Chen W, Hong Z, Zhang J, *et al.* Sparrow: expediting smart contract execution for blockchain sharding via inter-shard caching. *IEEE Trans. Parallel Distrib. Syst.* 2024, 36(3):377–390.

[38] Huang H, Lin Y, Zheng Z. Account migration across blockchain shards using fine-tuned lock mechanism. In *Proceedings of IEEE INFOCOM 2024—IEEE Conference on Computer Communications*, Vancouver, Canada, May 20–23, 2024, pp. 271–280.

[39] Wang K, Jia L, Song Z, Sun Y. Mitosis: a scalable sharding system featuring multiple dynamic relay chains. *IEEE Trans. Parallel Distrib. Syst.* 2024, 35(12):2497–2512.

[40] Ren Y, Zhou Z, Han Z, Ge C, Huang H. AdaptiveShard: enhancing throughput and security of sharded blockchain with adaptive verifiable coding. *IEEE Trans. Inf. Forensics Secur.* 2025, 20:7927–7939.

[41] Yang W, Chen W, Xue L, Zou W, Huang L. EZchain: a secure scalable blockchain protocol via passive sharding. *IEEE Trans. Dependable Secure Comput.* 2024, 22(3):2891–2908.

[42] Tian J, Jing C, Tian J, Li Y. PartChain: scaling blockchain through account-based partitioned sharding. *Comput. Networks* 2024, 254:110773.

[43] Che C, Li S, Wang X. Manifoldchain: maximizing blockchain throughput via bandwidth-clustered sharding. *arXiv* 2024, arXiv:2407.16295.

[44] Huang H, Ye G, Yang Q, Chen Q, Yin Z, *et al.* Blockemulator: an emulator enabling to test blockchain sharding protocols. *IEEE Trans. Serv. Comput.* 2025, 18:690–703.

[45] Cheng F, Xiao J, Liu C, Zhang S, Zhou Y, *et al.* SharDAG: scaling DAG-based blockchains via adaptive sharding. In *Proceedings of 2024 IEEE 40th International Conference on Data Engineering (ICDE)*, Utrecht, The Netherlands, May 13–16, 2024, pp. 2068–2081.

[46] Yin L, Xu J, Liang K, Zhang Z. Sidechains with optimally succinct proof. *IEEE Trans. Dependable Secure Comput.* 2023, 21(4):3375–3389.

[47] Yu M, Zhao Y, Wang J, Hu D, Liu X, *et al.* Enabling high-performance eov blockchains via transaction ordering exploration. In *Proceedings of 2024 IEEE 44th International Conference on Distributed Computing Systems (ICDCS)*, Jersey City, USA, July 23–26, 2024, pp. 356–366.

[48] Wadhwa S, Zanolini L, Asgaonkar A, D'Amato F, Fang C, *et al.* Data independent order policy enforcement: limitations and solutions. In *Proceedings of the 2024 on ACM SIGSAC Conference on Computer and Communications Security*, Salt Lake City, USA, October 14–18, 2024, pp. 378–392.

[49] Ciampi M, Kiayias A, Shen Y. Universal composable transaction serialization with order fairness. In *Proceedings of Annual International Cryptology Conference*, Santa Barbara, USA, August 18–22, 2024, pp. 147–180.

[50] Zhou Y, Yan Z, Chen Y, Ma F, Chen T, *et al.* Chord: towards a unified detection of blockchain transaction parallelism bugs. In *Proceedings of 2025 IEEE/ACM 47th International Conference on Software Engineering (ICSE)*, Ottawa, Canada, April 26–May 6, 2025, pp. 742–742.

[51] Wang S, Yang M, Cao J, Ling Z, Tang Q, *et al.* Core: transaction commit-controlled release of private data over blockchains. In *Proceedings of 2024 IEEE 44th International Conference on Distributed Computing Systems (ICDCS)*, Jersey City, USA, July 23–26, 2024, pp. 322–332.

[52] Kim D, Yun S, Lee S, Lee J, Niyato D. Intelligent transaction generation control for permissioned blockchain-based services. *IEEE Trans. Serv. Comput.* 2025, 18(2):828–838.

[53] Brahem A, Henry T, Bhiri S, Devogele T, Messai N, *et al.* Towards a trustworthy and adaptive execution of business process choreographies. *IEEE Trans. Serv. Comput.* 2024, 17(6):4383–4396.

[54] Kapoor S, Stroebl B, Siegel ZS, Nadgir N, Narayanan A. AI agents that matter. *arXiv* 2024, arXiv:2407.01502.

[55] Putta P, Mills E, Garg N, Motwani S, Finn C, *et al.* Agent Q: advanced reasoning and learning for autonomous AI agents. *arXiv* 2024, arXiv:2408.07199.

[56] Masterman T, Besen S, Sawtell M, Chao A. The landscape of emerging AI agent architectures for reasoning, planning, and tool calling: a survey. *arXiv* 2024, arXiv:2404.11584.

[57] Tomasev N, Franklin M, Leibo JZ, Jacobs J, Cunningham WA, *et al.* Virtual agent economies. *arXiv* 2025, arXiv:2509.10147.

[58] Zhang Y, Xiang Y, Lei Y, Wang Q, Qiu T, *et al.* SoK: blockchain agent-to-agent payments. *arXiv* 2026, arXiv:2604.03733.

[59] Motwani SR, Baranchuk M, Strohmeier M, Bolina V, Torr PH, *et al.* Secret collusion among AI agents: multi-agent deception via steganography. In *Proceedings of Advances in Neural Information Processing Systems (NeurIPS 2024)*, Vancouver, Canada, December 10–15, 2024, pp. 73439–73486.