AIL2 Whitepaper

The Largest AI L2 for Multi-Chain Ecosystem

Empowering developers to build deAI apps 100x faster across blockchains

Supported Chains: Ethereum | BNB Chain | XLayer | Base | Mantle | GIWA

Version: v1.5 (Official Release)
Date: 2025
Authors: AIL2 Core Contributors & Community

Disclaimer

This document is for informational and technical disclosure purposes only and does not constitute investment advice, legal advice, tax advice, or solicitation to buy or sell any assets.

Digital assets and blockchain systems carry high risks, including technical risks, market risks, regulatory risks, and liquidity risks. Users may suffer total loss. The AIL2 protocol and token economics may be adjusted based on audit conclusions, governance decisions, market changes, and regulatory requirements.

Before participating in any AIL2-related activities, please conduct independent due diligence and consult professional legal, financial, and tax advisors. Any forward-looking statements in this whitepaper are based on current expectations and assumptions, and actual results may differ.

The AIL2 team is not responsible for any direct or indirect losses arising from the use of information in this whitepaper. Investors should make informed decisions based on their own risk tolerance and investment objectives.

Table of Contents

Disclaimer 2
Abstract 4
Core Value Proposition 4
Three Target Customer Segments 5
Chapter 1 | Executive Summary & Positioning 6
1.1 One-Sentence Positioning 6
1.2 Core Problems AIL2 Solves 6
1.3 Target Customer Segments 7
1.4 Competitive Advantages 7
Chapter 2 | Industry Pain Points & Opportunities 8
2.1 Industry Status Analysis 8
2.2 Core Pain Points Analysis 8
2.3 Market Opportunity Assessment 9
2.4 Market Size Forecast 9
Chapter 3 | System Overview & User Tiers 10
3.1 Four-Layer Architecture 10
3.2 Architecture Diagram 11
3.3 Core Design Principles 11
Chapter 4 | Distributed GPU Network & Model Containers 12
4.1 Network Role Definitions 12
4.2 Model Container Lifecycle 13
4.3 Container Metadata Specification 13
Chapter 5 | Routing & Scheduling System (Mathematical Models) 14
5.1 Formal Definition of Scheduling Problem 14
5.2 Multi-Objective Scoring Function 15
5.3 Latency Prediction Model 17
5.4 Exploration-Exploitation Balancing 18
5.5 Load Balancing Algorithm 19
5.6 Routing Workflow Pseudocode 20
Chapter 6 | Metering Receipts & AI L2 Settlement 22
6.1 Receipt Data Structure 22
6.2 Billing Functions 23
6.3 Batch Settlement Mechanism 24
6.4 Cross-Chain Settlement Modes 25
6.5 Dispute Resolution Mechanism 26
6.6 Idempotency & Replay Protection 26
Chapter 7 | Launchpad: IMO / IAO 27
7.1 IMO (Initial Model Offering) 27
7.2 IAO (Initial Agent Offering) 28

7.3 Milestone Unlock Mechanism... 28
Chapter 8 | Security, Anti-Cheating & Verifiable Inference.. 29
8.1 Threat Model... 29
8.2 Risk Scoring System...30
8.3 Penalty & Demotion Mechanism.. 31
8.4 Slashing Mechanism..31
8.5 Challenge Window & Arbitration... 32
8.6 Verifiable Inference Roadmap.. 32
Chapter 9 | Governance & Roadmap..33
9.1 Governance Scope...33
9.2 Governance Structure..33
9.3 Three-Year Roadmap...34
Conclusion.. 35
References...36
Appendix..38
Appendix A: Glossary... 38
Appendix B: Mathematical Notation..38

Abstract

AIL2 is a decentralized AI application ecosystem based on a distributed GPU miner network and cross-chain AI L2 settlement layer. As the largest-scale AI Layer 2 network covering Ethereum, BNB Chain, XLayer, Base, Mantle, GIWA and other mainstream public chains, AIL2 is dedicated to providing developers with a one-stop decentralized AI infrastructure solution.

In recent years, the convergence of blockchain and artificial intelligence has become a research hotspot in both academia and industry. Sun et al. noted in their systematic review [1] that Decentralized Artificial Intelligence (DeAI) has emerged as a promising paradigm that leverages decentralization and transparency to improve the trustworthiness of AI systems. Despite rapid adoption in industry, the academic community lacks a systematic analysis of DeAI's technical foundations, opportunities, and challenges.

AIL2 abstracts complex underlying capabilities into a standardized service layer through innovative architectural design. Developers only need to provide their AI models and package them as model containers to deploy on AIL2's global GPU network to build distributed AI applications. User requests are sent to optimal GPU nodes for inference execution through AIL2's intelligent routing and scheduling system, and results are securely returned to users.

Core Value Proposition

AIL2's core value proposition is reflected in the following aspects:

- Decentralized GPU Computing Network: Integrates fragmented global GPU resources into an elastic, scalable distributed computing network
- Intelligent Routing & Scheduling: Multi-objective optimization-based request distribution system achieving low latency and high availability
- Cross-Chain Settlement Layer: Supports multi-chain native payments and unified reconciliation, lowering user barriers
- Demand-Bound Incentives: Innovative token economics deeply binding miner rewards with real demand, suppressing inflation
- One-Stop Launch Platform: IMO/IAO mechanisms reducing AI project fundraising and cold-start barriers

Three Target Customer Segments

AIL2 serves three core customer groups, covering the complete user profile from professional developers to ordinary creators:

AIL2 Core (Professional Developers)

For professional AI developers with smart contract development capabilities. These users can freely integrate core capabilities such as underlying computing, scheduling, and settlement through AIL2's complete technical documentation and SDK toolkit to build highly customized decentralized AI applications.

AIL2 Builder (IMO Mode)

For developers who lack smart contract development experience but have excellent AI models. Through the AIL2 Builder platform, developers can complete token issuance, model container creation, and IMO fundraising with one click, quickly monetizing models and gaining initial computing power support.

AIL2 Creator (IAO Mode)

For creators who don't understand smart contracts or AI model development but have creativity and operational capabilities. These users can combine existing model capabilities through a visual interface to create unique AI Agent applications and issue project tokens through the IAO mechanism to build communities.

Chapter 1 | Executive Summary & Positioning

1.1 One-Sentence Positioning

AIL2 is the largest AI L2 covering Ethereum, BNB Chain, XLayer, Base, Mantle, and GIWA, empowering developers to build decentralized AI applications 100x faster across chains.

This positioning contains three core meanings: First, AIL2 is a Layer 2 solution specifically optimized for AI application scenarios, providing higher throughput and lower transaction costs than the main chain; Second, AIL2 has cross-chain native properties, capable of running seamlessly on multiple mainstream public chains to provide users with a unified experience; Finally, AIL2 improves development efficiency 100x through standardized infrastructure, allowing developers to focus on AI model value creation.

1.2 Core Problems AIL2 Solves

The difficulty in scaling AI applications is often not training, but the delivery system. As Talaei Khoei et al. pointed out in their review [2], the integration of blockchain and decentralized AI in cybersecurity faces many challenges, including data security, privacy protection, and trust establishment. The core challenges facing current AI developers can be summarized in four aspects:

1.2.1 Distributed Execution Challenges

Building a reliable distributed AI inference system requires solving numerous technical challenges: container orchestration requires precise resource allocation and lifecycle management; scheduling systems need real-time trade-offs between latency, cost, and quality; elastic scaling needs dynamic resource adjustment based on load; fault tolerance and retry mechanisms need to handle various edge cases; canary releases need to ensure smooth version updates.

1.2.2 Metering & Billing Challenges

AI inference service metering and billing is a complex engineering problem. It requires precise statistics on multi-dimensional resource consumption such as token count, GPU usage time, and network bandwidth; clear service quality indicators (SLA) and corresponding penalty mechanisms need to be defined.

1.2.3 Revenue Sharing & Incentive Challenges

Decentralized AI networks need to establish fair and reasonable revenue distribution mechanisms. Miners providing computing power should receive reasonable compensation, developers contributing models should share in revenues, and the protocol itself needs ongoing funding support for operations and development.

1.2.4 Cross-Chain Payment Challenges

Web3 users are distributed across different blockchain ecosystems, holding different assets and using different wallets. To provide these users with seamless AI service experiences, unified cross-chain payment and settlement capabilities need to be established.

1.3 Target Customer Segments

1.3.1 Core: Professional Developers

Core users are professional teams with complete Web3 and AI development capabilities. AIL2 provides Core users with complete technical documentation, SDK toolkits, and API interfaces.

1.3.2 Builder: Model Developers

Builder users have excellent AI models but may lack smart contract development experience. The AIL2 Builder platform provides a one-stop solution.

1.3.3 Creator: Application Creators

Creator users have keen market insight, excellent product design capabilities, or strong community operation capabilities. The AIL2 Creator platform allows these users to participate in decentralized AI value creation.

1.4 Competitive Advantages

Dimension	AIL2 Advantage
Decentralization	True distributed GPU network, no single point of failure or censorship risk
Multi-Chain Support	Native cross-chain architecture, supporting 6+ mainstream chains
Developer Experience	Three product tiers covering full tech stack users
Incentive Mechanism	Demand-bound mining, suppressing inflation and wash trading
Fundraising Capability	Built-in IMO/IAO mechanism, one-stop project incubation

Chapter 2 | Industry Pain Points & Opportunities

2.1 Industry Status Analysis

Artificial intelligence is undergoing a critical transformation from laboratory to large-scale commercial applications. Generative AI technologies represented by large language models achieved breakthrough progress in 2023-2024, and the commercial value of AI applications has been fully validated by the market.

According to research published in MDPI Information [3], the synergistic combination of blockchain and AI introduces unique beneficial features with potential to enhance the performance and efficiency of existing ICT systems. The global blockchain market is expected to grow from $564 million in 2024 to $2.475 billion by 2030, with a CAGR of 27.9%.

2.2 Core Pain Points Analysis

2.2.1 Structural Problems of Centralized Platforms

The current AI inference service market is dominated by a few large tech companies. This centralized landscape brings supply concentration risks, billing transparency issues, and platform lock-in effects. As Mafrur et al. pointed out in IET Blockchain [4], many so-called 'decentralized AI' projects actually maintain centralized control over core operations, presenting an 'illusion of decentralized AI' problem.

2.2.2 High Barriers to Self-Built Systems

Facing the various problems of centralized platforms, some technically capable teams choose to build their own distributed GPU systems. However, this path is also challenging: high engineering complexity, high operational costs, and slow iteration speed.

2.2.3 Special Needs of Web3 Ecosystem

Web3 applications have unique needs for AI infrastructure: programmable micropayments, transparent revenue sharing, and incentive alignment. Existing solutions struggle to meet these needs. Research by Gürpinar in Frontiers in Blockchain [5] shows that the Web 4.0 era will see many autonomous AI agents emerging, which need to operate in decentralized infrastructure, posing new challenges for AI infrastructure.

2.3 Market Opportunity Assessment

Based on in-depth analysis of industry pain points, the AIL2 team has identified three key market opportunities on the supply side, demand side, and settlement side.

2.4 Market Size Forecast

Market	2025	2028	CAGR

Market	2025	2028	CAGR
Global AI Inference	$200B	$650B	48%
Decentralized AI	$5B	$50B	115%
Web3 AI Apps	$2B	$25B	132%

Chapter 3 | System Overview & User Tiers

3.1 Four-Layer Architecture

AIL2 adopts a layered architecture design, modularizing complex system functions to achieve separation of concerns and independent evolution. The overall architecture is divided into four layers:

3.1.1 Compute Layer (GPU Miner Network)

The compute layer is the computing foundation of the AIL2 network, consisting of globally distributed GPU miner nodes. Each miner node runs standardized AIL2 node software, responsible for pulling and running model containers, executing inference tasks, and generating metering receipts.

3.1.2 Network Layer (Router / Scheduler / Transport)

The network layer handles request ingestion, routing, and data transmission, serving as the bridge connecting users and computing power. Router handles authentication and rate limiting; Scheduler handles intelligent scheduling; Transport handles data transmission.

3.1.3 Settlement Layer (AI L2 Accounting + Cross-chain)

The settlement layer is AIL2's core differentiating capability as an AI L2, responsible for metering receipt verification, fee calculation, and multi-party settlement.

3.1.4 Ecosystem Layer (SDK / Builder IMO / Creator IAO)

The ecosystem layer is the product interface for developers and users, providing tools and services at different abstraction levels.

3.2 Architecture Diagram

User / dApp / Agent → SDK / Gateway → Router → Scheduler / Indexer → GPU Node A/B/C → Transport → User

GPU Nodes → Usage Receipts → Accounting → Split → Miners / Developers / Treasury / Ecosystem

3.3 Core Design Principles

- Decentralization First: All core components support distributed deployment, avoiding single points of failure and censorship risks
- Progressive Decentralization: Gradually opening key components to community operation while ensuring system stability
- Composability: Each layer module can be used independently or flexibly combined to meet different scenario needs
- Transparent & Verifiable: All metering and settlement data stored on-chain for anyone to independently audit

- Backward Compatible: API and protocol upgrades follow strict version management to protect existing integration stability

Chapter 4 | Distributed GPU Network & Model Containers

4.1 Network Role Definitions

AIL2's distributed GPU network operates through multiple roles working together, each bearing specific responsibilities. According to a survey published in ACM Computing Surveys [6], deep learning workload scheduling in GPU datacenters is a complex systems engineering problem that needs to comprehensively consider resource utilization, task completion time, energy consumption, and other objectives.

4.1.1 GPU Worker (Computing Miner)

GPU Workers are the computing power providers of the network. Their core responsibilities include: image pulling, container running, inference execution, receipt generation, heartbeat reporting, and benchmarking. Miners need to stake a certain amount of AIL2 tokens as service collateral.

4.1.2 Router

Router is the entry gateway for user requests, responsible for authentication, rate limiting, timeout management, and degradation strategies.

4.1.3 Scheduler

Scheduler is the brain of the network, responsible for intelligent scheduling decisions: node indexing, capability tagging, reputation scoring, risk assessment, and load balancing.

4.1.4 Settlement

Settlement handles all fund flow-related operations: batch settlement, revenue distribution, cross-chain synchronization, and dispute handling.

4.1.5 Verifier (Optional)

Verifier is an optional component for enhanced network security, responsible for random spot-checking, result comparison, and anomaly reporting.

4.2 Model Container Lifecycle

Model containers are standardized units carrying AI models in the AIL2 network. The complete lifecycle includes:

Build → Sign → Register → Pull → Benchmark → Health Check → Serve → Upgrade → Rollback

4.3 Container Metadata Specification

Each model container must declare a set of standardized metadata:

- model_id: Unique identifier for the model
- version: Semantic version number
- gpu_requirement: GPU requirements including VRAM size and CUDA version
- pricing: Pricing strategy supporting hybrid billing modes
- constraints: Constraints such as region restrictions and maximum latency requirements
- revenue_split: Revenue distribution ratios
- qos: Service quality parameters
- security: Security policies

Chapter 5 | Routing & Scheduling System (Mathematical Models)

5.1 Formal Definition of Scheduling Problem

AIL2's scheduling system needs to solve a constrained multi-objective optimization problem. According to research by Fu et al. published at ACM SIGCOMM 2024 [7], communication scheduling in GPU clusters is an NP-complete problem requiring efficient heuristic algorithms for approximate solutions. Formally, given an inference request r and candidate node set $N = \{n_1, n_2, ..., n_m\}$, the scheduler needs to find the optimal node n*:

$$n^* = argmax\_\{n \in N\}\ Score(n, r)\ \ subject\ to\ \ C(n, r) = true$$

Where Score(n, r) is the composite scoring function of node n for request r, and C(n, r) is the constraint set.

Constraints C(n, r) include hard constraints and soft constraints:

Hard Constraints (Must Satisfy):

- GPU VRAM constraint: VRAM(n) ≥ VRAM_required(r)
- CUDA version constraint: CUDA(n) ≥ CUDA_required(r)
- Region constraint: Region(n) ∈ Allowed_Regions(r)
- Availability constraint: Status(n) = AVAILABLE

Soft Constraints (Best Effort):

- Latency preference: Latency(n) ≤ Max_Latency(r)
- Cost preference: Cost(n) ≤ Budget(r)
- Quality preference: Quality(n) ≥ Min_Quality(r)

5.2 Multi-Objective Scoring Function

AIL2 uses a weighted multi-objective scoring function to evaluate each candidate node. The scoring function design follows three principles: decomposability, adjustability, and robustness:

$$Score(n, r) = \Sigma_i\ w_i \cdot \varphi_i(x_i(n, r))$$

Where w_i is the weight of the i-th objective, φ_i is the normalization function, and x_i is the raw metric value.

The specific scoring function expands to:

$$Score(n,r) = w_l \cdot \varphi_l(lat) + w_r \cdot \varphi_r(succ) + w\_c \cdot \varphi\_c(cost) + w\_q \cdot \varphi\_q(qual) + w\_p \cdot \varphi\_p(prox) - w\_s \cdot \varphi\_s(risk)$$

5.2.1 Latency Factor φ_l (lat)

The latency factor uses an exponential decay function, giving significantly higher scores to low-latency nodes:

$$\varphi_l(lat) = exp(-\lambda_l \cdot lat / lat\_baseline)$$

Where λ_l is the decay coefficient (default 2.0), lat_baseline is the baseline latency (default 500ms).

5.2.2 Success Rate Factor φ_r(succ)

The success rate factor uses a sigmoid function, producing steep score changes near the threshold:

$$\varphi_r(succ) = 1 / (1 + exp(-k_r \cdot (succ - succ\_threshold)))$$

Where k_r is the steepness coefficient (default 20), succ_threshold is the success rate threshold (default 0.95).

5.2.3 Cost Factor φ_c(cost)

The cost factor uses linear normalization, giving higher scores to lower-cost nodes:

$$\varphi\_c(cost) = 1 - (cost - cost\_min) / (cost\_max - cost\_min)$$

5.2.4 Quality Factor φ_q(qual)

The quality factor considers multiple quality dimensions:

$$qual = \alpha\_acc \cdot accuracy + \alpha\_rel \cdot relevance + \alpha\_coh \cdot coherence$$

$$\varphi\_q(qual) = qual\hat{}\gamma\_q$$

5.2.5 Proximity Factor φ_p(prox)

The proximity factor is calculated based on geographic distance using the Haversine formula:

$$d = 2R \cdot arcsin(\sqrt{}(sin^2((\varphi_2-\varphi_1)/2) + cos(\varphi_1)\cdot cos(\varphi_2)\cdot sin^2((\lambda_2-\lambda_1)/2)))$$

$$\varphi\_p(prox) = exp(-d / d\_scale)$$

5.2.6 Risk Factor φ_s(risk)

The risk factor is a penalty term calculated based on node's historical behavior:

$$risk = sigmoid(\beta_1\cdot z(burst) + \beta_2\cdot z(dispute) + \beta_3\cdot z(fail) + \beta_4\cdot z(fraud))$$

$$\varphi\_s(risk) = risk \cdot (1 + penalty\_multiplier \cdot violations)$$

5.3 Latency Prediction Model

Accurate latency prediction is key to scheduling decisions. According to research by Sharma et al. on arXiv [8], network-sensitive GPU cluster scheduling can improve end-to-end completion time by up to 69%. AIL2 uses a hybrid model combining queuing theory analysis and machine learning prediction:

5.3.1 Queuing Theory Base Model

For a single GPU node, we model it as an M/G/m queuing system (Poisson arrivals, general service time, m parallel servers):

$$\rho = \lambda / (m \cdot \mu)$$

Where λ is the request arrival rate, m is parallel processing capacity, μ is single-task service rate, ρ is system utilization (requiring $\rho < 1$).

Average queuing wait time uses the Kingman approximation formula:

$$W\_q \approx (\rho / (1-\rho)) \cdot (C\_a^2 + C\_s^2) / 2 \cdot (1/\mu)$$

Where C_a is the coefficient of variation of the arrival process, C_s is the coefficient of variation of service time.

Total predicted latency:

$$lat\_pred = RTT + W\_q + T\_compute$$

5.3.2 Compute Time Prediction

Compute time is related to input complexity, using the following model:

$$T\_compute = T\_base + \alpha \cdot tokens\_in + \beta \cdot tokens\_out + \gamma \cdot tokens\_in \cdot tokens\_out / context\_length$$

Considering GPU model differences, a performance factor is introduced:

$$T\_compute(GPU) = T\_compute\_ref \cdot (FLOPS\_ref / FLOPS\_GPU) \cdot (BW\_ref / BW\_GPU)\hat{}\delta$$

5.3.3 Adaptive Learning

The prediction model continuously optimizes through online learning using exponentially weighted moving averages:

$$\theta\_new = (1 - \alpha\_learn) \cdot \theta\_old + \alpha\_learn \cdot \theta\_observed$$

$$\alpha\_learn = \alpha\_base \cdot \exp(-t / \tau\_decay)$$

5.4 Exploration-Exploitation Balancing

To discover potentially undervalued quality nodes while avoiding performance loss from over-exploration, the scheduling system uses an improved UCB (Upper Confidence Bound) algorithm. The UCB algorithm was originally proposed by Lai and Robbins [9] in 1985 and further developed by Auer et al. [10] in 2002 as the UCB1 algorithm, achieving logarithmic regret bounds without prior knowledge of reward distribution parameters.

5.4.1 UCB Algorithm

The basic UCB1 formula:

$$UCB(n) = \bar{X}(n) + c \cdot \sqrt{(\ln(t) / N(n))}$$

Where $\bar{X}(n)$ is the average reward estimate for node n, t is total scheduling count, N(n) is selection count for node n, c is exploration coefficient.

AIL2 uses the improved UCB-V (UCB with Variance) algorithm, considering reward variance:

$$UCB\text{-}V(n) = \bar{X}(n) + \sqrt{(2 \cdot V(n) \cdot \ln(t)/N(n))} + 3 \cdot b \cdot \ln(t)/N(n)$$

Where V(n) is the variance estimate, b is the reward upper bound.

5.4.2 Softmax Temperature Sampling

Final node selection uses softmax temperature sampling rather than simple argmax:

$$P(n) = \exp(Score(n) / \tau) / \Sigma_j \exp(Score(j) / \tau)$$

Where τ is the temperature parameter controlling exploration degree.

Temperature parameter dynamically adjusts:

$$\tau = \tau\_min + (\tau\_max - \tau\_min) \cdot \exp(-t / \tau\_anneal)$$

5.4.3 Thompson Sampling Variant

For new node cold-start problems, Thompson Sampling is used:

$$\theta(n) \sim Beta(\alpha(n), \beta(n))$$

$$\alpha(n) = \alpha\_prior + successes(n)$$

$$\beta(n) = \beta\_prior + failures(n)$$

Sampling from the Beta posterior distribution naturally achieves exploration-exploitation balance.

5.5 Load Balancing Algorithm

Beyond node selection, global load balancing must be considered. According to research by Wang et al. published in Future Generation Computer Systems [11], the Graph Predictive Algorithm for Resource Scheduling (GPARS) can predict job duration by leveraging spatiotemporal correlations among jobs, minimizing wait time and maximizing computational resource utilization. AIL2 uses a variant of the Weighted Least Connections algorithm:

5.5.1 Load Metrics

Node load metrics comprehensively consider multiple dimensions:

$$Load(n) = w_1 \cdot (ActiveReqs(n)/Capacity(n)) + w_2 \cdot GPU\_Util(n) + w_3 \cdot Mem\_Util(n) + w_4 \cdot Queue\_Len(n)/Queue\_Max(n)$$

5.5.2 Load Balancing Objective

The global load balancing objective is to minimize maximum load:

$$minimize\ \ max\_n\ Load(n)$$

$$\text{subject to } \Sigma\_n\, Alloc(n, r) = 1 \ \forall r$$

Solved approximately using a greedy algorithm, selecting the lowest-load node satisfying constraints each time.

5.5.3 Overload Protection

When node load exceeds threshold, overload protection triggers:

$$\text{if } Load(n) > Load\_threshold \text{ then } Penalty(n) = (Load(n) - Load\_threshold)^2 \cdot k\_overload$$

5.6 Routing Workflow Pseudocode

Complete routing workflow:

```
function route_request(r, model m):
    # 1. Pre-checks
    auth(r); rate_limit(r)
    # 2. Candidate node filtering
    candidates = indexer.nodes_for(m, region=r.region, vram>=r.vram)
    candidates = filter(candidates, hard_constraints(r))
    # 3. Score calculation
    for n in candidates:
        lat_pred = predict_latency(n, r)
        score[n] = calc_score(lat_pred, n.succ_rate, n.cost, n.quality, n.proximity, n.risk)
        ucb[n] = calc_ucb(score[n], n.visits, total_visits)
    # 4. Node selection
    topK = top_k(ucb, 20); probs = softmax(score[topK], temperature)
    chosen = sample(topK, probs)
    # 5. Request dispatch & retry
    resp = dispatch(chosen, r)
    if fail(resp): chosen = retry_select(topK, exclude=chosen); resp = dispatch(chosen, r)
    # 6. Stats update
    update_stats(chosen, resp); return resp
```

Chapter 6 | Metering Receipts & AI L2 Settlement

6.1 Receipt Data Structure

After each inference request completes, the executing node generates a Usage Receipt recording detailed service information. Receipts are the atomic data units of AIL2's settlement system.

6.1.1 Receipt Field Definitions

Complete receipt data structure:

R = (receipt_id, project_id, model_id, version, timestamp, tokens_in, tokens_out, gpu_sec, bytes_in, bytes_out, price, chain_id, route_id, nonce, node_pubkey, sig_node, policy_hash)

- receipt_id: Unique receipt identifier, generated by hash(route_id || node_id || counter || timestamp)
- project_id: Project identifier, linked to revenue sharing rules
- model_id / version: Model identifier and semantic version number
- timestamp: Request completion timestamp (Unix milliseconds)
- tokens_in / tokens_out: Input and output token counts
- gpu_sec: GPU usage time (precise to milliseconds, stored as floating-point seconds)
- price: Fee calculated per billing rules (minimum units)
- chain_id: Blockchain ID where user paid

6.1.2 Receipt Signature Verification

Receipt signatures use ECDSA secp256k1 algorithm:

$$msg = keccak256(receipt\_id \,||\, project\_id \,||\, \ldots \,||\, policy\_hash)$$

$$sig\_node = ECDSA\_sign(node\_privkey, msg)$$

Verification:

$$valid = ECDSA\_verify(node\_pubkey, msg, sig\_node)$$

$$registered = NodeRegistry.isRegistered(node\_pubkey)$$

6.2 Billing Functions

AIL2 supports flexible hybrid billing modes, allowing projects to choose the most suitable billing strategy based on model characteristics.

6.2.1 Basic Billing Formulas

Token billing (for language models):

$$Fee\_token = a_0 + a_1 \cdot (tokens\_in / 1000) + a_2 \cdot (tokens\_out / 1000)$$

Time billing (for image/video models):

$$Fee\_time = b_0 + b_1 \cdot gpu\_sec$$

Hybrid billing (take maximum):

$$Fee\_raw = max(Fee\_token, Fee\_time)$$

6.2.2 Fee Boundary Protection

To prevent extreme fees from abnormal situations, boundary protection is introduced:

$$Fee\_final = min(max(Fee\_raw, Fee\_min), Fee\_cap)$$

6.2.3 Dynamic Pricing

Dynamic pricing adjustments when supply-demand imbalances:

$$Price\_multiplier = 1 + \eta \cdot (Demand / Supply - 1) \quad if\ Demand > Supply$$

$$Price\_multiplier = 1 - \eta \cdot (1 - Demand / Supply) \cdot (1 - floor\_ratio) \quad if\ Demand < Supply$$

6.2.4 Discount Mechanisms

Volume discount:

$$Discount\_volume = min(\delta\_max, \delta\_base \cdot log(1 + Volume\_30d / V\_scale))$$

Final fee:

$$Fee\_discounted = Fee\_final \cdot (1 - Discount\_volume) \cdot (1 - Discount\_loyalty)$$

6.3 Batch Settlement Mechanism

To reduce on-chain transaction costs, AIL2 uses batch settlement mode. Multiple receipts are aggregated into one batch for unified settlement submission.

6.3.1 Merkle Tree Construction

Receipt hash calculation:

$$leaf\_i = keccak256(serialize(receipt\_i))$$

Merkle Root calculation:

$$node\_\{i,j\} = keccak256(node\_\{i-1, 2j\} \,||\, node\_\{i-1, 2j+1\})$$

$$root = node\_\{log_2(n), 0\}$$

6.3.2 Batch State Machine

Batch settlement follows this state flow:

- COLLECTING → PROPOSED: Collection complete, settlement proposal generated
- PROPOSED → CHALLENGE: Proposal published, challenge window opens

- CHALLENGE → FINALIZED: Challenge window ends, no valid disputes
- FINALIZED → PAID: Revenue distribution executed, funds arrive

6.4 Cross-Chain Settlement Modes

AIL2 supports two cross-chain settlement modes, selectable based on specific needs:

6.4.1 Mode A: Multi-Chain Local Settlement + Unified Reconciliation

Independent SettlementAdapter contracts deployed on each chain. Users pay on their native chain, settlement completes locally.

$$GlobalRoot = keccak256(Root\_ETH \,||\, Root\_BSC \,||\, Root\_Base \,||\, \ldots)$$

6.4.2 Mode B: Main Settlement Chain + Cross-Chain Accounting

One chain selected as main settlement chain. Other chains serve only as payment entry points, receipts aggregated to main chain via cross-chain bridge.

$$Message = (source\_chain, batch\_root, total\_amount, timestamp, signatures[\,])$$

6.5 Dispute Resolution Mechanism

AIL2 has established comprehensive dispute resolution mechanisms to protect user and service provider rights.

Type	Description	Evidence Required
Metering Dispute	Abnormal token or gpu_sec metering	Receipt comparison + log hash
Result Dispute	Inference result incorrect or unavailable	Input/output records + spot-check results
Double Billing	Same request charged multiple times	route_id + nonce proof
Refund Fraud	User maliciously requests refund	Service completion proof + signature

6.6 Idempotency & Replay Protection

Settlement system must guarantee idempotency, preventing duplicate payments and replay attacks:

$$receipt\_id = hash(route\_id \,||\, node\_id \,||\, counter \,||\, timestamp)$$

$$if\ receipt\_id \in ProcessedSet\ then\ reject()$$

Chapter 7 | Launchpad: IMO / IAO

7.1 IMO (Initial Model Offering)

7.1.1 IMO Objectives

IMO is a one-stop fundraising and issuance mechanism designed by AIL2 for model developers, with core objectives including:

- Funding: Providing financing tools for model project R&D and growth
- Supply: Guiding GPU nodes to join and provide stable service for projects
- Standards: Providing an integrated foundation for billing, revenue sharing, risk control, and dashboards

7.1.2 IMO Full Process

1. Create Project → 2. Create Token → 3. Publish Container → 4. IMO Fundraising → 5. Incentive Bootstrap → 6. Operations Dashboard

7.1.3 IMO Contract Modules

- ProjectRegistry: Project registration and permission management
- ModelRegistry: Model version and policy registration
- TokenFactory: One-click token factory
- OfferingContract: Fundraising and refund logic
- VestingContract: Release and lock-up management
- TreasuryVault: Project treasury (multi-sig)
- MilestoneOracle: Milestone unlock trigger
- RevenueSplit: Revenue distribution executor
- MiningIncentives: Mining incentive distribution

7.2 IAO (Initial Agent Offering)

7.2.1 IAO Objectives

IAO targets non-technical creators, aiming to enable non-technical users to publish AI Agent applications and issue tokens, standardizing Agent workflows, permissions, billing, incentives, and growth components.

7.2.2 IAO Full Process

1. Select Template → 2. Combine Workflow → 3. Set Permissions → 4. Set Billing → 5. IAO Issuance → 6. Growth Components

7.3 Milestone Unlock Mechanism

To protect investor interests, IMO/IAO token releases are tied to project milestones:

- Technical Milestone: Container online + health check passed

- Stability Milestone: N nodes running continuously for T days
- Adoption Milestone: Daily call volume X or monthly revenue Y
- Risk Milestone: Dispute rate/refund rate below threshold

Chapter 8 | Security, Anti-Cheating & Verifiable Inference

8.1 Threat Model

AIL2 network faces multiple security threats. We use systematic threat modeling methodology for analysis. According to research by Liu et al. published in Automation in Construction [12], blockchain-AI integration frameworks can effectively address cybersecurity risks including AI data tampering, lack of transparency in training and usage, and single points of failure.

8.1.1 Attacker Classification

Attacker Type	Capability	Motivation	Risk Level
Malicious Miner	Controls single or few nodes	Illicit gains	High
Volume Farmer	Creates multiple accounts	Token rewards	High
Sybil Attacker	Creates many fake identities	Control network decisions	High
Arbitrageur	Cross-chain operation capability	Exploit price differences	Medium
External Attacker	Network attack capability	Disrupt system or ransom	Medium

8.1.2 Attack Vector Analysis

Volume Inflation Attack

Attackers inflate call volume through fake requests to gain mining rewards.

$$Profit\_attacker = Reward(fake\_volume) - Cost(fake\_requests) - Stake\_at\_risk$$

Defense: Multi-dimensional effective volume calculation making Cost(fake_requests) > Reward(fake_volume).

Sybil Attack

Attackers create many fake identities to control the network.

$$Control\_fraction = Sybil\_nodes / (Sybil\_nodes + Honest\_nodes)$$

Defense: Staking requirements making Sybil cost proportional to control ratio:

$$Cost\_sybil = Stake\_per\_node \cdot Sybil\_nodes$$

8.2 Risk Scoring System

AIL2 establishes a real-time risk scoring system, synthesizing multiple signals to assess participant risk levels:

$$Risk = \sigma(\Sigma_i\ a_i \cdot z(feature_i))$$

Where σ is sigmoid function, a_i is feature weight, z is z-score normalization.

8.2.1 Risk Level Classification

Risk Score	Level	Handling Strategy
0-0.2	Low Risk	Normal service
0.2-0.5	Medium Risk	Increased verification, reduced weight
0.5-0.8	High Risk	Limited functionality, manual review
0.8-1.0	Critical Risk	Service suspended, investigation initiated

8.3 Penalty & Demotion Mechanism

High-risk behaviors trigger penalties; penalty mechanisms follow progressive principles:

$$Penalty = \exp(-\xi \cdot violations)$$

Violation records decay over time, allowing correction opportunities:

$$violations\_effective(t) = \Sigma_i \; violations\_i \cdot \exp(-(t - t_i) / \tau\_decay)$$

8.4 Slashing Mechanism

Slashing is economic punishment for severe violations, directly deducting staked tokens.

Violation	Evidence Required	Slash Ratio
Forged Receipt	Signature verification failure	50-100%
Replay Attack	Nonce duplication proof	50-100%
Double Billing	Receipt comparison	30-50%
Spot-Check Failure	Result inconsistency	10-30%
Malicious Image	Security audit report	100% + permanent ban

$$SlashAmount = \min(Stake \cdot SlashRate, MaxSlash)$$

$$SlashRate = BaseRate \cdot SeverityMultiplier \cdot RepeatMultiplier$$

8.5 Challenge Window & Arbitration

Challenge window is the time period allowing objections to settlement proposals.

$$ChallengeWindow = BaseWindow + AdditionalTime(BatchSize, TotalValue)$$

Arbitration committee composed of randomly selected stakers:

$$Committee = RandomSelect(Stakers, CommitteeSize, weight=Stake)$$

$$Decision = majority\_vote(Committee)$$

8.6 Verifiable Inference Roadmap

AIL2 plans phased implementation of verifiable inference, progressively increasing security guarantees:

8.6.1 v1: Receipts + Random Spot-Checking

$$P\_verify = \min(p_0 + k \cdot \log(1 + fee / fee\_unit), p\_max)$$

8.6.2 v2: Redundant Execution Consistency Comparison

$$Result\_final = MajorityVote(Result\_1, Result\_2, \ldots, Result\_n)$$

8.6.3 v3: Zero-Knowledge Proofs / Trusted Execution Environments

$$Proof = ZK\_Prove(Input, Output, Model\_commitment)$$

$$Valid = ZK\_Verify(Proof, Input\_hash, Output\_hash, Model\_commitment)$$

Chapter 9 | Governance & Roadmap

9.1 Governance Scope

AIL2's on-chain governance covers the following key decisions:

- Fee rate and revenue sharing parameter adjustments
- Treasury budget allocation
- Security audits and protocol upgrades
- New chain expansion plans
- Incentive parameters and risk control thresholds
- Emergency response and vulnerability fixes

9.2 Governance Structure

9.2.1 Community Governance

AIL2 token stakers participate in routine governance decisions through on-chain voting. Voting weight correlates with staking amount and duration, encouraging long-term holding and deep participation.

9.2.2 Security Council

Security Council is controlled by multi-sig, responsible for emergency vulnerability response and attack handling. Council members are elected by the community with term limits and can be recalled.

9.2.3 Parameter Committee

Parameter Committee is responsible for professional analysis and proposals on technical parameters like routing and incentives. Committee proposals require community vote approval to take effect.

9.3 Three-Year Roadmap

Year 1: Infrastructure & MVP

- Routing/Scheduling/Receipt/Settlement system launch
- Core SDK release
- IMO/IAO MVP launch
- SLA/Reputation system v1
- First 3 chains deployed

Year 2: Feature Enhancement & Scale Expansion

- Dispute arbitration automation
- Subscription and credits system
- Full chain coverage and stable settlement
- Redundant verification launch
- All 6+ chains deployed

Year 3: Ecosystem Deepening & Standard Setting

- Verifiable inference optional layer
- AI micropayment standard proposal
- Ecosystem deep expansion
- Full decentralized governance transfer
- Cross-protocol interoperability standards

Conclusion

AIL2 integrates distributed GPU computing networks, cross-chain settlement, and incentive issuance tools into AI-native infrastructure, enabling developers to deploy model containers to global GPU networks without building their own networks, rapidly building, scaling, and monetizing decentralized AI applications, and achieving sustainable growth through AIL2's main token and project token systems.

We believe that decentralized AI infrastructure will become one of the most important infrastructures of the Web3 era. AIL2 is committed to becoming the standard setter and ecosystem leader in this field, creating value for global developers and users.

Thank you for reading this whitepaper. For more information, please visit our official website or join community discussions.

References

[1] Sun, R., et al. (2024). "SoK: Decentralized AI (DeAI)." arXiv preprint arXiv:2411.17461. https://arxiv.org/abs/2411.17461

[2] Talaei Khoei, T., et al. (2024). "Blockchain for secure and decentralized artificial intelligence in cybersecurity: A comprehensive review." Blockchain: Research and Applications, 5(1), 100006. https://doi.org/10.1016/j.bcra.2024.100006

[3] Konstantinidis, I., et al. (2024). "The Convergence of Artificial Intelligence and Blockchain: The State of Play and the Road Ahead." Information, 15(5), 268. https://doi.org/10.3390/info15050268

[4] Mafrur, R., et al. (2025). "AI-Based Crypto Tokens: The Illusion of Decentralized AI?" IET Blockchain. https://doi.org/10.1049/blc2.70015

[5] Gürpinar, T. (2025). "Towards web 4.0: frameworks for autonomous AI agents and decentralized enterprise coordination." Frontiers in Blockchain, 8, 1591907. https://doi.org/10.3389/fbloc.2025.1591907

[6] Gao, W., et al. (2024). "Deep Learning Workload Scheduling in GPU Datacenters: A Survey." ACM Computing Surveys, 56(6), 1-47. https://doi.org/10.1145/3638757

[7] Fu, B., Cai, D., & Zhai, E. (2024). "Crux: GPU-Efficient Communication Scheduling for Deep Learning Training." In Proceedings of ACM SIGCOMM 2024. https://doi.org/10.1145/3651890.3672268

[8] Sharma, A., et al. (2024). "GPU Cluster Scheduling for Network-Sensitive Deep Learning." arXiv preprint arXiv:2401.16492. https://arxiv.org/abs/2401.16492

[9] Lai, T. L., & Robbins, H. (1985). "Asymptotically efficient adaptive allocation rules." Advances in Applied Mathematics, 6(1), 4-22. https://doi.org/10.1016/0196-8858(85)90002-8

[10] Auer, P., Cesa-Bianchi, N., & Fischer, P. (2002). "Finite-time Analysis of the Multiarmed Bandit Problem." Machine Learning, 47(2-3), 235-256. https://doi.org/10.1023/A:1013689704352

[11] Wang, Y., et al. (2024). "GPARS: Graph predictive algorithm for efficient resource scheduling in heterogeneous GPU clusters." Future Generation Computer Systems, 152, 361-371. https://doi.org/10.1016/j.future.2023.11.008

[12] Liu, F., et al. (2024). "Decentralized artificial intelligence in construction using blockchain." Automation in Construction, 166, 105614. https://doi.org/10.1016/j.autcon.2024.105614

[13] Kumar, S., et al. (2024). "A Systematic Review of Blockchain Technology Assisted with Artificial Intelligence Technology for Networks and Communication Systems." Journal of Computer Networks and Communications, 2024, 9979371. https://doi.org/10.1155/2024/9979371

[14] Nartey, J. (2024). "Decentralized Finance (DeFi) and AI: Innovations at the Intersection of Blockchain and Artificial Intelligence." SSRN Electronic Journal. https://doi.org/10.2139/ssrn.4781328

[15] Adimora, K., et al. (2023). "GPU Job Scheduling based on Deep Reinforcement Learning." In Proceedings of HP3C 2023. https://doi.org/10.1145/3606043.3606049

[16] El-Sayed, M., et al. (2025). "A Survey of Advancements in Scheduling Techniques for Efficient Deep Learning Computations on GPUs." Electronics, 14(5), 1048. https://doi.org/10.3390/electronics14051048

Appendix

Appendix A: Glossary

Term	Definition
AI L2	Blockchain Layer 2 network optimized for AI application scenarios
GPU Worker	Miner node providing computing power
IMO	Initial Model Offering, first token issuance for AI models
IAO	Initial Agent Offering, first token issuance for AI agents
Slashing	Stake confiscation penalty mechanism
SLA	Service Level Agreement
UCB	Upper Confidence Bound algorithm
Merkle Tree	Hash tree for data integrity verification
Sybil Attack	Attack creating many fake identities
DeAI	Decentralized Artificial Intelligence

Appendix B: Mathematical Notation

Symbol	Meaning
λ	Request arrival rate / decay coefficient
μ	Service rate / quality weight
ρ	System utilization
θ	Anti-monopoly factor
κ	Anti-cheating penalty intensity
$\alpha, \beta, \gamma, \delta$	Revenue sharing ratio parameters
$\sigma()$	Sigmoid function
$z()$	Z-score normalization function
UCB(n)	Upper confidence bound value for node n
W_q	Queue waiting time