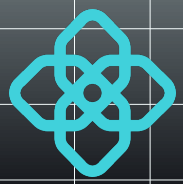

OpenGradient

Decentralized Infrastructure for Verifiable AI Execution

Whitepaper | March 2026

https://opengradient.ai
https://github.com/OpenGradient
https://docs.opengradient.ai

Abstract

Artificial intelligence infrastructure is rapidly consolidating into a handful of centralized providers, creating systemic risks of opacity, single points of failure, and unverifiable computation. When AI systems make decisions affecting financial markets, healthcare, governance, and critical infrastructure, there exists no mechanism to verify which model executed, what inputs were processed, or whether outputs were tampered with.

OpenGradient introduces a **vertically-integrated decentralized infrastructure stack** purpose-built for secure, verifiable AI execution. At its core is the **Hybrid AI Compute Architecture (HACA)** — a novel network design that separates execution from verification, enabling web2-like latency with blockchain-grade trust guarantees.

The architecture supports three verification methods spanning a trust spectrum: **Trusted Execution Environments (TEE)** for negligible-overhead hardware attestation, **Zero-Knowledge Machine Learning (ZKML)** for cryptographic mathematical proofs, and **Vanilla** mode for performance-critical workloads. This enables developers to select verification levels matching their risk profiles.

Built on CometBFT consensus with EVM compatibility, the OpenGradient network provides: payment-gated LLM inference via the **x402 protocol**, a decentralized **Model Hub** on Walrus storage, persistent AI memory through **MemSync**, on-chain ML execution via the **PIPE** engine, and a digital twins marketplace (**Twin.fun**).

The network currently hosts over 2,000 models, serves 100+ developers, and has processed over 1 million inferences on testnet.

Contents

1 Introduction **5**

2 Hybrid AI Compute Architecture (HACA) **6**

2.1 Why Traditional Blockchains Fail for AI 6

2.2 The HACA Solution: Separation of Execution and Verification 6

3 Node Architecture **7**

3.1 Full Nodes (Blockchain Validators) 8

3.2 Inference Nodes (GPU Workers) 8

3.2.1 LLM Proxy Nodes (TEE Enclaves) 8

3.2.2 Local Inference Nodes (Direct GPU Execution) 9

3.3 Data Nodes (Coming Soon) 9

3.4 Storage: Walrus Decentralized Storage 9

4 Verification Spectrum **9**

4.1 TEE Verification 10

4.1.1 TEE Node Registration 10

4.2 ZKML Verification 11

4.3 Vanilla Verification 12

5 Consensus and Settlement **12**

5.1 CometBFT Consensus 12

5.2 Technology Stack: Cosmos SDK + EVM 12

5.3 Settlement Flow 13

6 x402: Payment-Gated LLM Inference **13**

6.1 Overview 13

6.2 Payment Flow 13

6.3 Settlement Modes 14

6.4 Supported Models 14

7 PIPE: On-Chain ML Execution **14**

7.1 Parallelized Inference Pre-Execution Engine 14

7.2 Benefits 15

8 Products and Interfaces **15**
8.1 Model Hub: Decentralized Model Repository 15
8.2 MemSync: Long-Term AI Memory 16
8.3 Twin.fun: Digital Twins Marketplace 17
8.3.1 Economics 17
8.4 AlphaSense: Verifiable AI Workflows 18
8.5 Python SDK and CLI 18

9 Use Cases **19**
9.1 Verifiable AI Agents 19
9.2 Privacy-Preserving AI 19
9.3 DeFi and Financial Applications 19
9.4 Personalized AI with Persistent Memory 19
9.5 Decentralized Model Hosting 20

10 Design Principles and Trade-Offs **20**
10.1 Core Design Principles 20
10.2 Intentional Trade-Offs 20

11 Network Status and Deployments **20**
11.1 Current Metrics 21
11.2 Testnet Deployment 21
11.3 Payment Infrastructure 21

12 Conclusion **21**

1 Introduction

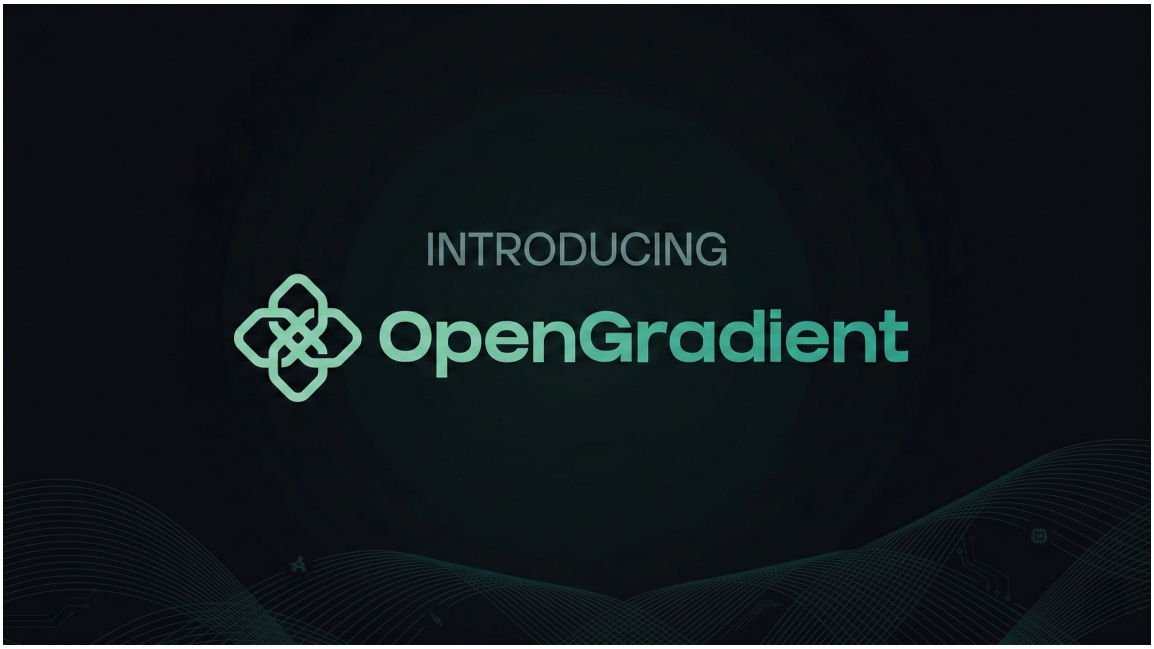

Figure 1: OpenGradient: Decentralized infrastructure for the AI economy.

The current AI landscape faces a fundamental tension. As artificial intelligence increasingly drives critical decisions — from autonomous trading to medical diagnostics to content moderation — the infrastructure delivering these capabilities grows ever more centralized and opaque. A handful of providers (OpenAI, Anthropic, Google, xAI) control the vast majority of AI inference, creating an ecosystem where:

- **Verification is impossible.** When an AI agent moves money, approves a transaction, or makes a healthcare recommendation, there is no way for external parties to verify which model version ran, what system prompt was used, or whether the output was silently modified.
- **Single points of failure abound.** Rate limits, outages, or unannounced model behavior changes can break mission-critical applications with no fallback path.
- **Privacy is assumed, not guaranteed.** Providers can log, analyze, and monetize prompts without users' knowledge. Operators can silently swap models, inject content, or censor outputs.
- **Lock-in deepens.** Proprietary APIs, non-standard interfaces, and opaque pricing create switching costs that compound over time.

OpenGradient addresses this by building an infrastructure layer where every AI computation can be *cryptographically verified* without trusting any single party. Models run on a permissionless network of specialized nodes, proofs are settled on-chain, and the entire pipeline is auditable — while inference remains fast enough for production workloads.

2 Hybrid AI Compute Architecture (HACA)

2.1 Why Traditional Blockchains Fail for AI

Conventional blockchains achieve consensus through **re-execution**: every validator independently runs every transaction and checks that all arrive at the same state. This model fails for AI inference for three fundamental reasons:

1. **Cost.** Running a 70-billion parameter LLM 100 times (once per validator) costs 100× more than a single execution, with zero additional value.
2. **Non-determinism.** LLMs with temperature > 0, models with dropout layers, and floating-point differences across hardware architectures all produce different outputs for identical inputs. Validators cannot simply compare results.
3. **Latency.** A single LLM inference takes seconds. If every validator must complete inference before the block advances, block times become impractical for any real-world application.

2.2 The HACA Solution: Separation of Execution and Verification

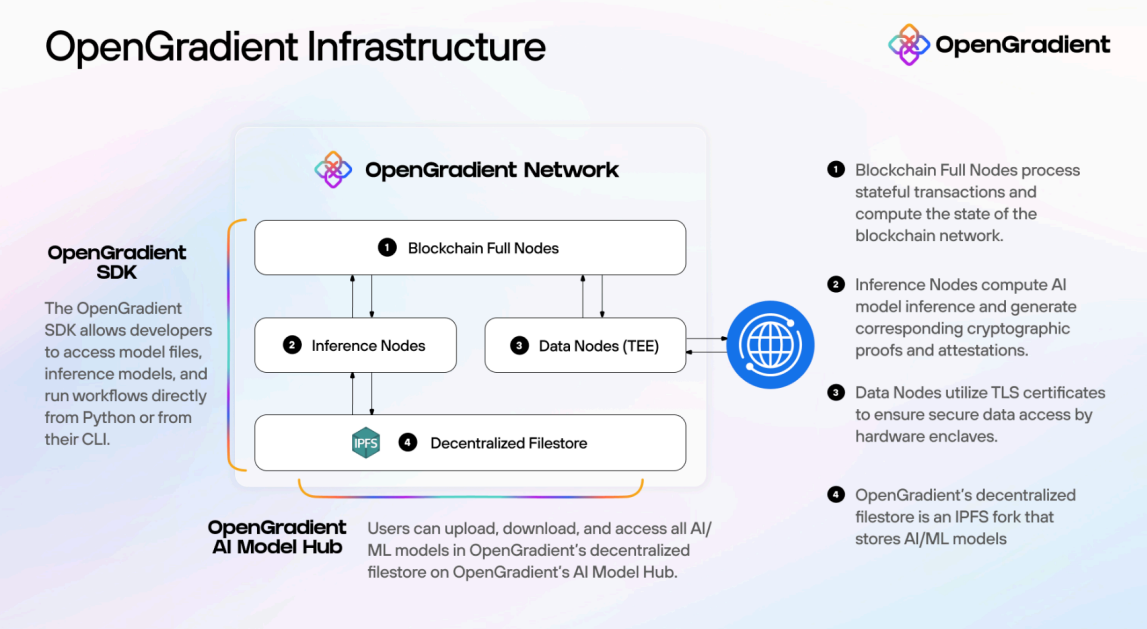

Figure 2: The Hybrid AI Compute Architecture separates execution from verification across specialized node types.

HACA's core principle is that **execution and verification are independent operations happening on separate timelines**:

Fast Path (milliseconds)	Verification Path (async)
User submits request	Proof/attestation generated
Inference node executes model	Full node verifies proof
Result returned immediately	Settlement recorded on ledger

This separation creates four critical properties:

- **Scalability.** Adding inference nodes increases throughput linearly without loading the verification layer.
- **Hardware heterogeneity.** Inference nodes require GPUs; full nodes operate on commodity hardware, improving decentralization.
- **Low latency.** Users receive responses immediately; settlement occurs asynchronously in the background.
- **Privacy preservation.** Full nodes verify proofs without accessing prompts, responses, or model weights.

3 Node Architecture

OpenGradient's network comprises specialized node types, each optimized for its role in the verification pipeline.

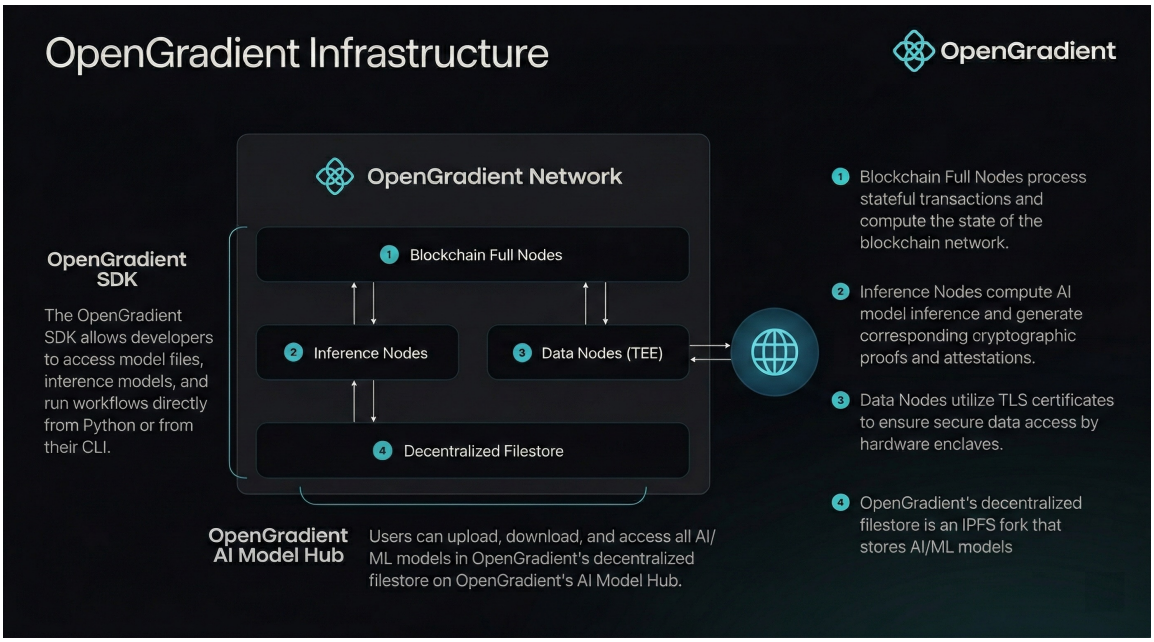

Figure 3: Infrastructure overview: specialized nodes collaborate to deliver verifiable AI inference.

3.1 Full Nodes (Blockchain Validators)

Full nodes maintain the OpenGradient ledger and form the consensus backbone of the network. Their responsibilities include:

- Running CometBFT consensus to produce and validate blocks
- Verifying all TEE attestations and ZKML proofs
- Managing payment settlement and fee distribution
- Maintaining the on-chain TEE node registry
- Performing node registration verification against approved code hashes

Full nodes operate on **commodity hardware** — no GPUs required. They never execute models or access user data directly, ensuring the verification layer remains lightweight and highly decentralized.

3.2 Inference Nodes (GPU Workers)

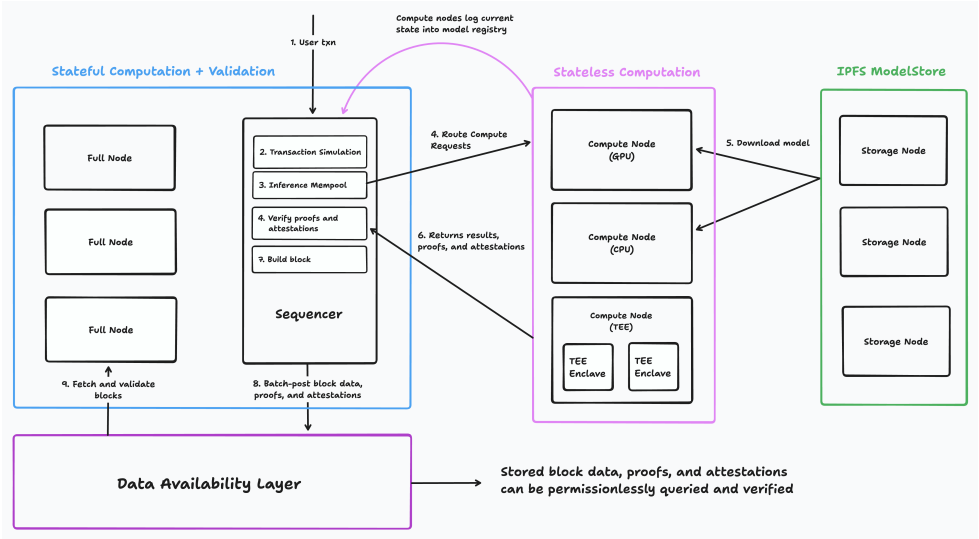

Figure 4: Inference node architecture: stateless GPU workers executing models and returning results directly to users.

Inference nodes are stateless workers that execute models and return results directly to users. They come in two types:

3.2.1 LLM Proxy Nodes (TEE Enclaves)

These nodes route requests to third-party LLM providers (OpenAI, Anthropic, Google, xAI) through **Trusted Execution Environments**. The TEE enclave guarantees:

- **Privacy:** Node operators cannot see, log, or manipulate request data.
- **Anonymity:** Requests are distributed across nodes and cannot be tied back to individual identities.
- **Integrity:** TEE attestations prove the enclave ran approved, untampered code.

3.2.2 Local Inference Nodes (Direct GPU Execution)

These nodes run open-source models directly on local GPU hardware. They download models from the Model Hub as needed and support all three verification methods (ZKML, TEE, and Vanilla). This type is ideal for custom fine-tuned models and workloads requiring full model control.

3.3 Data Nodes (Coming Soon)

Data nodes are TEE-secured nodes that fetch external data from APIs, databases, and oracles. They generate attestations proving data authenticity and enable multi-source aggregation within secure enclaves — ensuring the data pipeline is as verifiable as the inference pipeline.

3.4 Storage: Walrus Decentralized Storage

OpenGradient uses **Walrus**, a decentralized storage network, as its persistence layer:

- **Models** are stored on Walrus with content-addressed Blob IDs, making them permanently available and censorship-resistant.
- **Large ZKML proofs** are stored on Walrus with blob ID references recorded on-chain, keeping the blockchain lean.
- On-chain references enable verification while full data remains accessible on the decentralized storage layer.

4 Verification Spectrum

Not all AI inference requires the same level of trust. HACA supports a **spectrum of verification methods**, allowing developers to choose the trust level that matches their risk profile.

Method	Mechanism	Overhead	Best For
ZKML	Zero-knowledge proof	1000–10000×	High-stakes ML where cryptographic certainty is required
TEE	Hardware attestation	Negligible	LLM inference, privacy-sensitive workloads, production use
Vanilla	Signature only	None	Low-risk workloads, prototyping, non-critical inference

A key design choice: **different inferences within the same transaction can use different verification methods** — for example, TEE for LLM reasoning, ZKML for a risk model, and Vanilla for analytics, all within a single atomic operation.

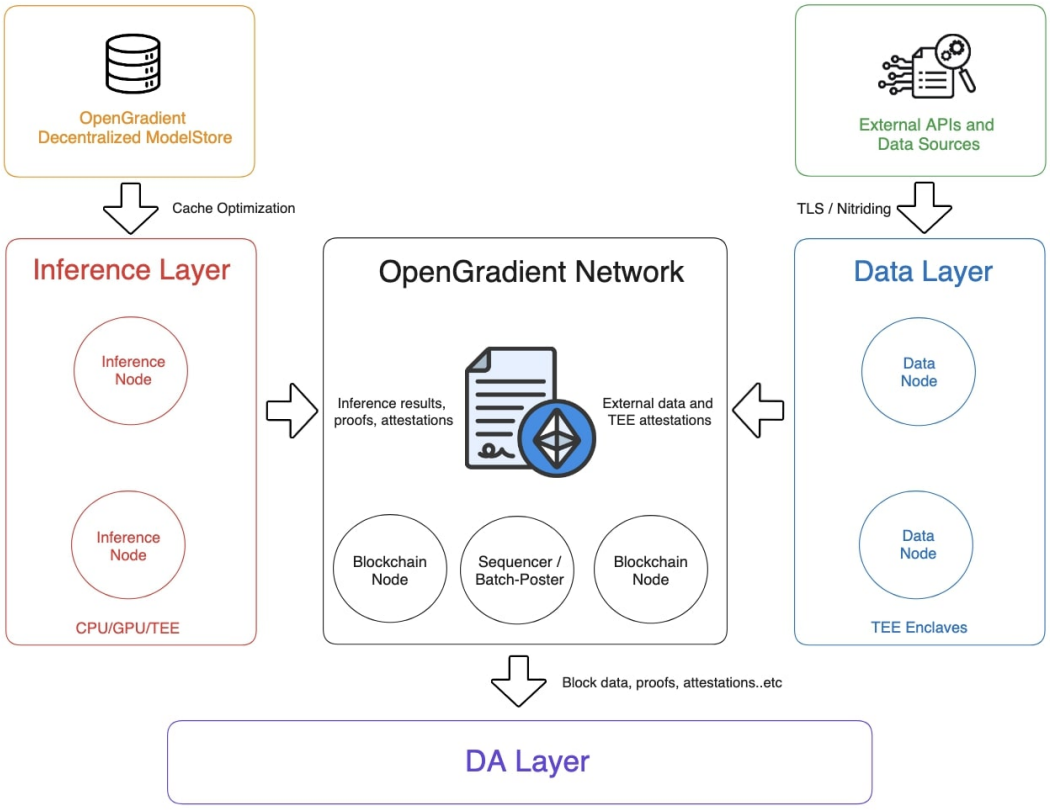

Figure 5: The verification spectrum: from mathematical certainty (ZKML) to hardware attestation (TEE) to lightweight verification (Vanilla).

4.1 TEE Verification

Trusted Execution Environments provide **hardware-level isolation** that guarantees code integrity and data confidentiality. The TEE verification flow for LLM inference:

1. An LLM Proxy Node receives a request inside its TEE enclave (AWS Nitro).
2. The enclave routes the request to the third-party LLM API.
3. The TEE generates a hardware attestation proving:
   - The enclave was running approved, untampered code (verified via PCR values)
   - The prompt was forwarded unmodified
   - The response was returned without tampering
4. The attestation is submitted to full nodes for on-chain verification and permanent recording.

4.1.1 TEE Node Registration

Before serving requests, every TEE node must register on-chain via the `ITEERegistry` smart contract. Registration includes the raw AWS Nitro attestation document, RSA signing public key, TLS certificate, payment address, and endpoint information.

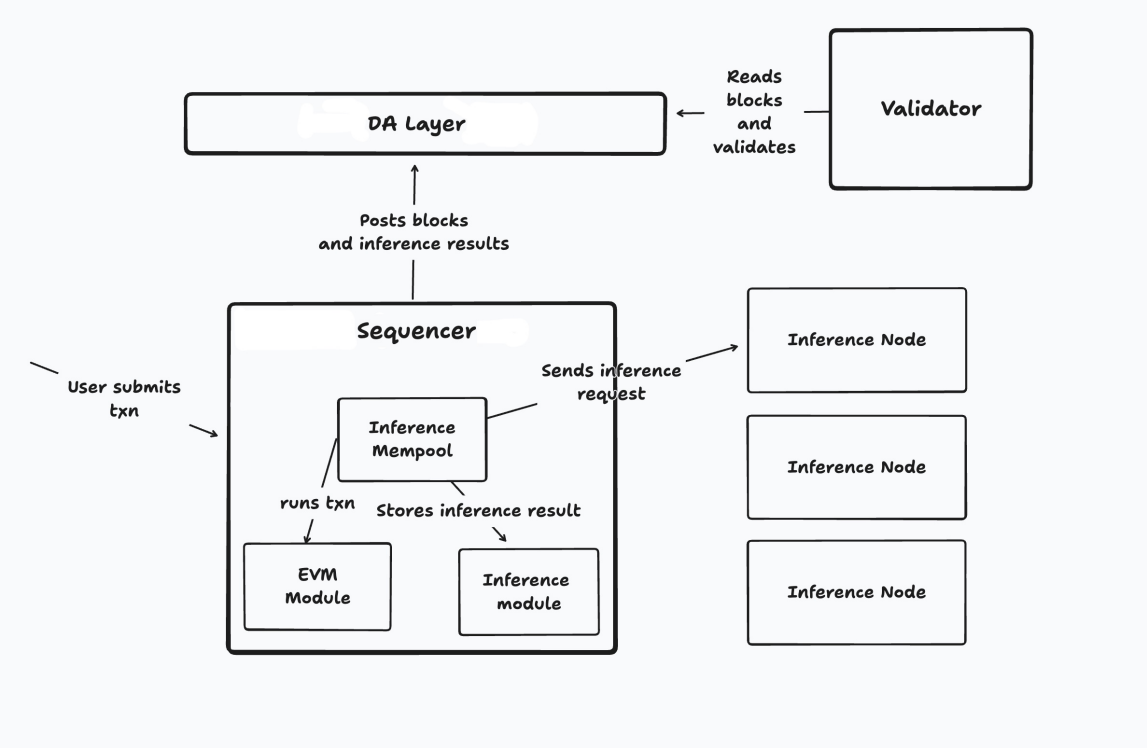

Figure 6: TEE verification flow: hardware attestation proves enclave integrity without re-executing inference.

On-chain verification checks include:

- Attestation authenticity against the AWS root certificate
- PCR values (PCR0, PCR1, PCR2) matched against on-chain approved code hashes
- TLS certificate binding: `SHA256(cert_public_key)` matches the hash in the attestation's `user_data`
- Signing key binding for settlement transaction verification

Users establish secure connections by downloading the TLS certificate from the on-chain registry — trust flows directly from **AWS Nitro hardware attestation → on-chain registry → TLS connection**, with no external certificate authorities required.

4.2 ZKML Verification

Zero-Knowledge Machine Learning provides the **strongest possible verification guarantee**: a mathematical proof that a specific model produced a specific output for a given input, without revealing model weights or input data to validators.

- The inference node runs the model and generates a zero-knowledge proof.
- The proof mathematically demonstrates correct execution.
- Full nodes verify the proof using purely cryptographic operations — no model re-execution required.

- **Trade-off:** Highest guarantees (mathematical certainty) but highest overhead (1000–10000×), currently best suited for smaller, high-stakes ML models.

4.3 Vanilla Verification

Vanilla mode provides **signature verification only** — no proof of correct execution. It is suitable for:

- Low-risk, non-critical inference workloads
- Prototyping and development environments
- Large generative AI models where performance is prioritized
- Scenarios where trust in the inference node is acceptable

5 Consensus and Settlement

5.1 CometBFT Consensus

OpenGradient uses **CometBFT** (formerly Tendermint) for its consensus layer, providing:

- **Instant finality:** Blocks are final once committed — no confirmation waiting required.
- **Byzantine fault tolerance:** The network remains secure with fewer than $\frac{1}{3}$ malicious validators.
- **Deterministic execution:** All validators verify the same proofs in the same order.
- **Efficient verification:** Validators verify proofs without re-executing inference.

Parameter	Value
Block Time	10 seconds
Finality	Instant
Consensus Threshold	$\geq \frac{2}{3}$ validators
Fault Tolerance	$< \frac{1}{3}$ Byzantine

5.2 Technology Stack: Cosmos SDK + EVM

The network is built on the **Cosmos SDK** with full EVM compatibility:

- **Cosmos SDK** provides modular architecture for AI-native custom modules, flexible state machines for AI operations, and interoperability via IBC (Inter-Blockchain Communication).
- **EVM compatibility** ensures familiar tooling (Hardhat, Foundry, ethers.js, MetaMask), smart contract integration via precompiles, and access to the broader DeFi ecosystem.

5.3 Settlement Flow

After inference completes, the settlement flow proceeds as follows:

1. **Inference completes:** The inference node returns the result to the user immediately (fast path).
2. **Proof submission:** TEE attestation or ZKML proof is submitted to the network.
3. **Payment settlement:** Settled on the OpenGradient chain (or Base Sepolia for x402 payments).
4. **Block proposal:** A validator includes the proof in its block proposal.
5. **Consensus verification:** Validators verify the proof during the consensus round ($\geq \frac{2}{3}$ agreement required).
6. **Finalization:** The proof is permanently and immutably recorded on the ledger.

6 x402: Payment-Gated LLM Inference

6.1 Overview

x402 is an open, neutral standard for internet-native payments. OpenGradient uses x402 to create a **payment-gated, verifiable LLM inference API** that operates over standard HTTP/REST:

- **Universal access:** Works from any programming language via standard HTTP.
- **Payment-gated:** Cryptographically-verified $OPG token payments before execution.
- **Verifiable:** Every inference produces a TEE attestation.
- **Decoupled chains:** Payment settles on Base Sepolia; execution and proof settlement on the OpenGradient network.

6.2 Payment Flow

1. **Permit2 Approval** (one-time): Client approves $OPG token allowance for Permit2 on Base Sepolia.
2. **Initial Request:** Client sends an LLM request to the x402 gateway.
3. **402 Response:** Server responds with payment details (amount, chain ID, payment ID, expiry).
4. **Payment Signing:** Client creates and signs the payment payload with their wallet.
5. **Resubmission:** Client resubmits the request with the payment signature in the `X-PAYMENT` header.
6. **Verification:** The facilitator contract verifies the payment signature on-chain.
7. **Execution:** The TEE node executes inference and returns the response with a verification proof.

8. **Settlement:** Payment settles on Base Sepolia; proof settles on the OpenGradient blockchain.

6.3 Settlement Modes

Clients choose how much data to record on-chain:

Mode	On-Chain Data	Best For
`SETTLE_INDIVIDUAL`	Input/output hashes	Agent reasoning affecting financial decisions
`SETTLE_BATCH`	Batch hashes	Secure chatbots, high-throughput apps
`SETTLE_INDIVIDUAL_WITH_METADATA`	Full metadata	Publicly auditable DeFi agents

6.4 Supported Models

The x402 gateway provides TEE-verified access to leading LLM providers:

Provider	Models
OpenAI	GPT-4.1, GPT-4o, o4-mini
Anthropic	Claude 4.0 Sonnet, Claude 3.7 Sonnet, Claude 3.5 Haiku
Google	Gemini 2.5 Flash/Pro/Flash-Lite, Gemini 2.0 Flash
xAI	Grok 3 Beta/Mini, Grok 4.1 Fast, Grok 2 Vision

7 PIPE: On-Chain ML Execution

7.1 Parallelized Inference Pre-Execution Engine

PIPE (Parallelized Inference Pre-Execution Engine) enables ML models to be called natively from blockchain applications with atomic execution guarantees:

1. User submits a transaction containing an inference request.
2. The transaction enters the **Inference Mempool**.
3. The mempool simulates all pending transactions and extracts inference requests.
4. Inference requests are dispatched to the inference network for **parallel execution**.
5. Once results are available, the original transaction resumes with pre-computed inference results.
6. The completed transaction is included in the next block.

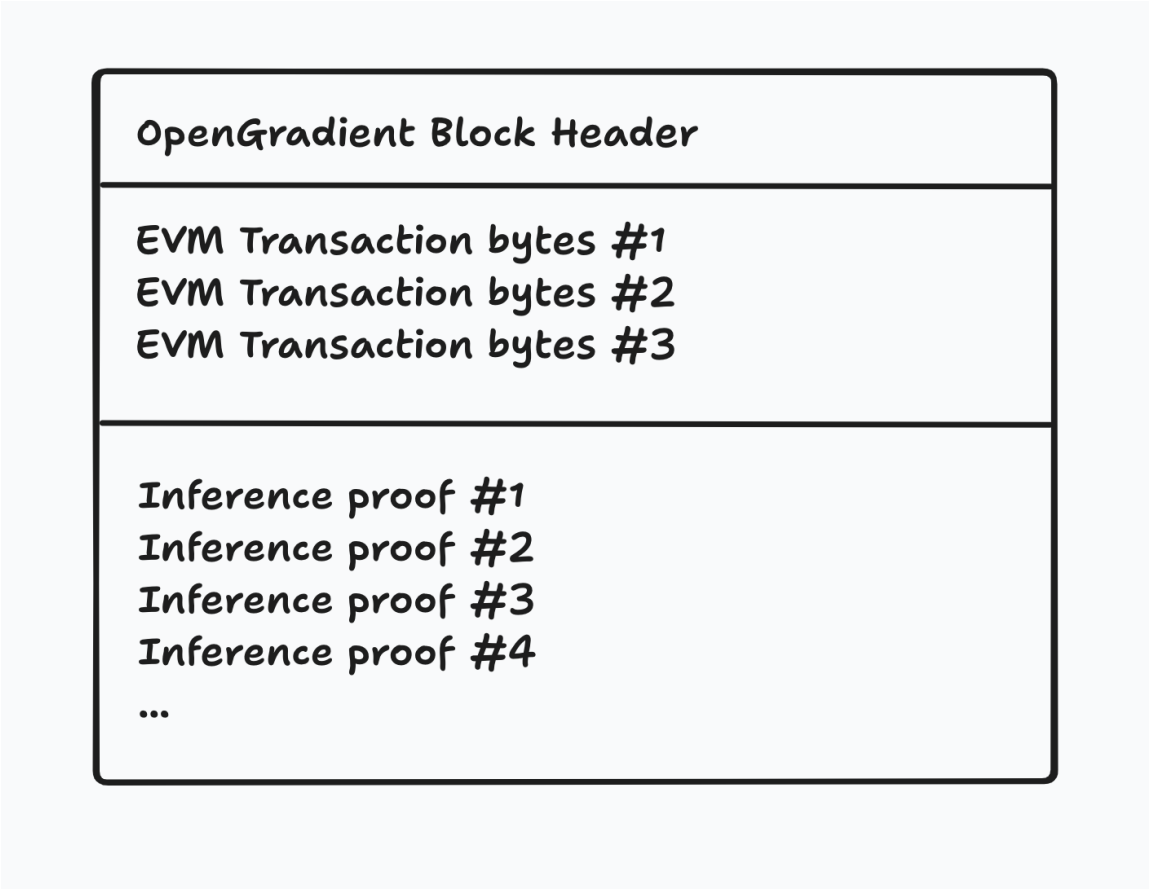

Figure 7: PIPE: the inference mempool enables parallel ML execution with atomic on-chain guarantees.

7.2 Benefits

- **Horizontal scalability:** Inferences for hundreds of pending transactions execute in parallel.
- **No bottleneck:** Expensive ML computations don't slow the blockchain.
- **Atomic execution:** Inference results are part of the same transaction — no oracle delay.
- **Fast block building:** Actual block construction remains extremely fast since inference is pre-computed.

8 Products and Interfaces

8.1 Model Hub: Decentralized Model Repository

The OpenGradient Model Hub is a **permissionless, decentralized model repository** built on Walrus storage:

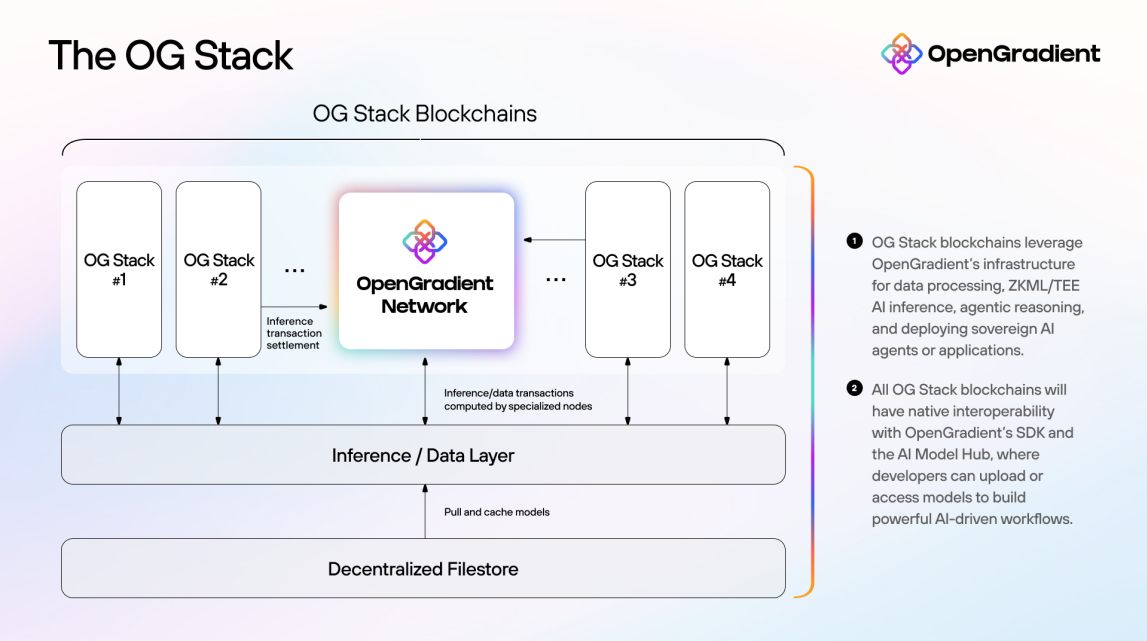

Figure 8: The OpenGradient product stack: from infrastructure to developer tools to end-user applications.

- **Permanent storage:** Models are content-addressed and cannot be censored or lost.
- **Versioning:** Repositories support semantic versioning (major.minor) with independent Blob IDs per version.
- **Inference-ready:** ONNX models are immediately available for verified on-chain inference.
- **Discovery:** Full-text search with filters by task, tags, author, and organization.
- **Interactive playground:** Test any model directly in the browser, with execution recorded on-chain.
- **Web3-native:** Wallet sign-in, content-addressed identifiers, community collaboration features.

The Model Hub supports any model format for storage, with ONNX format required for on-chain inference and ZKML support constrained by EZKL library compatibility (opset 9–18).

8.2 MemSync: Long-Term AI Memory

MemSync is a long-term memory layer for AI, built entirely on OpenGradient's verifiable inference infrastructure. All memory operations — extraction, classification, profile generation, and maintenance — are powered by TEE-verified LLM inference.

- **Automatic extraction:** Memories are extracted from conversations, documents, websites, and social profiles using verified LLM calls.
- **Semantic classification:** Memories are classified as *semantic* (lasting facts: "Software engineer at Google") or *episodic* (time-bound events: "Currently working on iOS app").

- **User profiles:** Auto-generated bios and insights, continuously updated.
- **Semantic search:** Query memories using natural language with embedding-based similarity.
- **REST API:** Simple integration into any application via standard HTTP endpoints.

8.3 Twin.fun: Digital Twins Marketplace

Twin.fun is a marketplace for AI-powered digital twins — agents modeled after real people or personas, each with a key market on a deterministic bonding curve.

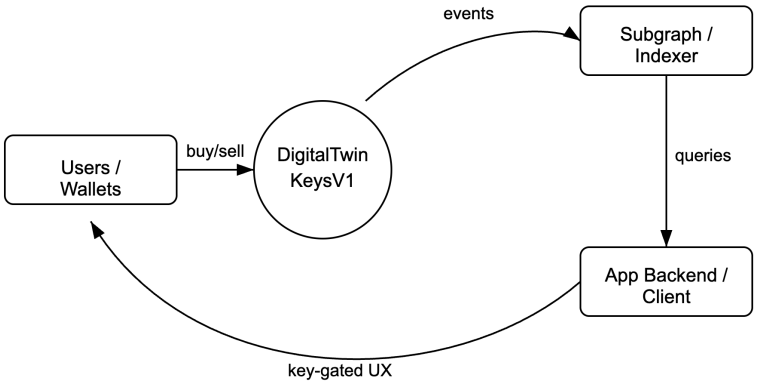

Figure 9: Twin.fun architecture: digital twins with bonding curve economics.

8.3.1 Economics

Each digital twin is referenced on-chain by a 16-byte ID (`bytes16`). Users acquire and redeem **keys** (shares) via a quadratic bonding curve:

$$\text{price}(s, a) = \frac{\sum_{i=s}^{s+a-1} i^2}{1{,}600{,}000} \text{ ETH} \tag{1}$$

where s is the current supply and a is the number of keys being purchased. Fees are assessed on every trade:

- **Protocol fee** → treasury address
- **Subject fee** → twin owner/creator

Holding ≥ 1 key unlocks gated experiences: chat, tools, and utilities powered by the twin's AI agent.

8.4 AlphaSense: Verifiable AI Workflows

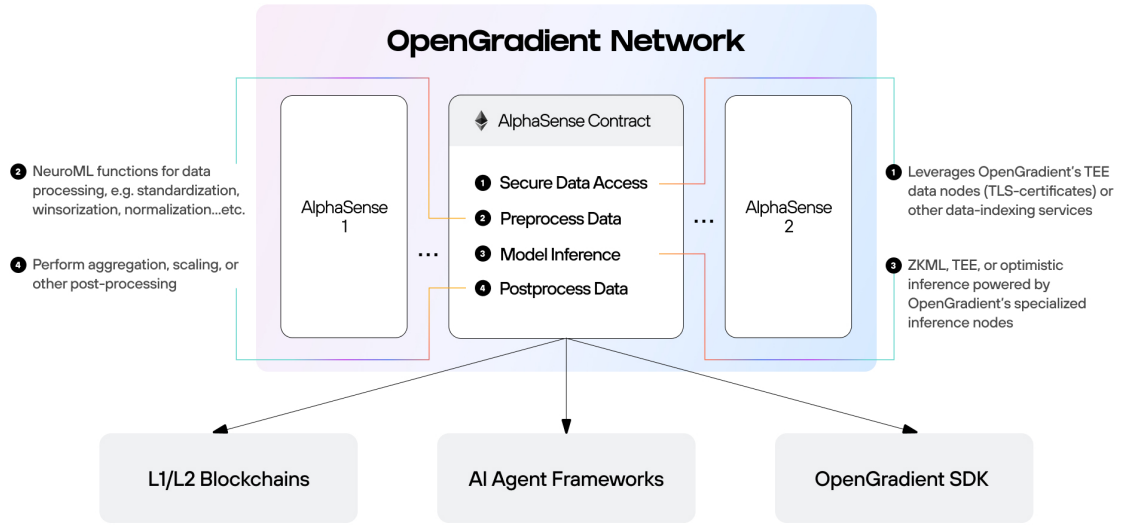

Figure 10: AlphaSense provides verifiable AI-powered financial signals and workflows.

AlphaSense wraps verifiable AI workflows that empower agents and applications with powerful signals:

- **Volatility AlphaSense:** Continuous volatility forecasts for risk management, AMM fee scaling, and LTV adjustment.
- **PriceForecast AlphaSense:** Spot returns forecasts using time-series ML models.
- **Sybil AlphaSense:** Wallet address analysis for Sybil account detection.
- **Markowitz AlphaSense:** Optimal portfolio positions via mean-variance optimization.

8.5 Python SDK and CLI

The OpenGradient Python SDK (`pip install opengradient`) is the primary developer interface:

```
import opengradient as og

client = og.Client(private_key="0x...")

# Verifiable LLM inference
result = client.llm.chat(
    model=og.TEE_LLM.GPT_4_1_2025_04_14,
    messages=[{"role": "user", "content": "Hello!"}]
)
print(result.chat_output["content"])
print(result.payment_hash)  # on-chain record
```

```
# ML inference with ZKML verification
result = client.alpha.infer(
    model_cid="<blob_id>",
    model_input={"input1": [1.0, 2.0, 3.0]},
    inference_mode=og.InferenceMode.ZKML
)
print(result.transaction_hash)
```

9 Use Cases

OpenGradient enables a new category of applications where AI decisions are auditable, verifiable, and decentralized.

9.1 Verifiable AI Agents

Autonomous AI agents where every LLM call is cryptographically signed with the exact prompt used. When an agent moves money, approves transactions, or makes decisions, anyone can verify the complete reasoning chain on-chain — providing audit trails for regulatory compliance and dispute resolution.

9.2 Privacy-Preserving AI

TEE nodes process prompts inside hardware enclaves where the node operator cannot see, log, or manipulate requests. This enables sensitive workloads: medical reasoning, financial analysis, private conversations, and enterprise applications requiring data confidentiality.

9.3 DeFi and Financial Applications

- **Dynamic fee optimization:** AMMs automatically adjust fees based on ML volatility predictions.
- **Smart risk assessment:** Lending protocols recalculate risk scores using verified ML models with real-time price feeds.
- **Portfolio management agents:** Autonomous portfolio managers with cryptographic proof of their decision-making process.
- **Trading bots:** Intelligent trading with verifiable model inference and full audit trails.

9.4 Personalized AI with Persistent Memory

MemSync enables AI applications that remember users across sessions, with the entire memory pipeline running on verified infrastructure. Use cases include personalized chatbots, educational AI tutors that track student progress, healthcare AI maintaining patient context, and CRM systems that remember interactions across touchpoints.

9.5 Decentralized Model Hosting

Upload any model to the Model Hub for permissionless access. Any model on Walrus is immediately available for inference on the network — no gatekeepers, no approval process, no vendor lock-in.

10 Design Principles and Trade-Offs

10.1 Core Design Principles

1. **Separation of concerns.** Execution and verification operate on separate timelines with specialized hardware, enabling independent scaling.
2. **Verification without re-execution.** TEE attestations and ZKML proofs eliminate the 100× compute waste of traditional blockchain consensus for AI workloads.
3. **Pragmatic trust spectrum.** Different applications have different trust requirements; forcing a single verification level would either over-tax low-risk workloads or under-serve high-stakes ones.
4. **Composability.** Multiple verification methods can be combined within a single transaction, enabling fine-grained trust optimization.

10.2 Intentional Trade-Offs

- **TEE relies on hardware trust.** If a fundamental TEE vulnerability were discovered, security would degrade. This is mitigated by supporting multiple verification methods and allowing ZKML requirements for critical operations.
- **ZKML is slow.** The 1000–10000× overhead makes ZKML impractical for large models. This is a limitation of current ZK technology that will improve as proof systems mature.
- **Asynchronous settlement creates temporary trust gaps.** Between result return and proof settlement, the result is not yet verified on-chain. For operations requiring immediate verification, PIPE provides atomic execution at the cost of higher latency.
- **Specialization adds coordination complexity.** Multiple node types require registration, routing, and coordination mechanisms more complex than a homogeneous validator set — but this is worthwhile because it enables capabilities impossible with uniform designs.

11 Network Status and Deployments

11.1 Current Metrics

Metric	Value
Models hosted	2,000+
Active developers	100+
Total inferences	1,000,000+

11.2 Testnet Deployment

Property	Value
Network Name	OpenGradient Testnet
RPC URL	`https://ogevmdevnet.opengradient.ai`
Chain ID	10740
Native Currency	ETH
Block Explorer	`https://explorer.opengradient.ai`
Faucet	`https://faucet.opengradient.ai`

11.3 Payment Infrastructure

x402 payments settle on **Base Sepolia** (Chain ID: 84532) using the $OPG token. The OpenGradient Facilitator contract is deployed at:

`0x339c7de83d1a62edafbaac186382ee76584d294f`

12 Conclusion

OpenGradient represents a fundamental shift in how AI infrastructure is delivered. By separating execution from verification and specializing nodes for their respective tasks, it achieves what was previously thought impossible: **web2-like performance with blockchain-grade trust**.

The architecture is not a one-size-fits-all solution but rather a **spectrum of capabilities** — from fast Vanilla inference for non-critical workloads to cryptographically-certain ZKML for high-stakes decisions. This pragmatic approach allows developers to choose trust levels matching their risk profiles.

With products spanning verifiable LLM inference (x402), decentralized model hosting (Model Hub), long-term AI memory (MemSync), on-chain ML execution (PIPE), digital twin markets (Twin.fun), and verifiable AI workflows (AlphaSense), OpenGradient provides

the complete stack needed to build a new class of AI applications: **verifiable, decentralized, and user-controlled**.

---

https://opengradient.ai | https://docs.opengradient.ai | https://github.com/OpenGradient