MODELOS LABS

modelOS

A Verifiable Proof-of-Useful-Work Layer 1 for Decentralised AI Inference

Technical Whitepaper

Version 1.0 · 2026

modeloslab.xyz

Abstract

modelOS is a proof-of-work Layer 1 blockchain in which the act of mining *is* real artificial-intelligence computation. Built on the zero-knowledge Proof-of-Useful-Work foundation introduced by the Pearl protocol, modelOS extends that foundation into a complete decentralised AI infrastructure layer: a live on-chain inference marketplace, Mixture-of-Experts proof circuits for frontier-scale models, a Zcash-inspired privacy architecture, a Turing-complete smart-contract language with native AI operations, an on-chain agentic runtime, a community-owned model trained by the network itself, and a consumer-facing application suite. The network's native asset, MDL, is earned exclusively by performing genuine AI inference: every coin in circulation is a cryptographic certificate that intelligence was produced. This paper describes the protocol's consensus mechanism, difficulty-adjustment algorithm, transaction model, inference marketplace, privacy stack, smart-contract environment, agentic layer, and first-party applications, and situates modelOS relative to existing decentralised-compute and AI-incentive networks.

Keywords: proof-of-useful-work, zero-knowledge proofs, verifiable inference, Mixture-of-Experts, decentralised AI, smart contracts, on-chain agents, shielded transactions

Contents

1 Introduction

Bitcoin changed the world by turning electricity into money. It demonstrated that energy expenditure, when it is scarce and independently verifiable, can serve as the foundation of a credible, decentralised currency. But Bitcoin's underlying operation, random SHA-256 hashing, was always a means to an end: the computation itself produces nothing beyond a number below a target threshold, and once a block is found, the work that found it is discarded.

The world now runs on a very different kind of computation. Artificial intelligence, and specifically the matrix multiplications that power every modern neural network, consumes a rapidly growing share of global electricity. Unlike Bitcoin hashing, AI inference produces something of genuine economic value: reasoning, answers, analysis, and creative output. The energy spent is not wasted; it becomes intelligence.

Pearl established a foundational insight: matrix multiplication can replace hashing as proof-of-work. Every block certificate on Pearl is a zero-knowledge proof that a real AI computation occurred, so the security budget of the network is useful compute rather than waste.

modelOS builds the complete infrastructure layer on top of that foundation. If the GPU securing the network is already running AI inference, the network itself should be the marketplace, the contract environment, and the privacy layer for that inference — verifiably, permissionlessly, and without a trusted intermediary.

> **modelOS is the first Layer 1 blockchain where every block is a zero-knowledge proof of real AI work, where any developer can write smart contracts that think, and where intelligence is the product rather than a feature.**

1.1 What modelOS Adds to Pearl

Pearl solves the mining problem; modelOS solves the infrastructure problem. The additions modelOS makes on top of Pearl are deliberate and non-overlapping, summarised in Table 1.

2 How modelOS Works --- The Big Picture

Before turning to technical detail, it is useful to view the system at a high level. modelOS is organised into three interlocking layers, each building on the one below (Figure 1).

Layer 1 — The Chain. The foundation is a proof-of-work blockchain. Miners run GPUs, and instead of performing arbitrary hashing, those GPUs execute real matrix multiplications — the core operation of neural-network inference. A zero-knowledge proof is attached to each block, proving the computation occurred correctly. The chain is merge-mined with Pearl, so miners earn both MDL and Pearl rewards from a single GPU run.

Layer 2 — The Market. The same GPUs that mine the chain serve AI inference requests. When a user wants to query a language model, they broadcast a transaction locking an MDL fee. Miners compete to serve the request and submit a proof of correct inference; the first valid proof claims the fee. Payment, work, and settlement all happen on-chain, with no API provider and no trusted

Capability	What it does
Verifiable inference market	Native transaction types turn the chain into a trustless compute marketplace. Users pay for inference; miners serve and prove; payment settles on-chain without a custodian.
Frontier-scale model support	Mixture-of-Experts (MoE) proof circuits extend verifiable inference from dense 70B models to the 100B–700B frontier class that defines the current state of the art.
Privacy layer	A Zcash-inspired shielded pool protects transfers, inference requests, model weights, fine-tuning data, and DeFi activity from public exposure.
Programmable AI contracts	Nova Script is a Solidity-like smart-contract language with `INFER`, `EMBED`, and `CLASSIFY` as native operations; contracts call AI models the same way they call any other function.
On-chain agentic infrastructure	Autonomous agents are Nova Script contracts with identity, on-chain memory, and MDL-funded operation. The network itself is the agent runtime.
Community-owned intelligence	The network trains, owns, and governs its own AI model — the first model created entirely through decentralised distilled learning from miner inference activity.

Table 1: *Capabilities modelOS adds on top of the Pearl protocol.*

intermediary.

Layer 3 — The Application Layer. On top of the inference market sits a complete application environment: Nova Script smart contracts with AI-native operations, an on-chain agentic layer, a privacy stack, and consumer-facing tools. Developers build *cognitive contracts* — programs that call AI models and act on the results, entirely on-chain.

The resulting economic loop is self-reinforcing: mining rewards attract GPU compute; more compute deepens the inference market; a deeper market attracts developers and users; developer and user activity generates inference fees; and inference fees flow back to miners. Every participant's incentive points in the same direction.

3 Proof-of-Useful-Work

3.1 The Core Idea

Traditional proof-of-work asks miners to find an input whose hash falls below a target. The operation is simple to verify and expensive to compute, but the computation itself produces nothing useful — once a block is won, the hashing work is thrown away.

modelOS, inheriting Pearl's protocol, replaces hashing with matrix multiplication: the same dense linear algebra that powers every modern AI model. Mining a block means producing a Plonky2 zero-knowledge proof that a large matrix multiplication was performed correctly, with a result that clears the difficulty target. The consequences of this design choice are significant:

- **Mining hardware is AI hardware.** The hardware that mines best is the hardware that runs AI

Figure 1: *The three layers of modelOS. Mining secures the chain (Layer 1); the same GPUs serve the inference market (Layer 2); developers and users build and interact on top (Layer 3). Activity at Layer 3 generates the fees that reward Layer 1 mining, closing the loop described below.*

best — GPUs are the optimal miners precisely because they are optimised for matrix operations, the same reason they run AI.

- **No wasted energy.** Miners are not burning energy on waste: the computation they perform is the same one users pay for in the inference marketplace. Mining and serving inference are the same operation.
- **Confidential computation.** Because the proof is zero-knowledge, miners can compute on private data — model weights, user prompts — without revealing those inputs to the network.

3.2 How It Works

A miner works with two matrices, A and B. The protocol adds carefully structured low-rank noise to both, $A' = A + N_A$ and $B' = B + N_B$, then requires the miner to compute the product $A' \cdot B'$ in a specific tile-by-tile order. At each tile, a running hash state is updated; if the final hash state clears the difficulty threshold, the miner has found a valid block.

The noise serves a dual purpose. It ensures that every pair of matrices is equally hard to multiply, preventing miners from gaming the system with trivial inputs, and it can be algebraically peeled away at the end to recover the useful product $A \cdot B$. The miner therefore produces something genuinely valuable — a real matrix-multiplication result — as a direct by-product of mining.

The zero-knowledge proof attached to the block is compact (under 60 KB after three rounds of recursive compression), verifies in milliseconds, and requires no trusted setup. It is post-quantum secure, relying only on the hardness of cryptographic hash functions.

3.3 Mixture-of-Experts Extension

Modern frontier AI models use a Mixture-of-Experts (MoE) architecture: rather than activating all parameters for every token, a routing function selects only the most relevant subset of specialised sub-networks. A model with 671 billion total parameters might activate only 37 billion per token, delivering frontier-scale intelligence at a fraction of the per-token compute cost.

The ZK-PoW circuit in modelOS includes a dedicated MoE proof type. It commits the routing

decision on-chain, enforces that the correct number of experts were activated, and verifies only the computation that actually occurred. This lets the network prove that a 671B MoE model ran correctly on a given prompt at a verification cost proportional to the *active* parameters, not the total model size. Section 8 discusses the proof system in detail.

3.4 Merge Mining with Pearl

modelOS uses AuxPoW (auxiliary proof-of-work) to merge-mine with the Pearl network. A single ZK proof can simultaneously satisfy the difficulty targets of both chains, so one unit of GPU work earns both MDL and Pearl rewards. Miners pay no additional compute cost to participate in both networks, and modelOS inherits security from Pearl's established miner base from the moment of launch.

4 Difficulty Adjustment --- Colossus 2.0

modelOS targets a 194-second (approximately 3 minute 14 second) average block time. Maintaining that target requires a difficulty-adjustment algorithm that responds quickly to changes in mining activity without overreacting and causing oscillations.

From block 10,000 onward, difficulty is governed by **Colossus 2.0**, which implements the absolute ASERT algorithm (`aserti3-2d`) used by Bitcoin Cash and Nexa. The algorithm fixes a single anchor block at activation and computes every subsequent block's difficulty target relative to that fixed reference point — never relative to the previous block's target:

$$\text{target}(h) \;=\; \text{target}_{\text{anchor}} \cdot 2^{\dfrac{(t_h - t_{\text{anchor}}) \;-\; \tau\,(h - h_{\text{anchor}} + 1)}{\lambda}}$$

where $\tau = 194$ s is the target block time, λ is the difficulty half-life, and h_{anchor}, t_{anchor}, $\text{target}_{\text{anchor}}$ are the height, timestamp, and target of the fixed anchor block.

This design has three important properties.

1. **Drift-free.** Errors do not accumulate over time, because the reference point never moves.
2. **Non-oscillating.** A sudden increase or decrease in mining activity is smoothly corrected over a half-life λ of approximately eight hours, without the overcorrection spikes that plagued earlier designs.
3. **Deterministic.** The calculation uses only integer arithmetic — no floating-point operations appear anywhere in the difficulty calculation, so every node on the network computes an identical result.

> **Why replace Colossus 1.0?** The previous algorithm anchored its exponent to the running difficulty target, which caused errors to compound without bound. Simulations showed oscillations exceeding 1,000% under a sudden change in hashrate, even with a median filter and a per-block cap in place. Absolute ASERT eliminates this failure mode by design.

5 Transactions

modelOS uses a UTXO (Unspent Transaction Output) ledger, the same accounting model as Bitcoin: coins exist as discrete, spendable outputs rather than account balances. Transactions consume existing outputs and create new ones, with the difference paid to the miner as a fee, and every transaction is signed with the sender's private key.

Three transaction versions are defined, summarised in Table 2.

Version	Name	What it does
v1	**Standard transfer**	Moves MDL between addresses. Functionally identical to a Bitcoin transaction.
v3	**Inference Request**	Locks an MDL fee on-chain and broadcasts an AI job to the inference market. The output script encodes everything the network needs to enforce payment rules, with no external lookup required.
v4	**Inference Proof**	Carries the miner's cryptographic proof of correct inference and claims the locked fee. The script releases payment only if the proof is valid and binds to the exact request.

Table 2: *Transaction versions defined by the modelOS protocol.*

Requesting AI inference on modelOS is as simple as sending a transaction: no API key, no account, and no trusted provider. The chain enforces everything.

6 The Inference Marketplace

modelOS turns the blockchain itself into a trustless AI compute market. Any user can pay for language-model inference, and any GPU miner can earn that fee by serving the request and proving correctness — an open, competitive market for AI compute with cryptographic guarantees that the work was actually done.

6.1 How a Request Flows

The lifecycle of an inference request has four steps, illustrated in Figure 2.

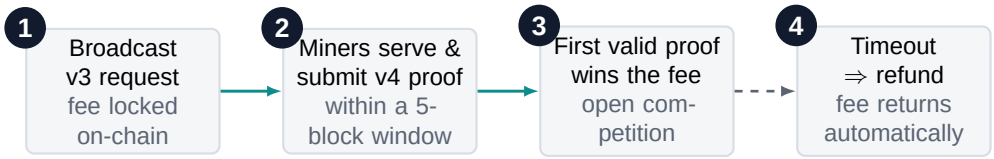

Figure 2: *Lifecycle of an on-chain inference request. If no valid proof arrives within the confirmation window, the locked fee is returned to the sender automatically.*

1. The user broadcasts a v3 transaction that locks an MDL fee under a script encoding the model tier, a commitment to the prompt, and the user's public key. The fee is now visible to all miners but unspendable until a valid proof arrives.

2. Miners serving the specified model run inference on the prompt and broadcast a v4 transaction carrying the cryptographic proof. The proof is bound to the exact request — the specific prompt, model, and nonce — making it impossible to recycle from a previous job. A five-block window (roughly sixteen minutes) begins at confirmation.

3. The first valid proof confirmed on-chain wins the fee. The competition is open to any miner serving the right model, not only the block producer, which keeps pricing competitive and prevents extraction.

4. If no valid proof arrives within the window, the fee unlocks and returns to the sender automatically. Funds are never permanently locked by a failed or ignored request.

6.2 Inference Futures

Users can pre-purchase inference capacity at a fixed MDL price for future use, similar to reserving GPU time in advance. These futures are settled as on-chain covenants and are redeemable as standard inference requests at any future block. For users, futures provide price certainty; for miners, they provide revenue predictability.

6.3 Private Inference

Some inference requests require complete confidentiality — the prompt may contain sensitive data that should never appear on a public ledger. For these cases, modelOS supports *shielded inference*: the prompt and response are encrypted end-to-end between user and miner, and only the MDL payment amount and the zero-knowledge proof of correct inference are published on-chain. The network records that valid AI work occurred and was paid for, without learning what was asked or answered.

7 Model Hosting and Support

7.1 Consensus Model Tiers

Supported models are represented on-chain as numbered tiers (Table 3). The model assigned to each tier is updated over time to reflect the current best open-weight reasoners, without changing the transaction interface: developers write against tier numbers, and the network improves the underlying models as better ones are released.

Tier	Size class	Current model	Base fee
v4	8B	DeepSeek R1 0528 (8B)	0.005 MDL
v3	14B	Qwen3 14B	0.020 MDL
v2	32B	Qwen3 32B	0.050 MDL
v1	70B	DeepSeek R1 (70B)	0.100 MDL

Table 3: *Consensus model tiers and indicative base fees. Actual market prices are set by supply and demand in the inference marketplace.*

Mixture-of-Experts tiers extend the ceiling to the frontier class: gpt-oss (OpenAI's open-weight MoE, 120B total / ~20B active per token), DeepSeek-V3 and DeepSeek-R1 (671B / ~37B active), Qwen3-MoE, Llama-4 MoE, and Mixtral. As new open frontier models are released, they are added as consensus tiers through network governance.

7.2 Permissionless Model Hosting

Not every model belongs in the consensus tier set. The network also supports any open-weight model through permissionless hosting: a host stakes MDL to register a model and pays miners directly, either per hour or in bulk, for the GPU capacity to serve it. Inference against hosted models uses the same v3/v4 transaction flow as consensus tiers, with the model identifier referenced in the request.

This opens the network to the full breadth of the open AI ecosystem — domain-specific fine-tunes, embedding models, code models, image-generation models such as Flux and Stable Diffusion, and any future open-weight release. Quality and uptime are enforced through staking: hosts who fail to maintain availability lose a portion of their stake.

7.3 LoRA Adapter Marketplace

Fine-tuned LoRA adapters extend the capability of base models for specific domains or tasks. The modelOS LoRA marketplace lets community contributors stake and publish adapters for any hosted base model; when a user attaches a LoRA identifier to an inference request, the serving miner applies the adapter at inference time.

Revenue sharing is automatic: every inference call through a LoRA adapter distributes a portion of the fee to the adapter's creator, defined in an on-chain revenue-share contract, so contributors are permanently rewarded for improvements that benefit the network.

7.4 The MDL-Native Model

The most significant long-term model initiative is the MDL-native model — the first AI model in history created, owned, and governed entirely by a decentralised network.

During normal inference activity, miners continuously generate input–output pairs across all hosted models. This stream of data, filtered and aggregated using privacy-preserving techniques, serves as the training signal for a distillation pipeline: the network trains progressively smaller, sharper models from larger ones, the same technique that has produced some of the most capable small models in the field.

The MDL-native model is governed by the community. Token holders vote on training-data standards, evaluation benchmarks, and when a new version is promoted to a consensus tier. No single entity controls it — every miner who has served an inference request has contributed to building it.

8 Mixture-of-Experts and Frontier Models

8.1 What Mixture-of-Experts Is

Most AI models are dense: every parameter is involved in computing every output. Scaling a dense model up means scaling compute up proportionally — a 70B model costs roughly ten times as much to run per token as a 7B model.

Mixture-of-Experts (MoE) breaks this linear relationship. An MoE model divides its feed-forward layers into many specialised expert networks and uses a lightweight router to select only the most relevant ones for each token. DeepSeek-V3, for example, has 671 billion total parameters but activates only around 37 billion per token: the model reasons at 671B quality while spending compute at roughly 37B cost. MoE is how the current generation of frontier models is built.

8.2 MoE in the Proof System

Supporting MoE in a verifiable inference system requires proving *sparse* routing, not just dense matrix multiplication. The modelOS proof circuit handles this natively: the MoE proof commits the routing decision on-chain, enforces that the correct number of experts were activated for each token, and verifies only the computation that actually occurred. Proof size scales with active parameters, not total parameters.

This means the network can prove that a 671B MoE model ran correctly on a given prompt at a cost proportional to the 37B parameters that actually fired, not the 671B that could have — frontier-scale intelligence at commodity verification cost.

8.3 What This Unlocks

- **A higher quality ceiling.** The supported model tier extends from dense 70B to the full frontier open-weight class. gpt-oss, DeepSeek-V3, DeepSeek-R1, Qwen3-MoE, and their successors become first-class inference targets on modelOS as they are released, so the network tracks the open ecosystem's quality curve rather than being frozen at a single model generation.
- **Broader miner participation.** Because only the active expert parameters are computed per token, a consumer-grade GPU — an RTX 4090 or 5090 — can serve a 671B MoE model at economically viable throughput. More eligible miners means a deeper, more competitive market and lower prices for users.
- **Trustless frontier inference.** Anyone can pay for a 671B-class reasoning answer and receive cryptographic proof that the real model ran on the real prompt — a guarantee that centralised API providers structurally cannot offer, since the modelOS chain enforces it by consensus.
- **Self-reinforcing compute value.** Every GPU-second simultaneously secures the chain and serves frontier inference. As MoE pushes served models toward the frontier, the economic value of network compute increases: the security budget and the product are the same resource.

9 Privacy Architecture

A blockchain is a public ledger: every transaction, every inference request, and every model interaction is visible to anyone who reads the chain. For many use cases — healthcare, legal, financial, competitive — this is unacceptable.

modelOS adds a comprehensive privacy layer modelled on Zcash's shielded-transaction architecture. The same cryptographic techniques that hide sender, receiver, and amount in Zcash's shielded pool are extended across every dimension of the modelOS stack: transfers, inference, model weights, fine-tuning, and DeFi.

9.1 The Shielded Pool

The foundation is a unified shielded pool. Private MDL exists as cryptographic notes — tuples of value, recipient key, and randomness — committed into an append-only Merkle tree. To spend a note, the holder produces a zero-knowledge proof of membership without revealing which note they own, and a nullifier prevents double-spending without identifying the note. Sender, receiver, and amount are all hidden.

Every other privacy feature in modelOS deposits into and spends from this shared pool, and users can move MDL between the transparent and shielded domains at any time.

9.2 Shielded Transfers and DeFi

Standard MDL payments can be routed through the shielded pool to hide amounts and counterparties. DeFi operations — token swaps, staking, liquidity provision — are also supported within the shielded environment, so on-chain financial activity does not expose trading strategies or positions.

9.3 ZK-Inference Proofs

Every inference request can optionally carry a zk-Inference proof: a compact zero-knowledge attestation that a specific model version ran on a specific input. The proof does not reveal the prompt or the response, only that the correct model was used and that the result is genuine. The model version is anchored to an on-chain commitment, making silent model substitution cryptographically impossible.

For regulated industries this is transformative: a healthcare application can prove that AI-assisted analysis occurred, on compliant infrastructure, using an approved model, without exposing any patient data. The proof *is* the compliance record.

9.4 zkML Layer

The zkML layer extends zero-knowledge proof generation to arbitrary model computations beyond the inference market. Applications can generate proofs of specific AI outputs for use in Nova Script contracts, compliance reporting, or cross-chain verification. zkML proofs are composable with the Nova Script contract environment described in Section 10.

9.5 Confidential Model Weights

Model owners who want to commercialise proprietary models without exposing their weights can host encrypted models on the network. Miners run inference inside a Trusted Execution Environment (TEE), where weights are decrypted and used without ever being accessible outside the secure boundary — not to the miner, not to the network. The inference result and the ZK proof exit the TEE; the weights never do.

9.6 Shielded Fine-Tuning

Fine-tuning a model on private data — patient records, financial history, proprietary corpora — has historically required trusting a third-party provider with that data. modelOS supports shielded fine-tuning through secure aggregation: miners compute and contribute gradient updates without ever accessing the raw training data. The data stays with the user; the model improves as if it had seen it.

10 Nova Script --- Smart Contracts for AI

Nova Script is modelOS's native smart-contract language. It borrows Solidity's familiar syntax — state variables, functions, events, modifiers — and adds one category of operations that Solidity does not have: native AI opcodes that call the inference market and return results to the contract.

A Nova Script contract does not merely compute. It can reason, classify, and generate, settling the AI work and the contract logic in the same atomic transaction.

10.1 AI-Native Opcodes

Opcode	What it does
`INFER(model_id, prompt)`	Calls the on-chain inference market for the specified model tier and returns the model's text response. The MDL fee is deducted from the contract's balance and paid to the winning miner automatically.
`EMBED(text)`	Generates a vector embedding of the input text using a hosted embedding model. The vector is returned to the contract for use in similarity search, clustering, or matching logic.
`CLASSIFY(input, labels)`	Performs zero-shot classification of the input against a list of candidate labels and returns the winning label for use in conditional logic and branching.

Table 4: *Native AI opcodes available to Nova Script contracts.*

Results from AI opcodes are first-class values in Nova Script: they can trigger transfers, update state, call other contracts, or feed into further AI operations. The contract does not trust the miner's output on faith — the v4 inference proof is verified by the script before any opcode returns a value.

10.2 Cognitive Swap --- The First Nova Script Contract

The first contract deployed at Nova Script launch is **Cognitive Swap**: the first token swap in blockchain history where the route and price are determined by an AI model running on-chain. The contract calls `INFER()` to assess current market conditions and determine the optimal swap route, and the model's output is verified by a zk-Inference proof before the swap executes.

Cognitive Swap is not merely a product feature — it is a proof of concept for the entire Nova Script paradigm: that contracts can make economically consequential decisions based on AI reasoning, entirely on-chain, without a trusted oracle or off-chain relay.

10.3 Contract Primitives

- **Reactive contracts.** Contracts that trigger automatically on the result of an inference call — for example, monitoring a data feed, running a sentiment model through `CLASSIFY()`, and executing a trade when sentiment crosses a threshold, with no off-chain component.
- **Model governance contracts.** DAO-controlled contracts through which MDL holders vote on consensus-tier updates, model safety ratings, pricing parameters, and when the MDL-native model's next distilled version is promoted to the network.
- **Inference escrow.** A contract holds MDL in escrow and releases payment to a miner only on receipt of a valid v4 proof matching the agreed model and prompt — the proof is the key, the contract is the lock, and there is no trusted escrow agent.
- **Revenue-share contracts.** Model owners define royalty splits at deployment, so every inference call against their model automatically distributes MDL across creator, miner, and treasury with no intermediary and no manual settlement.
- **Agent-as-a-contract.** An on-chain agent is a Nova Script contract with an MDL wallet, persistent contract-storage memory, and the ability to call `INFER()`, execute swaps, and manage funds autonomously. Described in detail in Section 11.

11 The On-Chain Agentic Layer

An AI agent is a system that perceives its environment, reasons about it, and takes action. On modelOS, agents are Nova Script contracts: they have an on-chain identity and address, they reason by calling `INFER()`, they remember by writing to contract storage, and they act by executing transactions. The chain is the runtime, and every action is permanent and auditable.

11.1 Agent Identity and the Registry

Every agent is a deployed Nova Script contract with a canonical MDL address. It publishes a capability manifest on-chain — a structured declaration of which models it uses, what task types it accepts, and what SLAs it commits to — and, to be listed in the agent registry, must stake MDL as a trust bond.

Interacting with an agent is as simple as sending it MDL: no API key, no permission, no account. Staked MDL is slashed for misbehaviour or missed SLAs, so agents are economically accountable for their outputs.

11.2 Contract-Storage Memory

Agents maintain persistent memory in Nova Script contract storage. Storage slots hold the agent's context — conversation history, task state, accumulated knowledge, and vector embeddings generated by `EMBED()` — and memory persists across sessions and contract calls without any off-chain server or database.

When an agent needs to retrieve a relevant memory, for example finding a past conversation related to a new request, it calls `EMBED()` on the query and performs a similarity lookup across vectors stored in its contract slots. The chain is the memory; retrieval is part of on-chain execution.

11.3 Multi-Agent Orchestration

Nova Script contracts coordinate multiple agents in structured workflows. The fundamental pattern is planner, executor, and verifier: one agent decomposes a complex task into steps, a second agent executes each step, and a third validates the results before payment is released. Each agent calls `INFER()` independently, and outputs flow between agents through contract function calls.

The full execution trace of a multi-agent workflow is recorded on-chain — every inference call, every agent invocation, and every state transition is visible and can be audited. There is no off-chain coordination layer, no trusted orchestrator, and no single point of failure.

11.4 The Bounty System

The bounty system is how work gets posted to the agent market. A user or contract publishes a task description, a deadline, and an MDL reward as a bounty contract. Agent contracts compete to complete the task, a verifier agent (or a ZK proof of task completion) validates the result, and the winning agent is paid automatically — no human arbiter, no trusted escrow.

The bounty system creates a permissionless market for cognitive work: any task that can be described and verified on-chain can be posted and fulfilled by autonomous agents operating entirely within the modelOS environment.

12 ModalCode and Modal Chat

Two first-party products make modelOS accessible to developers and end users without requiring knowledge of blockchain architecture.

12.1 ModalCode

ModalCode is a developer environment built specifically for the modelOS ecosystem. Every AI completion, suggestion, and tool call in ModalCode runs through MDL-hosted models on the modelOS inference network — the developer environment dogfoods the network it is designed to build

on.

- AI completions route through v3/v4 inference transactions: no external API, no API key, and usage that is transparent and priced in MDL.
- Native Nova Script support, including syntax highlighting, compilation, deployment, and on-chain interaction from a single interface.
- A model selector that lets developers choose any consensus tier or permissionlessly hosted model for completions, agent tasks, and inline queries.

ModalCode is the primary tool for writing Nova Script contracts, deploying on-chain agents, and interacting with the inference marketplace.

12.2 Modal Chat

Modal Chat is the consumer-facing interface to the modelOS network: a conversational AI assistant and direct alternative to ChatGPT, Claude, and similar products, powered entirely by open-source models hosted on modelOS. Nothing is routed through a centralised provider.

- Pay per message in MDL, or subscribe for a fixed monthly MDL allocation.
- Switch between model tiers — 8B, 14B, 32B, 70B, or MoE frontier — per conversation.
- No account required beyond a wallet; no data harvesting; no proprietary model dependency.

Modal Chat is the simplest way to experience what decentralised, verifiable AI looks like in practice, and the primary gateway for non-developer users to access the modelOS network.

13 EVM Bridge

modelOS is a UTXO-based Layer 1. To access the broader Ethereum and EVM developer ecosystem — its liquidity pools, lending protocols, and hundreds of millions of users — modelOS offers a bidirectional bridge to Ethereum, Base, and other EVM-compatible chains.

- MDL bridges to wMDL, an ERC-20 token, on EVM chains. wMDL is composable with any EVM DeFi protocol: AMMs, lending markets, liquid staking, and yield aggregators.
- The bridge is bidirectional: wMDL can be returned to native MDL at any time for use in inference payments, staking, and Nova Script contracts.
- EVM smart contracts can trigger modelOS inference by sending wMDL to a bridge adapter, which translates the instruction into a v3 inference transaction on the modelOS side — any Ethereum contract can access verifiable AI compute on modelOS.
- Bridge security uses a multi-signature model at launch, with a roadmap toward trustless light-client proof verification as the technology matures.

14 Competitive Landscape

The decentralised AI space includes compute marketplaces, incentive layers, API aggregators, and centralised platform alternatives. modelOS occupies a position that none of these projects currently fill: a Layer 1 blockchain with verifiable AI compute at the consensus level, a native inference market, AI-native smart contracts, and a community-governed model. The comparison in Table 5 is intended to be precise, not dismissive.

Protocol	Core offering	The gap modelOS fills
Bittensor (TAO)	Subnet-based AI incentive layer; validators vote on output quality, with no cryptographic proof that a model ran at all.	modelOS uses ZK-PoW: every block is a mathematical proof of real computation. Quality is enforced by cryptography, not reputation.
Akash Network	Decentralised cloud-compute rental; raw GPU access without an inference market, smart contracts, or model governance.	Compute is a means, not the product. modelOS provides the full stack above compute: verifiable inference, cognitive contracts, agentic infrastructure.
OpenRouter	Centralised API aggregator for language models; permissioned, custodial, with no on-chain settlement or proof of correctness.	modelOS is permissionless. No account, no API key; inference settles on-chain and every output is cryptographically verifiable.
Render Network	Decentralised GPU compute for rendering and AI; no inference market, smart-contract environment, or model-training layer.	Render provides raw compute. modelOS provides a complete AI infrastructure layer with programmability and governance.
Anthropic / OpenAI	Centralised, proprietary AI platforms with closed models, controlled access, and single points of failure.	modelOS is open-source, permissionless, and community-owned; no single entity controls the models or the infrastructure.

Table 5: *modelOS relative to existing decentralised-compute and AI-incentive networks.*

> modelOS is, to date, the only Layer 1 where every block is a proof of AI work, where model outputs are verified by consensus, and where the network itself trains and owns a model. Intelligence is not a feature of modelOS — it is the product.