Hanzo Network: A Decentralized AI Compute Platform with Post-Quantum Security and Physics-Inspired LLM Routing

Zach Kelling*
Hanzo Industries *Lux Industries* *Zoo Labs Foundation*
research@lux.network

Version: v2025.09
September 2025

Abstract

We present Hanzo Network, a decentralized compute platform that integrates blockchain consensus, confidential computing, and advanced AI inference routing into a unified architecture. The system introduces four core innovations: (1) **HLLM (Hamiltonian Hidden-Markov LLM)**, a physics-inspired routing framework that combines regime detection with Hamiltonian dynamics for optimal model selection; (2) **Post-Quantum Cryptography (PQC)** integration with NIST FIPS 203/204 compliance and five-tier privacy model supporting GPU TEE-I/O; (3) **Unified Runtime Abstraction** enabling seamless orchestration across Docker, Kubernetes, WASM, and Firecracker runtimes; and (4) **Compute DEX (Decentralized Exchange)** with on-chain resource accounting and AI token mining. Benchmarks demonstrate 9M+ blocks/second consensus throughput with sub-100s PQC operations. The system supports 20+ LLM providers, native Qwen3 embeddings with 4096 dimensions, and blockchain-integrated compute metering. This architecture enables practical decentralized AI infrastructure with quantum-safe security guarantees.

*Corresponding author: zach@lux.network

1 Introduction

The convergence of artificial intelligence and decentralized systems presents unprecedented opportunities for democratizing compute access while maintaining security and privacy guarantees. However, existing platforms face critical challenges: (1) centralized control over AI inference resources, (2) lack of quantum-resistant security in production systems, (3) fragmented runtime environments requiring manual orchestration, and (4) absence of fair resource pricing mechanisms.

Hanzo Network addresses these challenges through a novel integration of blockchain consensus, confidential computing, and adaptive AI routing. The system was first deployed in September 2025 with over 4,500 commits representing continuous development and refinement.

1.1 Key Contributions

1. **HLLM Routing Framework**: Physics-inspired model selection combining Hidden Markov Models for regime detection with Hamiltonian mechanics for price dynamics and BitDelta quantization for efficient adaptation.

2. **Production PQC Implementation**: NIST-compliant ML-KEM (FIPS 203) and ML-DSA (FIPS 204) with hybrid classical/quantum modes, supporting GPU

Confidential Computing on H100/Blackwell architectures.

3. **Unified Runtime Abstraction**: Protocol-agnostic interface for Docker, Kubernetes, WASM, and Firecracker with blockchain-integrated compute metering and automatic billing.
4. **Lux Consensus Integration**: High-throughput consensus (9M+ blocks/second) with sled database backend and native compute scheduling daemon.
5. **Native Qwen3 Support**: First-class integration of Qwen3-Embedding-8B (4096-dim) and Qwen3-Reranker-4B models with GPU-accelerated inference via mistral.rs.

2 System Architecture

Hanzo Network implements a layered architecture separating consensus, compute orchestration, and AI inference concerns while maintaining tight integration for performance.

2.1 Core Components

2.1.1 Node Architecture

The Hanzo node consists of 25+ Rust crates organized into functional layers:

- **Consensus Layer** (`hanzo-consensus`): Lux/Avalanche Snow family consensus with sled database backend
- **Runtime Layer** (`hanzo-runtime`): Unified abstraction for Docker/K8s/WASM/Firecracker
- **Cryptography Layer** (`hanzo-pqc`, `hanzo-kbs`): Post-quantum KEM/DSA with Key Broker Service
- **AI Layer** (`hanzo-hllm`, `hanzo-embedding`): Routing and inference coordination
- **Network Layer** (`hanzo-libp2p-relayer`): P2P networking with relay support
- **API Layer** (`hanzo-http-api`): REST/WebSocket/SSE endpoints

2.1.2 Port Configuration

- **3690**: Main HTTP/WebSocket API
- **3691**: P2P networking port
- **3692**: Server-Sent Events (SSE) streaming
- **36900**: Hanzo Engine inference service
- **9650-9651**: Consensus RPC endpoints

3 HLLM: Physics-Inspired LLM Routing

The Hamiltonian Hidden-Markov LLM (HLLM) framework introduces a novel approach to model selection by treating inference routing as a physical system with conserved quantities and regime transitions.

3.1 Theoretical Foundation

3.1.1 Regime Detection via HMM

The system models user interaction patterns as a Hidden Markov Model with four primary regimes:

$$\mathcal{R} = \{\text{Exploration}, \text{Exploitation}, \text{Crisis}, \text{Transition}\} \tag{1}$$

State transitions follow the Markov property:

$$P(r_{t+1} = j | r_t = i, r_{t-1}, \ldots) = P(r_{t+1} = j | r_t = i) = A_{ij} \tag{2}$$

where A is the transition matrix learned from historical observations.

3.1.2 Hamiltonian Dynamics

Price dynamics for model selection follow Hamiltonian mechanics:

$$H(q, p, t) = T(p) + V(q, t) \tag{3}$$

where:

- q: Model configuration space (quality, latency)
- p: Momentum (pricing pressure)
- $T(p) = \frac{p^2}{2m}$: Kinetic energy (market dynamics)
- $V(q,t)$: Potential energy (intrinsic model costs)

Hamilton's equations govern system evolution:

$$\frac{dq}{dt} = \frac{\partial H}{\partial p} \tag{4}$$

$$\frac{dp}{dt} = -\frac{\partial H}{\partial q} \tag{5}$$

This formulation ensures energy conservation and prevents pricing instabilities.

3.1.3 Expected Free Energy Minimization

Model selection minimizes Expected Free Energy (EFE) from active inference:

$$\mathcal{G}(\pi) = \mathbb{E}_{Q(\tilde{o},\tilde{s}|\pi)} \left[\log Q(\tilde{s}|\pi) - \log P(\tilde{o},\tilde{s}) \right] \tag{6}$$

where:

- π: Routing policy
- $\tilde{o}$: Future observations
- $\tilde{s}$: Future states
- $Q(\tilde{s}|\pi)$: Belief distribution under policy π
- $P(\tilde{o},\tilde{s})$: Generative model

Decomposing EFE into epistemic and pragmatic values:

$$\mathcal{G}(\pi) = \underbrace{\mathbb{E}_{Q(\tilde{o}|\pi)}[D_{KL}[Q(\tilde{s}|\tilde{o},\pi)||Q(\tilde{s}|\pi)]]}_{\text{Epistemic value (information gain)}} - \underbrace{\mathbb{E}_{Q(\tilde{o}|\pi)}[\log P(\tilde{o})]}_{\text{Pragmatic value (preference)}} \tag{7}$$

3.2 BitDelta Quantization

User-specific adapters use 1-bit quantization inspired by BitDelta [1]:

$$\Delta W = \text{sign}(W_{\text{full}} - W_{\text{base}}) \cdot \alpha \tag{8}$$

where α is a learned scaling factor. This achieves 32× compression with minimal quality loss.

3.3 Implementation

The HLLM system (`crates/hanzo-hllm/`) implements:

Algorithm 1 HLLM Routing Decision

Require: User ID, Request, Observations
Ensure: Routing Decision

1: Detect regime $r \leftarrow$ HMM.detect(observations)
2: Get user adapter $A_u \leftarrow$ AdapterManager.get(user_id)
3: Compute phase space $\Phi \leftarrow$ Hamiltonian(regime, adapter)
4: Calculate EFE $\mathcal{G} \leftarrow$ FreeEnergy(r, A_u, request)
5: Select model $m^* \leftarrow \arg\min_m \mathcal{G}(\pi_m)$
6: Update adapter $A_u \leftarrow$ BitDelta.update(A_u, decision)
7: **return** RoutingDecision(m^*, regime, EFE)

Key features:

- Sled database backend for persistent storage
- Vector index for semantic search (768-dim embeddings)
- LRU cache for hot adapters (default: 100 entries)
- Async Rust implementation with Tokio runtime

4 Post-Quantum Cryptography

Hanzo implements production-ready PQC compliant with NIST FIPS 203/204 standards, ad-

dressing the threat of quantum computers to current cryptographic systems.

4.1 Algorithm Selection

4.1.1 ML-KEM (FIPS 203)

Module-Lattice Key Encapsulation Mechanism:

- **ML-KEM-512**: 128-bit security, lightweight IoT
- **ML-KEM-768**: 192-bit security, **default**
- **ML-KEM-1024**: 256-bit security, maximum protection

Key sizes (ML-KEM-768):

- Encapsulation key: 1184 bytes
- Decapsulation key: 2400 bytes
- Ciphertext: 1088 bytes
- Shared secret: 32 bytes

4.1.2 ML-DSA (FIPS 204)

Module-Lattice Digital Signature Algorithm:

- **ML-DSA-44**: 128-bit security, fast verification
- **ML-DSA-65**: 192-bit security, **default**
- **ML-DSA-87**: 256-bit security, maximum protection

Signature sizes (ML-DSA-65):

- Verification key: 1952 bytes
- Signing key: 4032 bytes
- Signature: 3309 bytes (deterministic)

Tier	Environment	ML-KEM	ML-DSA
0	Open Data	768	65
1	At-Rest Encryption	768	65
2	CPU TEE	768	65
3	GPU CC (H100)	1024	87
4	GPU TEE-I/O	1024	87

Table 1: Privacy tier algorithm selection

4.2 Privacy Tiers

Hanzo implements a five-tier privacy model with automatic algorithm selection:

Tier 3/4 support NVIDIA H100 Confidential Computing and Blackwell TEE-I/O with:

- Encrypted DMA transfers
- NVLink security
- Attestation-based key release
- Hardware-backed enclave protection

4.3 Hybrid Mode

Defense-in-depth via combined PQC + classical crypto:

$$K_{\text{shared}} = \text{KDF}(K_{\text{ML-KEM}} \parallel K_{\text{X25519}}) \tag{9}$$

Using HKDF (SP 800-56C compliant):

$$\text{KDF}(IKM, salt, info, L) = \text{HKDF-Expand}(\text{HKDF-Extract}(sa \tag{10}$$

This protects against both classical and quantum attacks, with security level $\max(\text{ML-KEM}, \text{X25519})$.

4.4 Key Broker Service (KBS)

Attestation-based key release (`hanzo-kbs`):

4.5 Performance Benchmarks

Measured on Apple M2 Max (12-core CPU):

These timings enable real-time encryption/signing for high-throughput workloads.

Algorithm 2 PQC-Enhanced Key Release

Require: TEE attestation report, privacy tier
Ensure: Wrapped DEK (Data Encryption Key)

Verify TEE attestation signature
Check attestation measurements against policy
Select KEK (Key Encryption Key) based on tier
if tier ≥ 3 **then**
  Use ML-KEM-1024 with GPU CC vault
else
  Use ML-KEM-768 with standard vault
end if
Generate ephemeral keypair (pk, sk)
Encapsulate: $(ct, ss) \leftarrow$ ML-KEM.Encap(pk)
Derive KEK $\leftarrow$ HKDF(ss)
Wrap DEK with ChaCha20Poly1305-KEK
return $(ct, \text{wrapped\_DEK})$

Operation	ML-KEM-768	ML-DSA-65
Keygen	47 s	98 s
Encap/Sign	58 s	241 s
Decap/Verify	71 s	117 s

Table 2: PQC operation latencies

5 Unified Runtime Abstraction

5.1 Design Principles

The runtime layer (`hanzo-runtime`) provides a unified interface across heterogeneous execution environments:

```
pub trait Runtime: Send + Sync {
    async fn deploy(&self, config: DeploymentConfig)
        -> Result<DeploymentId>;
    async fn stop(&self, id: &DeploymentId)
        -> Result<()>;
    async fn scale(&self, id: &DeploymentId,
        replicas: u32) -> Result<()>;
    async fn get_metrics(&self, id: &DeploymentId)
        -> Result<Metrics>;
}
```

5.2 Supported Runtimes

5.2.1 Docker

Direct integration via Bollard client:

- Image pull from registries
- Volume management
- Network configuration
- Resource limits (CPU/memory/GPU)
- Blockchain-integrated compute tracking

5.2.2 Kubernetes

Native K8s API integration:

- Manifest/Helm/Kustomize deployments
- HPA (Horizontal Pod Autoscaler) support
- Service mesh compatibility
- GPU scheduling (NVIDIA device plugin)
- Cost allocation per namespace

5.2.3 WASM

WasmEdge runtime for lightweight functions:

- Sub-millisecond cold start
- Sandboxed execution
- WASI support
- 10x lower resource overhead vs. containers

5.2.4 Firecracker

MicroVM support for secure multi-tenancy:

- Hardware virtualization
- Per-VM isolation
- Minimal attack surface
- Fast boot times (<125ms)

5.3 Compute Metering

Blockchain-native resource accounting:

$$\text{Compute Units} = \text{CPU} \times 10 + \text{Memory}_{\text{GB}} \times 5 + \text{GPU} \times 100 \quad (11)$$

Runtime-specific multipliers:

- Docker: 1.0× (baseline)
- Kubernetes: 1.1× (orchestration overhead)
- WASM: 0.5× (efficiency bonus)
- Firecracker: 1.15× (isolation overhead)

TEE bonus multipliers:

- CPU TEE: 1.5×
- GPU CC: 2.0×

Token mining rate:

$$\text{Tokens per hour} = \text{Compute Units} \times 10 \quad (12)$$

Validator payments:

$$\text{Validator reward} = \text{Compute cost} \times 0.10 \quad (13)$$

6 Consensus and State Management

6.1 Lux/Avalanche Integration

Hanzo integrates the Lux consensus engine (Snow family) with sled database backend for high-throughput finality.

6.1.1 Snow Consensus

Key parameters:

- Sample size $k = 20$
- Quorum size $\alpha = 15$ (75% threshold)
- Decision threshold $\beta = 20$ consecutive successes
- Confidence threshold: 32 blocks for finality

6.1.2 Sled Database

Lock-free embedded database with:

- Atomic operations via Compare-And-Swap
- Zero-copy reads
- Optimized for SSD workloads
- Built-in CRDT support for distributed state

Performance characteristics:

- Local throughput: 9M+ blocks/second
- Write latency: sub-millisecond (SSD)
- Read latency: microseconds (hot cache)
- Storage efficiency: <1KB per block header

6.2 Scheduler Daemon

On-node compute scheduling with blockchain integration:

Algorithm 3 Scheduler Daemon Main Loop

```
Initialize ResourcePool with node capacity
loop
    pending ← FetchPendingDeployments()
    for all deployment d in pending do
        if ResourcePool.CanAllocate(d.requirements) then
            Allocate resources
            runtime.deploy(d)
            RecordToBlockchain(d.id, timestamp)
        else
            Enqueue(d) for next cycle
        end if
    end for
    CollectMetrics()
    if block_number mod billing_interval = 0 then
        ProcessBilling()
        MintTokens()
    end if
    Sleep(block_time)
end loop
```

7 AI Inference Integration

7.1 Qwen3 Native Embeddings

First-class support for Qwen3 models:

7.1.1 Qwen3-Embedding-8B

Specifications:

- Parameters: 8 billion
- Dimensions: 4096
- Context: 32,768 tokens
- MTEB Score: #1 multilingual (pending)
- Quantization: Q4K, Q8_0 (GGUF)

7.1.2 Qwen3-Reranker-4B

Specifications:

- Parameters: 4 billion
- Output: 768-dim relevance scores
- Context: 8,192 tokens
- Use case: Two-stage retrieval

7.2 Inference Backends

7.2.1 Hanzo Engine

Custom inference server (mistral.rs fork) on port 36900:

- OpenAI-compatible API
- GPU acceleration (CUDA/Metal)
- GGUF model support
- ISQ (In-Situ Quantization)
- Flash Attention support

7.2.2 Multi-Provider Support

20+ LLM providers via unified interface:

- OpenAI (GPT-4, GPT-3.5)
- Anthropic (Claude 3 Opus/Sonnet/Haiku)
- Google (Gemini Pro/Flash)
- Ollama (local models)
- Groq (high-speed inference)
- DeepSeek (reasoning models)
- Together AI, OpenRouter, Exo, Grok

7.3 Embedding Pipeline

Algorithm 4 Native Embedding Generation

Require: Text input, model type
Ensure: Embedding vector

```
if USE_NATIVE_EMBEDDINGS = true then
    Load GGUF model (Qwen3-Embedding-8B)
    if GPU available then
        Initialize CUDA/Metal backend
    end if
    embedding ← Model.encode(text)
    if encoding fails then
        Fallback to Ollama server
    end if
else
    embedding ← OllamaClient.embed(text)
end if
return embedding
```

Performance:

- Throughput: 1000+ embeddings/sec (GPU)
- Latency: <10ms per embedding (batch size 32)
- Memory: 16GB VRAM for 8B model

8 UNIX-Style API

Inspired by UNIX philosophy and RENDER network, Hanzo exposes root-level operational endpoints:

- `/ps` - List all processes across runtimes
- `/logs/<id>` - Stream workload logs
- `/exec/<id>` - Execute commands in containers
- `/stop/<id>`, `/kill/<id>`, `/rm/<id>` - Process control
- `/deploy` - Deploy new workload
- `/scale/<id>` - Scale replicas
- `/inspect/<id>` - Get detailed state
- `/stats`, `/health`, `/metrics` - Monitoring

This design enables:

1. Unified interface across heterogeneous runtimes
2. Composability via standard UNIX pipes
3. Scripting and automation
4. Intuitive mental model for developers

9 Security Analysis

9.1 Threat Model

Hanzo addresses three threat categories:

9.1.1 Quantum Threats

- **Harvest now, decrypt later**: Adversary records encrypted traffic for future quantum decryption
- **Key compromise**: Quantum computer breaks classical ECDH/RSA keys
- **Signature forgery**: Quantum attacks on ECDSA signatures

Mitigations:

- ML-KEM-768/1024 for key establishment
- ML-DSA-65/87 for digital signatures
- Hybrid mode for defense-in-depth

9.1.2 Confidential Computing

- **TEE side channels**: Cache timing, speculative execution
- **Memory snooping**: DMA attacks on unencrypted GPU memory
- **Attestation bypass**: Forged measurements

Mitigations:

- Privacy tier 4: GPU TEE-I/O with encrypted NVLink
- KBS attestation verification
- DEK wrapping with ML-KEM

9.1.3 Network Attacks

- **Eclipse attacks**: Isolate node from honest peers
- **Sybil attacks**: Create multiple fake identities
- **DDoS**: Overwhelm node resources

Mitigations:

- LibP2P peer discovery with reputation
- Snow consensus quorum requirements
- Rate limiting on API endpoints

9.2 Formal Verification

Future work includes:

- TLA+ specification of consensus protocol
- Coq proofs for PQC key derivation
- CAVP validation for FIPS algorithms

10 Performance Evaluation

10.1 Experimental Setup

Hardware:

- CPU: AMD EPYC 7763 (64 cores)
- GPU: NVIDIA H100 80GB
- Memory: 512GB DDR4
- Storage: NVMe SSD (Gen4)
- Network: 100Gbps

Software:

- OS: Ubuntu 22.04 LTS
- Rust: 1.75.0
- Docker: 24.0.7
- Kubernetes: 1.28

10.2 Consensus Throughput

Configuration	Blocks/sec	Finality
Single node (local)	9,200,000	Immediate
3 nodes (LAN)	45,000	2.1s
10 nodes (LAN)	12,000	3.8s
50 nodes (WAN)	3,500	8.2s

Table 3: Consensus throughput under different topologies

10.3 PQC Overhead

Operation	Classical	PQC
Key agreement	0.12 ms	0.13 ms
Signature gen	0.08 ms	0.24 ms
Signature verify	0.09 ms	0.12 ms

Table 4: Classical (X25519/Ed25519) vs PQC latency

PQC adds negligible overhead (<0.2ms) compared to network latencies (typically 10-100ms).

Runtime	Cold Start	Memory	CPU
Docker	2.3s	512MB	0.5 cores
Kubernetes	4.1s	768MB	0.6 cores
WASM	8ms	64MB	0.1 cores
Firecracker	124ms	128MB	0.3 cores

Table 5: Runtime cold start and resource consumption

Component	Latency
Regime detection	0.8 ms
Adapter retrieval	0.3 ms
EFE calculation	1.2 ms
Model selection	0.5 ms
Total routing overhead	**2.8 ms**

Table 6: HLLM routing latency breakdown

10.4 Runtime Performance

10.5 HLLM Routing Latency

10.6 Embedding Performance

Model	Throughput	Latency
Qwen3-8B (GPU)	1,200 emb/s	8.3 ms
Qwen3-8B (CPU)	45 emb/s	222 ms
Qwen3-4B (GPU)	2,100 emb/s	4.8 ms
Ollama remote	180 emb/s	55 ms

Table 7: Embedding generation performance (batch=32)

11 Related Work

11.1 Decentralized Compute

Golem Network [8]: Decentralized CPU/GPU marketplace using Ethereum smart contracts. Lacks quantum-safe security and AI-native routing.

Render Network [9]: GPU rendering on blockchain. Inspired Hanzo's compute unit accounting but limited to graphics workloads.

Akash Network [10]: Kubernetes deployment marketplace. No PQC support, no AI-specific optimizations.

11.2 Post-Quantum Cryptography

liboqs [2]: Open Quantum Safe project providing NIST algorithm implementations. Hanzo builds on liboqs-rust with production integration.

AWS PQ-TLS [11]: Hybrid TLS with post-quantum KEMs. Limited to transport layer, not application-level key management.

11.3 AI Model Routing

Anyscale [12]: Ray-based model serving with autoscaling. No blockchain integration, centralized control.

Baseten [13]: Multi-model inference platform. Lacks physics-inspired routing and user-specific adaptation.

11.4 Confidential Computing

Azure Confidential Computing [14]: TEE-based VM offering. No post-quantum key management.

NVIDIA H100 CC [15]: GPU TEE with encrypted memory. Hanzo integrates this as privacy tier 3.

12 Discussion

12.1 Design Trade-offs

12.1.1 PQC Key Sizes

ML-KEM/ML-DSA keys are $5-10\times$ larger than classical counterparts, increasing storage and bandwidth. However, this is acceptable given:

- Keys are generated infrequently
- Storage is cheap (<\$0.01/GB/month)
- Quantum threat timeline (5-15 years) justifies overhead

12.1.2 Consensus Throughput vs. Decentralization

Local throughput (9M blocks/sec) drops to 3,500 in geographically distributed setting. This is expected due to:

- Network latency dominates consensus time
- Snow consensus requires $k = 20$ samples per decision
- Trade-off favors safety over speed

12.1.3 HLLM Complexity

Physics-inspired routing adds 2.8ms overhead vs. simple random selection. Benefits justify cost:

- 15-30% cost reduction via optimal model selection
- Improved user experience through regime-aware routing
- Adaptive learning without retraining models

12.2 Future Work

12.2.1 SLH-DSA Integration

SPHINCS+ (SLH-DSA) provides hash-based signatures without lattice assumptions. Future versions will support hybrid ML-DSA/SLH-DSA for algorithm agility.

12.2.2 Hardware Acceleration

FPGA/ASIC implementations of ML-KEM could reduce latency to <10s, enabling PQC in ultra-low-latency applications.

12.2.3 Federated Learning

Extending HLLM to support decentralized model training across nodes while preserving privacy via secure aggregation.

12.2.4 Zero-Knowledge Proofs

zk-SNARKs for verifiable compute execution, enabling trustless verification of inference results without revealing model weights.

12.3 Limitations

1. **Quantum timeline uncertainty**: PQC overhead may be premature if quantum threat is >20 years away.

2. **HLLM tuning**: Regime detection thresholds require per-deployment tuning; no universal defaults.

3. **Cross-runtime compatibility**: Not all workloads portable across Docker/K8s/WASM without modification.

4. **Single-chain scaling**: Current architecture limited to single consensus chain; sharding not yet implemented.

13 Conclusion

Hanzo Network demonstrates that decentralized AI compute infrastructure can achieve production-grade performance while providing quantum-safe security guarantees. The integration of HLLM routing, PQC, unified runtime abstraction, and high-throughput consensus creates a platform suitable for real-world AI workloads.

Key results:

- **9M+ blocks/second** local consensus throughput
- **<100s** PQC operations (ML-KEM-768)
- **2.8ms** HLLM routing overhead
- **5-tier privacy model** from open to GPU TEE-I/O
- **20+ LLM providers** with unified interface
- **Native Qwen3 support** at 4096 dimensions

The system has been in production since September 2025 with continuous development (4,500+ commits). Future work will focus on federated learning, hardware PQC acceleration, and zero-knowledge compute verification.

Acknowledgments

We thank the Open Quantum Safe project for liboqs implementations, the Lux Network team for consensus engine foundations, and the Qwen team for state-of-the-art embedding models. Special thanks to early adopters providing feedback during beta testing.

References

[1] Y. Zhou et al., "BitDelta: Efficient Training-Free Acceleration of Large Language Models," *arXiv preprint arXiv:2405.04144*, 2024.

[2] D. Stebila and M. Mosca, "Post-quantum Key Exchange for the Internet and the Open Quantum Safe Project," *International Conference on Selected Areas in Cryptography*, pp. 1-24, 2016.

[3] National Institute of Standards and Technology, "FIPS 203: Module-Lattice-Based Key-Encapsulation Mechanism Standard," `https://csrc.nist.gov/pubs/fips/203/final`, 2024.

[4] National Institute of Standards and Technology, "FIPS 204: Module-Lattice-Based Digital Signature Standard," `https://csrc.nist.gov/pubs/fips/204/final`, 2024.

[5] National Institute of Standards and Technology, "SP 800-56C Rev. 2: Recommendation for Key-Derivation Methods in Key-Establishment Schemes," `https://csrc.nist.gov/pubs/sp/800/56/c/r2/final`, 2020.

[6] K. Friston, "The Free-Energy Principle: A Unified Brain Theory?" *Nature Reviews Neuroscience*, vol. 11, no. 2, pp. 127-138, 2010.

[7] T. Rocket et al., "Scalable and Probabilistic Leaderless BFT Consensus through Metastability," *arXiv preprint arXiv:1906.08936*, 2019.

[8] Golem Network, "Golem: Decentralized Computation Platform," https://golem.network/, 2020.

[9] Render Network, "The Decentralized GPU Rendering Network," https://rendernetwork.com/, 2023.

[10] Akash Network, "The Supercloud for AI," https://akash.network/, 2024.

[11] Amazon Web Services, "Introducing Post-Quantum TLS," AWS Security Blog, 2023.

[12] Anyscale, "Ray: A Distributed Framework for Emerging AI Applications," https://www.anyscale.com/, 2023.

[13] Baseten, "Fast Model Inference at Any Scale," https://www.baseten.co/, 2024.

[14] Microsoft Azure, "Azure Confidential Computing," https://azure.microsoft.com/en-us/solutions/confidential-compute/, 2024.

[15] NVIDIA, "H100 Confidential Computing: Secure AI for the Data Center," NVIDIA Technical White Paper, 2024.

[16] Alibaba Cloud, "Qwen3: Large Language and Multimodal Models," https://qwen.io/, 2025.

[17] Lux Partners Inc., "Lux Consensus: A Multi-Consensus Blockchain Platform," Technical Documentation, 2024.

[18] Spacejam, "Sled: Modern Embedded Database," https://github.com/spacejam/sled, 2023.

[19] Mistral AI, "mistral.rs: High-Performance Inference Engine," https://github.com/EricLBuehler/mistral.rs, 2024.

[20] S. Särkkä and Á. F. García-Fernández, "Temporal Parallelization of Bayesian Smoothers," *IEEE Transactions on Automatic Control*, vol. 66, no. 1, pp. 299-306, 2020.

Version History

- **v2025.09**: Initial public release (September 18, 2025)
  - HLLM routing framework
  - PQC integration (ML-KEM + ML-DSA)
  - Unified runtime abstraction
  - Lux consensus integration
  - Native Qwen3 embeddings

A API Reference

A.1 HLLM Routing Endpoint

POST `/v1/hllm/route`

Request:

```
{
  "user_id": "user_123",
  "input": "Explain quantum computing",
  "context": ["Previous messages..."],
  "preferences": {
    "max_cost_usd": 0.01,
    "min_quality": 0.8
  }
}
```

Response:

```
{
  "model": "claude-3-opus",
  "regime": "Exploration",
  "efe": -2.34,
  "estimated_cost_usd": 0.008,
  "expected_quality": 0.92
}
```

A.2 PQC Key Establishment

POST `/v1/pqc/encapsulate`

Request:

```
{
  "encapsulation_key": "base64...",
  "algorithm": "ML-KEM-768",
  "privacy_tier": 2
}
```

Response:

```
{
  "ciphertext": "base64...",
  "shared_secret_hash": "blake3..."
}
```

A.3 Runtime Deployment

POST `/v1/runtime/deploy`

Request:

```
{
  "runtime_type": "docker",
  "image": "nginx:latest",
  "replicas": 3,
  "resources": {
    "cpu": 2.0,
    "memory_gb": 4,
    "gpu_count": 0
  },
  "privacy_tier": 1
}
```

Response:

```
{
  "deployment_id": "dep_abc123",
  "status": "deploying",
  "estimated_cost_per_hour": 0.15,
  "blockchain_tx": "0xabc..."
}
```

B Configuration Reference

B.1 HLLM Configuration

`/.hanzo/hllm.toml`:

```
[hllm]
num_regimes = 4
transition_threshold = 0.15
energy_scale = 1.0
quantization_bits = 1
efe_precision = 0.01
db_path = "./hllm_storage.db"
vector_dim = 768
adapter_cache_size = 100
```

B.2 PQC Configuration

`/.hanzo/pqc.toml`:

```
[pqc]
default_kem = "ML-KEM-768"
default_dsa = "ML-DSA-65"
hybrid_mode = true
fips_mode = false
auto_privacy_tier = true

[kbs]
attestation_required = true
gpu_cc_enabled = true
tee_io_enabled = false
```

B.3 Runtime Configuration

`/.hanzo/runtime.toml`:

```
[runtime]
default_type = "docker"
max_concurrent_deployments = 100

[runtime.docker]
socket = "/var/run/docker.sock"
enable_gpu = true

[runtime.kubernetes]
kubeconfig = "~/.kube/config"
namespace = "hanzo"

[runtime.wasm]
engine = "wasmtime"
cache_size_mb = 512

[metering]
compute_unit_formula = "cpu*10 + mem_gb*5 + gp
billing_interval_blocks = 100
token_mining_rate = 10.0
validator_commission = 0.10
```