



Master Project Report

Obsuidian: a fully decentralized RPC solution for the Sui network

Loris Tran and Alexandre Mourot

MSc in Computer Science
École Polytechnique Fédérale de Lausanne

Rachid Guerraoui
Thesis Advisor

Gauthier Voron
Thesis Supervisor

Distributed Computing Laboratory
School of Computer and Communication Sciences
École Polytechnique Fédérale de Lausanne

May 2025

Abstract

This report examines the challenges of Remote Procedure Call (RPC) access in the Sui blockchain ecosystem and proposes a decentralized solution using the Lava Network. We analyze the current limitations of centralized RPC providers, including reliability issues, centralization risks, and lack of economic incentives for Full Node operators. Our solution leverages Lava's decentralized protocol to create a robust, incentivized network of Sui RPC providers, ensuring high availability, censorship resistance, fault-tolerant and economic sustainability. The implementation details, including specification files, provider configuration, and security mechanisms are thoroughly explored and benchmarked to demonstrate the viability of this approach for large scale blockchain infrastructure.

1 Introduction to Sui Blockchain

Sui is a high-performance Layer 1 blockchain designed for scalable, decentralized applications. Built by Mysten Labs, Sui introduces a novel architecture that departs from traditional blockchain designs by implementing a directed acyclic graph (DAG) structure and parallel transaction execution. This architecture enables Sui to achieve high throughput, low latency, and horizontal scalability.

Sui's object-centric data model treats on-chain assets as distinct objects with unique identifiers, enabling parallel execution of transactions that operate on different objects.

1.1 Object-Centric Architecture in Sui

Sui implements an object-centric data model that differs from traditional account-based or UTXO-based systems. In Sui's architecture, the fundamental unit of state is the object, which is a discrete entity with a unique identifier, owner, and data payload. This object-centric approach enables Sui's parallel execution model, which is central to its scalability proposition.

Objects in Sui are classified into two primary categories with significantly different execution characteristics. First, there are Owned Objects, which are exclusively owned by a single address. These owned objects exhibit several key characteristics that make them particularly efficient to process. Each owned object has exactly one owner (an address) that maintains exclusive control over the object, enabling transactions involving only owned objects from different owners to be processed in parallel without causal ordering constraints.

Furthermore, transactions operating exclusively on owned objects can achieve immediate finality without consensus overhead, requiring only a quorum of validator signatures. The execution outcome of transactions on owned objects is also fully deterministic based on the transaction inputs, without dependencies on global state.

This architecture enables Sui's "fast path" execution, where transactions involving only owned objects can bypass the traditional consensus bottleneck, achieving high throughput and low latency. This design choice represents a fundamental optimization in Sui's object model, allowing for significant performance improvements in scenarios where objects have clear, singular ownership.

Then, we also have Shared objects with fundamentally different properties. Multiple addresses can access and modify shared objects, requiring coordination mechanisms to prevent conflicts. Transactions involving shared objects must go through consensus to establish a canonical ordering, as concurrent modifications could lead to inconsistent states. Operations on the same shared object must be processed sequentially to maintain consistency, introducing a synchronization point in Sui's

otherwise parallel execution model. The execution outcome of transactions involving shared objects depends on the current state of those objects, which may change due to other transactions. Shared objects enable critical functionality such as DEXs, shared counters, and other coordination-dependent applications, but at the cost of increased execution complexity.

1.2 Full Node vs Validator Node Roles

Sui separates its consensus and its data layer into Full Nodes and Validator Nodes. Validator Nodes are responsible for participating in consensus protocol. These validators execute transactions involving shared objects and collectively maintain the integrity and ordering of such transactions within the system. They act as authoritative sources, issuing certificates for valid transactions, and produce effects that represent the resulting state transitions. They are configured to enable quick confirmation of transactions, with a time to finality of around 300ms.

In contrast, Full Nodes do not participate in consensus but instead focus on maintaining a replica of the global state derived from the transaction outputs confirmed by validators. They handle data dissemination, transaction submission, and state queries for clients. Full Nodes play a crucial role in enabling scalability and data availability by serving as intermediaries between clients and the consensus layer, caching verified data, and allowing clients to fetch on-chain object states efficiently. Full nodes relay clients transactions requests into Validator Nodes, as can be seen in Figure 1

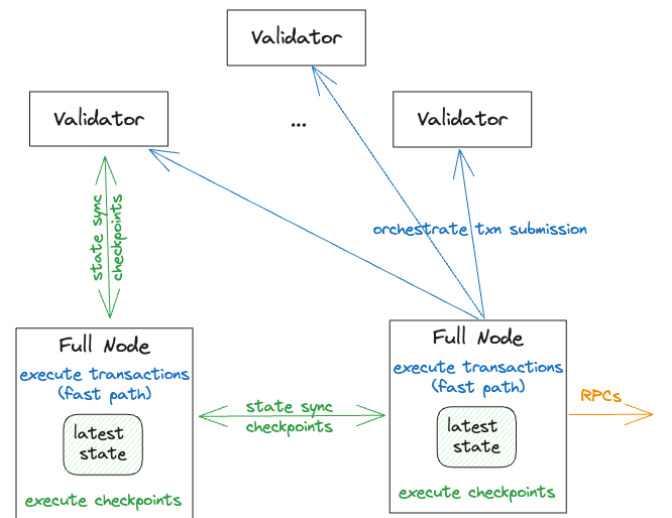


Figure 1: Communication Schema between Full Nodes and Validator Nodes.

The dual object nature of Sui’s object model creates specific dependencies on full node state that directly impact scalability: For transactions involving shared objects, full nodes must maintain an accurate and up-to-date view of the object state to support several critical functions. Full nodes need to validate transaction preconditions, as many transactions specify preconditions on object states that must be verified before execution. They must also detect potential conflicts between concurrent transactions accessing the same shared objects. Additionally, to provide clients with expected transaction outcomes, full nodes must simulate execution against their current state. For shared objects, full nodes track the causal history to ensure correct sequential processing. This state-dependent validation creates a fundamental scalability challenge: as transaction volume increases, the state management burden on full nodes grows correspondingly.

Sui addresses this challenge through several state partitioning strategies. By tracking dependencies at the object level rather than globally, Sui enables parallel processing of transactions affecting different objects. Transactions involving only owned objects can be processed with minimal state dependencies, maximizing throughput. Shared objects can be sharded across validator subsets, reducing the consensus bottleneck for any single shared object. Full nodes can implement aggressive state pruning strategies for historical data while maintaining the active object set. These approaches collectively enable Sui to scale more effectively than traditional blockchain architectures.

1.3 Full Node Specialization in Data Access

The complexity of Sui’s object model necessitates specialized full node configurations to efficiently serve different query patterns. Full nodes must maintain specialized indices beyond the core state to efficiently query objects by owner, type, or other attributes. These nodes must also maintain additional historical data to support applications requiring access to historical object states, while implementing efficient event propagation mechanisms for real-time applications dependent on event subscriptions. Furthermore, DApp developers heavily rely on transaction simulation capabilities to predict execution outcomes, requiring full nodes to implement complex simulation logic. These specialized requirements create significant operational overhead for full node operators, exacerbating an underlying economic misalignment problem.

In blockchain architectures, scalability remains one of the three cornerstone challenges, alongside decentralization and security, that directly impacts network performance and user experience. To address this challenge, most blockchain networks implement a multi-tiered

architecture that separates validator nodes from full nodes, a pattern particularly evident in high-throughput blockchains such as Sui, Aptos, and other next-generation networks. This architectural separation enables validators to focus their computational resources on achieving consensus on the canonical state of the blockchain, while full nodes handle data storage and client accessibility through RPC endpoints. Full nodes can be horizontally scaled to accommodate increasing query loads from applications and users, independent of the consensus layer’s scaling constraints, while creating a more efficient network topology where relatively few validators maintain consensus as a larger number of full nodes disseminate blockchain data. Additionally, this separation enhances overall network security by isolating query handling from consensus participation, effectively reducing the attack surface of validator nodes.

However, while this architectural pattern effectively addresses scalability concerns, it introduces a critical dependency: client applications must rely on Remote Procedure Call (RPC) endpoints provided by full nodes to interact with the blockchain. This reliance creates a single point of failure, as full node operators currently lack direct incentives to maintain their infrastructure. While solutions like Lava or dRPC exist for EVM chains, newer blockchain architectures implementing novel programming models like Move lack compatible decentralized RPC solutions, creating a significant gap in their infrastructure ecosystem. This limitation particularly affects next-generation blockchains that have moved beyond traditional EVM compatibility, highlighting the need for more versatile and adaptable decentralized RPC solutions.

1.4 The Full Node RPC Dependency Problem

The reliance on full node RPC endpoints creates several significant challenges for blockchain ecosystems:

1. **Single Points of Failure:** Applications typically connect to a single RPC endpoint or a small set of endpoints, creating critical dependencies that compromise the inherent decentralization benefits of blockchain technology.
2. **Lack of Economic Incentives:** Unlike validators who receive protocol rewards, full node operators lack direct economic incentives for providing reliable RPC services, resulting in underinvestment in this critical infrastructure layer.
3. **Quality of Service Variability:** Without economic incentives tied to service quality, RPC endpoints often exhibit inconsistent performance, availability, and data freshness.

4. **Centralization Pressure:** The economic burden of operating high-quality full nodes leads to centralization around well-funded entities, contradicting the decentralization ethos of blockchain networks.

This problem is particularly seen in production environments where applications require enterprise-grade reliability, performance, and data consistency. The absence of a sustainable economic model for full node operation creates a fundamental misalignment between the infrastructure needs of the ecosystem and the incentives for infrastructure providers. In turn, this pushes builders of decentralized applications on the Sui Network to rely on a few, expensive, and centralized data providers such as Sentio or Tritton One. This creates significant barriers to entry for newcomers who might not have the resources to afford these providers.

1.5 Limitations of Existing Solutions for Non-EVM Blockchains

For Ethereum Virtual Machine (EVM) compatible blockchains, several decentralized RPC solutions have emerged to address these challenges. Protocols such as Lava Network and dRPC have developed frameworks that create marketplaces for RPC services, aligning economic incentives with service quality through token-based mechanisms.

However, these solutions are about non-existent when applied to non-EVM blockchains, particularly those implementing new programming models such as Move-based blockchains (Sui, Aptos) or other next-generation architectures:

1. **API Incompatibility:** Non-EVM chains implement fundamentally different API structures and RPC methods, request-response patterns, and data models that are incompatible with EVM-oriented solutions.
2. **Determinism Challenges:** The verification mechanisms used in decentralized RPC networks often rely on deterministic response validation, which becomes more complex in blockchains with different execution models and state representations.
3. **Specialized Requirements:** Move-based blockchains like Sui implement unique features such as object-centric data models and parallel transaction execution that require specialized handling in RPC infrastructure.
4. **Protocol Evolution:** Newer blockchains are evolving rapidly, with frequent API changes and protocol upgrades that require more adaptive RPC infrastructure.
5. **Transition to gRPC:** Many next-generation blockchains are transitioning from JSON-RPC to

gRPC for performance reasons, requiring RPC infrastructure that can support both protocols during transition periods.

The absence of decentralized RPC solutions for these non-EVM blockchains creates a significant barrier to their adoption and ecosystem growth. Applications built on these platforms must either rely on centralized RPC providers or invest substantial resources in operating their own full nodes, neither of which represents an optimal solution for a decentralized ecosystem.

This infrastructure gap highlights the need for blockchain-agnostic decentralized RPC protocols that can adapt to the unique characteristics of diverse blockchain architectures, particularly those implementing novel programming models and execution environments beyond the EVM paradigm.

2 Problem Statement: Centralized and Vulnerable RPC Infrastructure in Sui

The current RPC infrastructure for the Sui blockchain suffers from fundamental structural issues that undermine the network's decentralization principles and create significant operational and security risks:

1. **Centralization of Access Points:** Despite Sui's decentralized architecture, most applications rely on a small number of centralized RPC providers (such as Mysten Labs' official endpoints, Sentio, and Tritton One), creating critical single points of failure that compromise application resilience.
2. **Absence of Economic Incentives:** Unlike validators who receive protocol rewards, Full Node operators have no direct economic incentives to provide reliable RPC services, resulting in under-investment in this critical infrastructure layer and limiting the geographic distribution of available endpoints.
3. **Quality of Service Inconsistency:** Without economic alignment, RPC endpoints exhibit variable performance, availability, and data freshness, particularly during periods of high network activity such as token launches or NFT mints.
4. **Censorship Vulnerability:** Centralized RPC providers represent potential censorship vectors, as demonstrated by recent incidents where Infura (Metamask) and Alchemy, two popular RPC providers, censored access to Tornado Cash contracts through their RPC. Venezuelan users of Infura RPC were also censored in 2022, citing law compliance.

5. **Security Vulnerabilities:** The current model of trusting individual RPC providers creates significant security risks. Malicious providers can act as "man-in-the-middle" attackers, potentially manipulating transaction data, returning falsified state information, front-running transactions, or selectively censoring specific users—all while appearing as legitimate infrastructure providers.
6. **Lack of Verification Mechanisms:** Users currently have no systematic way to verify that the data received from RPC providers is accurate and unmodified, forcing them to place complete trust in their chosen provider.
7. **Lack of Move-Compatible Solutions:** While decentralized RPC solutions exist for EVM chains, these are incompatible with Sui's Move-based architecture, unique object model, and specialized API requirements.

These issues create significant barriers to Sui's adoption and compromise the network's security and decentralization. As the ecosystem grows, the demand for reliable RPC access will only increase, potentially exacerbating centralization pressures and security risks. A decentralized, economically sustainable, and security-focused RPC solution specifically designed for Sui's architecture is therefore essential for the network's long-term success.

3 Lava Network: A Decentralized RPC Solution

To address all the RPC challenges in the Sui ecosystem, we propose leveraging the Lava Network—a decentralized protocol specifically designed to provide reliable, censorship-resistant, and incentivized access to blockchain data.

3.1 Lava Network Architecture

Lava Network functions as a decentralized marketplace for RPC services, connecting blockchain applications (consumers) with infrastructure providers through a protocol layer that ensures quality, reliability, and fair compensation. This Network is RPC agnostic and consists of several key components. **Providers** are node operators who run Sui Full Nodes and offer RPC services through the Lava protocol, forming the backbone of the network's infrastructure. **Consumers** represent applications or services that require access to Sui blockchain data, ranging from wallets and dApps to analytics platforms. **Validators** are entities that validate transactions on the Lava blockchain and maintain consensus and pairings lists for Providers and Consumers, ensuring the integrity of the network. The **Pairing Mechanism** is a

pseudo-random algorithm that matches consumers with multiple providers based on stake, geolocation, and quality metrics, optimizing for both performance and decentralization. Finally, **Specification Files** are governance-approved configurations that define the specifications, compute costs, and verification parameters for each supported blockchain, enabling the protocol to adapt to different blockchain architectures.

3.2 Benefits for Sui Ecosystem

Implementing Lava as the RPC layer for Sui offers several significant advantages. **Decentralization** is achieved by distributing RPC requests across multiple independent providers, eliminating single points of failure that plague centralized infrastructure. **Economic Incentives** allow providers to earn rewards for delivering reliable RPC services, creating sustainable economics for Sui Full Node operators who previously lacked direct compensation for their services. **Geographic Distribution** is enhanced through Lava's geolocation-aware pairing, which ensures consumers can access providers in their region, reducing latency and improving application performance. **Censorship Resistance** is strengthened as the random pairing of consumers with multiple providers makes targeted censorship extremely difficult, protecting the network from regulatory or political interference. **Quality Assurance** is maintained through Lava's data reliability system, which verifies provider responses to ensure accuracy and consistency across the network. Finally, **Scalability** is inherent to the design as demand increases, economic incentives attract more providers to the network, naturally scaling capacity to meet growing ecosystem needs.

4 Technical Implementation

4.1 Sui Specification File

The foundation of integrating Sui with Lava is the specification file—a detailed configuration that defines how Sui's RPC Methods are exposed through the Lava protocol. Below is an excerpt from the Sui JSON-RPC specification:

This specification defines:

- The unique identifier for Sui JSON-RPC services (SUI_JSONRPC)
- Security parameters for data verification
- Minimum stake requirements for providers
- Supported RPC methods with their compute costs
- Determinism classification for each method

The determinism classification is particularly important for Lava's security model. Methods marked as

deterministic should return identical responses across all providers for the same query at the same block height, enabling cross-verification of responses.

Sui JSON-RPC Specification

```

1 {
2   "proposal": {
3     "title": "Add Specs: Sui Full Node
4       JSON-RPC",
5     "description": "Adding new
6       specification support for relaying
7       Sui Full Node JSON-RPC data on Lava",
8     "specs": [
9       {
10        "index": "SUI_JSONRPC",
11        "name": "Sui Full Node JSON-RPC",
12        "enabled": true,
13        "reliability_threshold":
14          268435455,
15        "data_reliability_enabled": true,
16        "block_distance_for_finalized_data":
17          1,
18        "blocks_in_finalization_proof":
19          1,
20        "average_block_time": 1000,
21        "allowed_block_lag_for_qos_sync":
22          10,
23        "shares": 1,
24        "min_stake_provider": {
25          "denom": "ulava",
26          "amount": "5000000000"
27        },
28        "api_collections": [
29          {
30            "enabled": true,
31            "collection_data": {
32              "api_interface": "jsonrpc",
33              "internal_path": "",
34              "type": "POST",
35              "add_on": ""
36            },
37            "apis": [
38              {
39                "name": "
40                  suix_getAllBalances",
41                "block_parsing": {
42                  "parser_arg": ["latest
43                    "],
44                  "parser_func": "DEFAULT
45                    "
46                },
47                "compute_units": 100,
48                "enabled": true,
49                "category": {
50                  "deterministic": true,
51                  "local": false,
52                  "subscription": false,
53                  "stateful": 0
54                }
55              },
56              // Additional APIs...
57            ]
58          }
59        ]
60      }
61    ]
62  }
63 }

```

4.2 Provider Setup for Sui

To become a Sui RPC provider on Lava, node operators must:

1. **Run a Sui Full Node:** Configure and maintain a Sui Full Node following the official documentation.
2. **Install Lava Software:** Install the Lava protocol software (lavap) and configure it to connect to the Sui Full Node.
3. **Stake Tokens:** Stake LAVA tokens to the Sui specification to become eligible for pairing with consumers.
4. **Configure Provider Service:** Set up the provider service to listen for requests and forward them to the Sui Full Node.

The provider configuration involves creating a YAML file or using command-line arguments to specify the connection details:

Provider Configuration Example

```

1 network-address: 0.0.0.0:8080
2 # Provider Local Listen Address
3 chain-id: SUI_JSONRPC
4 # Specification file for Sui
5 api-interface: jsonrpc
6 # RPC interface type (here JSON-RPC)
7 node-urls: http://localhost:9000
8 # Sui RPC Source Address

```

Once configured, providers stake tokens using the lavap command-line tool:

Provider Staking Command

```

1 lavap tx pairing stake-provider "
2   SUI_JSONRPC" \
3   "5000000000ulava" \
4   "provider-domain.com:443,USC" USC \
5   --from "provider_wallet" \
6   --chain-id lava-mainnet-1 \
7   --gas="auto" \
8   --gas-adjustment "1.5"

```

Where provider-domain.com:443,USC is the local provider IP or Address, and USC the geolocation of the provider (Here US-Central).

where -from provider_wallet indicates which provider address to stake from,

where -chain-id lava-mainnet-1 is the chain ID of the lava node you stake to.

4.3 Docker Compose Deployment

To simplify the deployment and management of the entire Sui-Lava integration stack, we have implemented a

comprehensive Docker Compose solution. This approach enables operators to deploy the complete infrastructure using a single command, with proper configuration and inter-service communication handled automatically.

4.3.1 Root Docker Compose Architecture

The deployment uses a hierarchical Docker Compose structure, with a root `docker-compose.yml` file that orchestrates the entire stack:

Root docker-compose.yml

```
1 name: master-project
2
3 volumes:
4   checkpoints_data:
5
6 include:
7
8   - path: ./sui_indexer_checkpointTx/
9     docker-compose.yml
10    project_directory: ./
11    sui_indexer_checkpointTx
12    name: sui
13
14   - path: ./supabase/docker/docker-
15     compose.yml
16    project_directory: ./supabase/docker
17    name: supabase
18
19   - path: ./full-node/docker-compose.yml
20    project_directory: ./full-node
21    name: full-node
22
23   - path: ./lava/docker/simple-provider/
24     docker-compose.yml
25    project_directory: ./lava/docker/
26    simple-provider
27    name: lava
```

This root compose file leverages Docker Compose's `include` feature to include services defined in the individual project repositories, while also defining the necessary network connections and dependencies between services. A `docker-compose.override.yml` file also extends some definitions and acts as a setup on top of the local `docker-compose.yml` file of each repository.

4.3.2 Component-Specific Compose Files

Each component maintains its own Docker Compose configuration:

1. **Sui Node Compose:** Configures the Sui Full Node with appropriate volume mounts for persistence and network ports.
2. **Lava Compose:** Sets up the Lava node service along with two provider services and one consumer service, along with a nginx proxy. This setup along with the stake commands is fully automated and

results in having a local Lava Blockchain running, which communicates with the two providers and the consumer setup.

3. **Sui Indexer Compose:** Configures the Sui-Indexer-Alt-Framework from the Sui Indexer 2.0 platform to ingest its checkpoints from the full node, perform logic, and dump data into a PostgreSQL database.
4. **Supabase Compose:** Configures a Supabase Docker Instance, an open-source alternative to firebase with a built-in PostgreSQL database, dashboard, analytics, editor, and REST + GraphQL endpoints automatically generated from the database content.

4.3.3 Environment Configuration

Configuration is managed through a combination of environment files and volume mounts, this environment file is the combination of all the single environment files from each repository.

.env File Example

```
1 # Sui Indexer Configuration
2
3 POSTGRES_PASSWORD=sui-indexer
4 START_CHECKPOINT=138216332
5 LOCAL_MODE=true
6 CHECKPOINT_DIR=/checkpoints_indexer
7 PACKAGE_ADDRESS=0x000
8
9
10 # Lava Configuration
11 LAVAP_VERSION=v5.2.1
12 LAVAD_VERSION=v5.2.1
13 PROVIDER_WALLET=service1
14 GEOLOCATION=2
15 KEYRING_BACKEND=test
16 LAVA_RPC_NODE=tcp://lava-node:26657
17 PROVIDER_SUI_PORT=2220
18
19 # Supabase Configuration :
20 POSTGRES_PASSWORD=sui-indexer
21 JWT_SECRET=xxx
22
23 ANON_KEY=xxx
24 SERVICE_ROLE_KEY=xxx
25 DASHBOARD_USERNAME=supabase
26 DASHBOARD_PASSWORD=xxx
27 SECRET_KEY_BASE=xxx
28 VAULT_ENC_KEY=xxx
```

4.3.4 Deployment Workflow

The complete deployment process is streamlined to a few commands:

Deployment Commands

```
1 # Clone repositories
2 git clone git@github.com:Loris-EPFL/
  Monorepo.git
3
4 # Configure environment
5 cp .env.example .env
6 # Edit .env with appropriate values
7
8 # Start the stack
9 docker compose up -d
10
11 # Monitor logs
12 docker compose logs -f
13
14 # Close the stack
15 docker compose down -v
```

This Docker Compose integration significantly reduces the operational complexity of running a Sui RPC provider on the Lava Network, making it accessible to a wider range of operators and thereby enhancing the decentralization of the RPC infrastructure.

4.4 Decentralized Provider Network

A key advantage of Lava’s approach is the creation of a decentralized network of Sui RPC providers. This network offers several benefits over traditional centralized providers. Redundancy is achieved as multiple providers serve the same APIs, eliminating single points of failure that plague centralized solutions. Geographic distribution is enabled as providers can specify their geolocation, allowing consumers to connect to nearby providers for lower latency, which is particularly important for latency-sensitive applications. The pairing mechanism implements intelligent load balancing by distributing requests across providers based on their stake and capacity, preventing any single provider from becoming overwhelmed during usage spikes. Additionally, automatic failover ensures that if a provider becomes unresponsive, consumers can seamlessly switch to alternative providers in their pairing, maintaining continuous service availability even during partial network outages.

5 Evaluation of Lava Network for Sui RPC Infrastructure

This section evaluates the effectiveness of the Lava Network as a decentralized RPC solution for Sui blockchain. We examine both the economic sustainability through incentive mechanisms and the technical performance through comprehensive benchmarking.

5.1 Economic Incentives and Provider Participation

5.1.1 Lava Incentive Pools

The Lava Network implements a sophisticated economic model to incentivize RPC providers and ensure sustainable infrastructure. The incentive structure consists of several key components that work together to create a balanced ecosystem. During the network’s initial growth phase, a pre-allocated Bootstrap Rewards Pool of LAVA tokens supplements provider rewards to ensure economic viability before consumer demand reaches sustainable levels. This pool gradually diminishes as organic demand increases, creating a smooth transition to market sustainability. Consumer Subscription Fees form another critical revenue stream, as applications and users purchase subscription plans that grant access to RPC services across multiple chains. These subscription fees flow directly to providers based on their service provision, measured in Compute Units (CUs).

Provider rewards are not distributed equally but follow a Quality-Based Distribution system weighted according to several factors. Providers with higher stake receive proportionally more pairing opportunities, incentivizing capital commitment to the network. Those with higher Quality of Service (QoS) metrics—including better uptime, lower latency, and more accurate responses—receive higher rewards, creating a meritocratic system that rewards technical excellence. Additionally, providers in underserved regions may receive boosted rewards to encourage global coverage, ensuring the network maintains geographic diversity.

Lava’s unique Dual Staking Mechanism allows the same token stake to simultaneously secure the validator network and the provider network, optimizing capital efficiency while maintaining security. This approach reduces the capital requirements for participation while maintaining strong security guarantees. Finally, Slashing Conditions ensure accountability, as providers who deliver incorrect data, experience excessive downtime, or engage in malicious behavior face slashing penalties, creating strong disincentives for poor service. The economic model is designed to create a virtuous cycle: as more consumers join the network, provider rewards increase, attracting more providers and improving service quality, which in turn attracts more consumers.

5.1.2 Cross-Chain Provider Participation

To evaluate the effectiveness of Lava’s incentive model, we analyzed provider participation across multiple blockchain networks supported by Lava.

The data demonstrates robust provider participation across multiple chains, with established networks attracting between 4 to active providers each. This level of participation ensures sufficient decentralization and redundancy for reliable RPC service. The correlation

Table 1: Chain Statistics and Provider Rewards

Chain	# Pr.	# Req.	Rew.(\$)
Near	28	60.72B	26,620
Arbitrum	24	3.09B	15,208
Base	15	3.01B	11,404
Solana	5	5.14B	11,404
Polygon	10	28.94M	11,404
BSC	13	1.01B	11,404
Optimism	14	1.17B	11,404
Ethereum	27	17.6B	11,404
Lava	25	822.68M	9,124
BSC Testnet	5	8.90M	2,273
Base Sepolia	5	4.28M	2,273
Optimism Sepolia	7	4.08M	2,273
Arbitrum Sepolia	12	528.32M	1,513
Cosmos Testnet	4	133.92M	1,133
Movement Testnet	8	24.67M	1,133
Axelar Testnet	30	3.47B	1,133
Stargaze	15	105.27M	373
Evmos	16	4.65B	373

between request volume and incentives shows that the economic model successfully directs rewards to chains with higher demand.

5.2 Benchmarking Methodology

To evaluate the performance and reliability of Lava’s decentralized RPC infrastructure for Sui, we conducted comprehensive benchmarking using `k6`, an open-source load testing tool. Our testing framework was designed to compare Lava’s RPC performance against existing RPC provider, and measure Lava Network overhead when redirecting existing RPC. We measured **Latency** through Response time for various RPC methods : `multiGetObjects`, which fetches multiple objects in a single call, `getLatestCheckpoint`, which retrieves the latest checkpoint information, and `getReferenceGasPrice`, which Retrieves the current reference gas price.

We also benchmark the **Throughput** with Maximum requests per second without degradation, and the **Consistency** with the variation in response times across multiple requests

The benchmarking utilized the `k6_test.js` script, which implements a structured testing methodology that defines metrics for tracking performance, configures test scenarios with controlled request rates, executes parallel requests to both Lava and centralized endpoints, measures response times and error rates, and generates comprehensive performance reports.

Tests were conducted using a controlled environment with one local full node running under Docker (Provider 1) and an external RPC provider (Provider 2) using <https://sui-mainnet.nodeinfra.com>. The test executed

300 RPC calls at a rate of 10 requests per second over a 30-second period.

5.3 Benchmarking Results and Analysis

Table 2: Performance Comparison: Lava vs. Centralized Provider

Metric	Lava	Others	Diff (%)
Overall Latency (ms)			
Average	46.20	48.27	-4.28%
Minimum	20.85	24.22	-13.91%
Maximum	122.38	401.86	-69.55%
Median	42.35	35.48	+19.36%
Method-Specific Avg. Latency (ms)			
<code>multiGetObjects</code>	49.35	47.60	+3.67%
<code>getLatestCheckpoint</code>	42.50	48.49	-12.35%
<code>getReferenceGasPrice</code>	46.77	48.73	-4.03%
Error Rate (%)	0.00	0.00	0.00%

The performance comparison between Lava Network’s decentralized RPC infrastructure and the centralized alternative for Sui blockchain reveals several key insights. Across 300 total RPC calls with a 0% error rate, Lava achieved an average latency of 46.20ms compared to 48.27ms for the centralized provider, representing a 4.28% performance advantage.

While the centralized provider showed a slightly better median latency (35.48ms vs 42.35ms), Lava demonstrated significantly better consistency with a maximum latency of 122.38ms compared to the centralized provider’s 401.86ms—a 69.55% improvement that indicates Lava avoids the severe performance spikes that can disrupt application experiences.

Method-specific analysis revealed varying performance characteristics across different RPC calls. For the `multiGetObjects` method, Lava was marginally slower with a 3.67% overhead (49.35ms vs 47.60ms). However, Lava demonstrated clear advantages in other critical methods, with `getLatestCheckpoint` showing a 12.35% performance improvement (42.50ms vs 48.49ms) and `getReferenceGasPrice` performing 4.03% faster (46.77ms vs 48.73ms). These performance advantages in state-critical methods are particularly valuable for applications requiring real-time blockchain state information.

The overall performance metrics indicate that Lava Network provides a viable and sustainable decentralized alternative to centralized RPC providers for the Sui blockchain ecosystem. The combination of competitive performance, reduced maximum latency spikes, and consistent reliability across different method types demonstrates that decentralization does not come at the cost of performance.

6 Conclusion

The integration of Sui blockchain with Lava Network represents a significant advancement in decentralized RPC infrastructure. By addressing the critical challenges of centralization, reliability, and economic sustainability, this solution enables the Sui ecosystem to scale while maintaining its decentralization principles.

The technical implementation leverages Lava’s sophisticated protocol to create a network of incentivized Sui Full Node operators, ensuring high availability, geographic distribution, and fault tolerance. The security mechanisms, including cross-verification and conflict resolution, protect against malicious behavior and ensure data integrity.

Most importantly, the economic model creates sustainable incentives for Sui Full Node operators, solving the fundamental problem of RPC infrastructure funding. As the Sui ecosystem continues to grow, this decentralized RPC layer will become increasingly valuable, supporting the next generation of decentralized applications with reliable, censorship-resistant blockchain access.

This approach not only benefits the Sui ecosystem specifically but also demonstrates a viable model for decentralized infrastructure that could be applied to other blockchain networks facing similar challenges.

A APPENDIX: Lava Security and Reliability Mechanisms

A.1 Data Reliability System

Lava implements a comprehensive data reliability system to ensure the accuracy and consistency of RPC responses. For deterministic APIs, consumers can verify responses by comparing data from multiple providers through cross-verification. The specification defines a reliability threshold that determines how frequently verification occurs, approximately 1 in 16 requests for Sui. Additionally, the specification includes parameters for block finality and synchronization requirements to ensure data consistency. All provider responses include cryptographic signatures, creating undeniable proof of the data they provided, which forms the foundation of Lava's accountability system.

A.2 Conflict Resolution

When conflicting responses are detected, Lava employs a structured resolution process. Consumers can submit a "conflict transaction" containing the original query and multiple signed responses, providing evidence of the discrepancy. A jury of providers is then selected to evaluate the conflict and determine the correct response based on consensus rules. Providers found to have provided incorrect data face penalties, including slashing of staked tokens and potential jailing, creating strong economic disincentives for dishonest behavior.

A.3 Quality of Service Monitoring

Lava continuously monitors provider performance through multiple mechanisms. Response time tracking measures the latency of provider responses, while availability checks detect when providers are unresponsive. The system also incorporates consumer reports, allowing users to report quality issues they encounter. These metrics are combined into an Excellence Quality of Service Metric, a composite score that influences provider pairing probability and ultimately affects provider rewards.

B Provider Misbehavior Detection

B.1 Types of Provider Misbehavior

Lava's security mechanisms address several types of provider misbehavior. Data manipulation involves altering transaction data or responses to benefit the provider or harm users. Censorship occurs when providers selectively refuse to process certain transactions, undermining network neutrality. Eclipse attacks attempt to isolate a consumer by controlling all their provider connections. Denial of service happens when providers refuse to respond or deliberately respond slowly. Inconsistent data provision involves providing different responses to different consumers for the same query, which can lead to consensus failures.

B.2 Cross-Verification Process

The cross-verification process works through a decentralized verification mechanism. A consumer sends a request to a provider in their pairing list, and based on the reliability threshold, some requests are randomly selected for verification. For these verification requests, the consumer sends identical queries to multiple providers and compares the responses to detect inconsistencies. If inconsistencies are found, the consumer has cryptographic evidence of misbehavior. This process is entirely decentralized, with verification occurring at the consumer level rather than relying on a central authority.

B.3 Cryptographic Evidence

Each response from a provider includes a cryptographic signature, creating undeniable evidence of the data they provided. This signature serves multiple purposes: it prevents providers from denying they sent a particular response, allows consumers to prove they received conflicting data, creates an audit trail for conflict resolution, and enables the blockchain to verify the authenticity of reported conflicts.

C Conflict Resolution Mechanism

C.1 Conflict Transaction Submission

When a consumer detects conflicting responses, they create a "conflict transaction" containing the original query, multiple signed responses from different providers, block height information, and a consumer signature. This transaction is submitted to the Lava blockchain, triggering the conflict resolution protocol and initiating the formal dispute process.

C.2 Jury Selection and Voting

The conflict resolution protocol involves a structured jury process. A set of providers within the relevant specification is selected to serve as jurors. The process uses a commit-reveal scheme where jurors first commit to their vote (correct response) using a cryptographic commitment scheme, then reveal their votes along with the commitment opening. Votes are tallied to determine the correct response, and the provider(s) found to have provided incorrect data face penalties. This commit-and-reveal voting scheme prevents jurors from simply copying each other's votes and ensures the integrity of the voting process.

C.3 Byzantine Fault Tolerance

The jury mechanism is designed to be Byzantine fault-tolerant, meaning it can reach correct conclusions even if some jurors are malicious, as long as the majority are honest. This is achieved through random jury selection to prevent collusion, sufficient jury size to ensure statistical security, economic incentives for honest voting, and penalties for non-participation or provably dishonest voting.

D Penalty Mechanisms

D.1 Slashing

Providers found guilty of providing incorrect data may face slashing penalties where a portion of their staked tokens is confiscated. The severity of slashing depends on the nature and frequency of the offense, with repeated offenses resulting in progressively harsher penalties. This economic disincentive helps maintain the integrity of the network.

D.2 Jailing

Providers may be "jailed" (temporarily or permanently removed from the provider pool) under certain conditions: providing provably incorrect data, failing to respond to a significant number of requests, accumulating too many negative Quality of Service reports, or claiming more compute units than verified by consumers. Jailed providers cannot receive new pairings until they are "un-jailed" through a governance process or automatic time-based release.

D.3 Reputation Impact

Beyond direct penalties, provider misbehavior affects their reputation through reduced Excellence Quality of Service metric, lower probability of being selected in future pairings, decreased attractiveness to delegators, and reduced rewards due to lower compute unit claims. This reputation system creates long-term incentives for honest behavior.

E Economic Model and Incentives

E.1 Provider Rewards

The economic model incentivizes Sui Full Node operators to provide reliable RPC services through multiple reward mechanisms. Compute Unit Rewards allow providers to earn based on the compute units processed for consumers. Subscription Revenue from consumer subscription fees is distributed to providers based on their participation. The delegation system enables token holders to delegate to providers, increasing their effective stake and rewards. Quality Bonuses ensure that providers with higher quality metrics receive more pairings and thus more rewards.

E.2 Sustainable Economics for Full Node Operators

This economic model addresses the fundamental issue of sustainable Full Node operation by creating a viable business model. Cost recovery helps offset the infrastructure costs of running a Sui Full Node. Profit potential enables high-quality providers to earn beyond their operational costs. Capital efficiency through dual staking allows the same tokens to secure both the Lava network and provider services. Scalable incentives ensure that as demand for Sui RPC services grows, so do the rewards for providers.

F Future Developments

F.1 Sui gRPC Support

As Sui transitions from JSON-RPC to gRPC, Lava is preparing to support this evolution with several initiatives. A new gRPC specification for Sui is being developed to support the new API format. Streaming support will enhance Lava's protocol to support gRPC streaming for real-time data access. During the transition period, backward compatibility will be maintained with both JSON-RPC and gRPC specifications supported.

F.2 Enhanced Provider Features

Future enhancements to the provider ecosystem include advanced load balancing for more sophisticated load distribution across provider nodes. Specialized API support will allow providers to offer premium services for specific API subsets. Historical data services will extend storage and indexing of historical Sui data. Custom data services will provide value-added services beyond basic RPC access, further expanding the utility of the Lava network.