Spheron: On-Demand DePIN for GPUs

Spheron Network

June 16, 2024

Abstract

The digital landscape is undergoing a profound transformation, with the surge in demand for GPU resources for AI and machine learning applications revealing critical supply chain bottlenecks. These challenges have led to increased costs and restricted access, impeding innovation and development in these critical technological areas. This whitepaper introduces a groundbreaking decentralized architecture designed to revolutionize the traditional GPU compute market. By leveraging a novel incentive mechanism for the sharing of GPU resources, this framework aspires to democratize access to computational power, mitigate costs, and stimulate innovation across the AI and machine learning sectors. This architecture not only promises to alleviate current market constraints but also paves the way for a more inclusive and efficient computational ecosystem. Through the decentralization of GPU resources, we envision a future where developers, researchers, and innovators have unfettered access to the computational resources needed to drive progress and breakthroughs in AI and machine learning. This initiative marks the inception of the first Ethereum Virtual Machine (EVM)-based GPU and compute marketplace, leveraging the EVM Chain to introduce a novel economic model that incentivizes both suppliers and users. This work:

- Presents a comprehensive overview of the proposed decentralized architecture, detailing its operational mechanisms and the benefits it offers to both GPU resource providers and consumers.
- Discusses the integration of smart contracts for transaction management, escrow services, and reward distribution, ensuring a transparent and equitable ecosystem.
- Explores future research directions aimed at expanding the network's capabilities, including the integration of underutilized compute resources from smaller devices and enhancing network security and reliability.
- Formalizes the concept of decentralized GPU resource sharing and its potential to disrupt traditional compute markets, offering a new paradigm for resource allocation and utilization in the digital age.
- Examines potential use cases and the broader implications of the architecture for the development and deployment of AI and machine learning applications.

By charting a course toward the decentralization of GPU resources, this whitepaper lays the foundation for a more accessible, cost-effective, and innovative computational landscape, heralding a new era of development in AI and machine learning technologies.

Note: Spheron is a work in progress. Active research is underway, and new versions of this paper will appear at https://spheron.network. For comments and suggestions, contact us at info@spheron.network.

1. Introduction

The rapid evolution of Artificial Intelligence (AI) and Machine Learning (ML) technologies has brought to light the critical need for scalable and accessible computational resources. In response to this demand, we introduce the Spheron protocol, a groundbreaking decentralized architecture designed to revolutionize the allocation and utilization of GPU and compute resources. This protocol, leveraging blockchain technology, aims to democratize access to these resources, mitigate supply chain bottlenecks, and establish a transparent and equitable economic model for all participants. Through the Spheron protocol, we envision a future where computational power is not a bottleneck but a catalyst for innovation and growth in AI and ML domains.

1.1 Elementary Components

The Spheron protocol is underpinned by four innovative components, each contributing to the robustness and efficiency of the Decentralized Compute Network (DCN).

1. **Decentralized Compute Network (DCN):** The DCN forms the backbone of the Spheron protocol, offering a distributed framework where independent providers supply GPU and compute resources. This network ensures resilience, scalability, and accessibility, catering to the diverse needs of AI and ML projects.
2. **Onchain Supply Market:** Central to the protocol is the Onchain Supply Market, a platform where compute and GPU resources are listed, traded, and allocated. This market operates transparently on the blockchain, facilitating fair and open access to computational power.
3. **Transparent Economy:** The Spheron protocol introduces a transparent economy underpinned by a native token system. This economy ensures clear and fair compensation for resource providers and straightforward expense management for users, fostering a trust-based ecosystem.

1.2 Protocol Overview

1. **Decentralized Network Infrastructure:** At its core, the Spheron protocol is built as a Decentralized Compute Network (DCN) on a blockchain framework, complete with a native token. This structure enables clients to utilize tokens for accessing vital compute and GPU resources, while providers earn tokens by contributing these resources to the network.
2. **Node Operation and Deployment Orchestration:** The protocol's efficiency is driven by the Matchmaking Engine and Provider nodes, which execute crucial functions to orchestrate deployments seamlessly via onchain transactions. This permissionless operation ensures that deployments are managed efficiently and transparently.
3. **Onchain Payment Mechanism:** Payments within the Spheron protocol are executed onchain, ensuring timely compensation for providers and maintaining the transparency economy's integrity. This system streamlines the financial interactions within the network, providing clarity and trust in transactions.
4. **Incentivization and Rewards:** A key feature of the protocol is the incentivization of Provider nodes. By contributing compute and GPU resources, and facilitating efficient provider selection through matchmaking engine, providers earn rewards, encouraging active participation and investment in the network's growth.

The Spheron protocol, with its innovative components and operational framework, sets a new standard for decentralized compute networks. By addressing current limitations and providing a scalable, secure, and transparent platform, it paves the way for the next generation of AI and ML development.

2. Background and Motivation

As Artificial Intelligence (AI) and Machine Learning (ML) continue to evolve, the demand for high- powered Graphics Processing Units (GPUs) surges, revealing stark inefficiencies in the current supply and cost structure. These challenges disproportionately affect smaller entities and independent researchers, hindering innovation due to the scarcity and high expense of essential computational resources. This whitepaper introduces the Spheron protocol, a decentralized network designed to revolutionize GPU resource allocation. By enabling efficient distribution, Spheron aims to democratize access and foster an ecosystem ripe for AI advancements.

The **Computational Resource Dilemma** faced today is characterized by:

- **Limited Availability:** Access to necessary hardware through cloud services like AWS, GCP, or Azure is often delayed, with high-demand GPU models frequently unavailable.
- **Restricted Options:** Users are confined to the limited selections provided by centralized services, affecting their ability to tailor projects to specific needs.
- **Prohibitive Costs:** The financial burden of acquiring quality GPUs for AI tasks is significant, potentially running into hundreds of thousands of dollars monthly.

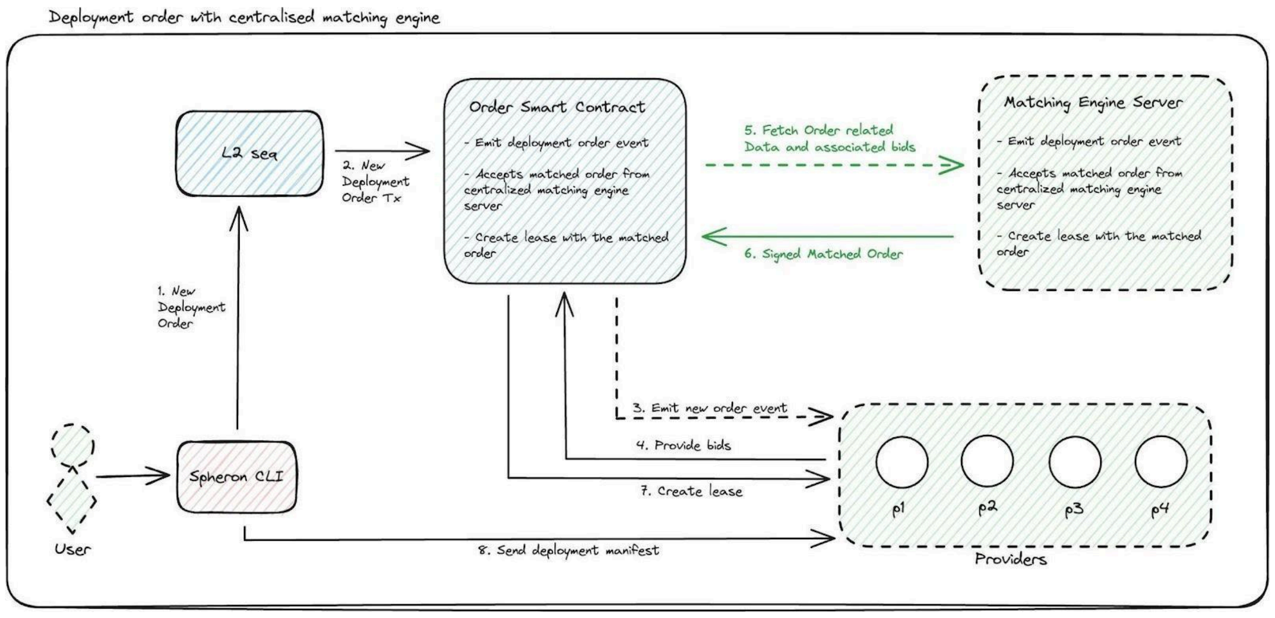
Spheron's decentralized computing network proposes a solution by aggregating underutilized GPUs from diverse sources, forming a Decentralized Physical Infrastructure Network (DePIN). This initiative promises scalable, customizable, and cost-effective access to computational power, addressing the critical challenges of availability, choice, and expense. Envisioned as the "digital oil" of technology, Spheron aims to make $SPON the currency of compute, creating a new paradigm for accessing and leveraging computational resources. This decentralized approach seeks to break down barriers to innovation, making GPU resources more accessible and enabling a new era of AI and ML development.

3. Architecture Overview

3.1. Overview

In developing a robust and fully functional decentralized compute and GPU network, our deliberations have focused on various critical components such as provider registration, order matching, provider incentivization, and payment settlements. A key aspect of our discussion also revolves around the classification of providers within the network, which comprises two primary types: Fizz Node and Provider Nodes. The subsequent sections will detail the architecture involved in deployment and matching orders specifically for Provider Nodes.

3.2. Deployment Order Workflow without Eigen AVS based matching engine



Deployment order with centralised matching engine
L2 seq
2. New Deployment Order Tx
Order Smart Contract
- Emit deployment order event
- Accepts matched order from centralized matching engine server
- Create lease with the matched order
5. Fetch Order related Data and associated bids
6. Signed Matched Order
Matching Engine Server
- Emit deployment order event
- Accepts matched order from centralized matching engine server
- Create lease with the matched order
1. New Deployment Order
3. Emit new order event
4. Provide bids
7. Create lease
8. Send deployment manifest
Spheron CLI
User
p1
p2
p3
p4
Providers

Figure 1: Deployment order with a centralized matching engine

This architecture is centered around a matchmaking engine, a pivotal component within the network. This engine processes order details and provider bids, selecting the optimal provider for each deployment based on specific parameters outlined in Section 4.2.3. After a match is made, the order smart contract initiates a lease with the selected provider. Subsequently, the user transmits the deployment manifest to initiate workload deployment on the designated provider over MTLS connection. Upon completion, the user can verify the deployment status and access all relevant deployment details such as connection URLs and port mappings.

Although this architecture offers an efficient order-matching mechanism, its major drawback is its centralized nature. The matchmaking engine, introduces potential points of failure—any server downtime or scaling issues could result in a backlog of stagnant deployment orders. To address this vulnerability and achieve true decentralization, Section 3.3 introduces an Eigenlayer AVS-based architecture for the matching engine.

3.3. Deployment Order Creation with Eigen AVS based matching engine

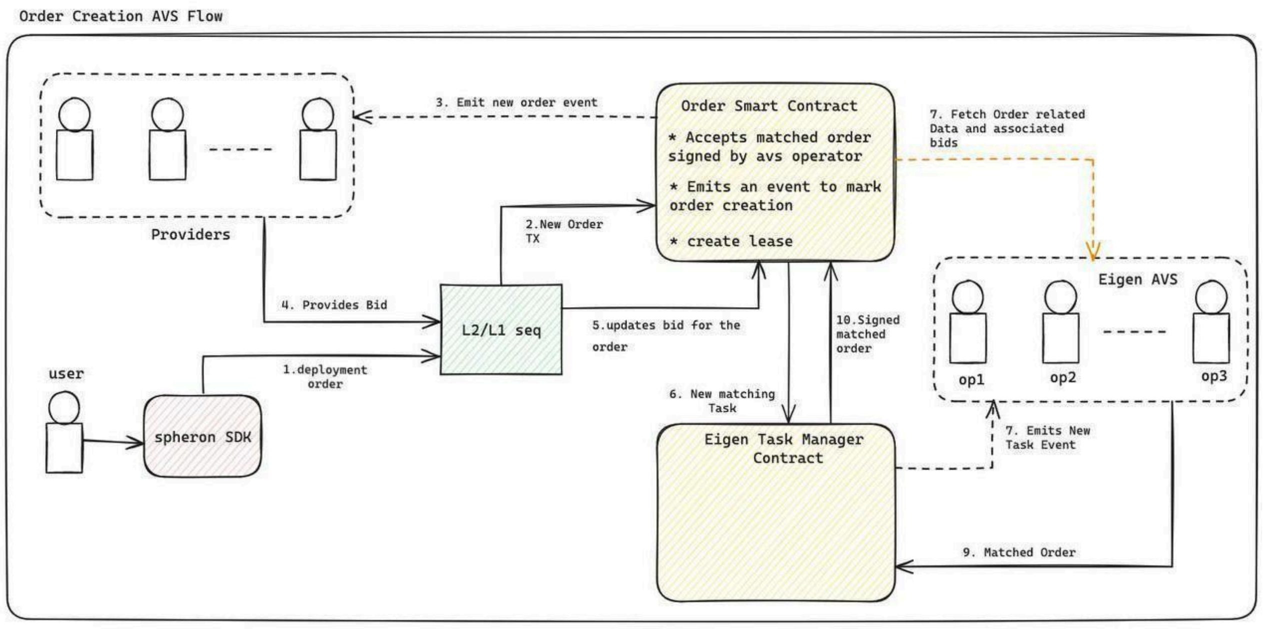
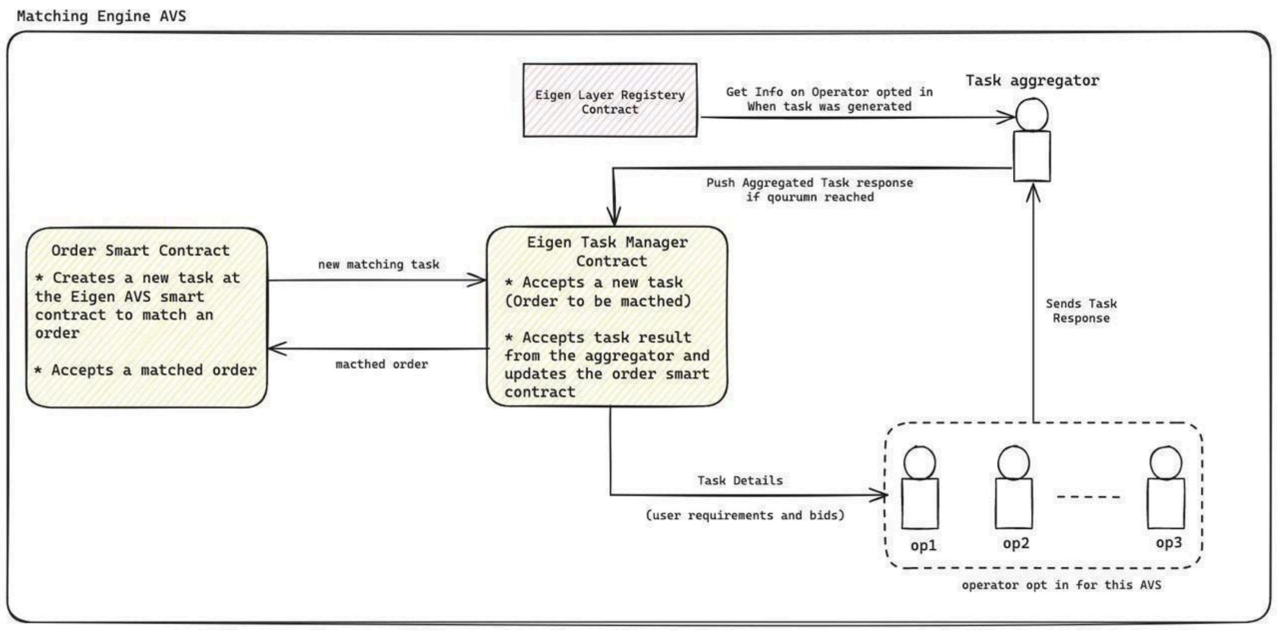


Figure 2: Deployment Order creation with Eigen AVS architecture

This architecture represents a fully decentralized evolution of the previous system, incorporating a matching engine built with the Eigen AVS architecture. This system not only provides economic efficiency but also integrates Ethereum-based trust into the network of matching engine operators.

The matching engine extends the previous architecture with operators who join the Spheron Matching AVS. When a new matching task arises, these operators activate the matching algorithm, which is derived from the parameters mentioned in Section 4.2.3. The algorithm evaluates bids from various providers against the deployment order and selects the most suitable provider based on user requirements. Once a quorum is achieved within the AVS, the validated matched order is sent back to the Order Smart Contract, which then establishes a lease with the provider on behalf of the user. The user then sends the deployment manifest via MTLS to initiate workload deployment on the cluster. Following completion, the user confirms the deployment status and retrieves all necessary deployment details such as connection URLs and port mappings.

3.4. Matching engine with Eigen AVS

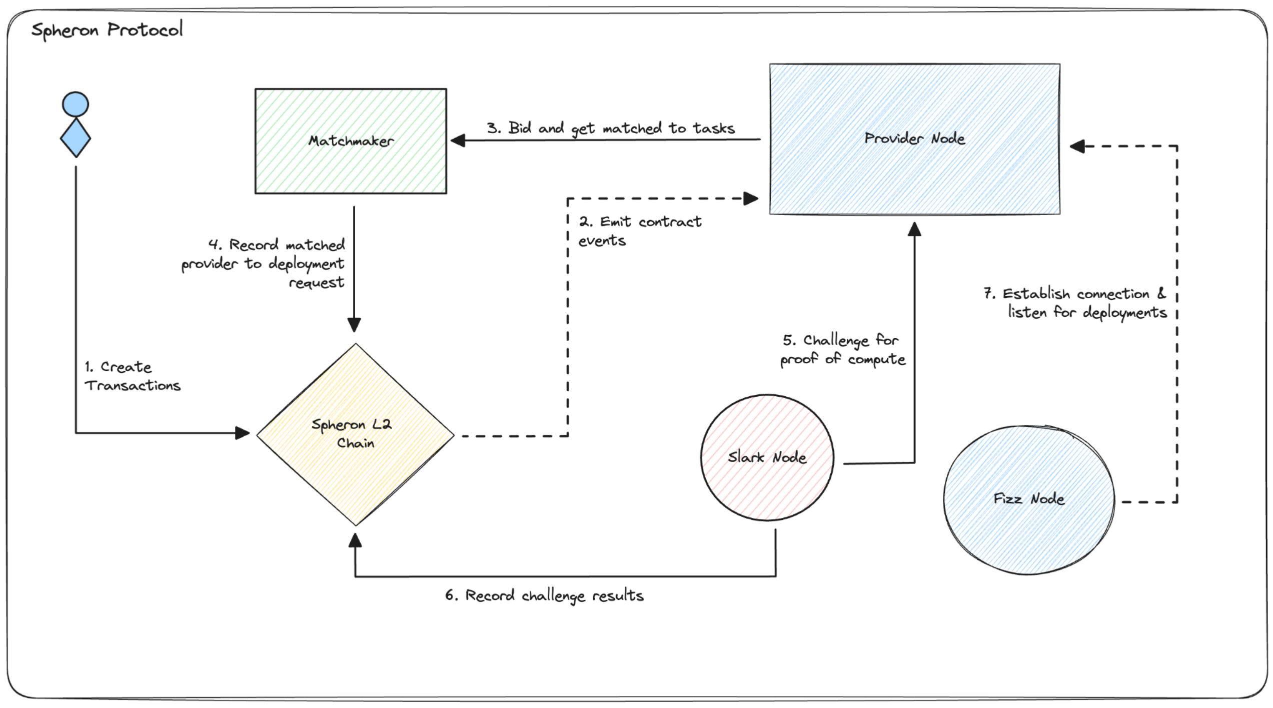


Matching Engine AVS
Eigen Layer Registery Contract
Get Info on Operator opted in When task was generated
Task aggregator
Push Aggregated Task response if qourumn reached
Order Smart Contract
* Creates a new task at the Eigen AVS smart contract to match an order
* Accepts a matched order
new matching task
macthed order
Eigen Task Manager Contract
* Accepts a new task (Order to be macthed)
* Accepts task result from the aggregator and updates the order smart contract
Sends Task Response
Task Details
(user requirements and bids)
op1
op2
op3
operator opt in for this AVS

Figure 3: Matching engine architecture in AVS

This detailed architecture showcases the decentralized matching engine operating under the Eigen AVS framework. The Eigen Task Manager Contract receives a new task, which is a deployment order that needs to be matched with a provider, and emits an event for all operators who have opted into this AVS to start the matching algorithm. Once a match is made, the task response is sent to the Task Aggregator. The aggregator gathers all relevant information on the opted-in operators from the Eigen Layer Registry when the task was generated, and checks if the quorum has been met. If the quorum threshold is achieved, the aggregated task response is pushed back to the Eigen Task Manager Contract, which then updates the Deployment Order smart contract with the matched provider for the corresponding deployment order.

3.5. Overview of Spheron Protocol Architecture and Component Interactions



Spheron Protocol
Matchmaker
3. Bid and get matched to tasks
Provider Node
2. Emit contract events
4. Record matched provider to deployment request
7. Establish connection & listen for deployments
5. Challenge for proof of compute
1. Create Transactions
Spheron L2 Chain
Slark Node
Fizz Node
6. Record challenge results

Figure 4: Spheron Protocol - Full flow & components

This section outlines the complete workflow and interaction of all components within the Spheron Protocol. These components work in synergy to ensure the protocol functions effectively. Users or application builders interact with the Spheron Chain to initiate transactions for deployment or payments. Provider Nodes, listening on the chain, respond by executing the necessary actions within their Kubernetes clusters.

Slark nodes play a critical role in maintaining the quality of service and uptime for Provider Nodes and their associated Fizz Nodes. This oversight ensures that users experience no disruptions with their deployments, encouraging continuous use of the protocol.

The matchmaking process is pivotal in pairing deployments with the most suitable Provider Node. The matchmaker ensures that the selected provider is the best fit for the deployment, enhancing the user's experience by ensuring that their deployment is managed securely and efficiently. This system is fully automated and abstracted, allowing users to concentrate solely on their deployment needs and specifications adjustments.

Fizz Nodes, a lightweight version of Provider Nodes, form an essential part of the network. Anyone with a smaller device, such as a laptop or a small server, can operate as a Fizz Node. These nodes establish connections with Provider Nodes to receive deployments and workloads from users. Multiple Fizz Nodes can link to a single Provider Node, forming a subnet. Each subnet can develop its own economy, govern itself, and earn rewards from the network, thereby contributing to the overall dynamism and scalability of the Spheron Protocol.

4. Detailed Component Analysis

4.1. Decentralization and Blockchain Infrastructure

4.1.1. Overview

The advent of blockchain technology heralds a transformative era for computing networks, marked by an unequivocal shift towards decentralization. This paper outlines the architectural foundations of an EVM based GPU supercompute network, employing the EVM chain to deliver a lower gas fees, increased gas limits, and significantly higher TPS. Furthermore, we are building a novel decentralized matching engine utilizing the Spheron protocol with **Actively Validated Services** (AVS) allows for both economic efficiency and the decentralized trust inherent in Ethereum Restaking.

4.1.2. Integrating DePIN within an EVM-based Ecosystem

The foundational aspiration to forge a sovereign, permissionless blockchain, underpinned by the security guarantees of the Ethereum ecosystem, is actualized through the integration of a Decentralized Public Infrastructure Network (DePIN) protocol. This integration not only epitomizes our commitment to fostering a decentralized network architecture but also leverages the intrinsic advantages of EVM compatibility, thereby facilitating seamless interoperability with extant decentralized applications (dApps) and infrastructure. This strategic amalgamation is anticipated to catalyze network expansion, engendering a fertile ground for liquidity augmentation through diverse dApps.

4.1.3. Network Operational Dynamics

At the operational core of the network are base modular contracts, deployed at genesis, which afford a foundation for the development of modular dApps, specifically tailored for compute and GPU resource allocation within the network. These contracts encapsulate essential functionalities, notably Provider Proposal & Registry, Deployment Order, Staking Manager, Payment, and Rewarder modules, each meticulously engineered to address distinct facets of network operations:

- **Provider Proposal:** These contracts are responsible for the initial proposal and subsequent registry of new providers within the network. They act as the gatekeeper, ensuring that only qualified providers are accepted into the network based on assessments conducted by the governance body, DAO, or Admins. This mechanism helps curate the provider base, preventing bad actors from entering the network.

  Note: *Initially, to lower entry barriers and encourage network growth, all provider proposals will be automatically approved. This policy may be revisited and adjusted via governance proposals once a sufficient number of providers are bootstrapped and capable of performing curation roles.*

- **Provider Registry:** This contract serves as a comprehensive repository for all provider-related information within the network. It meticulously records essential details such as the provider's capacity, geographic region, availability, type, and service tier, among other attributes. This structured storage of provider data ensures that the network can efficiently manage and access critical information needed for the optimal matching of resources with user requests.
- **Staking Manager:** This module facilitates the staking of $SPON tokens within the network. While primarily utilized by providers, its functionality could be extended to support other significant staking activities within the ecosystem.
- **Deployment Contracts**: These contracts manage the logistics of deployments, including the preservation of state and records of user ownership on the blockchain. They are equipped with event emission capabilities, which are crucial for synchronizing nodes and initiating deployments.
- **Payment Contracts:** Serving as the financial backbone, these contracts manage all financial transactions related to deployments. They ensure a transparent and efficient allocation of funds, predicated on compute resource utilization, thereby safeguarding against premature deployment termination due to insufficient funds.
- **Challenge Contracts:** These contracts serve as records of all the challenges and their outcomes conducted by Slark Nodes and Provider Nodes within the network. Based on these record, the rewards and slashes will be performed every Era.
- **Rewarder Contracts:** These are intrinsically linked to the Escrow Fund contracts and are responsible for the equitable distribution of rewards, in the form of $SPON tokens or alternative currencies, to providers and matchmakers. This mechanism incentivizes network participation and contribution, with an automated reward calculation system facilitated by Oracle integration.

4.2. Matchmaking Mechanism: Facilitating Decentralized Resource Allocation

4.2.1. Overview

Within the ecosystem of a decentralized compute network, the Matchmaker Mechanism stands as a crucial component, orchestrating the dynamic allocation of compute resources between demand (deployment requests) and supply (provider nodes). This mechanism is underpinned by an EigenLayer's **Actively Validated Services** (AVS) framework, which incorporates a sophisticated matchmaking consensus algorithm designed to selectively pair each deployment request with the most suitable provider. This paper details the operational dynamics, economic incentives, and the complex consensus parameters that define the functionality of the Matchmaking Engine.

4.2.2. Operational Dynamics of the Matchmaker Operator

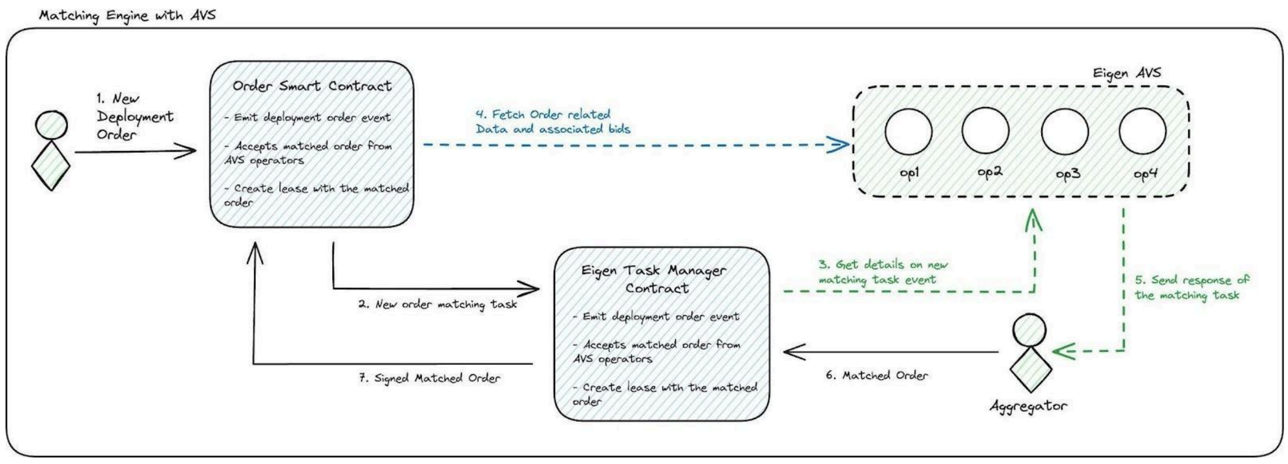

Matching Engine with AVS
1. New Deployment Order
Order Smart Contract
- Emit deployment order event
- Accepts matched order from AVS operators
- Create lease with the matched order
4. Fetch Order related Data and associated bids
Eigen AVS
op1
op2
op3
op4
2. New order matching task
Eigen Task Manager Contract
- Emit deployment order event
- Accepts matched order from AVS operators
- Create lease with the matched order
3. Get details on new matching task event
5. Send response of the matching task
7. Signed Matched Order
6. Matched Order
Aggregator

Figure 5: Matchmaking engine full workflow

The Matchmaker Mechanism is vital for facilitating the deployment of computing resources on the network. While there are various approaches to constructing decentralized matching engines on EVM chains, significant challenges include limitations in TPS, scalability, and the time constraints associated with deployment orders. To address these issues, we have adopted an AVS-based architecture for our matching engine, leveraging the **Economic and Ethereum Inclusion Trust** afforded by the Eigen Layer AVS protocol. This protocol is trusted economically due to ETH restaking and offers decentralization through a broad network of ETH validator operators.

When a user initiates a new deployment request on our protocol, they execute a function on the **Deployment Order Smart Contract** which then emits a **`PROVIDER_LOOKUP`** event. Providers listen for this event and submit their bids to the Deployment Order Smart Contract. After all bids have been collected, the contract initiates a matching task that captures the attention of all operators who then begin the process of selecting a provider based on predefined matching parameters. Once a quorum is reached, the aggregator consolidates the matching information from all operators and updates the matched order in the Task Manager contract. This contract then returns the signed matched order back to the Deployment Order contract which will create a lease for the matched order and emit **`CREATE_LEASE`** event.

The user can then verify the lease and then send the Deployment Manifest (containing all the configurations and secrets) to the matched provider, which the provider listens for and begins the server deployment on their cluster.

***Note:** As proponents of progressive decentralization, our initial approach involves a simple version of the fully decentralized matching engine. Initially, a single matching engine will process orders and identify the optimal provider for each order. As our network of Providers and Fizz Node matures, we plan to advance towards a fully decentralized matching engine, enhancing the network's autonomy and self-sufficiency.*

4.2.3. Matching Parameters for Provider Selection

The consensus mechanism at the heart of the Matchmaker Node is engineered to evaluate a myriad of parameters to ascertain the optimal provider node for each deployment request:

- **Region / Availability Zone:** Prioritizes geographical proximity to minimize latency and adhere to data residency regulations.
- **Price Delta:** Ensures economic efficiency by matching deployment budgets with competitive provider bids.
- **User Feedback:** Feedback given by the user regarding the provider, calculated in average of 10.
- **Resource Availability:** Assesses the current capacity of providers to fulfill the deployment requirements.
- **Slashes:** Considers any penalties incurred by providers for previous contract violations or failures.
- **Trust Tier:** The higher the trust tier of Provider Node, the greater the likelihood of securing a deployment lease.
- **Stakes:** The higher the stake amount, the greater the likelihood that a provider will be selected for deployments. This mechanism ensures that those who commit more significantly to the network are rewarded with increased opportunities for engagement and revenue.
- **Randomness:** Incorporates an element of unpredictability to safeguard against deterministic biases in the selection process.

These parameters are synthesized through a robust algorithm, culminating in the selection of a provider node that aligns with the deployment's stipulated criteria. Following this, the Matchmaker Node executes an on-chain transaction `matchOrder`, that creates a lease between order request and the provider selected, officially documenting the allocation and commencing the deployment process.

4.2.4. Economic Model and Incentive Structure

Matchmaking operators within the network currently engage in restaking to bolster economic trust among participants. To further enhance the trustworthiness and security of the network, we propose the introduction of dual staking involving the $SPON token. This dual staking mechanism will leverage both the existing restaking framework and $SPON tokens, thereby integrating an additional layer of commitment and reliability from the operators.

Moreover, the Spheron network recognizes the need to incentivize active participation in the matchmaking process. While the current system effectively rewards participation, there is an absence of a penalty mechanism for non-compliance or malperformance. To address this, we plan to implement a slashing mechanism for matchmaking operators once it becomes available. This slashing mechanism will serve to ensure that operators not only benefit from their active participation through rewards but also adhere strictly to network protocols to avoid penalties. This balance of incentives and penalties will promote a more robust and reliable matchmaking process within the Spheron network.

4.2.5. Network Event Synchronization and Transactional Integrity

Matchmaker Operator will have access to public Spheron Sequencer RPC node, enabling real-time synchronization with blockchain events such as `OrderCreated & BidPlaced` emitted by the Order Request & Bid contract. This ensures the Matchmaker Operator's decisions are informed by the most current data, thereby upholding transactional integrity and facilitating the execution of on-chain transactions.

The Matchmaker Operators represents a cornerstone of the decentralized compute network, embodying the principles of fairness, decentralization, and efficiency. Through its nuanced operational framework and consensus algorithm, it ensures that compute resources are allocated judiciously, fostering an environment where innovation and collaboration thrive. As the network evolves, the Matchmaker Node will continue to play a crucial role in building a fair market for providers and light providers and balancing the intricate dynamics between supply and demand within the decentralized computing landscape.

4.3. Provider Node: The Network's Computational Backbone

4.3.1. Overview

At the core of the decentralized compute network lies the Provider Node (PN) Node, a pivotal entity responsible for contributing both CPU and GPU resources to a communal pool. This section explores the architecture, operational mechanics, and incentive framework designed to sustain high network integrity and reliability. Through a combination of staking, rewards, tier system and a sophisticated deployment orchestration system based on Kubernetes, Provider Nodes are incentivized to offer their computational resources, thereby ensuring the network's robustness and scalability.

4.3.2. Architectural Overview

Provider Nodes form the backbone of the network, offering essential GPU and compute resources vital for powering a wide array of applications. Utilizing Kubernetes, these nodes orchestrate deployments, enabling the network to scale dynamically according to fluctuating computational demands. This orchestration capability not only streamlines deployment management but also significantly enhances the network's overall efficiency and responsiveness. Kubernetes, an open-source system at its core, provides a comprehensive range of tools that facilitate seamless service deployment and connectivity, and offers a composable system that enhances the protocol with advanced features.

We leverage the full potential of the Kubernetes system to manage user services in a multi-tenant environment. This includes numerous security measures and isolation between different user services. Although Kubernetes inherently addresses many of these concerns, extensive customizations have been implemented to perfect the system. The Provider Node binary interacts with the Kubernetes API to manage tasks such as creating pods with reserved resource specifications, restarting pods if a service fails, terminating pods, or updating them. These actions are all facilitated through the Kubernetes API. Moreover, the provider binary manages pricing details and interacts with the blockchain to record bids or close leases when services are not running or are restarting abnormally.

The provider binary also addresses challenges posed by Slark Nodes. It responds to these challenges by creating a pod with the specifications outlined in the challenge and then shares the results with the Slark Node.

With the introduction of the Fizz Node, a Provider Node (PN) can function as a subnet gateway for the Fizz Node (FN), beyond just managing deployment requests from the network. Any FN can establish a connection with an PN by delegating a minimum amount of $SPON tokens as determined by the PN, enabling it to receive deployment requests. Further details will be discussed in the Fizz Node section. The PN and all connected FNs will collectively be referred to as a Subnet.

4.3.3. Provider Registration

For providers to join the network, they must undergo a comprehensive verification process. This procedure is designed to ensure that only legitimate providers with adequate compute resources and no malicious intent are integrated into the network. The verification process involves several steps and is overseen by the decentralized autonomous organization (DAO) or governance body through a voting system.

1. **Provider Registration Proposal:** Providers interested in joining the network must first submit a registration proposal. This proposal includes detailed specifications about the provider's resources, such as region, compute capacity, tier, and type of compute.

2. **Governance Review:** The governance body reviews the proposal. Upon approval, the provider's details are registered in our Provider Registry smart contract. This registration triggers the creation of a new smart contract specific to the provider, updating the proxy address in the registry contract.

3. **Staking Requirement:** Following registration, providers are required to stake $SPON tokens. This stake activates the provider's bidding engine, allowing them to start submitting bids for new deployments. The stake depends on the Tier requirement.

Note: Initially, provider registration proposals will be automatically approved to lower barriers to entry and accelerate network growth. However, once the network reaches a certain level of maturity, we plan to introduce a curation process by submitting a governance proposal to disable the automatic approval mechanism.

4.3.4. Provider Verification Challenges

A persistent challenge within Decentralized Physical Infrastructure Networks DePIN is verifying that providers are genuinely offering the resources they claim rather than exploiting the system to farm tokens. This issue is particularly acute in compute and GPU networks, where providers might spoof resource availability to mine tokens without actually providing the promised hardware.

To protect our network against fraudulent activities and ensure that providers genuinely contribute the advertised compute and GPU resources, we propose implementing proof of capacity mechanisms with the help of **Slark Node Network** that we will be discussing in the next sections. The Slark nodes are responsible for challenging the Provider Nodes to verify the quality and reliability of their service. The Provider Nodes have to respond to the Slark nodes challenges and respond with a signed proof of capacity. These proof of capacity challenge is small random task with a specific spec for deployment in the provider and if the deployment failed then provider fails the challenge and the Slark get a chance to earn from the liveness reward slashing.

4.3.5. Tiering System for Provider Node

In our decentralized compute network, providers are categorized into two complementary tier systems: **Provider Attributes Tiers** and **Trust and Performance Tiers**. These systems are designed to ensure that the Matchmaker can select the most appropriate providers for user requests based on high trust and reliability, efficiency in service delivery, and the type of compute resources offered. This approach helps to allocate better liveness reward multipliers based on the provider's trust level within the network and the type of compute resources they provide.

By incorporating these tiering systems, the network creates an incentive structure that encourages providers to achieve better results, move up the tier ladder, and earn more rewards. This system also ensures that providers can recover their capital expenditure (CapEx) invested in hardware and maintain their infrastructure over the long term.

All the multipliers from both tier systems stack up together to calculate the rewards for each Era. However, these multipliers are not compounded. For example, if the attribute multiplier is 1.5x and the trust multiplier is 2x, the total reward calculation will be based on (1.5 + 2.0)x.

4.3.6. Provider Attributes Tiering

In our decentralized compute network, providers can specify various attributes that assist the Matchmaker in selecting an appropriate provider based on user requests. During the initiation phase, providers are encouraged to detail these attributes to optimize the matchmaking process. Here is a systematic breakdown of the different types of attributes that providers can declare:

4.3.6.1. Compute Tier

Providers categorize their compute servers into distinct tiers based on CPU performance scores, enabling a precise match with the computational demands of user projects:

- **High Performance Compute (HPC) Tier:** These are robust servers equipped with modern next- generation EPYC processors, characterized by CPU scores exceeding 50,000.
- **General Purpose Tier:** This tier includes older generation servers with CPU scores below 50,000, widely available and capable of handling a variety of general computing tasks.
- **Basic Tier:** Comprising servers with no more than 48 threads and CPU scores below 20,000, these units offer essential computing power at a lower cost.

4.3.6.2. Storage Tiers

Focused on data storage capabilities, this category is subdivided into:

- **HDD Storage Tier:** Servers predominantly equipped with Hard Disk Drives.
- **SSD Storage Tier:** Servers predominantly equipped with Solid State Drives.
- **NVME Storage Tier:** Servers outfitted with Non-Volatile Memory Express drives, offering faster data transfer rates.

4.3.6.3. GPU Tiers

GPU providers specify the performance tier of their hardware, which influences their suitability for different computational tasks:

- **Entry Tier:** Suitable for basic model inferencing, these GPUs are priced below approximately $1,000 and offer modest performance.
- **Low Tier:** Priced under approximately $2,000, these units are adequate for less demanding machine learning tasks and inferencing.
- **Medium Tier:** Commonly used in commercial applications for distributed training and model inferencing, with a price point under $5,000.
- **High Tier:** These premium GPUs, priced above $7,500, are scarce and predominantly utilized for training large language models (LLMs) and other intensive tasks.

- **Ultra High Tier:** These premium GPUs, priced above $15,000, are high capital intensive and predominantly utilized for training large language models (LLMs) and other intensive tasks.

There are ten distinct tiers for the GPU, based on the effectiveness of the AI workloads. These are further subdivided from five high-level tiers into more specific categories such as Entry 1, Entry 2, Low 1, etc., each with its own reward boost multiplier.

4.3.6.4. Connectivity Tier

Providers can also declare their connectivity capabilities, which are crucial for data-intensive tasks:

- **Low Speed:** Download speeds less than 100 Mbps and upload speeds less than 75 Mbps.
- **Medium Speed:** Download speeds less than 500 Mbps and upload speeds less than 400 Mbps.
- **High Speed:** Download speeds less than 1 Gbps and upload speeds less than 800 Mbps.
- **Ultra High Speed:** Download speeds exceeding 5 Gbps and upload speeds exceeding 5 Gbps.

4.3.6.5. Compliance

Providers should disclose their compliance with industry standards and regulations, enhancing trust and security for end-users:

Providers can indicate whether they meet standards such as **SOC2, HIPAA, or GDPR**. Choosing a compliant provider offers users enhanced security and data protection, and compliant providers receive incentives through a better utilization multiplier.

These attributes are integral to the protocol, guiding the allocation of resources and incentivizing providers based on their hardware capabilities and compliance status. The detailed provider attribute tier system and associated multipliers, which will be elaborated in phase one of our network deployment, are designed to reward providers proportionately to their investment in quality and compliance, thereby enhancing the overall efficiency and reliability of the network.

4.3.7. Trust and Performance Tiers

Provider Nodes are categorized into 7 tiers, with each tier representing a higher level of service quality and reliability. The Trust and Performance Tiers classify providers based on their reliability, performance, and trustworthiness. Higher tiers indicate greater trust and performance, leading to higher prioritization and rewards.

4.3.7.1. Tier 1 - Most Trusted Providers

- **Requirements**:
    - Maintain a 99% uptime every Era.
    - Maintain a 98% task completion rate every Era.
    - Maintain a avg. of < 100 ms response time every Era.
    - User Feedback should be > 4.5 (when implemented)
    - Staking requirement is > X1 (e.g., 500 tokens).
    - Required to submit valid compliance (SOC II) and audit reports of the servers.
- **Benefits**:
    - Highest priority in resource allocation
    - **Liveness reward boost: Highest (2.0x)**
    - Rebate in fees for transactions.

4.3.7.2. Tier 2 - Highly Trusted Providers

- **Requirements**:
    - Maintain a 98% uptime every Era.
    - Maintain a 96% task completion rate every Era.
    - Maintain a avg. of < 150 ms response time every Era.
    - User Feedback should be > 4.2 (when implemented)
    - Staking requirement is > X2 (e.g., 400 tokens).
- **Benefits**:
    - High priority in resource allocation
    - **Liveness reward boost: High (1.7x)**

4.3.7.3. Tier 3 - Trusted Providers

- **Requirements**:
    - Maintain a 97% uptime every Era.
    - Maintain a 94% task completion rate every Era.
    - Maintain a avg. of < 200 ms response time every Era.
    - User Feedback should be > 4.0 (when implemented)
    - Staking requirement is > X3 (e.g., 300 tokens).
- **Benefits**:
    - Moderate priority in resource allocation
    - **Liveness reward boost: Medium (1.5x)**

4.3.7.4. Tier 4 - Somewhat Trusted Providers

- **Requirements**:
    - Maintain a 95% uptime every Era.
    - Maintain a 90% task completion rate every Era.
    - Maintain a avg. of < 250 ms response time every Era.
    - User Feedback should be > 3.8 (when implemented)
    - Staking requirement is > X4 (e.g., 200 tokens).
- **Benefits**:
    - Moderate priority in resource allocation
    - **Liveness reward boost: Low (1.2x)**

4.3.7.5. Tier 5 - Less Trusted Providers

- **Requirements**:
    - Maintain a 90% uptime every Era.
    - Maintain a 85% task completion rate every Era.
    - Maintain a avg. of < 300 ms response time every Era.
    - User Feedback should be > 3.5 (when implemented)
    - Staking requirement is > X5 (e.g., 150 tokens).
- **Benefits**:
    - Lower priority in resource allocation
    - **Liveness reward boost: Low (1.1x)**

4.3.7.6. Tier 6 - Untrusted Providers

- **Requirements**:
    - Maintain a 85% uptime every Era.
    - Maintain a 80% task completion rate every Era.
    - Maintain a avg. of < 400 ms response time every Era.
    - User Feedback should be > 3.2 (when implemented)
    - Staking requirement is > X6 (e.g., 100 tokens).
- **Benefits**:
    - Low priority in resource allocation
    - **Liveness reward boost: No boost (1.0x)**

4.3.7.7. Tier 7 - Most Untrusted Providers

- **Requirements**:
    - Maintain a 75% uptime every Era.
    - Maintain a 75% task completion rate every Era.
    - Maintain a avg. of < 500 ms response time every Era.
    - User Feedback should be > 3.0 (when implemented)
    - Staking requirement is > X7 (e.g., 50 tokens).
- **Benefits**:
    - Lowest priority in resource allocation
    - **Liveness reward boost: No rewards**

Provider Tier Table

Tier	Total Staked Amount	Uptime (%)	Task Completion Rate (%)	Response Time (ms)	User Feedback (Rating)	Rewards Multiplier
1	**> X1**	**> 99**	**> 98**	**< 100**	**> 4.5**	**2.0**
2	**> X2**	**> 98**	**> 96**	**< 150**	**> 4.2**	**1.7**
3	**> X3**	**> 97**	**> 94**	**< 200**	**> 4.0**	**1.5**
4	**> X4**	**> 95**	**> 90**	**< 250**	**> 3.8**	**1.2**
5	**> X5**	**> 90**	**> 85**	**< 300**	**> 3.5**	**1.1**
6	**> X6**	**> 85**	**> 80**	**< 400**	**> 3.2**	**1.0**
7	**> X7**	**> 75**	**> 75**	**< 500**	**> 3.0**	**0.0**

4.3.6. Tier Transition Mechanism

- **Promotion:**
  - Provider Nodes can move up to a higher tier by consistently meeting or exceeding the requirements of the desired tier over 3 consecutive eras and then meeting the staking requirements.
- **Demotion:**
  - Provider Nodes can be downgraded to a lower tier if they fail to meet the requirements of their current tier for 5 consecutive eras or fail to maintain the required staking amount.

4.3.7. Incentive and Staking Mechanism

To actively participate in the network, Provider Nodes are mandated to stake $SPON tokens. This staking mechanism fulfills several critical roles: it secures a vested interest in the network's prosperity, establishes a method for penalizing misconduct, and synchronizes the objectives of providers with the overarching goals of the network. Acts of malfeasance such as prolonged downtime, delays in system upgrades, or misuse of user deployments incur penalties, including the slashing of stakes. This robust accountability framework is instrumental in bolstering network security and maintaining trust among participants.

4.3.7.1. Reward System for Provider Nodes

- **Utilization compensation**: Provider Nodes receive compensation based on the utilization of their computational resources by the users. These amounts are paid any token that Provider Node and User agreed upon during lease creation.
- **Liveness Reward:** The providers are rewarded with liveness rewards based on tiers and multipliers to ensure that providers remain active within the network, thereby preventing potential exits due to insufficient returns on investment (ROI) or inability to cover operational expenses (OpEx).
- **Bonus Reward:** The providers are incentivized to achieve certain lease hours milestones which will be given in the Era. For e.g. 100 Lease Hours / Era.

4.3.7.2. Foundation Delegation and Incentivization

To attract new providers, especially those with high-tier compute and GPU capabilities, foundation delegation serves as an initial incentive. This strategy is particularly impactful for providers who may require support to scale operations or achieve profitability. The delegated funds can be reclaimed by the foundation once the provider achieves a threshold of network utilization, effectively offsetting the initial support provided.

4.3.7.3. Delegation of $SPON Tokens and Rewards

The system also enables the delegation of $SPON tokens to Provider Nodes by token holders and Fizz Node aiming to strengthen network security. This mechanism increases a node's likelihood of being selected for deployments and also move them to higher tiers, enhancing potential earnings. The rewards given to the provider will be distributed between all the delegators based on their delegation on the provider . The Annual Percentage Rate (APR) for stakers is dynamic, influenced by factors such as the provider's liveness rewards during a given Era, and the network's inflation rate.

This comprehensive incentive and staking mechanism ensures that Provider Nodes are not only motivated to maintain high standards of operation but are also financially supported to sustain participation in the network, thereby aligning individual success with the network's overall health and longevity.

4.3.8. Slashing Mechanism

Providers are subject to penalties for any misconduct within the network, such as user data misuse or spoofing, as identified by the Spheron Team or the broader ecosystem. In such instances, the provider's $SPON collateral—which includes both staked and delegated funds—along with any accrued rewards, will be slashed. This measure is intended to provide restitution for any data breaches or other damages inflicted upon consumers.

Liveness Reward Slashing occurs under 2 different conditions of operations for nodes on the network. Either the node has degraded performance and failure to meet uptime guarantees during the deployment period for an end user or the node has uptime failure during the standby mode of operation. To prevent nodes from constantly connecting, disconnecting, and earning Liveness Rewards without committing resources (an indirect denial of service), there is a 1-hour cooldown period on Liveness Reward earnings whenever a node is terminated or paused and subsequently reconnected.

Whereas, when a user workload has been assigned to the node, and the user experiences downtime or breach of service conditions, the node operator loses the Liveness rewards for the time period and is also subject to slashing equivalent to the fee income/hour of the workload from the collateral posted.

Simply explained, the Slashing conditions and amounts can be expressed as follows:

- **During Standby Mode**
  - Less than 100% uptime during an hour leads to losing liveness rewards for that hour
  - If the node goes offline for more than an hour, the equivalent of one hour's liveness reward is slashed from the stake
- **During Task Deployment Mode**
  - Slash the node for the higher value of - one hour of liveness rewards or one hour of Fee income associated with the users' compute requirements.

4.3.9. Operational Dynamics

The operational life cycle of a Provider Node encompasses several key stages:

- **Registration and Bidding:** Initially, Provider Nodes register their compute specs, region, tier, etc on the Provider Registry. Utilizing a bid engine, these nodes can compete for deployments by submitting bids to the Order contract against each deployment that reflect their capacity to deploy the order and pricing model for the compute specs. Successful bids lead to user choosing the provider to deploy their workloads, with all pertinent details shared on-chain, marking the beginning of payment settlement based on the pricing set by the provider.

- **Deployment and Management:** Provider Nodes are responsible for the continuous management of deployments, including the initiation (`OrderCreated`), update (`UpdateLease`), and termination (`LeaseClosed`) of services. Each action is meticulously recorded on the blockchain, ensuring transparency and traceability.

- **Synchronization and Reliability:** To maintain operational alignment with the network, Provider Nodes can run a dedicated RPC nodes to listen to the contract events. This synchronization is critical for participating in the bidding process and for ensuring the timely and successful execution of deployments.

  The Provider Node is instrumental in achieving the decentralized compute network's vision of an accessible, scalable, and secure computational resource ecosystem. Through a balanced approach combining technological sophistication with an incentivized participation model, Provider Nodes ensures the network remains adaptive and resilient. As the network evolves, the continuous refinement of Provider Node operations and governance mechanisms will be paramount in sustaining the growth and vitality of the decentralized computing landscape.

4.4. Fizz Node: The Community Compute

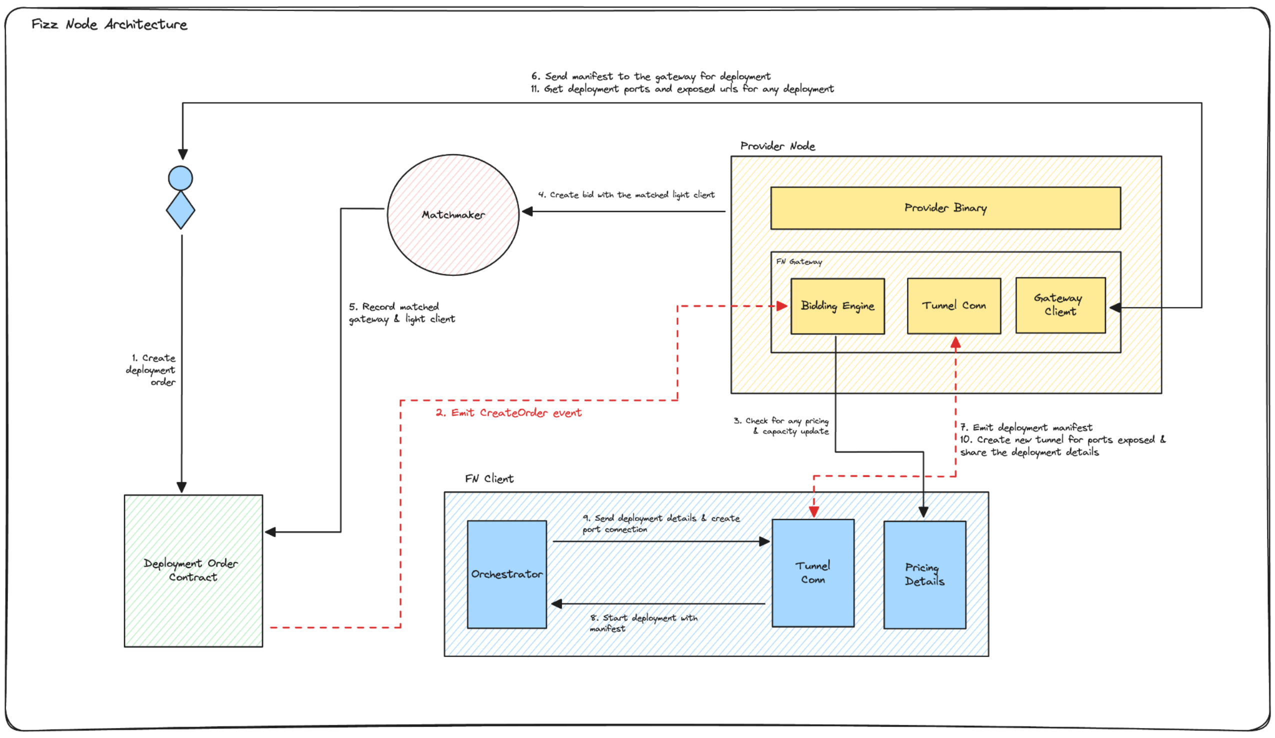

Fizz Node Architecture
6. Send manifest to the gateway for deployment
11. Get deployment ports and exposed urls for any deployment
Provider Node
Provider Binary
FN Gateway
Bidding Engine
Tunnel Conn
Gateway Climt
Matchmaker
4. Create bid with the matched light client
5. Record matched gateway & light client
1. Create deployment order
2. Emit CreateOrder event
3. Check for any pricing & capacity update
7. Emit deployment manifest
10. Create new tunnel for ports exposed & share the deployment details
FN Client
Deployment Order Contract
Orchestrator
9. Send deployment details & create port connection
8. Start deployment with manifest
Tunnel Conn
Pricing Details

Figure 7: Fizz node full architecture

4.4.1. Overview

The concept of a light node introduces a Fizzweight version of a provider node that can be operated on small devices, enhancing the network's flexibility and accessibility. This document outlines the development, challenges, and operational mechanisms of light nodes within our decentralized network, which are designed to run efficiently on less powerful devices such as personal computers and laptops.

Fizz nodes can collaborate to form a subnet within the network. A subnet consists of a Provider Node and multiple Fizz Nodes. This structure allows subnets to create their own economies, improving their performance in the network and earning better rewards. Additionally, subnets can establish their own governance systems to make decisions about their operations, further empowering them to optimize their contributions to the network.

***Note:** Fizz nodes can be operated by community members, offering a scale of compute or GPU resources that is unprecedented, fully decentralized, and highly scalable. However, the expansion of this network introduces certain challenges that potential users must be aware of. Those utilizing this global network for extensive computations, distributed computations, or training should understand that workloads should not contain any sensitive data or secrets. Since these nodes operate on external computers, there is a possibility that the data could become visible to the host. While we cannot guarantee absolute security for community-run providers, we are continuously developing methods to enhance privacy and restrict unauthorized access to private information from both the host and user sides. This ongoing effort is crucial to maintaining the integrity and trustworthiness of our decentralized network.*

4.4.2. Fizz Node Architecture and Challenges

Fizz nodes are streamlined versions of full provider nodes and are designed to run in environments with limited computational resources:

- **System Architecture:** We are developing a Docker-based light node system where each node operates in a new Docker container per user request. This approach minimizes the computational overhead on small devices compared to more resource-intensive systems like Kubernetes.

- **Connectivity Challenges:** Addressing the connectivity issues involves managing dynamic IP addresses and NAT routing. These nodes often reside on networks using DHCP, which complicates consistent access and communication.

- **Isolation and Security:** Ensuring the isolation of individual Docker containers (pods) is critical. We must implement robust security practices to prevent resource exhaustion on the host system and ensure that containers do not interfere with each other's operations.

- **Maintenance and Updates:** Regular updates are essential for maintaining security and functionality. The light node architecture must support seamless upgrades to address vulnerabilities as they arise.

A significant challenge for light nodes is accessing deployed services. This architecture addresses these challenges while ensuring decentralization. The main components of the light node system are:

- **Fizz Client**: Run by individuals on small devices (e.g., laptops or small servers) to provide underutilized compute resources to the network for monetary rewards.

- **Gateway Service**: Managed by providers, it facilitates user interactions with light clients and handles most communication.

4.4.3. FN Client Overview

The light client comprises three components working together for deployments, orchestration, and creating tunnel connections:

1. **Pricing Configuration**: Fizz clients require a pricing script or configuration provided by the user at startup. This configuration determines the cost of the resources the light node offers to the network and can be adjusted anytime through the portal or locally from the configuration file. This setup ensures flexibility and responsiveness to market conditions or resource availability.

2. **Orchestrator**: Once a deployment bid is successful, the orchestrator within the light client takes over to manage the deployment process. This involves spinning up a Docker container to handle the deployment. The orchestrator handles the requests to start deployments, manages the lifecycle of each deployment, and ensures that the service is running smoothly. It coordinates with the service tunnel to establish necessary connections and expose the deployed service.

3. **Service Tunnel**: The service tunnel functions as a crucial connection between the gateway server and the light client. Given that light clients do not have public IP addresses, exposing services running within them to the internet presents a challenge. To overcome this, the system uses a tunnel mechanism inspired by ngrok. This mechanism establishes a TCP connection between the light client and the gateway server, enabling the forwarding of requests to the internal server running on the light client. The service tunnel listens for client-related requests, such as retrieving server details or deploying manifests, and creates new tunnels for any services deployed on the light client.

Fizz clients need to specify the pricing for their resources and the extent of exposure they are willing to provide to the network. Only the specified resources are tracked and utilized for new tasks or deployments in the system.

4.4.4. FN Gateway Overview

The gateway service acts as the interface for all light clients connected to a provider. It ensures security by mediating user access and offers provider selection based on trust levels. The gateway service continuously polls light clients for pricing and resource availability and handles deployment requests from users.

1. **User Interaction**: Users interact with the gateway service to send manifests, deploy services, and access services running inside light clients. The gateway manages these requests and communicates with light clients to initiate and manage deployments. It ensures that users do not directly access host machines, thereby maintaining security and providing an optional layer of trust by selecting providers.

2. **Pricing and Capacity Monitoring**: The gateway service includes a component dedicated to continuously checking the current pricing and capacity of connected light clients. This ensures that the gateway has up-to-date information on available resources and their costs, facilitating efficient and fair deployment decisions.

3. **Proof of Capacity**: The gateway service coordinates with light clients to aggregate proofs of compute capacity when challenged. Upon receiving a challenge, the gateway requests all its connected light clients to submit their proofs of compute for a specific task. Once collected, the gateway aggregates these proofs and submits them to the Slark. Successfully handling these challenges rewards the provider cum gateway with liveness rewards.

4.4.5. Deployment Flow

The deployment flow in the light node system involves the following steps:

1. **Initiate Deployment**: The user starts a deployment by making a contract function transaction.

2. **Listening to Requests**: All gateways listen to the new deployment order request.

3. **Selecting Fizz Client**: Gateways, which always know the current state of light client pricing and capacity, choose a light node based on several factors:
    - Number of successful challenges performed within the subnet
    - Total delegation on the subnet
    - Pricing set by the client
    - Remaining resource capacity
4. **Acknowledgment**: The gateway selects a light client based on these parameters and gets an acknowledgment from the light client.
5. **Bidding Process**: The gateway then makes a bid for the deployment, and after 2 blocks of waiting, the matchmaker engine starts the process.
6. **Manifest Submission**: Once the light client and gateway are chosen by the matchmaker, the user can send the manifest to the gateway.
7. **Deployment Initiation**: The gateway receives the manifest and passes it to the light client to start the deployment.
8. **Orchestration**: The light client takes the manifest from the tunnel connection (simple TCP connection) and starts the deployment orchestration.
9. **Completion**: Once the deployment is done, the light client shares the URLs with the gateway.
10. **Monitoring and Management**: The user can then ask the gateway client to get status, logs, events, shell access, etc., directly from the light client through the gateway client.
11. **Disconnection Handling**: If the light client disconnects, the gateway cannot connect to it, and all running tasks on the light client will be temporarily turned off. The gateway needs to track the system status of all light clients so users can check it anytime.
12. **Duration and Payment**: The user specifies how many hours or days the task needs to run. If the task is closed prematurely, the locked amount is adjusted based on compute hours used and transferred to the light client.
13. **Extension**: The user can extend the duration by locking more amount in escrow, equivalent to the bid price times hours.
14. **Data Sensitivity**: Users should not provide very sensitive data during deployment, as the task/server runs on someone else's device and can be accessible to the host.

4.4.6. Delegation, Incentives & Governance

The system incentivizes both providers and light clients through a mix of delegation, liveness rewards, and task earnings:

1. **Delegation**: Fizz nodes delegate to Provider Node which it want to establish connection with. The Provider Node may have their own delegation requirements. The rewards given to the provider are distributed among light nodes based on their delegation amount. This mechanism not only secures the network but also incentivizes light clients to remain active and reliable.
2. **Task Earnings**: Fizz nodes earn from tasks and leases initiated by users. When a user deploys a task, the light node performing the task receives a portion of the fee. This direct earning mechanism ensures that light nodes are compensated for their compute resources and efforts.
3. **Subnet Performance**: The collective performance of light clients and their provider affects the subnet's tier and rewards. High performance across consecutive Eras can lead to the subnet moving up tiers, resulting in increased rewards. Conversely, poor performance or the presence of bad actors can result in lower tiers and reduced rewards, motivating the subnet to perform well and maintain integrity.
4. **Subnet Governance**: Subnets can form decentralized autonomous organizations (DAOs) to manage incentives, economy, and collective decision-making. This governance model allows subnets to self-regulate, remove bad actors, and optimize their performance and rewards. DAOs provide a democratic framework for subnet members to propose and vote on changes, ensuring that the subnet's interests align with its members.

By fostering collaboration and efficient resource utilization, the Fizz Node System aims to create a decentralized, scalable, and rewarding network for light nodes and providers alike. This comprehensive approach ensures that light nodes can participate meaningfully in the network, providers can manage and support their light clients effectively, and users can deploy and access services seamlessly.

4.4.7. Node Registration

To become part of the network, Fizz Nodes (FNs) need to join an existing subnet or create a new subnet with a Provider Node (PN). Users start by installing a binary on their local systems, during which they configure pricing and specify the compute and GPU resources they intend to offer.

Once the setup is complete, the node registers this information, including a unique peer ID, with the network. It then establishes a connection with the Provider Node gateway to accept deployment tasks and create a gateway for the services it will run. After completing the setup, users can register their details in the Fizz Node Registry under the associated Provider Node. They must also delegate $SPON tokens to the Provider Node to activate their node, enabling it to accept workloads from users.

4.4.8. Conclusion

The light node mechanism within our decentralized network aims to expand the accessibility of computational resources by enabling small device owners to participate as providers. By addressing the outlined challenges and leveraging a structured lifecycle for registration, deployment and traffic management, light nodes can significantly contribute to the network's resilience and scalability. This system not only enhances the network's capability but also ensures that resource allocation is efficient, secure, and user-friendly.

4.5 Slark Node: Proof of Compute Capacity & Uptime

4.5.1. Overview

We are implementing an advanced Proof of Compute system that features the introduction of Slark Nodes, a tier-based system for providers, reward delegation mechanisms, and a comprehensive licensing system for Slark Nodes. These Slark Nodes play a critical role in maintaining the integrity and performance of both Providers and Fizz Nodes within the Spheron network. Their primary function is to verify the specifications of providers and light nodes to uphold the network's Quality of Service (QoS).

The verification focuses on key parameters such as Liveness, Capacity, and overall Quality of Service, which are crucial for ensuring that the network operates efficiently and reliably. To effectively assess these parameters, the system employs a variety of checking methods. These include:

- **Heartbeat Collection:** Regularly checks the "heartbeat" or signal from nodes to ensure they are active and responsive.
- **Benchmark Testing:** Measures the performance capabilities of nodes by putting them through rigorous computational tests to assess their processing power and efficiency.
- **Link Data Collection and Analysis:** Gathers and analyzes data transmitted over the network to evaluate the stability and speed of connections, further contributing to assessments of network quality and service reliability.

These methods provide a robust framework for continuously monitoring and validating the operational standards of all participating nodes, thereby safeguarding the network against inefficiencies and potential vulnerabilities.

4.5.2. Challenge Mechanism

4.5.2.1. Slark Nodes

- **Role:** Challenge Provider Nodes to verify the quality and reliability of their service.
- **Responsibilities:**
    - Regularly challenge PNs to ensure compliance.
    - Provide proof of successful challenges.
    - Receive economic incentives based on the outcomes of challenges.
- **Earnings**: Slark nodes will earn from the slashing of the provider node rewards and receive a reward for each successful challenge completed. These rewards will be pooled and distributed among all participating Slark nodes in an Era.

4.5.2.2. Challenge Process

1. **Challenge Initiation**:
    - Slark nodes will randomly select providers to challenge.
    - Slark nodes will specify a simple task with specific requirements, including CPU, RAM, Disk, and GPU.
2. **Random Task Allocation**:
    - If the challenged provider has resources totaling (`x`) CPU, (`y`) GB RAM, (`z`) GB Disk, and (`k`) GPU, the Slark will pick random values within the ranges:
        - CPU: 0 to `90% of (a-x)`
        - RAM: 0 to `90% of (b-y)` GB
        - Disk: 0 to `90% of (c-z)` GB
        - GPU: 0 to `90% of (d-k)`
    - Where `a, b, c, & d` is CPU, RAM, Disk, GPU registered onchain
    - The challenge, including the provider's peer ID and a hash of a random puzzle, will be registered in a smart contract to prevent collusion.

3. **Task Execution and Verification**:
    - The challenged provider will perform the task and sends it back to the Slark nodes.
    - After a Slark Node completes a task, it signs the results with its private key and delivers them to the Referee. The Referee will receive 2N+1 results for each Provider, and each Node that delivered the same result as the majority, will be rewarded.

4.5.3. Slark Nodes Subset Selection Process

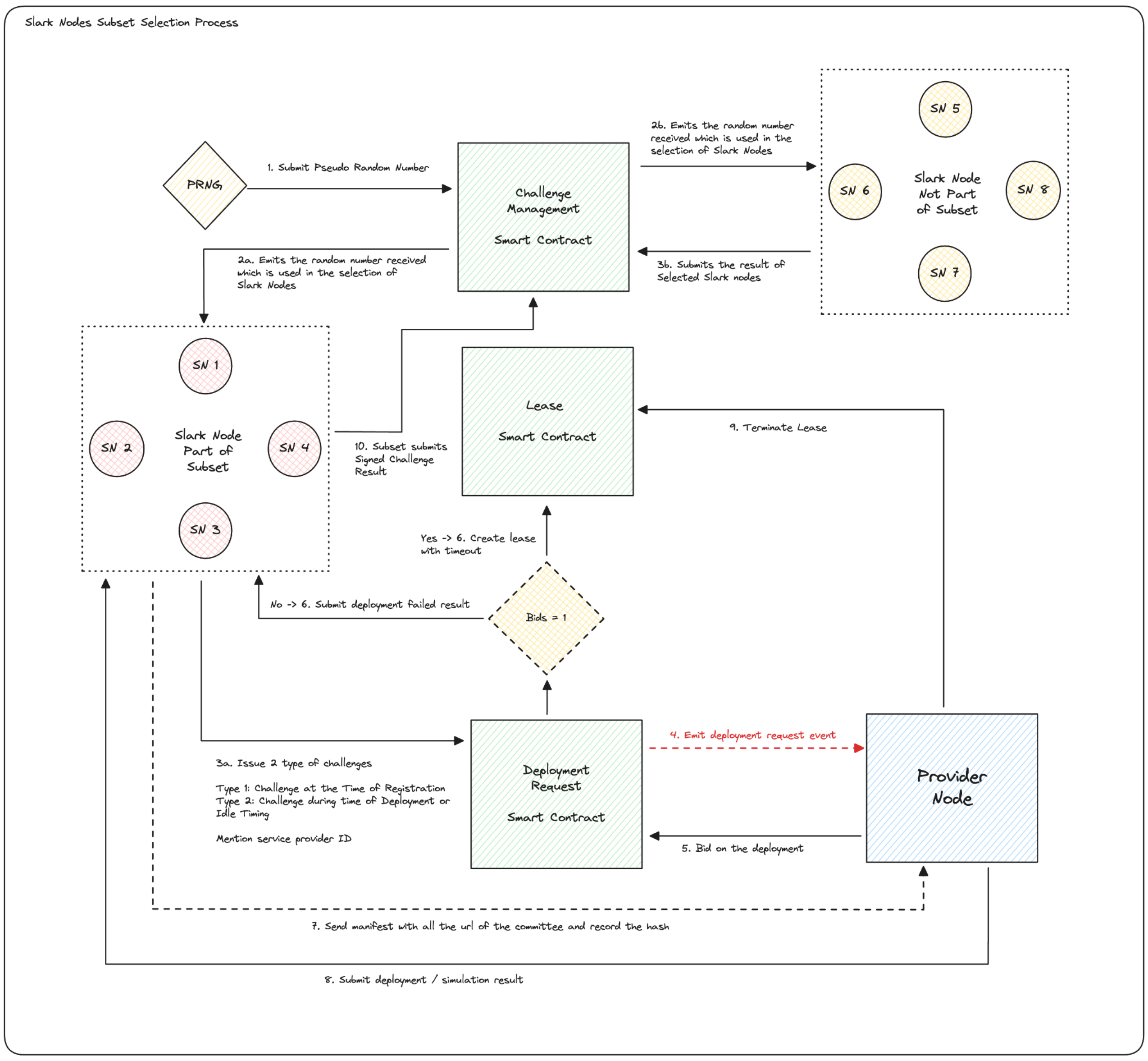

Slark Nodes Subset Selection Process
PRNG
1. Submit Pseudo Random Number
Challenge Management Smart Contract
2b. Emits the random number received which is used in the selection of Slark Nodes
SN 5
SN 6
Slark Node Not Part of Subset
SN 8
SN 7
3b. Submits the result of Selected Slark nodes
2a. Emits the random number received which is used in the selection of Slark Nodes
SN 1
SN 2
Slark Node Part of Subset
SN 4
SN 3
10. Subset submits Signed Challenge Result
Lease Smart Contract
9. Terminate Lease
Yes -> 6. Create lease with timeout
No -> 6. Submit deployment failed result
Bids = 1
3a. Issue 2 type of challenges
Type 1: Challenge at the Time of Registration
Type 2: Challenge during time of Deployment or Idle Timing
Mention service provider ID
Deployment Request Smart Contract
4. Emit deployment request event
Provider Node
5. Bid on the deployment
7. Send manifest with all the url of the committee and record the hash
8. Submit deployment / simulation result

Figure 8: Slark nodes subset selection process

Building on the concept analogous to Ethereum's beacon chain, our system incorporates Slark Nodes that may either employ an inbuilt consensus mechanism similar to the Beacon Chain or use a smart contract to achieve consensus, akin to a sequencer in Layer 2 technologies.

4.5.3.1. Proposed System Overview

In this system, a pseudo-random number is either generated by the sequencer or submitted to the sequencer using systems such as Chainlink VRF or the RANDAO mechanism or with a verifiable server. This number is transmitted to the Challenge Management Smart Contract as a transaction for each designated slot, which typically spans a 1-hour window. Utilizing this pseudo-random number, the Challenge Management Smart Contract emits an event which includes the random number. This event informs various Slark Nodes whether they have been selected for that particular slot. The selected Slark Nodes then commence their challenges to the respective nodes based on user feedback, utilizing the same random number that was submitted to the smart contract.

Nodes that are not selected for a slot are responsible for calculating which Slark Nodes have been chosen and subsequently sign this subset before submitting it to the Challenge Management Smart Contract. This submission ensures that the Challenge Management Smart Contract maintains a comprehensive list of selected Slark Nodes for the epoch.

4.5.3.2. Epoch and Node Selection

An epoch is defined as a collection of multiple slots, specifically 24 slots, designed to ensure comprehensive coverage of all provider nodes at least once throughout the entire epoch.

The selection of Provider Nodes for challenging is prioritized as follows:

1. New registrations of Provider Node Nodes.
2. User feedback on existing Provider Node Nodes.
3. The pseudo-random number submitted to the smart contract.

Slark Nodes submit their signed results using the BLS signature scheme to the Challenge Smart Contract, signaling the completion of challenges for selected Provider Node Nodes.

The Challenge Smart Contract then verifies all the submitted signatures and seeks to reach a quorum on the results. If a quorum is achieved, the smart contract marks the challenge as completed for the specific Provider Node node. This process not only ensures the integrity and reliability of the network but also enhances the accountability of the Provider Nodes through transparent and verifiable challenges.

4.5.4. Slark Node Tier System

Slark nodes also have a tier system, with tiers up to Tier 6. The tier of a Slark node determines the range of provider tiers it can challenge and the reward and incentive it can get for successful challenges:

4.5.4.1. Tier 1 Slark Node

- **Requirements**:
  - Maintain a 90% verification accuracy every Era.
  - Maintain a avg. of < 350 ms response time every Era.
  - Buy amount decided during Node Sale
- **Benefits**:
  - Can challenge providers from Tier 6 to Tier 7 and earn from Slashing
  - **Reward multiplier: 1.0x**

4.5.4.2. Tier 2 Slark Node

- **Requirements**:
  - Maintain a 92% verification accuracy every Era.
  - Maintain a avg. of < 300 ms response time every Era.
  - Buy amount decided during Node Sale
- **Benefits**:
  - Can challenge providers from Tier 5 to Tier 7 and earn from Slashing
  - **Reward multiplier: 1.2x**

4.5.4.3. Tier 3 Slark Node

- **Requirements**:
  - Maintain a 95% verification accuracy every Era.
  - Maintain a avg. of < 250 ms response time every Era.
  - Buy amount decided during Node Sale
- **Benefits**:
  - Can challenge providers from Tier 4 to Tier 7 and earn from Slashing
  - **Reward multiplier: 1.4x**

4.5.4.4. Tier 4 Slark Node

- **Requirements**:
  - Maintain a 97% verification accuracy every Era.
  - Maintain a avg. of < 200 ms response time every Era.
  - Buy amount decided during Node Sale
- **Benefits**:
  - Can challenge providers from Tier 3 to Tier 7 and earn from Slashing
  - **Reward multiplier: 1.6x**

4.5.4.5. Tier 5 Slark Node

- **Requirements**:
  - Maintain a 98% verification accuracy every Era.
  - Maintain a avg. of < 150 ms response time every Era.
  - Buy amount decided during Node Sale
- **Benefits**:
  - Can challenge providers from Tier 2 to Tier 7 and earn from Slashing
  - **Reward multiplier: 1.8x**

4.5.4.6. Tier 6 Slark Node

- **Requirements**:
  - Maintain a 99% verification accuracy every Era.
  - Maintain a avg. of < 100 ms response time every Era.
  - Buy amount decided during Node Sale
- **Benefits**:
  - Can challenge providers from Tier 1 to Tier 7 and earn from Slashing
  - **Reward multiplier: 2.0x**

Tier Summary Table

Tier	Verification Accuracy (%)	Responsiveness (ms)	Buy Amount	Provider Node Challenge	Rewards Multiplier
1	> 90	< 350	> Y1	≥ 6	1.0
2	> 92	< 300	> Y2	≥ 5	1.2
3	> 95	< 250	> Y3	≥ 4	1.4
4	> 97	< 200	> Y4	≥ 3	1.6
5	> 98	< 150	> Y5	≥ 2	1.8
6	> 99	< 100	> Y6	≥ 1	2.0

4.5.5. Tier Transition Mechanism

- **Promotion:**
  - Slark Node can move up to a higher tier by consistently meeting or exceeding the requirements of the desired tier over 3 consecutive eras and when there is any slot is available in the higher Tier. The Slark node owner needs to opt in to higher tier when available from the Portal.
- **Demotion:**
  - Slark Node can be downgraded to a lower tier if they fail to meet the requirements of their current tier for 2 consecutive eras. This will happen automatically.

4.5.6. Reward Delegation Mechanism

- **Challenge Reward:** The Era reward is only given to the Slark node based on how many challenges it has done during the era.
- **Slashing Reward:** The more it challenges the provider, the more chances it can earn from the provider reward slashing. The higher the tier of provider is, the more amount a high tier Slark node can earn from the slashing.
- 80% of the era reward will be automatically delegated to the Slark node's chosen provider when setting it up.
- The remaining 20% will be given to the Slark node as a liquid token.
- Rewards can be undelegated, but if the delegated amount exceeds a certain threshold, there will be a vesting period.

4.5.7. Licensing System for Slark Nodes

4.5.7.1. Slark Node License (NFT)

- **Ownership and Operation**: Anyone can run a Slark Node Client if they hold a Slark Node License (NFT) or receive delegation from an NFT owner.
- **Resource Management**: Resources scale linearly; one operator can run up to **x** licenses on a single machine. Insufficient resources may cause licenses to go offline or produce incorrect task results, leading to reduced rewards or banned licenses.

4.5.7.2. Delegation Process

- **NFT Owners**:
  - **Delegation**: Can choose which Slark Node Client to delegate to via a Portal.
  - **Self-Operation**: Running the node themselves requires verification of the wallet address inside the Slark Node Client.
  - **Rewards**: Delegation ensures rewards for valid calculations and uptime. Delegation can be managed anytime through the Portal.
- **Node Runners**:
  - **Acceptance**: Can accept or reject delegation requests within the Slark Client.
  - **Termination**: Can end a delegation at any time. Verifying the correct owner wallet address is crucial to earning rewards correctly.

- **Reward Distribution**:
    - **Licensee Rewards**: Rewards can be given to the licensee who provided the license to the node operator.

This comprehensive system ensures that all participants are incentivized to maintain high standards of performance and reliability, fostering a trustworthy and efficient network.

4.5.8. Security Measures

To bolster the stability of the Slark node network and counteract potential misuse, the Slark Node Non-Fungible Token (NFT) will be soulbound for a period of one year. This precaution ensures the integrity of the ecosystem by minimizing rapid changes in node ownership, thereby maintaining a consistent and reliable network environment.

Once a Slark node begins accruing tokens, these tokens are allocated exclusively to the NFT's owner, irrespective of who operates the Slark node client. Moreover, during the initial setup of the Slark node client, the rewards earned by the node are automatically delegated to a provider selected by the NFT owner. This automated delegation process not only streamlines the distribution of rewards but also enhances token earnings through potential compounding effects.

NFT owners are granted the flexibility to redelegate their tokens as needed. In instances where an owner opts to withdraw their tokens and the amount delegated exceeds a pre-determined threshold, a vesting period is initiated to manage the withdrawal process smoothly and securely. This structured approach to managing rewards and withdrawals aims to encourage sustained engagement and investment in the network, promoting its long-term growth and stability.

4.6. Payment System: Facilitating Equitable Transactions

4.6.1. Overview

The decentralized compute network introduces an advanced Payment System, underpinned by a token-based economy, specifically crafted to ensure transparent and equitable compensation for providers of GPU resources. This section delves into the mechanisms and structures of the payment system, highlighting its role in addressing inefficiencies within the current market and ensuring that providers are appropriately rewarded for their contributions.

4.6.2. Introduction to the Token-Based Economy

At the heart of the network's Payment System lies a token-based economy, utilizing any token including $SPON tokens as the medium of exchange. This innovative approach is designed to streamline transactions within the network, enabling a seamless flow of value between users and providers. This will also enable a great ecosystem of contributors & will open a huge opportunity for the foundations or even other player to ask them bring the utility into the network. Any token as payment must be get enabled with the token governance model. The system's foundation on blockchain technology ensures that all transactions are transparent, auditable, and secure, thereby enhancing trust and reliability across the network.

4.6.3. User Payment Contributions

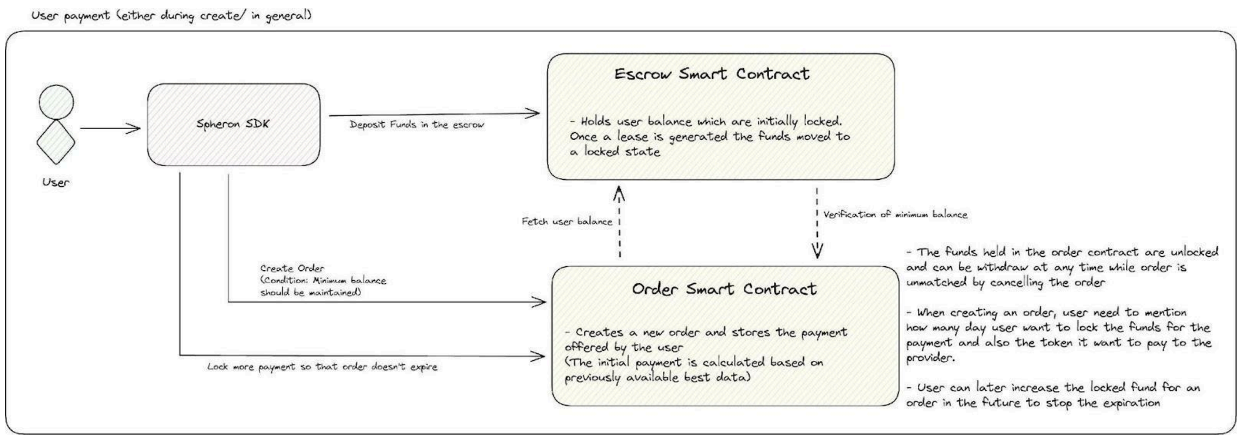

User payment (either during create/ in general)
User
Spheron SDK
Deposit Funds in the escrow
Escrow Smart Contract
- Holds user balance which are initially locked. Once a lease is generated the funds moved to a locked state
Fetch user balance
Verification of minimum balance
Create Order (Condition: Minimum balance should be maintained)
Lock more payment so that order doesn't expire
Order Smart Contract
- Creates a new order and stores the payment offered by the user (The initial payment is calculated based on previously available best data)
- The funds held in the order contract are unlocked and can be withdraw at any time while order is unmatched by cancelling the order
- When creating an order, user need to mention how many day user want to lock the funds for the payment and also the token it want to pay to the provider.
- User can later increase the locked fund for an order in the future to stop the expiration

Figure 9: User payment workflow during the create order / in general

The workflow of user payment from the above Figure 7 shows that for user to start a deployment in the protocol, the user need to first deposit their initial fund to our Escrow Smart Contract which will hold their balance (some in locked and some in unlocked). User then can make create order to start a new deployment to the Order Smart Contract, which will fetch user balance (which are unlocked) to verify minimum balance of the user before starting order matching process.

The user also need to mention how many days / hour it want to lock the initial fund for the deployment order and also the token user want to pay to the provider for the workload. The locking of initial fund for the order will be calculated once the order is matched with the provider and the contract know what bid was selected. Once lease is created the fund will be locked in the escrow and user won't be able to access these funds unless user closes the lease. As the lease progresses the amount in locked state is debited from the user and assigned to the provider based on the time passed (calculated using block.timestamp). If user close the lease prematurely, the balance of both user and provider are updated, and the final balance is pushed to both accounts and the lease is marked closed.

In short, the system delineates a clear and flexible mechanism for user payments:

- **Compute Escrow Wallets:** Users establish contract wallets on the blockchain, serving as compute escrows into which they deposit a minimum amount. This escrow system allows users to deposit an array of tokens, including $SPON, offering the flexibility to manage their funds according to compute usage.

- **Withdrawal and Fund Management:** Users have the capability to withdraw any remaining funds from the escrow wallet, ensuring they only pay for the compute resources utilized. The escrow wallet also facilitates the transfer of instance ownership and funds between users, providing a layer of versatility and control over resource allocation and usage.

- **Maximum Resource Allocation Mechanism:** Upon the deposition of funds into the escrow wallet, the user's allocation of computational resources is governed by a predefined bracket system. This system systematically determines the quantity of resources allocated based on the amount of funds deposited, ensuring a structured and equitable distribution of computing power. For instance, a deposit of $15 entitles the user to a specific allocation of computational resources, encapsulated as follows:
    - **For standard compute allocations:** The user is entitled to access 8 CPUs, 32 GB of RAM, and 512 GB of Disk space.
    - **For GPU-intensive tasks:** The allocation includes access to a mid-tier GPU (such as the A4000 or A6000 models), complemented by 32 CPUs, 128 GB of RAM, and 1 TB of Disk space.

4.6.4. Mechanisms for Provider Compensation

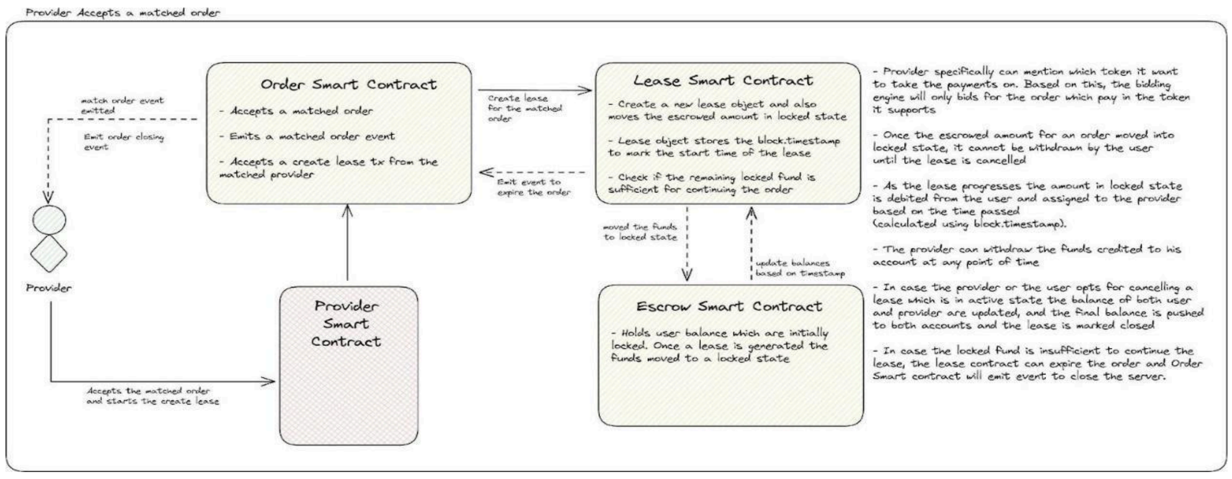

Provider Accepts a matched order
Order Smart Contract
- Accepts a matched order
- Emits a matched order event
- Accepts a create lease tx from the matched provider
match order event emitted
Emit order closing event
Provider
Accepts the matched order and starts the create lease
Provider Smart Contract
Create lease for the matched order
Emit event to expire the order
Lease Smart Contract
- Create a new lease object and also moves the escrowed amount in locked state
- Lease object stores the block.timestamp to mark the start time of the lease
- Check if the remaining locked fund is sufficient for continuing the order
moved the funds to locked state
update balances based on timestamp
Escrow Smart Contract
- Holds user balance which are initially locked. Once a lease is generated the funds moved to a locked state
- Provider specifically can mention which token it want to take the payments on. Based on this, the bidding engine will only bids for the order which pay in the token it supports
- Once the escrowed amount for an order moved into locked state, it cannot be withdrawn by the user until the lease is cancelled
- As the lease progresses the amount in locked state is debited from the user and assigned to the provider based on the time passed (calculated using block.timestamp).
- The provider can withdraw the funds credited to his account at any point of time
- In case the provider or the user opts for cancelling a lease which is in active state the balance of both user and provider are updated, and the final balance is pushed to both accounts and the lease is marked closed
- In case the locked fund is insufficient to continue the lease, the lease contract can expire the order and Order Smart contract will emit event to close the server.

Figure 10: Provider payment acceptance architecture

We will be discussing the provider payment mechanism. From Figure 8, you can clearly see that after a Deployment Order is successfully matched with a provider, a lease is established. This lease automatically locks a portion of funds, estimated to cover the execution fees for the duration agreed upon (days or months). These funds are then gradually debited from the user to compensate the provider, starting from the lease's initiation timestamp. Until the lease expires or is canceled by either party, the locked funds remain inaccessible. However, providers have the ability to withdraw funds that have accrued in the escrow associated with their deployments. Moreover, providers are obligated to contribute 20% of the total deployment payment to the Spheron Foundation.

In the event that either the provider or user decides to cancel an active lease, the balances of both parties are updated. The final balances are then pushed to their respective accounts, and the lease is officially closed. Additionally, if at any point the locked funds become insufficient to continue supporting the active lease, the Lease Smart Contract will automatically terminate the lease. Subsequently, the Order Smart Contract will emit an event to shut down the server.

To promote the use of $SPON in transactions and enhance the network's economic efficiency, Spheron has implemented a specific fee structure:

User Fees:

- Payments made with tokens other than $SPON attract a 2% facilitation fee.
- Payments made with $SPON are not subject to any fees.

Provider Fees:

- Liveness rewards do not incur any fees.
- For utilization payments:
    - Payments to providers in tokens other than $SPON or $SPON incur a 2% fee.
    - Payments made in $SPON are free from any fees.

This fee strategy is designed to incentivize the utilization of the network's native token, $SPON, thereby reducing transaction costs for both users and providers while supporting the sustainability and growth of the network.

4.6.5. Compensation Model

The compensation model for providers is meticulously structured to ensure fairness and incentivize participation:

- **Utilization Payments:** Providers will receive payment in any token that user have used to start the deployment. Provider can mention all the token it want to support and will be matched with order which give payments in the token the provider supports.

  Note: Foundation will be taking commission on utilization rewards from the Deployment Order. Initially it will be 20%, but it can be increased or decreased by the DAO / Governance based on utilization of the network.

- **Network Rewards:** Provider nodes are incentivized by the foundation to continue to be a part of the network if there is less utilization of the provider capacity in any Era timeframe. This will make sure the providers are not leaving the network, in case of providers are not able to make enough ROI on their hardware and wasn't able to pay for their OpEx charges. The rewards will be given in $SPON token, over a defined period known as an Era.

- **Era Duration: Era** occur based on use of the network, generally a 24 hour period and can be adjusted by the Foundation with input from Governance as needed in order to respond to changes in network usage. This mechanism ensures that rewards are distributed in a timely and predictable manner.

- **Auto-Staking:** To further incentivize providers, the system incorporates an auto-staking feature whereby rewards can be directly staked, enhancing the provider's stake in the network and increasing their chances of being selected for future deployments.

4.6.6. Wallet Infrastructure

A comprehensive wallet infrastructure supports the Payment System, enabling the creation, management, and transfer of escrow wallets:

- **Multi-Escrow Wallet Creation:** Users can create numerous escrow wallets from a single liquidity wallet, streamlining the process of funding and managing compute deployments.
- **Ownership and Fund Management:** Escrow wallets are designed with functionalities to deposit, withdraw, view funds, and manage ownership, ensuring users have complete control over their resources within the network.

The Payment System embodies the decentralized compute network's commitment to fostering a fair, transparent, and efficient marketplace for computational resources. Through its token-based economy, the system not only addresses prevailing market inefficiencies but also ensures that all participants are fairly compensated for their contributions, thereby reinforcing the network's integrity and promoting sustained growth and innovation.

5. Deployment Lifecycle

The deployment lifecycle within a decentralized compute network represents a cornerstone process, essential for the provisioning and management of GPU resources. This paper elucidates the sequential steps and smart contract interactions that govern this lifecycle, from the initial GPU resource request to the ultimate fulfillment and potential modifications of the deployment. By dissecting the operational framework, this exposition aims to provide a deep understanding of the system's underlying mechanics, highlighting the pivotal role of smart contracts in orchestrating transactions, managing escrows, and distributing rewards.

5.1. Introduction

The deployment lifecycle in a decentralized compute network is a structured process that encompasses several key stages. It begins with a user's request for server deployment and progresses through various states, including bidding, order matching, lease creation, deployment activation, and optional updates or termination. Each step is facilitated by the network's smart contracts, ensuring a transparent, secure, and efficient pathway from request to resource allocation.

5.2. Operational Steps of the Deployment Lifecycle

5.2.1. Initiation and Provider Lookup

- The lifecycle begins with a user initiating a server deployment request and placing it in the `OrderCreated` state. This state triggers all eligible Provider Nodes to start the bidding process, actively competing to fulfill the request.

5.2.2. Provider Selection and Bidding

- During this phase, providers that meet the deployment criteria are invited to submit their bids for the requested resources. Bidding is allowed for a duration of 1-2 blocks, after which no further bids are accepted. The order is then processed by the matchmaking engine, which selects the optimal provider based on the bids received. The result is forwarded to the Order Smart Contract to initiate lease creation.

5.2.3. Lease Creation and Provider Matching

- Following the selection of a provider, the matchmaking engine will match the order with the provider and create a lease in the smart contract alongside it, issuing an event that prompts the provider to reserve the specified resources for the deployment. This step formalizes the agreement between the user and the provider through the execution of the smart contract, ensuring the resources are allocated for the user's needs.

5.2.4. Deployment Activation

- Upon reservation confirmation, the Provider Node is ready for receiving the deployment manifest necessary for server setup.

- The user then transmits the deployment manifest via an MTLS connection, enabling the provider to commence server deployment.

- The provider configures and activates the server, ensuring it meets all specified requirements. User can then call the provider fetch the deployment detail respective to the Lease ID (LID) like connection URL, Exposed Ports, IP assigned, etc.

5.2.5. Termination and Closure

- Users retain the ability to terminate their deployments by calling `closeLease` method in the Lease Contract which will make a `LeaseClosed` event. Upon listening the event, the provider node will cease the operation of the server in the kubernetes cluster, thus concluding the resource allocation.

5.2.6. Deployment Updates

- The lifecycle also accommodates the modification of existing deployments. Users may request updates concerning general configurations, specifications, or replicas via a `updateLease` transaction which will close the existing lease and create a new lease with the updated specs and new prices and emit the event for Provider Node to update the pod with the new pod specification.

At the heart of the deployment lifecycle are the network's smart contracts, which automate the management of transactions, escrows, and rewards. These contracts not only enforce the terms of the deployment agreements but also ensure the secure and efficient transfer of funds per resource usage. By leveraging blockchain technology, the system guarantees transparency and trust, pivotal for the network's integrity and user confidence.

The deployment lifecycle within a decentralized compute network exemplifies the intricate orchestration required to dynamically allocate compute & GPU resources.

Through a systematic process underpinned by smart contracts, the network achieves a harmonious balance between demand and supply, ensuring that resources are allocated efficiently and equitably. This lifecycle not only represents a technical achievement but also encapsulates the network's broader mission to democratize access to computational resources, thereby fostering innovation and progress in GPU- intensive applications.

6. Future Work

The rapid advancement and increasing demands of Artificial Intelligence AI and Machine Learning ML applications present unique challenges and opportunities for decentralized compute networks. This section outlines a forward-looking research agenda aimed at harnessing underutilized compute and GPU resources from smaller devices. By expanding the network's capabilities, we aspire to create a more comprehensive and scalable infrastructure that can support a broader spectrum of AI and ML innovations. This research will delve into several critical areas, including integration strategies for smaller devices, networking solutions, reliability and capacity proofs, GPU cluster coordination, and enhanced security protocols.

6.1 Introduction to Future Research Directions

As AI and ML applications continue to evolve, the need for diverse and scalable computational resources becomes increasingly critical. Recognizing this, our future work is dedicated to exploring several key areas that promise to extend the network's functionality and capacity, particularly focusing on the integration of smaller, underutilized devices. These devices, often overlooked, hold significant potential to augment the network's computational power.

6.2 Research Areas and Objectives

6.2.1. Integration of Underutilized Resources

- **Objective:** Develop methodologies to seamlessly integrate compute and GPU resources from smaller devices into the network. This involves creating a robust framework that can identify, manage, and allocate these resources efficiently, thereby expanding the network's overall capability.

6.2.2. Networking Solutions for Smaller Devices

- **Objective:** Innovate networking protocols that facilitate reliable and efficient communication between smaller devices within the decentralized network. This includes overcoming challenges related to bandwidth, latency, and device heterogeneity to ensure seamless resource pooling.

6.2.3. Proof of Uptime and Capacity

- **Objective:** Establish verification mechanisms for provider uptime and resource capacity, enhancing the Service Level Agreements (SLAs) and automation within the network. This entails developing proof systems that accurately reflect the availability and computational power of both providers and smaller devices, ensuring optimal deployment allocation.

6.2.4. GPU Cluster Coordination

- **Objective:** Engineer advanced coordination techniques for GPU clusters, focusing on enterprise-grade training tasks. This includes optimizing resource allocation, synchronization, and parallel processing capabilities to support intensive AI training and inference workloads.

6.2.5. Security Enhancements

- **Objective:** Strengthen the security framework to protect user servers and sensitive environments from unauthorized provider access. This involves devising comprehensive access control mechanisms, encryption protocols, and audit trails to safeguard data integrity and privacy.

7. Conclusion

This whitepaper has introduced a novel decentralized architecture designed to address the significant challenges associated with GPU supply and cost in the field of Artificial Intelligence (AI) development and research. By advocating for a shift towards a decen tralized network, we envision a future where access to GPU resources is democratized, costs are significantly reduced, and distributionis optimized for efficiency. Such advancements are poised to unlock a new era of AI innovation, making cutting- edge research and development accessible to a broader spectrum of individuals and institutions. The proposed framework not only promises to alleviate current supply chain constraints but also aims to foster a more inclusive and collaborative environment for AI exploration. As we embark on this journey towards redefining the GPU marketplace, the potential for transformative impact on AI development and the broader technological landscape is immense. This initiative marks a critical step forward in ensuring that the computational resources necessary for driving the next wave of AI advancements are within reach of the global research and development community.

Acknowledgment

This research and these ideas were conducted by Mitrasish Mukherjee and Prashant Maurya, co-founders of Spheron (https://spheron.network/). We extend our gratitude to Aditya Rola for his valuable contributions to the ideation, discussions, and architecture.

References

- Filecoin - https://filecoin.io/filecoin.pdf
- Proof of aliveness - https://dl.acm.org/doi/pdf/10.1145/3359789.3359827
- Proof of spacetime : PoSt
- Proof of execution by Risc zero:The RISC Zero STARK Protocol | RISC Zero Developer Docs
- Hardware Attestation: Introducing Zero-Knowledge Proofs for Private Web Attestation with Cross/Multi-Vendor Hardware
- Remote attestation of hardware or hardware/software hybrid - https://arxiv.org/pdf/2005.12453.pdf Cloud
- Attestation - http://palms.ee.princeton.edu/system/files/cloud_attest_1c.pdf
- Constellation Confidential Kubernetes) Confidential Kubernetes | Constellation TPM
- Explainer - https://webthesis.biblio.polito.it/24507/1/tesi.pdf
- Fluence - https://fluence.dev/docs/build/introduction
- Akash: Intro to Akash | Akash Guidebook | Akash Guidebook
- Vistara: Welcome to Vistara
- Aethir: Home | Aethir
- Bittensor: Introduction | Bittensor
- IONet - https://developers.io.net/docs/overview
- Render - https://rendernetwork.com/
- Eigen Layer - https://eigenlayer.xyz