deMEC

(It's a Temporary Name)

The decentralized one-click deployment *PaaS* and *IaaS* powered by multi-access edge computing

Disclaimer

deMEC is a work in progress and not to be taken as financial or technical advice or reference

Introduction:

Public clouds are designed to provide general-purpose computing resources and services to a wide range of users over the internet. They are not optimized for edge computing, which involves running applications and services closer to the end user in order to reduce latency and increase performance. In order to provide edge native or OpenRAN capabilities, public clouds would need to invest in infrastructure and resources that are specifically designed for edge computing, which may not be a viable option for them due to the specialized nature of edge computing and the need for close proximity to the end user. Traditional public clouds typically do not have the same level of control and custodial sovereignty over the network and infrastructure as edge native solutions, which can make it difficult for them to provide the necessary level of performance and reliability for edge computing applications.

***First**, public clouds are centralized, meaning that they are owned and operated by a single entity. This can create potential security vulnerabilities, as the central authority has control over the infrastructure and can potentially access or misuse user data.*

***Additionally**, public clouds are not designed with the same level of flexibility and customization as private or on-premises infrastructure. This can make it difficult to integrate web3 applications, which often require specialized infrastructure and configurations.*

***Furthermore**, public clouds can be expensive, as users are charged for the resources they consume on a pay-as-you-go basis. This can make it difficult for web3 applications to scale and may not be a cost-effective solution for some use cases.*

***Overall**, while public clouds can certainly be used for web3 applications, they may not provide the same level of security, flexibility, and cost-effectiveness as other options.*[1]

The cloud infrastructure industry is worth $210 billion as of 2022.

By 2021, it is predicted that 94% of internet applications and computer instances will be processed by Cloud Service Providers (CSPs) with only 6% being processed by traditional data centers.

The main driver for this growth is the low utilization rates of IT resources in traditional data centers, as on average they only deliver 6% of their maximum computing output over the course of the year and up to 30% of servers are idle. There are currently 8.4 million data centers globally, with an estimated 96% of server capacity going to waste.

The three leading CSPs, Amazon Web Services (AWS), Google Cloud, and Microsoft Azure, dominate the market with 71% market share and this is expected to increase. These providers are complex, inflexible, and restrictive, and come with a high recurring cost and vendor lock-in agreements. Cloud cost optimization has been the top priority for cloud service users for the past three years. Apart from the large incumbent providers, organizations do not have many options for cloud computing. DeMEC aims to

---

[1]*Note: This lite paper represents a continuous work in progress. We will endeavor to keep this document up to date with the latest development progress. However, due to the ongoing and iterative nature of the deMEC development process, the resulting code and implementation may differ from what is outlined in this paper.*

create efficiencies in the cloud hosting market by repurposing compute resources that are currently being wasted.

By using a blockchain, DeMEC introduces decentralization and transparency into an industry currently dominated by monopolies, making cloud computing a commodity that is accessible and affordable anywhere in the world. DeMEC is the world's first and only decentralized supercloud for serverless computing, allowing anyone with a computer to become a cloud provider by offering their unused compute cycles in a safe and seamless marketplace. In this paper, we present an economic system that uses a native currency to achieve economic sovereignty in our decentralized computing ecosystem. We also propose an inflation design to address the inherent adoption challenges faced by early market economies, such as a lack of demand from tenants due to a lack of supply. We also present a mechanism for a stable medium of exchange by addressing token volatility, a major challenge for the adoption of decentralized ecosystems.

Considerations

In a perfect world, a decentralized edge cloud service using a P2P network and WebAssembly could potentially be highly scalable. The decentralized nature of the network would allow it to potentially handle a large number of nodes and a high volume of requests without the need for a central authority to manage the network.

However, it is important to note that there are many factors that can affect the scalability of a decentralized network, including the efficiency of the underlying protocols and algorithms, the amount of resources (such as bandwidth and storage) available to each node, and the ability of the network to handle failures or disruptions.

Additionally, the scalability of a decentralized edge cloud service would also depend on the specific workloads and applications it is designed to support. Some workloads may be more suitable for a decentralized network than others, and the network may need to be optimized or designed differently in order to support different types of workloads effectively.

Overall, while a decentralized edge cloud service using a P2P network and WebAssembly could potentially be highly scalable in a perfect world, there are many challenges and considerations that would need to be addressed in order to achieve this level of scalability in practice.

It is possible that a decentralized edge cloud service with some nodes running on traditional servers could be more scalable than one that relies solely on browser-based nodes. Traditional servers typically have more resources (such as processing power, memory, and storage) available than web browsers, which could allow them to handle more workloads and process more requests.

However, it is important to consider the trade-offs involved in using traditional servers as part of a decentralized network. While they may provide more resources, they also introduce a central point of failure, as the network would depend on the availability and reliability of these servers. This could undermine the decentralized nature of the network and make it less resilient to failures or disruptions.

Additionally, using traditional servers as part of a decentralized network could also raise concerns around security and confidentiality, as the data being processed by these servers could be more vulnerable to unauthorized access or tampering.

Overall, whether or not a decentralized edge cloud service with some nodes running on traditional servers would be scalable would depend on the specific requirements and workloads of the service, as well as the efficiency of the underlying protocols and algorithms used to manage the network. It may be possible to achieve good scalability with this approach, but it would depend on how the network is designed and implemented.

Our decentralized Multi-Access Edge Computing (deMEC) platform leverages a distributed hash table (DHT) to power edge computing for functions and containers. This allows for more reliable connectivity and improved performance compared to traditional centralized cloud providers. In contrast to Akash Network, which uses the Cosmos SDK, our platform is intended to first be compatible with the Ethereum Virtual Machine (EVM) and can support other standards such as Solana, cosmWASM and Substrate with further modification. This allows for seamless integration with existing smart contracts and decentralized applications (dApps) on the Ethereum, Polygon, NEAR Aurora, Avalanche, Telos, and zkEVM based network. Additionally, our platform addresses the lack of Telco interoperability and unreliable Container Network Functions (CNFs) that often plague traditional edge computing solutions. By using signed messages and events for public keys, including JWTs, our platform is able to securely verify, create, and gate resource access. This added layer of security, combined with the benefits of a DHT-based network and EVM compatibility, make our deMEC platform a superior choice for decentralized edge computing.

In addition to providing decentralized, cost-effective computing resources, our DHT edge network is also proposing the implementation of storage and code execution using WebAssembly. This allows for the execution of functions and services on the network to be compiled into a lightweight, portable format that can be run on any device with a compatible runtime.

Furthermore, we are also proposing the use of service workers in the browser via websockets and WebRTC on top of the Hypercore Protocol. This enables the creation of a fully decentralized edge computing network, with the ability to execute code and access data directly from the client-side in a reliable and efficient manner. This is particularly useful for applications that require low-latency access to compute resources, such as real-time data processing or interactive content delivery.

Overall, our DHT edge network aims to provide a more reliable and flexible alternative to traditional cloud computing solutions, with the added benefits of decentralization, security, and cost-effectiveness. By leveraging the power of Hypercore and WebAssembly, we are able to create a scalable, interoperable platform that can support a wide range of use cases across multiple industries.

Hypercore and Lit Protocol are two open-source technologies that are revolutionizing the way that decentralized functions and applications are built and deployed on the Internet. Hypercore Protocol is a peer-to-peer (P2P) protocol for distributing and replicating data streams, while Lit Protocol is an Ethereum Virtual Machine (EVM) based protocol for verifying, creating, and gating access to resources via signed messages and events.

Together, these two technologies enable the creation of a decentralized compute edge network, where functions and applications can be executed on remote P2P nodes in a secure and scalable manner. This is accomplished through the use of WebAssembly code execution, micro Virtual Machines (microVMs), and off-chain verification systems.

One key advantage of this approach is the ability to execute functions and applications closer to the end user, reducing latency and improving performance. This is especially important for applications that require low-latency access to data streams, such as real-time analytics, streaming media, and IoT devices.

Another advantage is the ability to create and manage access to resources in a decentralized manner. With Lit Protocol, resource owners can create public keys that represent their resources, and grant or revoke access to these resources based on signed messages from users.

This enables the creation of new types of decentralized applications and business models that are not possible with traditional server-based architectures.

Finally, the use of microVMs and off-chain verification systems helps to improve security and reliability. By executing functions in isolated environments, it is possible to mitigate the risk of code injection attacks and other types of vulnerabilities. Additionally, off-chain verification allows for the creation of service level agreements (SLAs) that can be enforced on-chain, ensuring that functions and applications are executed as expected.

In summary, the combination of Hypercore Protocol and Lit Protocol represents a powerful new approach to decentralized computing. With the ability to execute functions and applications closer to the end user, manage access to resources in a decentralized manner, and improve security and reliability, these technologies have the potential to create a new era of decentralized applications and business models.

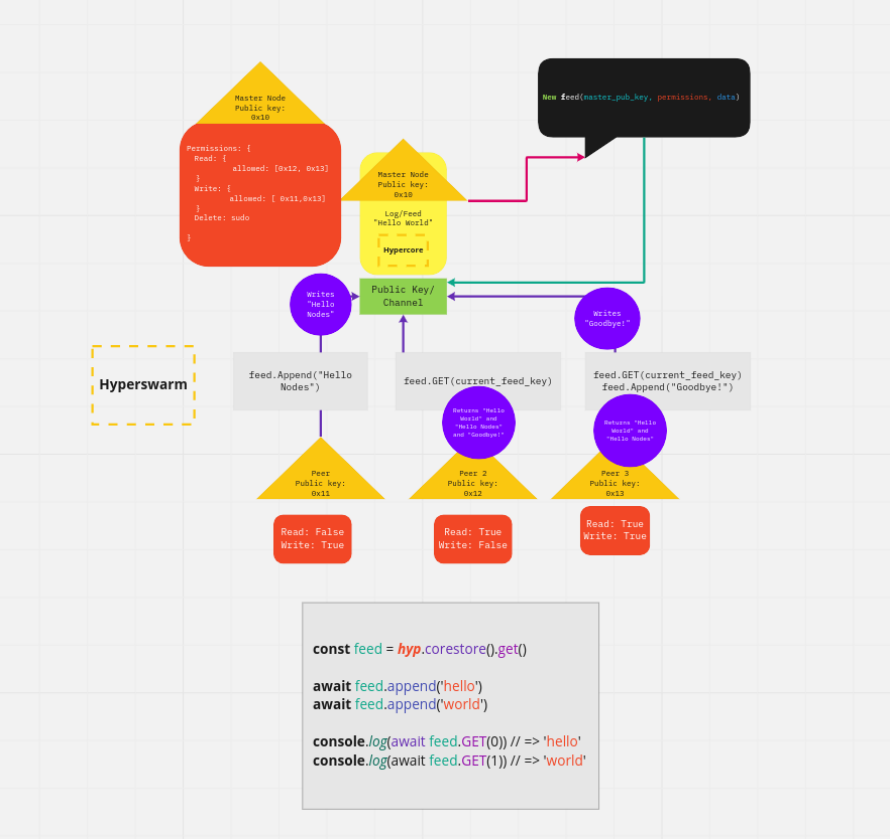
Through the proposed POC, we aim to demonstrate the viability of this approach by building a basic proof-of-concept that demonstrates how P2P nodes using Hypercore Protocol's hyperwarm can call and execute off-chain compute functions in a given region based on their signatures, which are signed and verifiable with Lit Protocol contracts. The POC will use microVMs provided by Firecracker to execute these functions in a secure and isolated manner.

Abstract

BFT (Byzantine Fault Tolerance) state machine replication is a technique used to ensure the consistency and integrity of data stored in a distributed system, even when some of the nodes in the system may be faulty or behaving in a malicious manner. In a DHT (Distributed Hash Table) based RPC (Remote Procedure Call) system for multi-tenant decentralized clouds and serverless functions, BFT state machine replication can be used to ensure that all nodes in the system are processing requests and storing data in a consistent manner. One way to implement BFT state machine replication in this context is to use EVM (Ethereum Virtual Machine) signatures and Rollups for consensus. EVM signatures are cryptographic signatures that can be used to verify the authenticity and integrity of data and transactions in the Ethereum blockchain. Rollups are a layer 2 scaling solution for Ethereum that allows for the batching and compression of multiple transactions into a single "rollup" transaction, which can then be included in the Ethereum blockchain.

To use EVM signatures and Rollups for BFT state machine replication in a DHT based RPC system for multitenant decentralized clouds and serverless functions, the following steps could be followed:

Each node in the system maintains a copy of the state machine, which is responsible for processing requests and storing data.



When a node receives a request to perform an action, it processes the request and generates a new state transition. The node then signs the new state transition using an EVM signature, and broadcasts the signed state transition to all other nodes in the system. Other nodes in the system verify the EVM signature to ensure the authenticity and integrity of the state transition.

If the signature is valid, the other nodes apply the state transition to their own copies of the state machine. If a node discovers that it has received conflicting state transitions from different nodes, it can use the EVM signatures and the consensus rules of the Rollup to determine which state transition is the correct one to apply.
By using EVM signatures and Rollups for BFT state machine replication, it is possible to ensure the consistency and integrity of data in a distributed system, even when some nodes may be faulty or behaving in a malicious manner. This can be particularly useful in a multitenant decentralized cloud or serverless functions environment, where multiple tenants may be interacting with the system and it is important to maintain the integrity of their data.

Universal Routing and Cross Platform NAT Solution using unique identifiers, public keys and signatures to create Namespaces for Endpoint (Nodes) and Service Discovery

How peers across various node configurations can connect to each other using the decentralized edge computing network:

- ➔ Peer A and Peer B both want to connect and use the network to access computer resources.
- ➔ Peer A and Peer B both run a node on their respective devices, using a decentralized protocol such as Hypercore to discover and connect to other nodes in the network.
- ➔ The nodes of Peer A and Peer B use the decentralized protocol to exchange information about the compute resources they have available and their capabilities.
- ➔ Based on the information exchanged, Peer A and Peer B can determine if they want to establish a connection and utilize each other's computer resources.
- ➔ If a connection is established, Peer A and Peer B can begin streaming data and executing code on each other's devices, using WebAssembly for code execution and Hypercore for data streaming.
- ➔ The connection between Peer A and Peer B is secured and authenticated using signed messages and keys managed by the Lit Protocol contracts.
- ➔ As long as the connection between Peer A and Peer B remains active, they can continue to stream data and execute code on each other's devices, using the decentralized edge computing network to power their compute needs.

To measure the throughput needed across light and full node clients on a protocol, you could use the following expression:

Throughput = (*Number of light node clients + Number of full node clients*) * *Average request rate per client*

This expression takes into account the total number of clients making requests to the protocol, as well as the average request rate per client. The request rate per client can be measured in requests per second, minute, or hour, depending on the desired level of granularity.

For example, if you have 50 light node clients and 50 full node clients making requests to the protocol at an average rate of 100 requests per second per client, the total throughput needed would be (50 + 50) * 100 = 10,000 requests per second.

It is important to note that this expression is a rough estimate of the throughput needed and may not take into account other factors such as the size of the requests, the complexity of the processing required, and the overall load on the system. A more detailed analysis may be needed to accurately determine the throughput needed for a specific protocol and client setup.

Measuring the throughput needed across light and full node clients on a protocol:

- ➔ Determine the total number of light node clients and full node clients that will be making requests to the protocol.
- ➔ Measure the average request rate per client over a period of time, using tools such as load testing software or monitoring systems.
- ➔ Calculate the total throughput needed by using the following formula:
- ➔ Throughput = (Number of light node clients + Number of full node clients) * Average request rate per client
- ➔ Monitor the actual throughput of the system over time to ensure that it is sufficient to handle the load and adjust the system as needed to meet the desired throughput.

It is also worth noting that there may be other factors that impact the throughput of the system, such as the complexity of the requests, the resources available on the server, and the overall load on the system. These factors should be taken into account when determining the throughput needed and making adjustments to the system.

Light clients via browsers play a role in decentralized systems by allowing users to interact with the system without the need to download and sync the full blockchain or other data locally. This can be particularly useful in situations where users do not have the resources or bandwidth to download and store large amounts of data, or where the data is constantly changing and would need to be constantly re-downloaded.

Light clients via browsers typically rely on a trusted full node to provide them with the necessary data and verify the authenticity of the data. The full node acts as a proxy for the light client, serving as a point of contact between the light client and the decentralized system.

Some examples of the role that light clients via browsers can play in decentralized systems include:

- Allowing users to access and interact with decentralized applications (dApps) without the need to download and sync the full blockchain.
- Providing a more user-friendly interface for accessing decentralized systems, particularly for users who may not be familiar with the technical details of the system.
- Enabling users to interact with decentralized systems from any device with a web browser, without the need to install specialized software.
- It is important to note that light clients via browsers may not have the same level of security and functionality as full nodes, as they rely on a trusted full node for their data and may not have the ability to fully validate the authenticity of the data. In some cases, it may be necessary for users to run a full node in order to participate in the decentralized system in a more secure and fully-featured manner

Binding a tail scale IP to a WebAssembly (Wasm) virtual iptable in the DOM (Document Object Model) can help provide missing TLS (Transport Layer Security) functionality in native browsers in the following ways:

Tailscale is a secure network overlay that allows users to connect their devices and networks in a secure manner over the internet. By binding a tailscale IP to a Wasm virtual iptable in the DOM, it is possible to establish a secure connection between the browser and the decentralized system, even if the system is running on a different network.
TLS is a security protocol that provides encryption and authentication for communications over the internet. By binding a tailscale IP to a Wasm virtual iptable in the DOM, it is possible to establish a secure TLS connection between the browser and the decentralized system, helping to protect against data tampering, interception, and other security threats.
Native browsers do not always provide the necessary TLS functionality for interacting with decentralized systems. By binding a tailscale IP to a Wasm virtual iptable in the DOM, it is possible to add the missing TLS functionality to the browser, enabling users to securely access and interact with decentralized systems.
It is important to note that binding a tailscale IP to a Wasm virtual iptable in the DOM is just one way to add missing TLS functionality to native browsers. There are also other approaches that may be used, depending on the specific requirements and constraints of the decentralized system and the users' needs.

Proof of Service Consensus

In order to incentivize the provision of reliable compute resources and to accurately price the use of these resources, the deMEC platform utilizes a combination of Proof of Stake, Proof of Zero Knowledge, and Proof of Signature. This consensus mechanism, known as Proof of Service, allows for the creation of dynamic pricing models using bonding curves, which are defined using Bezier formulas. These curves allow for the predictable and transparent determination of prices based on the current demand for compute resources and the available supply. The use of bonding curves helps to ensure that the cost of using the deMEC platform is fair and sustainable for both providers and consumers of compute resources.

There are a few different ways that proof of service could use dynamic pricing for individual services and invocations via bonding curves to determine the price of services. A given stake is issued to pair value to a redeemable service in the form of an access or utility token. Upon ownership, Lit Protocol Contracts, can be utilized to provide redeemability in a non-custodial manner. Refunds/Redemption may also be sold back into a supply curve such that one could “pinpoint” or deterministically retrieve a market position to sell based on current thresholds. This allows real-time, liquid markets to be closely integrated One approach would be to use a Bezier curve to model the relationship between the amount of service being provided and the price of that service.

The curve could be defined by a set of control points, which would be used to shape the curve and determine how the price changes as the amount of service increases. The equation for a Bezier curve is as follows:

$$\left[B(t) = \sum_{i=0}^{3} (1-t)^{3-i} t^i P_i\right]$$

In this equation, B(t) represents the position on the curve at a given value of t (which ranges from 0 to 1), and P0, P1, P2, and P3 are the control points that define the curve. By adjusting the values of these control points, you can create a variety of different curve shapes, each of which will have a different relationship between the amount of service being provided and the price of that service.

One of the key benefits of using a proof of service model, particularly in conjunction with dynamic pricing or bonding curves, is the ability to offer on demand computing and pay as you go models for cloud services in a more fair and transparent manner compared to traditional cloud providers like AWS.

With traditional cloud providers, pricing can be opaque and difficult for users to understand, with costs often tied to upfront commitments and long-term contracts. In contrast, the use of proof of service and dynamic pricing allows for real-time pricing that accurately reflects the current demand and supply of compute resources. This can lead to significant cost savings for users, as they only pay for the resources they actually use and are not locked into expensive long-term contracts. Furthermore, the use of bonding curves and other mechanisms for managing the price of compute resources allows for a more balanced and sustainable ecosystem, as it incentivizes providers to offer high quality services in order to attract and retain customers. This can lead to improved reliability and performance, further enhancing the value

proposition for users. Overall, the integration of proof of service and dynamic pricing in the provision of cloud services has the potential to revolutionize the industry and provide a more equitable and efficient alternative to traditional cloud providers.

Bonding curves are mathematical functions that can be used to model the relationship between the supply and demand of a particular resource, such as compute power or network bandwidth. They can be used to calculate the price of this resource based on various factors, such as the number of bit operations per floating point or the number of megabytes per second of bandwidth.

One way to model the price of compute using bonding curves is to measure it in terms of bit operations per floating point. This can be done by using a formula such as:

Price = a * bit operations + b * floating points in combination with a standardization of kWh with respect to a universally composable medium for that given resource.

Where "a" and "b" are constants that can be adjusted to reflect the relative value of bit operations and floating points.

Similarly, the price of network bandwidth can be calculated using bonding curves by measuring it in terms of megabytes per second and taking into account factors such as latency and ping. This can be done using a formula such as:

Price = c * megabytes + d * seconds + e * latency + f * ping

Where "c", "d", "e", and "f" are constants that can be adjusted to reflect the relative value of each factor.

By using bonding curves to model the price of compute and network resources, it is possible to create a more fair and efficient market for these resources, particularly in the context of on-demand computing and pay-as-you-go models.

Proof of Zero Knowledge (PoZK) is a consensus mechanism that allows for the verification of a statement without revealing any information about the statement itself. This can be achieved through the use of zero-knowledge proofs, which allow a prover to demonstrate the truth of a statement to a verifier without revealing any additional information about the statement.

Proof of Stake (PoS) is a consensus mechanism in which the right to create a new block is determined by the stake, or ownership, of the validating node. In a PoS system, validating nodes (also known as "validators") are chosen to create new blocks based on their stake in the network. The higher the stake, the higher the probability of being chosen to create a new block.

Proof of Signature (PoSign) is a consensus mechanism that uses digital signatures to verify the authenticity of transactions. In a PoSign system, validating nodes use their private keys to sign transactions, and the signatures are verified by other nodes using the corresponding public keys.

By combining PoZK, PoS, and PoSign, it is possible to create a consensus mechanism for code execution that is both secure and efficient. The use of PoZK allows for the verification of statements without

revealing any additional information, while PoS and PoSign provide additional security and authenticity to the process. This combination of consensus mechanisms can be used to ensure the integrity and reliability of code execution on a decentralized network.