Utility: A Verifiable Computing Network Based on Instruction Signatures

utnet.org
UTNET research@utnet.org

About UtilityNet Foundation

GRAND TECHNOLOGY INTERNATIONAL LIMITEL is an ecology-building fund invested by the UtilityNet development community.

Assets/funds under management are used for, but not limited to, the following purposes:

1. On-chain ecological project investment/incubation
2. Off-chain ecological project investment/incubation
3. AI Computing network development and upgrade maintenance
4. Metaverse open source project support
5. Frontier technology and thinking subject research investment
6. Special incentive for miner nodes
7. Development related commercial and public welfare investment
8. Blockchain carbon neutral field project investment
9. On-chain ecological project investment/incubation

Abstract

In this paper, we will discuss in detail Utility Network, a new blockchain architecture that builds consensus based on verifiable computational instructions. We will look at the the scaling of parallel computation, and the difficulties of its implementation. Then this paper will propose a new consensus based on computing instructions (Proof Of Computation Integrity, POCI) and analyze various aspects of POCI and Computing Avaliability Over Time (CAT). This paper also presents the concepts of blockchain and data structure, token cycle, and computational ecology that match the POCI consensus. The next section immediately follows with an in-depth discussion of designing execution VMs. Chapter 6 demonstrates the artificial intelligence(AI) computational Network based on the Utility network blockchain. The article then proceeds to describe the features and advantages of the Utility network in various aspects. Finally, at chapter 8 the future and possibilities of Utility network will be discussed, including technological innovation, application scenario expansion, and market potential. The summary will reiterate the importance of Utility network as an innovative blockchain network and computing consensus technology, and its impact on the overall computation and financial ecology.

1 Introduction

Utility network is a new distributed blockchain network technology based on Tensor Processing Unit (TPU) chip computational power that combines peer-to-peer payments and verifiable decentralized computation for parallel computation, revolutionizing the blockchain network ecosystem and computing market. Utility uses a consensus mechanism based on verifiable computational power with command signatures to enable secure, transparent and efficient transaction processing. At the same time, Utility provides more powerful hardware and software support for distributed computation by introducing execution virtual machines, Kubernetes, WASM and PyTorch integration, and cross-domain data scheduling.

The Utility network builds consensus based on chip verifiable computation, combining proof of computational power with blockchain cryptography to further improve security and transparency of the system. With the new computational consensus and proof of computational power, Utility achieves efficient performance and various availability, while ensuring distributed mining, decentralized finance (Defi), computation scheduling, big data processing and artificial intelligence (AI) areas are satisfied.

The Utility network provides a more flexible programming environment for developers by integrating technologies such as Solidity VM, Kubernetes, WASM, and PyTorch and other technologies to facilitate cross-domain data scheduling and collaboration and AI development. Our team is committed to making Utility network as a promising blockchain network with widespread application and market potential blockchain network that will have a profound impact on the entire computation and financial ecosystem [1-4].

1.1 Current advantages and disadvantages of large-scale computation

As the founding team of the technology, we are always focused on the trends of large-scale computation and its real-world applications. Currently, the Large-scale computation is mainly supported by two architectures: Intel general-purpose computing and NVIDIA's heterogeneous parallel computing architecture [9, 10]. These two architectures have their respective advantages and disadvantages, and we will analyze them in details.

1.1.1 Intel's General Computation

Intel's general-purpose computation is based on a multi-core CPU architecture [9], emphasizing the performance of single-core processors while enabling parallel multi-task parallel processing. General-purpose computation is highly flexible and can be applied to a variety of computational requirements, especially for tasks that require highly serialized processing. However, the disadvantage of this architecture is that it is less energy-efficient and less able to cope with massively parallel computing scenarios, such as graphics rendering, deep learning, and scientific computation, etc.

1.1.2 NVIDIA's Heterogeneous Parallel Computing Architecture

NVIDIA's heterogeneous parallel computing architecture uses Graphical Processing Unit(GPU) as computational unit[11], with a large number of parallel processing cores. This architecture offers significant advantages in handling massively parallel tasks, such as image processing, machine learning, and big data analysis, etc. Compared to general-purpose computation, heterogeneous parallel computation offers energy efficiency and performance. However, it also has the limitation that for highly serialized tasks, its performance is not as good as general-purpose computation.

1.1.3 Advantages of Heterogeneous Parallel Architecture

By comparative analysis, heterogeneous parallel computation architectures have more potential. In more and more scenarios, the demand for massively parallel computation is constantly increasing, especially in the domain of artificial intelligence(AI), big data processing and scientific computation. The heterogeneous parallel computation architectures can effectively improve computational performance and energy efficiency to meet the needs of these areas.

However, the challenge of large-scale computing devices and the ability to network services still exists. Firstly, the investment cost of large-scale computing devices is high, especially in the deployment of distributed computing devices. In addition, in terms of network service capability, large-scale computational networks is required to handle huge data transmission, coordination, and security. These issues need to be solved in the network construction.

In summary, heterogeneous parallel computation architecture is the development direction of large-scale computation, yet it still needs to overcome the challenges of investment cost and network service capability.Utility network is a blockchain network designed to solve these problems, which uses blockchain technology and computation consensus mechanism to achieve efficient utilization and coordination of distributed computing resources. By introducing technologies such as verifiable computation, execution virtual machines and cross-domain data scheduling, the Utility network provides powerful support for large-scale computation.

The decentralized nature of UT helps reduce the investment cost of large-scale computational equipment, enabling more efficient resource sharing between providers and demanders of computing resources through incentives and rental markets. In addition, the Utility network uses advanced P2P technology and computational power certification mechanisms to achieve high performance and secure computation of high performance, security and reliability. As an innovative blockchain network, Utility network has a wide range of applications and market potential. By solving the financial problem of large-scale computing equipment deployment and the problem of network service capability, Utility has a great potential to become the future support for large-scale computational development.

1.2 Scalization of parallel computation

Scalization of parallel computation is the future trend of computing development, which will bring about more efficient, economical and flexible utilization of computing resources. Filecoin, as a decentralized storage public chain, effectively integrates storage resources worldwide through the token mechanism to effectively integrate storage

resources worldwide, achieving a storage scale of over 20 EB, which is difficult to be achieved by any single data center. The success of Filecoin has enlightened us to integrate and optimize massively parallel computation through blockchain networks and token economies. [4, 5]

Through the blockchain network, Utility network is expected to achieve ultra-large scale chip deployment and to solve the computation problem. Utility network utilizes the proof of computational power and computational consensus mechanisms to integrate and coordinate computing resources around the world, enabling more efficient resource utilization. Meanwhile, UtilityNet coin(UNC) economy cycle model and application development provide a platform for both providers and demanders for computing resources, pushing to the scalized development of parallel computation.

In the field of AI, especially the SoA (State of Art) general-purpose Transformer models have made significant breakthroughs in a number of fields, including natural language processing, computer vision, and reinforcement learning [23]. However, these models are extremely demanding on computing resources, especially in AI training and inference. The massively parallel computation achieved by the Utility network can provide the necessary resources for these models, driving the development and innovation of AI technology.

Utility network can apply the blockchain network and token economy to the scalization of parallel computation, which offers great potential for future AI computation. potential for the future of AI computation. By effectively integrating global computing resources, this network system is expected to drive innovation in computation, open up new paths and opportunities for AI technology development

1.3 Blockchain's difficulties on execution

As a revolutionary technology, blockchain brings great potential for applications in various industries with its features of decentralization, high transparency and security. However, in practical applications, blockchain faces many difficulties in landing. The following will elaborate several blockchain landing challenges from the perspective of resource efficiency and service scenarios.

First, cryptography does not perfectly solve many problems. Although cryptography provides security guarantees for blockchains [6-8], in practical applications, it introduces additional computation and storage costs. For example, in order to verify the computation, a large number of invalid computations are required to complete the Proof of Work (PoW). Similarly, in order to validate storage, a space proof technique similar to Filecoin is required to fill up the storage space [1]. The resources consumed in these validation processes often far exceed the actual demand, resulting in serious inefficiencies in the utilization of resources.

Second, blockchain technologies often waste a lot of resources while satisfying consensus. For example, in blockchain networks such as Bitcoin and Ethereum that use the PoW mechanism, miners need to perform a large number of hash calculations to find blocks that satisfy the consensus conditions. The power and computing resources consumed during these calculations are not available for other valuable computational tasks, resulting in a waste of resources.

Finally, there is a contradiction between the demand for efficient service of resources in practical application scenarios and blockchain technology. In many scenarios, such as finance, Internet of Things and supply chain management, there are strict requirements for data processing speed, real-time and resource utilization. However, blockchain technology currently still has limitations in terms of performance, scalability and resource efficiency to meet the needs of these application scenarios.

To sum up, the difficulties of blockchain implementation are mainly manifested in the contradiction between resource efficiency and the demand of practical application scenarios. To solve these problems, we need to continuously explore and innovate to find a more efficient, scalable and resource-efficient blockchain technology solution. Utility Network is a new type of blockchain network born to solve these thorny problems. By introducing technologies such as efficient computing consensus, blockchain data structure, token cycle and computing ecological model analysis, isolated execution environment and execution virtual machine, integration of different language environments, and cross-domain data scheduling, it realizes efficient utilization of distributed computing resources and coordination, which provides unlimited possibilities for blockchain technology in practical application scenarios.

1.4 P2P network bridging computing networks

The tight combination of P2P networks and computational networks can bring higher performance, scalability and resource utilization to blockchain technology. This section will explain the process of combining computational consensus-based and P2P computation from a professional perspective, as well as the implementation of instructions and computational verification.

Computational consensus mechanism is the core of bridging P2P networks with computational networks. By introducing computational consensus, computing resources can be integrated and coordinated globally to achieve more efficient resource utilization. Compared with the traditional Proof of Work (PoW) and Proof of State (PoS) mechanisms, the computational consensus mechanism can better meet the needs of practical application scenarios while reducing resource waste.

P2P computation can provide more powerful computational capabilities for blockchain networks [3]. Blockchain networks based on P2P computation can distribute computational tasks among nodes, thus improving computational efficiency and network performance. In addition, P2P computation enables dynamic allocation of computing resources to suit the needs of different application scenarios.

In terms of instruction and computation validation, blockchain networks based on a combination of computational consensus and P2P computation can employ techniques such as verifiable instruction execution and isolated execution environments. Verifiable instruction execution techniques can verify the correctness of computation results without revealing the content of the computation. Isolated execution environments (e.g., execution virtual machines) can ensure the security and reliability of the computation process. With these techniques, blockchain networks can achieve effective verification of instructions and computations, further improving the performance and resource utilization of computing networks [13, 16]. These techniques will be developed in detail in the following sections.

The blockchain network based on the combination of computational consensus and P2P computation can achieve an effective bridge between P2P networks and computational networks. By introducing computational consensus mechanism, P2P computational technology, and instruction and computational verification technology, this network can provide higher performance, scalability, and resource utilization for practical application scenarios, thus creating new opportunities for the development of blockchain technology.

2 Computational Consensus and Proof of Computational Power

Computational consensus is an important mechanism in blockchain technology for solving data consistency and trust problems in a distributed environment. Proof of Work(PoW)is one of the commonly used consensus algorithms, the basic principle of which is to prove one's contribution to the network by consuming a large amount of computing resources to gain bookkeeping rights and rewards.

PoW plays an important role in ensuring the security of blockchain networks, but there are also some problems, such as consuming large amounts of energy and computing resources, and easily triggering computation competitions and centralization. To solve these problems, some new consensus algorithms such as Proof of Stake(PoS), Proof of Authority(PoA), etc. have emerged in recent years [4, 5, 18]. In this section, we introduce our brand new proof of computation mechanism: Proof Of Computation Integrity (POCI).

Different consensus algorithms are applicable to different scenarios and applications, and need to be selected and used according to the actual situation. Computational consensus and proof of computational power is one of the core mechanisms in blockchain technology, which is important to ensure the reliability and security of blockchain.

2.1 Instruction set encryption

Instruction Set Encryption (ISE) is a technology that encrypts a computer instruction set to protect program code and data. Its main purpose is to improve the security of the system and resist attacks such as code tampering and code injection. As a high security feature of the chip instructions, cryptographic security that guarantees the absolute safety of the chip owner or miner, the computation energy chip is also equipped with the basic instruction set encryption technology. In systems that use instruction set encryption, the processor uses a key to encrypt and decrypt the CPU instruc-

tions. The encrypted instruction set is stored in memory or on disk, and the decryptor is responsible for decrypting the encrypted instructions and passing them to the CPU for execution. It is important to note here that only authorized decryptors can decrypt and execute the encrypted instructions, and non-authorized decryptors cannot execute the encrypted instructions, thus protecting the security and integrity of the program and data. Instruction set encryption is usually used in systems with highly secure requirements, such as financial transactions, military equipment, smart cards, etc. The advantage of instruction set encryption over traditional software encryption is its ability to protect code and data at runtime, making it more difficult to be cracked by attackers. In the future, instruction set encryption will also play an important role in protecting data and core digital assets, etc. in our future Utility mining or AI training services.

2.2 Verifiable instruction execution

Verifiable instruction is a special type of computer instruction whose purpose is similar to the instruction set encryption mentioned in the previous section, and it is an important technique to guarantee the security of computer programs, which has wide application in systems with high security requirements. In contrast to instruction set encryption, verifiable instruction execution focuses more on the security and correctness of program operation, so that the execution process of a computer program can be verified and the result of the program can be ensured to be correct. Verifiable instructions are often used in systems with high security requirements, such as smart cards, encrypted communications, financial transactions, and other areas. Among the chip instructions, there are basic instruction sets such as the following ones:

1.SGX instructions: Intel SGX is a security enhancement technology that can be used to protect confidential information and code. the SGX instruction set includes enclave creation, enclave entry, and enclave exit instructions to ensure the confidentiality and security of the computing process.

2.ARM TrustZone Instructions: ARM TrustZone is a security technology that can be used to secure processors and systems. The TrustZone instruction set includes instructions such as Secure Monitor calls, TrustZone registers, and other instructions to ensure the security and integrity of the computing process.

3. MIPS TrustZone instructions: MIPS TrustZone is a security technology that can be used to protect the security and integrity of the system. The MIPS TrustZone instruction set includes Secure Monitor call, Secure Monitor return and other instructions to ensure the confidentiality and security of the computing process.

4. RISC-V PMP instructions: RISC-V PMP (Physical Memory Protection) is a memory protection technology that can be used to protect the security and integrity of a system. the PMP instruction set includes instructions such as permission management for physical memory access, which is used to ensure the security and integrity of programs.

The implementation principle of verifiable instructions is based on the zero-knowledge proof technique and computable security technique in cryptography. Specifically, verifiable instructions can achieve the following important program protection functions: 1. Prevention of malicious tampering: verifiable instructions can ensure that the execution process of instructions is not tamperable and prevent malicious tampering with the execution process of programs. 2. Confirmation of program correctness: verifiable instructions can prove that the execution process of programs is correct, thus avoiding system crashes and data corruption caused by program errors. 3. Prevention of Information leakage: Verifiable instructions can ensure the security of programs and data, and prevent information leakage and malicious attacks. 4. Improvement of performance efficiency: Verifiable instructions can reduce the execution time and computation of programs, and thus improve the performance efficiency of programs. Overall, verifiable instructions play a critical role in the security, integrity and correctness of program operation.

2.3 Isolated execution environment (execution virtual machine) environment proof

Proof of Isolated Execution Environment (PIEE) is a proof method based on mathematical and cryptographic techniques to prove the correctness and security of isolated execution environments in processors. We know that every cloud server is at risk of being attacked, and it is extremely important to ensure the security of user code running and information data storage on it. Introducing the concept of Trusted Execution Environment and

Confidentail Computation (CC) can better address the execution environment proof challenge. In the CC approach, the execution environment guided by the cloud system software is generally considered unreliable, and it is recommended to run security sensitive workloads in an isolated Trusted Execution Environment (TEE). the security assurance of TEE is rooted in the deep hardware layer of the platform, and remote authentication will be used to verify and declare security.

The key words of PIEE here are isolation and remote proof. The main isolation means include memory isolation, CPU isolation, network isolation and file isolation, etc. These means are good to ensure that the workload runs in a reliable TEE, and the next remote proof will be in the form of cryptography to fully prove the security of TEE operation. Kubernetes (K8s for short) will be used in the Utility network architecture to deploy the runtime environment and obtain a segment of TEE for proof by means of isolation, then the specific proof of security is as follows:

A random task function (computational operator)θ is obtained for a node $\xi(x)$ that needs to perform PIEE in the language framework (e.g. PyTorch) specified in the deployment environment and a task U is extracted from the Kubernetes execution container to perform a verification operation to obtain the result:

$$Q_\theta = \xi(U_\theta) \tag{1}$$

where U_θ is the task specific input value at that node and Q_θ is the result of the calculation.

1.The computation results are cryptographically signed and packaged and published on the chain along with the function operators and tasks. 2.Each node obtains this function operator $\xi(x)$ and the encrypted result of Q_θ, re-decrypts it with the public key to obtain the original result Q_θ, and performs the computation in its respective environment:

$$Q_1 = \xi(U_1) \quad Q_2 = \xi(U_2)... \, Q_i = \xi(U_i) \tag{2}$$

where U_i and Q_i are the task input and final computation result corresponding to node i, respectively. If the computation result is the same as Q_θ, it proves that the PIEE of the original node passes, and the PIEE verification of each node is done as mentioned analogously.

2.4 POCI (Proof Of Computation Integrity)

Blockchain consensus mechanisms are commonly found in the proof of computing power, where the computational power of the miner will be tied to block-keeping rights and rewards. Traditional proofs of computing power first relied on PoW, which is implemented by calculating an intractable mathematical puzzle to consume computing resources, and the solution to the puzzle must satisfy certain conditions, then it is considered to have the computing power resources to satisfy the conditions. For example, the puzzle in Bitcoin is to hash a random number to obtain a specific hash value H, which must satisfy the difficulty requirement Diff set by the Bitcoin network ($H{>}Diff$) [15, 25]. Therefore, solving this difficulty requires a large amount of computation, which consumes a lot of time and computing resources, which is the core idea of computation proof. However, this idea suffers from a serious drawback, both in terms of huge energy and computational resource consumption, and in terms of failure to generate practical social value implications (e.g., the permanent storage feature of the Arweave network), generating negative impacts on the environment and the economy that cannot be ignored. In addition, proof of computation is also prone to computation competitions and centralization, because only large mines and pools have enough resources to participate in the network, and small nodes have difficulty competing, which may lead to the concentration of power in the network in the hands of a few people, which is suspected to be contrary to the original decentralized idea.

The proof of computation demonstrated in this subsection will use a new proof mechanism to indirectly prove computational power through trusted computation ownership, i.e., Proof Of Computation Integrity (POCI). The proof mechanism completely abandons the high-cost, high-consumption computing model and replaces it with the chip computation ownership model. In our Utility network chain, high-performance Sophon TPU computational chips are utilized to form a powerful computing network that will be used to provide users with decentralized mining and artificial intelligence training services in the future. These include: high performance up to trillion level floating point operations per second; wonderful up to trillion level floating point operations; low power consumption; opti-

mized hardware design and flexibility compatible with a variety of development frameworks such as TensorFlow, PyTorch and other software libraries.

Figure 1 below shows a Sophon TPU with Security Engine (SE). The serial number is etched on the chip form a unique identification for the chip. What is the most importance is the etching of "secure key", which is a 128-bit AES(Advanced Encryption Standard) key, on a small storage area, called efuse. It is usually regarded as the "secret identity" for each TPU chip. The secure key can only be sent to do cryptological calculation (AES encryption and decryption) inside the SPACC(security protocol accelerator) model of the chip(shaded area of figure 1) without exposure, hence it proves the confidentiality and uniqueness. On the other hand, at the PKA(public key accelerator) model, it allows signature by private key and verification by public key. ECC(Ellipse Curve Cryptography) algorithm is adopted in PKA model. By taking a general base point P and a big random seeds as private key $PriK$, with the ECC encryption equation, one is able to get the public key $PubK$ [7, 8]:

$$PubK = P \times PriK \tag{3}$$

With these tools and keys, the following is basically the principle of how every chip owner can prove its ownership of computation with this secure key etched, SPACC model, along with PKA model:

User will first etch the secure key S on the TPU chip one time, and active the SPACC model. Chip provider usually provides the private key $PriK$ and public key $PubK$ to user for every unique chip, with its encryption and decryption algorithm. At SPACC model, user can get the encrypted private key $PriKEnc$ by AES encryption:

$$PriKEnc = \mathcal{A}_{\mathcal{E}}(S, PriK) \tag{4}$$

User then upload the $\{PriKEnc,\ PubK\}$ key pairs to the blockchain to claim its ownership of the chip, spreading to all nodes for verification. Noted that the information is cryptographically protected and recorded on the blockchain, ensuring tamper-proof security. Origin $PriK$ can be dropped permanently for safety or stored by user locally in a secret place. Thanks to the SPACC model in the chip, malicious node is unable to steal $PriK$ from $PriKEnc$ without the correct chip, since:

$$PriK = \mathcal{A}_{\mathcal{D}}(S, PriKEnc) \tag{5}$$

where $\mathcal{A}_{\mathcal{D}}$ is the AES decryption function, so a wrong or void chip fails to give the correct S, thus wrong $PriK$, making $PriK$ absolute confidential and signature absolute authorized to the chip. Therefore, the user or node that obtains or possesses the chip can claim ownership of the chip's computational power by POCI, without having to perform high-powered mathematical calculations to prove the computational power. The next part will show how POCI is performed by a simple algorithm.

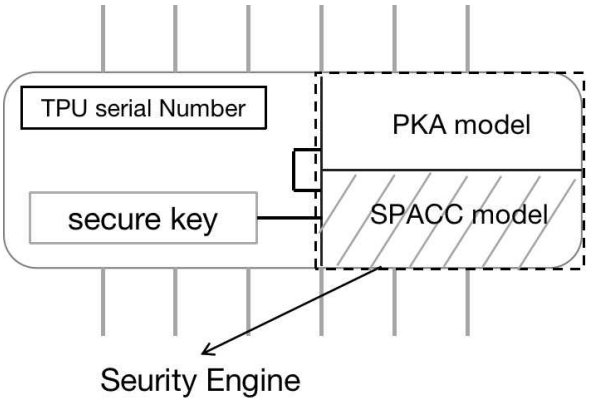

Figure 1: simple illustration based on TPU Security Engine

Suppose the node θ obtains a TPU with the serial number α, then he will obtain an encrypted instruction set σ(decoder and decoding algorithm provided by the chip vendor) from which the signature function $Sign_\alpha(x)$ can be obtained and user can sign a message m [7, 8]:

$$Sign_\alpha = \mathcal{ECC}_{\mathcal{E}}(\mathcal{A}_{\mathcal{D}}(S, PriKEnc)) \tag{6}$$

$$E_\alpha = Sign_\alpha(m) \tag{7}$$

$\mathcal{ECC}_{\mathcal{E}}$ is the ECC encryption signature based on original $PriK$ recovered by SPACC model and E_α is the signed encrypted message by $Sign_\alpha$. Other verifying nodes verify it by the verification function $V_\alpha(x)$, obtained from the ECC decryption $\mathcal{ECC}_{\mathcal{V}}$ based on public key, if it satisfies:

$$V_\alpha = \mathcal{ECC}_{\mathcal{V}}(PubK) \tag{8}$$

$$m = V_\alpha(E_\alpha) \tag{9}$$

Then POCI succeeds and proves that θ legitimately owns α, then it acquires the computational power of α, which is marked as ϕ and can claim to possess the computational power of size ϕ.

2.5 Computing Availability over Time (CAT)

According to the core idea inside the previous subsection POCI, we can then define the cluster computing power and online availability of computational power (Computing Availability over Time, CAT), the real sense of allocating the whole network computational power and mining interests. Again back to the node θ, if it claims to have obtained N computational power ϕ, then according to the principle mentioned above, it will accept N signatures to verify POCI, which is obviously a bit tedious, especially when this N is above the order of a thousand, it will greatly increase the proof workload. Therefore, the following integration optimization is proposed to integrate N signature functions into a new signature function $Sign\chi$, which is generated by the regular integration operator τ, i.e:

$$E\chi = Sign\chi(m) = \tau(Sign_1(m)+ (Sign_2(m) + ... + (Sign_N(m)) \quad (10)$$

$E\chi$ is the signed message after integration of N signatures, and other verifying nodes verify it by the integration verification function $V\chi$ of N public keys, where:

$$V\chi(x) = \tau^{-1}(V_1(x) + (V_2(x) + ... + (V_N(x)) \quad (11)$$

Considering the regularity and commutativity among signature function, verification function and operator τ:

$$\tau^{-1}V_i(\tau Sign_i) = V_i(Sign_i) \quad (12)$$

$$\tau^{-1}V_i(\tau Sign_j) = 0(i \neq j) \quad (13)$$

It is not difficult to find out that if the following are satisfied:

$$Nm = V\chi(E\chi) = V_1(E_1)+ V_2(E_2) + ... + V_N(E_N) \quad (14)$$

then it is proved that θ has gained the computational power of $N\phi$. If the total computational power of the current network is C, then the ratio η of θ to the total network computational power is:

$$\eta = \frac{N\phi}{C} \quad (15)$$

This is the computational power under POCI. The computational power of each node will be serialized into segments, and the computational power of a chip is counted as a unit segment. $N\phi$ is abstractly sliced into N segments and labeled with serial numbers, as shown in Figure 2 below. A total segment consists of 1, 2, ... , n nodes form a total segment λ, and the length of the segment obtained by each node depends on the computational power size. And by applying VRF (Verifiable Random Function) function, the total fragment is random sampled at time interval T to get the serial number κ, then the node corresponding to the fragment of the chip serial number searched is the burst block node:

$$\kappa = VRF(T, \lambda) \quad (16)$$

$$\kappa \to \phi(\kappa) \quad \phi \to \theta(\phi) \quad (17)$$

This mechanism shows that the node fragment with large computational power occupies more space and has a higher probability η of being chosen by VRF, thus achieving POCI.

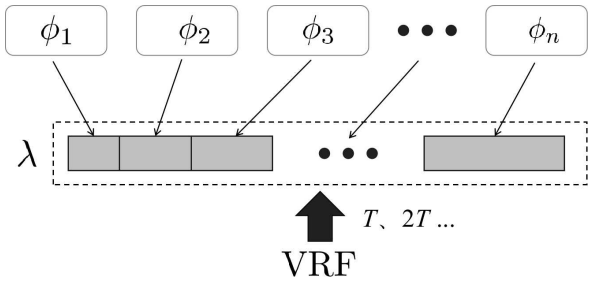

Figure 2: Random sampling process based on VRF for computational fragments composed of nodes

However, in order to incentivize nodes to maintain their computational power, nodes that obtain the right to burst blocks need to perform CAT verification, which repeats previous POCI challenge, to prove again that they are in a highly computational power state at any moment t. Similarly, the digital signature of the computational chip and the verification session of each node are performed again, and the block reward is obtained after completion. In this way, POCI computational proof and CAT complete the implementation of the POCI mechanism of the entire Utility network in a way of low-energy, low-cost and high-efficiency.

2.6 RPOI (Rent Proof Of Instructions)

After completing the POCI, the computation owners can consider how to further utilize the computational power efficiently, they either consider on-chain mining to earn tokens (reward distribution will be described in the next subsection), further

purchase computation, or perform lease (rent) service clusters to clients for distributed mining or AI training tasks, etc. In this section, we will define the proof of lease ownership, namely Rent Proof Of Instructions (RPOI), to complete each secure and complete computational lease agreement.

Define β as the computation leaser and R as the user. Suppose R sends a request to β to lease M TPU chips ϕ, the request $\gamma(R \to \beta, M\phi)$ is signed with a private key[7, 8] along with timestamp T, and the transaction is packaged and recorded on the block along with the current deposit ρ and rental ω, the transaction hash $H(R \to \beta, M\phi, T, \rho, \omega)$ is obtained and broadcast on the blockchain, and the funds are pledged to the public ledger address π with the encryption by the public key of π. After β receiving the message, decrypting and verifying, he will add the chip encryption instruction set and the relevant set of encrypted files of computational power lease σ to the message string, sign, and encrypt it with the public key of R and send it to R. R decrypts its own private key to obtain the lease and usage of computational power:

1. R sends a request to β: $\gamma(R \to \beta, M\phi)$

R completes signature:

$$E_R = sign_R(m(\gamma(R \to \beta, M\phi), T, \rho, \omega))$$

R submits a transaction on the chain:

$$H(R \to \beta, M\phi, T, \rho, \omega) = SHA(E_R)$$

2. R pledges funds: $K_\pi(E_R, \rho, \omega) \to \pi$

3. β signature check: $m(\gamma, T, \rho, \omega) = V_R(E_R)$

β signs and re-encrypts:

$$Enc = K_R(sign_\beta(m'(m + \sigma(M\phi \to \beta))))$$

4. R decrypts to obtain computational usage: $m'(m + \sigma(M\phi \to \beta)) = V_\beta(D_R(Enc)) \to \sigma(M\phi \to \beta)$

The lease message string $m(R \to \beta, M\phi, T, \rho, \omega)$ and the encrypted computation file $\sigma(M\phi \to \beta)$are the first phase of RPOI(RPOI phase I, Figure 3) completed by R.

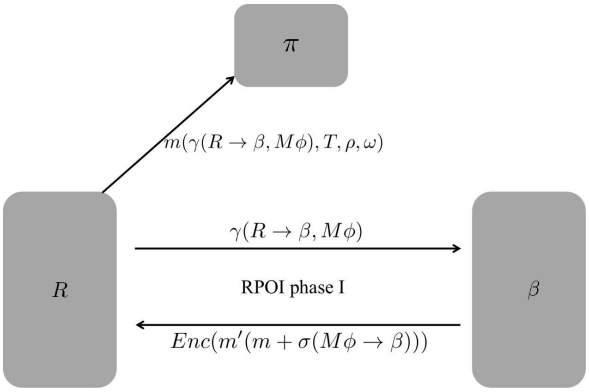

Figure 3: Schematic diagram of RPOI phase I process

When R has finished using and want to return the computing power, a mutual rating (Rating) and sign is needed to complete for the refund of the deposit on R and the rental income on β. The steps are listed as following:

1.Generate a rating message $Ra_R(R \to \beta)$ for β and integrate $m(R \to \beta, M\phi, T, \rho, \omega)$ and $\sigma(M\phi \to \beta)$ into a new message string, signed and sent to β:

$$E'_R = sign_R(m'(m + Ra_R)) \tag{18}$$

2.The public key verifys the signature E'_R and generates the message rating $Ra_\beta(\beta \to R)$ for R, combines, signs and encrypts using the public key of π to get the "final message":

$$mfinal = K_\pi(sign_\beta(m''(m' + Ra_\beta))) \tag{19}$$

3. Finally, the $sign_\beta(m''(m' + Ra_\beta))$ is submitted to the blockchain and the encrypted "last message" $mfinal$ is sent to π, where π decrypts it with its own private key to confirm the completion of RPOI, and then generates the scheduling operators ζ and ξ to act on the encrypted request message $E(m(\gamma(R \to \beta, M\phi), T, \rho, \omega))$ decrypted by public key of R, and dispatches funds from the pool to send the deposit and rent to R and β, respectively, to complete the process of computing power leasing:

$$mfinal \to \zeta(\gamma(R \to \beta, M\phi), \rho, \omega) \Rightarrow \rho \to R \tag{20}$$

$$mfinal \to \xi(\gamma(R \to \beta, M\phi), \rho, \omega) \Rightarrow \omega \to \beta \tag{21}$$

Here $mfinal$ contains information about the previous lease request, the amount of funds, and the mutual rating of the two, which is verified by π to complete the second phase of RPOI (RPOI phase II, Figure 4). π can be considered as an impartial

adjudicator, and the absence of any part of the information in $mfinal$ will result in the failure of fund allocation. In the first step above, if fails to generate a rating message, $mfinal \rightarrow \zeta(\gamma(R \rightarrow \beta, M\phi), \rho, \omega) \Rightarrow \rho \rightarrow R$ will fail to execute; and in the second step, if fails to generate a rating message, $mfinal \rightarrow \xi(\gamma(R \rightarrow \beta, M\phi), \rho, \omega) \Rightarrow \omega \rightarrow \beta$ will fail to execute, and the deposit ρ and the rental ω will all be returned to the address R . Each of the above message delivery is encrypted and decrypted by signatures and public-private keys to guarantee the process tamper-evident and prevent malicious listening [19, 20].

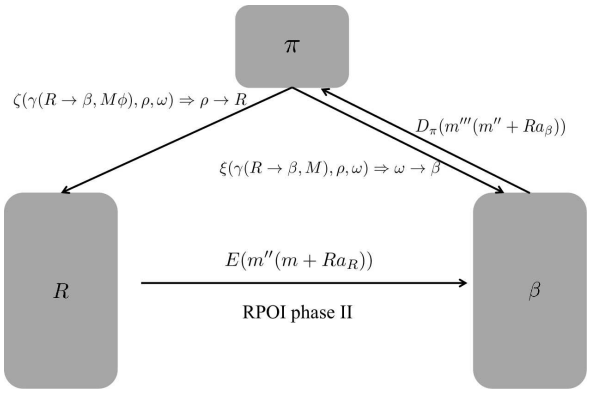

Figure 4: Schematic diagram of the RPOI phase II process

2.7 Block rewards and on-chain information

We stipulate that each node mining burst blocks will receive UtilityNet coins(UNC) as block rewards, in order to incentivize the rapid clustering of computational power, we set the first n height burst block rewards for 100 UNC, n to n_1 for 75 UNC, n_1 to n_2 for 50 UNC, from the height of n_2 it becomes 30 UNC, and follow the regular exponential decreasing mechanism: each height reward decreases $0.01exp(-\beta(H - n_2))$ UNC, as shown in the following figure (where H is the block height and β is the decay factor). This mechanism incentivizes high and rapidly decaying rewards in the early stage, which motivates miners to enter quickly and build up the computational ecosystem; in the later stage, the token rewards are in a steady and slow decreasing process, maintaining a stable ecosystem [15, 22].

Node miners who mine blocks and are rewarded with UNC tokens are required to provide on-chain transaction or data information bookkeeping services. For example, the purchase transaction of chip computation and the POCI process in section 2.4, the two stages of PROI we defined in the previous section will also be recorded on the block as important digital information.

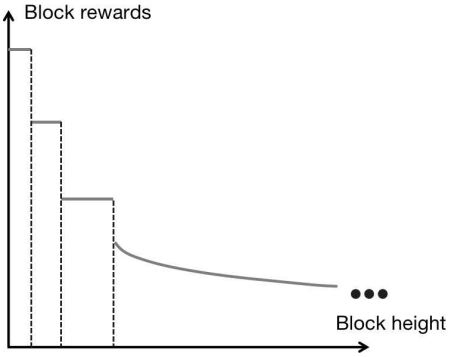

Figure 5: Relationship between block reward and block height

$$Reward\,(H) = \begin{cases} 100 & H \leq n \\ 75 & n < H \leq n_1 \\ 50 & n_1 < H < n_2 \\ 30 & H = n_2 \\ Reward(H-1) - 0.01exp(-\beta(H - n_2)) & H > n_2 \end{cases}$$

Figure 6: Relationship between block reward and block height

3 Blockchain and data structure

Blockchain technology, as a decentralized distributed data storage and management system, is centered on its unique data structure design. The blockchain data structure achieves security, integrity and tamper-evident data through cryptography and distributed consensus algorithms. Merkle Tree (MT) [7] is a common data structure for storing part of blockchain data. MT is a binary tree structure whose leaf nodes store the hash of data and non-leaf nodes store the hash of their children's hash.The main role of MT is to provide data integrity verification and fast data retrieval. When the number of data blocks is large, by verifying the hash value of the root node it is possible to determine whether all data blocks have been tampered with. In addition, due to the uniqueness and irreversibility of the hash value, the hash value of any data block can be found or a data block can be verified quickly by Merkle tree. This structure ensures data integrity and consistency, while improving the efficiency of data validation and query.

In practical applications, blockchains often need to handle large amounts of off-chain data. In order to ensure the security and integrity of off-chain data, blockchain networks can adopt on-chain verification technology for off-chain data. This technique achieves effective verification of off-chain data by storing the hash value or Merkle tree root node [7] of off-chain data on the blockchain. This approach reduces the storage pressure of the blockchain network and ensures the security and integrity of the off-chain data.

Blockchain is closely related to data structure. Through technologies such as Merkle state machine, block composition, and off-chain data on-chain checksum, blockchain technology achieves data security, integrity, and tamper-evident, providing a reliable data foundation for various application scenarios. The following subsections will introduce the relationship between blockchain and data structure in details, including Merkle state machine, block composition, and off-chain data on-chain checksum.

3.1 Merkle state machine

As a key component of blockchain data structure, Merkle State Machine (MSM) enables orderly storage and fast verification of data through Merkle trees. MSM can maintain multiple sets of storage systems to meet the needs of different application scenarios. This section will introduce how to employ MSM to maintain three sets of storage systems: address account system, contract account system and off-chain data, and combine with Bloom filters to achieve efficient data retrieval.

Address account system: the Merkle State Machine maintains the address account system, which enables efficient management of user assets and transaction records. The Merkle State Machine generates a unique hash value for each address account and stores it in the Merkle tree. This approach ensures data integrity and consistency, while improving the efficiency of data validation and query.

Contract account system: the Merkle State Machine can also be used to maintain the account system of a smart contract. By generating a unique hash value for each contract account and storing it in a Merkle tree, efficient management of the state, variables, and function results of smart contracts can be achieved. Also, this approach helps to reduce the pressure of data storage and computation during the execution of smart contracts [4].

Off-chain data: the Merkle State Machine can also be used to maintain a storage system for off-chain data. By storing the hash value or Merkle tree root node of off-chain data on the blockchain, effective verification of off-chain data can be achieved. This approach not only reduces the storage pressure of the blockchain network, but also ensures the security and integrity of the off-chain data.

To achieve fast retrieval of these three storage systems, the Bloom Filter technique can be used. Bloom filter is a probabilistic data structure that can efficiently detect whether an element is in a set or not. By applying Bloom Filter to Merkle State Machine, efficient retrieval of data such as address variables of contracts on the chain, function results, address funds and off-chain operation call records can be achieved. Using the Merkle State Machine combined with Bloom filter technology, efficient management and retrieval of multiple storage systems can be realized, providing powerful data support for the development and innovation of blockchain technology.

3.2 Block Composition

A blockchain consists of a series of interlinked blocks of data. Each data block contains a certain amount of transaction data, a hash of the previous block, a timestamp, and other metadata. This data is encrypted and verified to form a complete block. Nodes in the blockchain network link these blocks in chronological order to form a growing chain data structure, ensuring data security and consistency across the network.

3.2.1 Block-head

The block header contains the key information about the block that must be made public, listed below:

1. Random Number $Nonce$: Nonce is used to ensure prevention of double-spend problem/branch chain fork problem sent by the Proof of Work(PoW) consensus algorithm to adjust the random number so that the block header hash value meets a specific difficulty condition.

2. $Mixhash$: Mixhash constructs are used to ensure verifiable computation of multiple signatures, providing additional security to the blockchain. For example, in Ethernet, Mixhash is used to prevent long-range attacks and ensure the uniqueness of each block.

3. Transaction Hash $TxRoot$: TxRoot is used to ensure that the order and structure of transactions

are unique in the MSM. A Merkle tree root node is generated by hashing the hash values of all transactions and stored in the block header.
4. Other security fields: for example, timestamp (Ts) records the time of block generation and ensures the timing of the block chain. The hash of the previous block ($preH$) is used to link the blocks together to form the blockchain structure.

Thus the new block hash can be expressed as [3]:

$$H(n) = \mathcal{H}(H(n-1), TxRoot, Ts, Nonce) \tag{22}$$

where $\mathcal{H}$ denotes the hash function, n denotes the height of the current block, $H(n)$ denotes the hash value of the current block, $H(n-1)$ denotes the hash value of the previous block, TxRoot denotes the hash value of all the transaction information contained in the current block, Ts denotes the timestamp, and $Nonce$ denotes a random number. It is worth noting that the hash function $\mathcal{H}$ is a one-way function in this formula, where the original data cannot be reversed by the hash value, thus ensuring that the information in the block is not tampered with. At the same time, the introduction of $Nonce$ increases the computational complexity and makes the blockchain network more secure and reliable.

3.2.2 Block-body

The block body contains all the transaction records, and each transaction consists of the following parts:
1. Transaction initiator signature: The transaction initiator digitally signs the transaction to ensure the security and non-repudiation of the transaction.
2. TargetAddress: The address of the recipient of the transaction.
3. Transaction Data TxData: Transaction data contains all smart contract releases, computational operators call stacks, data stack routing information, and custom binary data. These data are used to guide the execution of smart contracts and handle the business logic on the chain.
4. Value: The number of tokens (amount) involved in the transaction.

In summary, the basic block structure contains two parts, the block header and the block body, which carry the transaction records and key information in the blockchain network. This structure ensures the integrity, security and uniqueness of blockchain data and provides a reliable infrastructure for decentralized distributed applications.

3.3 On-chain checksum of off-chain data

In order to solve the scalability problem of blockchain, off-chain data on-chain verification adopts a scaling expansion and consensus mechanism similar to Layer-2 scheme to move some data and computation tasks to off-chain processing, while ensuring the security and integrity of on-chain data. This section will briefly introduce how to implement off-chain data on-chain verification by rollup scheme.

Rollup is a Layer-2 extension scheme, which aggregates the transaction data under the chain and then submits the aggregated data to the chain. Specifically the rollup scheme consists of the following steps:
1. Off-chain data aggregation: In the off-chain environment, individual nodes handle the task of transferring and computing services for data sets. These nodes aggregate the transaction data to form a data structure called rollup.
2. On-chain data submission: The rollup data after aggregation is submitted to Layer-1 networks, such as the Ethereum main chain. These submitted data include meta-data of aggregated transactions (meta-data), such as Merkle tree root node hash, etc.
3. On-chain data verification: Layer-1 network nodes verify the submitted rollup data to ensure the correctness of the data under the chain. This process is mainly achieved by verifying the Merkle tree root node hash value.

The entire rollup off-chain data validation process can also be illustrated as follows:

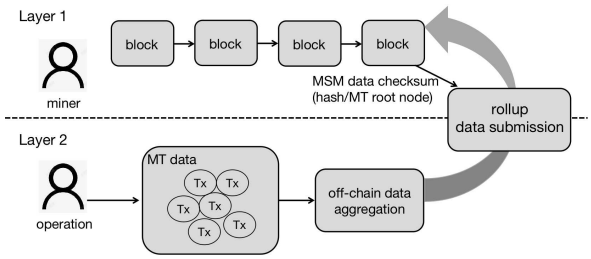

Figure 7: Rollup off-chain data validation process

With the rollup scheme, a large amount of data and computation tasks can also be migrated from on-chain to off-chain processing, reducing the burden on the on-chain network and improving the throughput and scalability of the blockchain. At the same time, the rollup scheme ensures the security and integrity of off-chain data by submitting aggregated data to the Layer-1 network and performing on-chain checksum.

4 Token cycle and computing ecology

The token cycle and computational ecology are key components of the blockchain network system, involving the mining mechanism, token reduction, Gas Fee, Burn Fee, machine time rental orders, and inferred deployment orders. Among them, the mining mechanism is the core component of the blockchain network system, responsible for creating new tokens and allocating them to the participants who provide computing resources to the network. The mining mechanism in the Utility network uses the Proof of Computational Power consensus algorithm, which ties mining rewards to the computational power provided by the participants. The token reduction mechanism is also particularly important in order to maintain the scarcity and value of UNC [15, 20]. The reduction of tokens is used in the form of Burn Fee [16]. To take the advantage of the computational ecology, a decentralized mechanism for leasing computing resources and generating inference deployment orders would allow users to lease idle computing resources for performing complex computational tasks and deploy AI models on computational nodes to accomplish model training and inference tasks, respectively.

In summary, the token cycle and computational ecology thus play a key role in the Utility network, working together through multiple mechanisms to maintain the security, stability, and scalability of the network ecology and token economy. In the following, we will expand on these concepts and their applications in the Utility network, and finally show our model of token economics.

4.1 Mining Mechanism

The mining mechanism of the Utility network uses the Proof Of Computation Integrity(POCI) consensus algorithm, which ties the mining reward to the computing power provided by the participants. In this process, the network-wide computing power is used by all participants to submit computing power to jointly confirm transactions and reach consensus. To ensure the fairness of the mining process, the Utility network uses an on-chain randomness decision mechanism to generate random numbers through a Verifiable Random Function (VRF) [17], which selects the miners among all participants without bias.

The miner needs to complete the block mining task based on the random number generated by VRF and his own computational power. The higher the computational power, the higher the probability that the miner will burst a block in the network will increase linearly. Once a block is successfully mined, the miner needs to submit a proof and broadcast it to show that he/she has completed the block mining task. Other nodes in the network will verify this proof to ensure its validity.

UNC, the native tokens of the Utility network, can be obtained by mining. In the Utility network token issuance plan, 97% of the UNC will be distributed to participants through mining. This mining mechanism incentivizes more participants to join the Utility network and provide computing resources, thus improving the security and stability of the network.

The mining mechanism of the Utility network relies on proof of computational power and on-chain randomness decisions that together ensure fairness and transparency in the mining process. Through this mechanism, UNC can be widely distributed in the network, incentivizing more participants to provide computing resources to the Utility network, thus promoting its security, stability and scalability.

4.2 Token Reduction

The Utility network employs a token reduction mechanism similar to Bitcoin [3] to control the total number of UNC and maintain their scarcity and value. Specifically, every 2 years (or some predetermined block height), the mining reward of the Utility network will be halved. Mathematically, this can be expressed as:

$$q = q_0 0.5^{\frac{t}{2}} \tag{23}$$

where t is the elapsed time (in years), q is the reward at moment t, and q_0 is the initial reward, which is the mining reward per block when the network is first started.

The token reduction mechanism keeps the total number of UNC limited, thus ensuring their long-term scarcity and value. Over time, mining rewards continue to diminish and the number of newly generated tokens gradually decreases. This design helps offset the effects of inflation and keeps the value of tokens growing steadily.

The token reduction mechanism of the Utility network complements the supply and demand in the market. As the scarcity of UNC increases,

the demand for them in the market is likely to rise, thus driving up their value. In addition, token reduction provides an incentive for participants to join at the beginning of the network for higher mining rewards, thus contributing to the growth and development of the network.

As mentioned above, the token reduction mechanism of the Utility network maintains the scarcity and value of UNC by limiting the total number of tokens. This design helps offset the effects of inflation, maintains a steady increase in token value, and motivates more participants to join the network.

4.3 Gas and Burn Fee

In the Utility network, the execution of smart contracts consumes a certain amount of fuel (Gas) to measure the usage of computing resources. The design of Gas is borrowed from the scheme of Ethereum [4, 5], where the computational overhead of contract operators is measured so that the corresponding computing resources are allocated for the execution of smart contracts.

Each operator has a predefined Gas consumption value, which depends on the computing resources required for its execution. When a user submits a smart contract transaction, there needs to be enough Gas cost attached to the transaction to ensure smooth execution of the contract. The execution virtual machines (VMs) in the Utility network keep track of the Gas consumption while running the contract to ensure that it does not exceed the limit provided by the user. Once the Gas consumption exceeds the limit, contract execution will be aborted to avoid malicious behavior such as infinite loops.

In addition, the Utility network introduces a Burn Fee mechanism, whereby a portion of each transaction and message transmission is burned and permanently removed from on-chain circulation. This design will also help reduce the inflation rate of tokens, increase the scarcity of tokens, and maintain a stable growth of token value. the exact value of Burn Fee can be adjusted according to the actual operation of the network to ensure the economic health of the network.

In summary, the Gas and Burn Fee mechanisms play an important role in the Utility network; Gas is used to measure the computing resources consumption of contract execution and ensure fair and efficient computational resource allocation in the network, while Burn Fee assists to reduce inflation and increase the scarcity of tokens, thus maintaining their value. Together, these two mechanisms above provide a sustainable economic foundation for the Utility network.

4.4 Machine Time Rental Orders

The machine-hour rental order in the Utility network allows users to rent computing resources on the chain. This process can be described in mathematical notation as follows:
Suppose $\mathcal{O}$ represent a computation rental order, which includes the following main elements:
u: User
t: Rental time in seconds
r: Computing resource requirements (e.g. number of CPU cores, memory size, etc.)
p: User bid (in UNC)
e: Computing environment (e.g., VM type, operating system, etc.)
s: Service provider
f: Scoring transactions
v: Validation result set
The process of sending an order to the chain can be expressed as follows:

$$\mathcal{O}(u, t, r, p, e, s, f, v) \rightarrow UtilityNet \tag{24}$$

which is the detailed expression of request γ at section 2.4.

Once the order is sent to the chain, the computing resources providers in the network will decide whether to accept the order based on the user's demand and bids. After accepting the order, the service provider needs to allocate the resources according to the user's computing environment requirements, completing the encrypted file signature and providing the service during the rental period.

After the service is completed, users and service providers engage in a rating transaction. The rating transaction f includes information on service quality and user experience, which helps build a fair and transparent rental market. Both parties can ensure the correctness and integrity of the calculation results of the rental process by validating the result set v. The validation result set contains information such as calculation outputs, execution logs, etc., so that allow users to evaluate the quality of the leasing service and the deployment of funds to users and service providers.

To summarize, the machine time rental order achieves efficient allocation and utilization of

computing resources in the Utility network, providing users with flexible and reliable computing services. Through on-chain transactions, scoring mechanisms and verified result sets, the Utility network ensures fairness, transparency and security in the decentralized and leasing market.

4.5 AI inference deployment orders

The TPU-based deployment of the Utility network possess the mature AI model training and inference environment, and therefore also it can be introduced to the inference deployment order service. The inference deployment order allows users to use TPU resources on the chain to perform daily inference algorithms on the AI models being trained.
Suppose $\mathcal{R}$ represent an inference rental order, which includes the following main elements:
u: User
t: Rental time in seconds
r: Computing resource requirements (e.g. number of CPU cores, memory size, etc.)
m: The trained algorithm model
d: trained data corpus
p: user bid (in UNC)
s: Service provider
f: Scoring transactions
v: Validation result set
The process of sending an order to the chain can be expressed as follows:

$$\mathcal{R}(u, t, r, m, d, p, s, f, v) \rightarrow UilityNet \qquad (25)$$

Similarly, once an order is sent to the chain, the computing resources providers in the network will decide whether to accept the order based on the user's requirements and bids. After accepting the order, the service provider needs to allocate resources according to the user's model, data and environment requirements, complete the encrypted file signing, and provide specific AI training inference services during the rental period.

After the user receives the inference results, the system will verify the results to ensure the accuracy and security of the results. The validation process includes digital signature verification of results, data integrity verification, algorithm interpretability verification, etc. Only the results that pass the validation will be considered as valid inference results.

Once the inference deployment order is completed, like the machine time rental order, the user and the service provider for scoring transactions, verification, the system will automatically carry out the settlement, and the cost will be allocated to the nodes involved in the inference deployment and deposit returned to the user. From request to liquidation, all the processes are automatically executed by the blockchain smart contract order, and the security and fair performance of the fees are guaranteed to the maximum.

4.6 UT Token Economics Model

The Utility network token has a complete and mature economic system model. According to the plan, th model is separated into four Age periods: Talos Age, Vajra Age, Golem Age, and Maria Age. These are described in detail below.

4.6.1 Talos Age

Until the release of this whitepaper, the Utility network environment is in the first Talos Age, during which UtilityNet coins(UNT) are issued, the Public Chain Foundation is established, and the community and public chain are in the development phase. During this phase, the first internal test will be launched, and the incentive model will be tested based on BEP20 (Binance Smart Chain, BSC). Participants transact through a decentralized network and then update the network state by writing and processing messages in their blocks. Running transfers of data and messages consumes computing and storage resources on the network. Accordingly, the GAS fee consumed is BNB. The internal testing phase also requires a certain amount of UNC to be filled as fuel (Fuel) to be burned for mining. The planned fuel will be burned in 200 days, and 5% of the total amount burned each day will go into the project foundation wallet as operation and technology development funds. The production of UNC coins mined will be released directly, without locked positions. The production will be based on every 200 days as a mining cycle, with each cycle reducing production by 5%, and it is expected to be able to produce 9.7 billion pieces in 90 cycles. The initial cycle yield is set at 500 million coins, with 2.5 million coins produced per day. The details are shown in the following table:

	1st period	2nd period	3rd period	4th period	
total release	500 million	475 million	451.25 million	428.6865 million	.....
daily release	2.5 million	2.375 million	2.25625 million	2.143438 million	.....
reduce 5% per cycle period					

Table 2: UNC releases versus cycles

Based on this output rule, Utility expects a minting limit of 10 billion UNC, which is the token for Utility network. Finally UNC will be burned at 95% of the total amount, with the total amount deflated to 500 million. UNC will be allocated to UtilityNet miners as mining rewards for providing AI computation, blockchain maintenance, data distribution, running contracts, and other services. These rewards will support multiple types of mining in the future.

4.6.2 Vajra Age

In the second epoch, the Vajra Age, the test network UT Testnet will be deployed online and an early incentive program will be announced. Computation miners are the only group of miners that will be rewarded when the network goes online, and it is the earliest group of miners that is also responsible for maintaining the core functions of the protocol. The purpose of the incentive program is to reward these early miners to achieve a rapid build-up of computational power.

4.6.3 Golem Age

Significant events in the third Golem Age includes the deployment of the UT mainnet to be pushed online, the mapping of testnet UNC token to the mainnet UNC token and it is the unfolding of the era of decentralized computation for artificial intelligence. A complete UNT ecological network is officially established.

4.6.4 Maria Age

In the Maria Age, the fourth epoch, the largest man-made parametric models are trained and run in the UNT network. A powerful UNT ecological network continues to grow and enlarge the scale of AI training, building models with trillions of parameters and super trillions of computational networks.

5 Execute the virtual machine

This chapter focuses on the implementation and integration of the Utility Execution Virtual Machine (VM). The execution VM is a key component of the entire Utility network, responsible for handling various computational tasks and executing smart contracts. This chapter will focus in detail on the following areas:

1. Virtualization and Isolated Execution Environments: i.e., discusses how virtualization technologies can be used to create secure, isolated execution environments to ensure that computing tasks are performed in a controlled and secure sandbox.

2. Environment proof: Prove how to generate and verify the execution environment to ensure the correctness and reliability of the computational tasks.

3. Solidity VM: Introduce the role of Solidity VM in the Utility network and how to integrate with other components of Utility network.

4. Kubernetes Integration: Explore how to integrate the Utility network VM with Kubernetes to maximize efficient management and scheduling of distributed computing tasks.

5. WASM integration: WebAssembly(WASM) technology to improve the performance and compatibility of the Utility network VM possible.

6. PyTorch Outreach Computing Support: Integrates the Ute virtual machine with PyTorch to support a wider range of artificial intelligence and machine learning applications.

The next section will delve into the above key technologies and explain how they can be organically combined with the Utility network to form an efficient, secure and scalable computing ecosystem.

5.1 Virtualization and isolation of the execution environment

Virtualization and isolated execution environments are critical to guarantee the security and reliability of computing tasks in the Utility network. Utility uses technologies based on Arm framework drivers and memory virtualization support, combined with memory virtualization on TPU chips, to achieve efficient resources management and scheduling. With the Arm framework driver, Utility network can fully utilize the underlying hardware resources to achieve high performance processing of computing tasks. At the same time, the memory virtualization technology enables the TPU chip to flexibly divide and share memory resources among different computational tasks to improve the overall computational efficiency [27]. The framework is shown in the following figure:

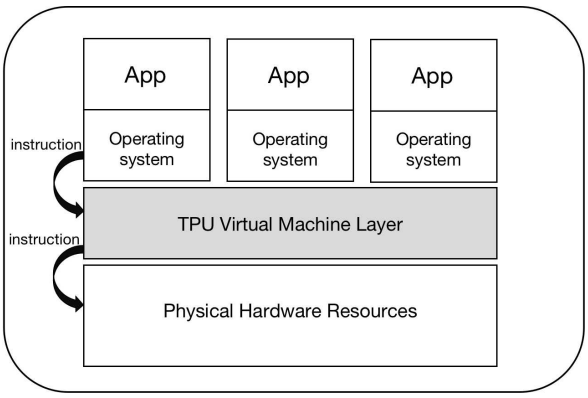

Figure 8: TPU Virtualization Architecture

In the Utility network, each isolated execution environment is located within a container that contains both standard images and data algorithm mounts. The standard images provide the basic runtime environment and dependency libraries for the execution environment, while the data algorithm mounts contain the specific algorithms and data used to process the computational tasks. By combining the execution environment with container technology, Utility network is able to ensure that computational tasks are performed in a controlled and secured sandbox environment, providing effective intervention against potential security risks and data breaches.

5.2 Environmental certification

The Utility network uses a decentralized environment proof mechanism to ensure the correctness of the execution environment. The mechanism is implemented through a consensus algorithm and a block-bursting process that enforces that miners have standard environmental execution and the corresponding computational power in each consensus round.

In the Utility network, the generation of new blocks requires miners to complete a series of tasks in a short period of time (e.g., a few seconds). These tasks include running random code in a standard execution environment to generate PyTorch operator. In this process, miners need to ensure that their execution environment meets the requirements of the network, while having enough computational power to train and reason about the tasks.

When a miner completes a task and is ready to submit a new block, other miners need to validate their results. This validation process involves checking that the PyTorch operator generated by the pre-blown block party is correct or not. If the majority of miners agree that the operator is correct, the new block is approved and added to the blockchain. Conversely, if the operator fails to pass validation, the exploded block is considered invalid and the consensus process proceeds to the next round.

This environment proof mechanism effectively deters potential evildoers. This is because in the Utility network, the evildoer is unable to initiate a compliant execution environment to complete the task in a very short period of time, and thus misses the opportunity to burst the block. In this way, the Utility network ensures the correctness of the execution environment, while providing a reliable computational basis for training and inference tasks.

5.3 Solidity VM

The Utility network supports smart contracts coding using the Solidity language and executes them in its own VMs. To enable efficient connectivity with hardware, operations and data, a range of new operations and structures can be allowed to be introduced in the Solidity VM[28], including underlying data types. These extensions can help build applications that are more tightly integrated with the underlying hardware.

By employing the rollup technology described in Section 3, the Utility network enables efficient transmission and verification of on-chain and off-chain data. Under this architecture, the layer 1 network is responsible for aggregating transactions and ensuring token support on both sides of the execution, while the layer 2 network is responsible for the actual computing and data processing.

The decentralized nature of the Utility network ensures distributed management of the computational state, making the algorithm's computing process tightly bound to the miner proper. By providing a standard suite of services, the Utility network ensures a best-in-class service experience for all participants, from large data centers to single-unit miners. This design makes the Utility network highly scalable and powerful for computation, providing a solid foundation for future distributed computing applications.

5.4 Kubernetes Integration

Utility Network's Volcano Engine integrates with common Kubernetes to jointly build a verifiable environment and service infrastructure on the miner

side. The Volcano Engine, a highly optimized computing engine, works seamlessly with Kubernetes to enable powerful computational capabilities and high scalability.

By integrating API Gateway, Scheduler, Tensor Processing Unit (TPU), and fast cache virtualization technologies, Utility builds isolated sandbox environments on the Kubernetes platform. These isolated sandboxes are implemented through container technology and standard images, enabling Utility to support mega-scale parallel computing tasks such as training and inference deployment of Transformer models.

In the Utility network, Kubernetes handles container-based application deployment, scaling, and management. This enables the Utility network to leverage the benefits of the Kubernetes ecosystem [29], including flexible resource management, high scalability, and support for a variety of hardware and software platforms. The Kubernetes integration also provides a range of advanced tools and services, including Autoscaling, service Discovery, and Fault Recovery to ensure high availability and performance in distributed computing scenarios. Utility also integrates virtualization technology with Kubernetes to automate the scheduling and management of computing tasks, while Kubernetes dynamically allocates and adjusts the memory resources of TPU chips based on task demand and resource status to ensure efficient and stable operation of computing tasks in the Utility network.

As one can see, the integration of Utility network and Kubernetes provides a powerful foundation for building distributed computing environments. Through this integration, Utility can provide highly scalable, high-performance, secure and reliable computing resources for various computing tasks to meet the needs of future artificial intelligence and big data applications.

5.5 WASM Integration

WebAssembly (WASM) integration in the Utility network is key to enabling cross-platform algorithmic code to run, and WASM is a common binary instruction set designed to provide an efficient and compact VM format for a variety of computing devices [30]. Through WASM integration, the Utility Network is able to support a variety of different hardware and software environments to ensure cross-platform compatibility and associated performance.

WASM is tightly integrated with the algorithm code behind rollup operation in the Utility network, where the algorithm code is first compiled into WASM bytecode, and then packaged and optimized by the rollup technique. In this way, the WASM bytecode can be executed quickly on each node in the Utility network, improving the overall computation efficiency.

In addition, the Utility network works seamlessly with Kubernetes' WASM execution environment. kubernetes can support multiple languages and platforms using the WASM Runtime, enabling algorithmic code in the Utility network to run on different computing devices. This seamless integration facilitates reduced resource consumption and improved computational efficiency while ensuring code security and reliability.

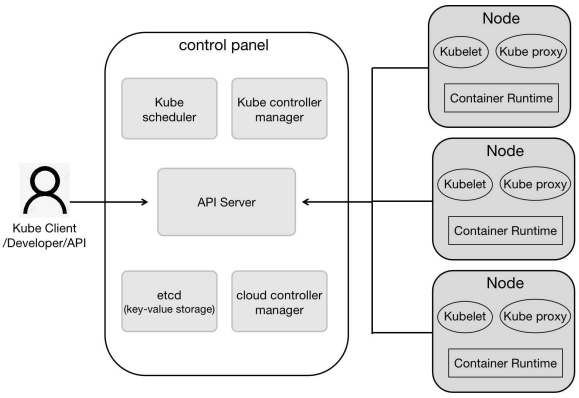

Figure 9: Integration of Kubernetes framework with WASM environment

With WASM integration, the Utility network enables a highly flexible and scalable computing ecosystem. Whether handling complex AI model training tasks or performing lightweight inference deployments, the Utility network is able to provide powerful computational capabilities in a variety of hardware and software environments.

5.6 PyTorch Outreach Computing Support

The PyTorch outreach computing support of the Utility network solves the problem that Solidity contracts cannot directly embed PyTorch code based on Python. And the Utility network allows efficient collaborative computing between on-chain contracts and off-chain PyTorch code through rollup technology [31], thus enabling the demonstration of service capabilities for AI.

In the Utility network, training data and function inputs are transmitted to the miner side via

rollup technology, combined with Layer1 contractual conventions. This structure allows miners to perform extended computations or AI model training and inference in an off-chain environment. After the computation is completed, the results are transmitted back to the chain via rollup for interaction and validation with the on-chain contract.

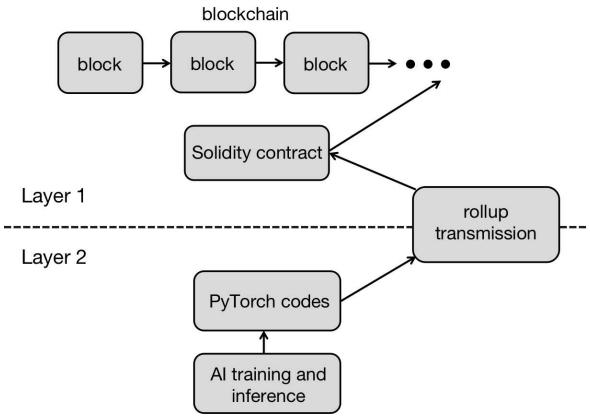

Figure 10: PyTorch outreach computing support

By introducing PyTorch outreach computing support, the Utility network enables a seamless connection between on-chain and off-chain. This hybrid computing model enables the Utility Network to leverage existing AI frameworks, such as PyTorch, to further improve computational efficiency and performance. In addition, this approach brings greater scope for innovation in AI, supporting researchers and developers to develop more complex and powerful AI applications in the Utility network.

6 Artificial intelligence(AI) computation network based on the Utility blockchain

The operation model of the Utility network allows the emergence of AI computation service providers. AI computation service providers build their own AI computation centers by leasing IDC server rooms, and through the Ute blockchain can fairly authenticate the type of AI training and inference chips, their computational size, whether they are online, and other information on the chain, thus confirming their total computational power on the Utility network.

However, AI computation is not equal to AI computational services. What a computation order buyer needs for AI model training and deployment is AI computation service with comparable service quality. The online AI computation in the chain is only one of the production factors of AI computation service. The computational service provider also needs to purchase CPU servers, hard disk, rent network bandwidth, maintain stable power supply of AI servers, and other production factors to provide AI computational service.

How to measure and guarantee the quality of service of AI computational services? An efficient solution to this problem will determine whether the Utility blockchain is productive and whether Utility coins can form a closed loop of circulation. We have designed an under-chain AI computational network solution to efficiently connect AI computation centers around the computational service provider to achieve efficient sharing and circulation of data, computation, algorithms, models, services and other elements. And with the AI computational service quality scoring mechanism based on Utility blockchain, the chain can trace and publicize the transactions between buyers and sellers of computation services. The combination of on-chain and off-chain provides efficient productivity in the AI era. The combination of on-chain and off-chain can maximize the incentive for AI computational service providers to join the Utility computing network, reducing the cost of using computation, promote the unified settlement of heterogeneous computation in different places, make the production, distribution, circulation and consumption of AI computation smoother, and improve the efficiency of market operation. The goal is to build a worldwide AI computational infrastructure with easy access to computing resources, efficient and unified scheduling of tasks, convenient and inexpensive use by users, and a sustainable operation model and mechanism.

The platform layer of Utility AI computational network is designed with four central subplatforms based on Utility blockchain, namely, the AI computational network task unified scheduling and management platform, the AI computational network data unified storage and management platform, the one-stop platform for debugging and training deployment of AI computing centers, and the AI computing network data market.

6.1 AI computational network task unified scheduling and management platform

Artificial intelligence network task unified scheduling and management platform schedules and manages different types of online resources from different wise computing centers effectively to maximize the utilization and efficiency of the entire wise computing network.

6.1.1 Functions

Collecting resource information: computing resources include CPU, GPU, TPU, NPU, GPGPU, FPGA, ASIC, etc., network resources include supercomputing intranet with dedicated Internet, public network, controlled and managed supercomputing intranet, controlled and managed AI computing intranet, etc., data resources include various public datasets of AI, private datasets of users, etc. Each resource on the network has its unique advantages and applicable Each resource on the network has its unique advantages and application scenarios.

Integration of resources: According to the minimum granularity of each heterogeneous resource is not the same, to conveniently use AI, it often needs to combine a single computational resource and with different kinds of heterogeneous resources to form a resource package, the resource package will be allocated to the most appropriate task requirements to optimize the task completion efficiency and resource utilization.

Managing jobs: Each AI job often contains multiple nested tasks, which can be called sub-task, and each sub-task can be scheduled to different wise computing centers in a distributed manner, and successful job scheduling will constitute the network of sub-task required for the job. In due course, it is necessary to monitor and manage the operation status and performance of the jobs and sub-task, to detect problems and deal with them in a timely manner.

Scheduling jobs: The platform schedules the jobs submitted by users, and schedules each subtask included in the job to the appropriate AI computing center based on the load of each AI computing center, data location, computing power price factor, communication efficiency, job resource demand and other factors (referred to as "scheduling factors", which will be outlined below) to run, to maximize the efficiency of task completion and resource utilization.

Scheduling strategies: A number of scheduling factors are used as inputs, and candidate wise computing centers that meet the job resource requirements are processed using certain processing logic to output a normalized score for each wise computing center. The output of each scheduling strategy will be used as the input of the scheduling evaluation model for the next step of comprehensive decision making. The scheduling policies are determined according to the actual scenario requirements, and the optional scheduling policies are load minimum priority, resource idle priority, data affinity, computational power lowest price priority, computational power highest performance priority, network performance priority, etc.

Scheduling factors: Scheduling factors are selected according to the actual scenario requirements, and custom scheduling factors are also available. Optional scheduling factors include: job requirements, resource packet demand specifications, data location, computational price, computational performance, wise computing center load, etc.

Scheduling evaluation model: The output of multiple scheduling policies will be used as the input of the evaluation model, and the optimal wise computing center will be calculated by the evaluation model to obtain the final scheduling results. The following evaluation model is used: for a job waiting to be scheduled, scheduling policies S_1, S_2,....., S_n are used, respectively, and assign weights W_1, W_2, ..., W_n to these scheduling policies. At the kth wise computing center C_k, each scheduling policy outputs a score of G_{k_1}, G_{k_2}, ... , G_{k_n}, then the final score of the wise computing center C_k is G_k:

$$G_k = \sum_{i=1}^{n} G_{k_i} \times W_i \tag{26}$$

From all wise computing centers C_1, C_2,...., C_n, the highest rated one is selected as the final scheduling result, and the task is dispatched to the target wise computing center.

Data scheduling factor: The platform can obtain the Id, name, version, data source distribution location, data cache distribution location, access rights and other related information of the target data set through the query interface of the AI data unified storage and management platform (see section 6.2 for details). It can also query the interface whether the wise computing center allows downloading, average downloading time, maxi-

mum downloading concurrency, maximum downloading bits, recent downloading failure rate and other information through the artificial intelligence data unified storage and management platform. The platform uses the above target data set and the information about the target AI computing center as one of the scheduling factors.

Data Scheduling Policy: The scheduling engine decides whether to migrate datasets across the AI computing center based on the output of the data scheduling policy. The platform has the following two data scheduling policies.

Calculate with the number of moving policy: the existence of the target data set of intelligent computing center can be exported as the target job can be scheduled to the candidate intelligent computing center. When the target wise computing center is selected by the job, the optimal data source is selected based on the multidimensional data scheduling factor of the candidate data source, then the target data set is migrated to the target wise computing center and cached, and the target job is scheduled after the data migration task is completed.

6.1.2 Architecture

The core of the unified scheduling and management platform for artificial intelligence tasks is the inter-cloud management and scheduling system, which is responsible for assigning user tasks to different heterogeneous intelligent computing centers.

When designing the platform architecture, we adopted the design idea of storage and computation separation, splitting the inter-cloud resource management into two major components: computation resource management and data management, in order to achieve a high degree of scalability and flexibility.

In this architecture, the inter-cloud management and scheduling system is divided into four main modules: job management module, scheduling engine module, heterogeneous computing power resource management module, and data management module. The main functions of each module are as follows:

Job management module: it is responsible for completing the functions of adding, deleting, changing and checking job objects, log management, running load monitoring, etc.

Scheduling engine module: it is based on the comprehensive scoring of internal scheduling policy and scheduling evaluation program, the computing jobs are distributed to the target AI computing center to maximize the task completion efficiency and resource utilization.

Heterogeneous computing power resource management module: it is responsible for unified access and management of heterogeneous computing, network and data resources in the AI computing center, periodically collecting resource information, classifying, storing and managing them for use in subsequent job scheduling.

Data management module: it gives the Real-time sensing of the target data resources of the intelligent computing network, when the target intelligent computing center lacks the target data set required for AI operations, the data management module can sense the data source with the best network location and provide the scheduling engine with a data migration strategy to realize the effect of scheduling with the number of calculations and calculations.

6.2 AI computational network data unified storage and management platform

AI computing network needs to schedule a large number of tasks to meet the conditions of computing resources of the AI computing center, different computing tasks rely on a variety of data sets, these data sets may be distributed in one or more AI computing center. While each AI computing center may use different storage systems, there is a large variability between storage architecture and external interface. Other problems like the complexity of use by users, the overhead costs of such as bandwidth, latency time, and copy storage space will be paid for multiple data set migrations between clouds, also exist. Therefore, the Wise Computing Network needs to build a unified storage and management platform for inter-cloud data, to collaboratively manage the heterogeneous storage media and data sets of the Wise Computing Center, to improve the efficiency of resource sharing among the participants of the Wise Computing Network ecosystem, to give the Wise Computing Network higher scalability, and to provide more convenience for more Wise Computing Centers and data to join the Wise Computing Network ecosystem in the future.

6.2.1 Functions

Fast access to heterogeneous storage systems: The storage system of each AI computing center may be heterogeneous, using inconsistent data storage so-

lutions. The application interface protocols may be S3, OBS, OSS, MINIO, FTP, and various custom storage system APIs. To achieve efficient management of network data, it is necessary to adapt and unify the interfaces of the heterogeneous storage systems of each AI computing center to form a set of The internal unified storage management interface.

System high availability: the access to the wise computing center is attributed to different computing power service providers, and has geographical isolation and management isolation. Access to the heterogeneous storage system can not be considered completely reliable and always online. AI computing center suddenly offline its storage system should not affect the normal other AI computing center to continue service.

Sense the dataset: To improve the efficiency of computing tasks, realize the scheduling strategy of calculating with the number. The intelligent computing network needs to generate a network-wide unique id for the dataset and sense the storage of the dataset in each intelligent computing center to form a global data storage view. Based on this global view, the AI task scheduling and management platform can query whether the target data exists in the target AI computing center. In the case that multiple computing centers meet the computing power demand of a computing task at the same time, the user computing task can be dispatched to the computing center where the dataset already exists in order to improve the task activation efficiency.

Verifying dataset integrity: Datasets in the AI domain may contain a large number of folders and multiple files, and AI training tasks are strongly dependent on the dataset, and inconsistencies in the dataset will affect the effectiveness of AI model generation, so quickly verifying the dataset integrity will be a challenge for the platform.

Support for multiple versions of datasets: datasets in the field of artificial intelligence hold a variety of annotated data. As society develops and commercial companies invest in development, similar datasets tend to keep adding new content, in which case new versions of different datasets are needed to distinguish and archive the contents of datasets from different periods.

Inter-cloud migration of datasets: The computing service provider may deploy the wise computing center as a private cloud or may deploy the wise computing center to a public cloud. When a computational task will be scheduled to the wise computing center N, the target dataset needs to be migrated between clouds when the target dataset specified by the task is not available in that center. The platform senses the existence of this target dataset in the wise computing center M and can migrate it to the storage system of the target wise computing center N from the wise computing center M as the data source.

Data set compression: Most of the data sets in the field of artificial intelligence are unstructured data, which are compressed and then stored in the AI computing center, which will enable the AI computing network to save a lot of transmission bandwidth, transmission time and storage space.

Data set caching: Migration of data sets between clouds requires four steps: sensing the data source, downloading from the data source to an intermediate proxy server, and uploading the data set to the target wise computing center by the proxy server. Each migration requires uploading and downloading through the public network, with huge bandwidth and time costs. Building a memory cache network for dataset in the AI Computing Network will greatly improve the efficiency of dataset transfer, greatly enhance the efficiency of AI task scheduling, and greatly improve the experience of using the entire AI computing network.

Support multi-tenancy: Wise Computing Network will serve multiple users and multiple organizations. The dataset has privacy attributes, requiring the platform to have a complete set of user role permission management system, supporting users to manage access rights to the dataset for private, limited domain public, and network-wide public.

External unified API: The AI data unified storage and management platform and the AI task unified scheduling and management platform adopt the storage and calculation separation architecture, and the AI data unified storage and management platform supports multi-tenancy, which requires the development of a set of external unified API interfaces to provide multi-party calls for dataset awareness and migration.

6.2.2 Data structure

Before storing the dataset in this system, we need to compress and format the original data to generate two parts: DataSet Metadata Part and Dataset Blobs Part. Among them, the metadata part is responsible for storing the description information

and folder hierarchy of the original dataset files, and the original data block part is a chunk of all the files in the dataset. Each data chunk in the original data block part generates a hash digest, and the digest algorithm is designed to be variable and configurable (optional Sha256, Blake3, etc.) to adapt to the increasingly powerful computational cracking attacks and higher performance hash algorithm innovations. Based on the folder hierarchy of the original dataset, a hash summary Merkle tree data structure is formed in the metadata section and a unique id of the dataset is generated. Due to the presence of the original data block section, some or all of the data blocks are selected randomly within a single dataset with the same flexibility and the dataset integrity is verified using the corresponding summary algorithm.

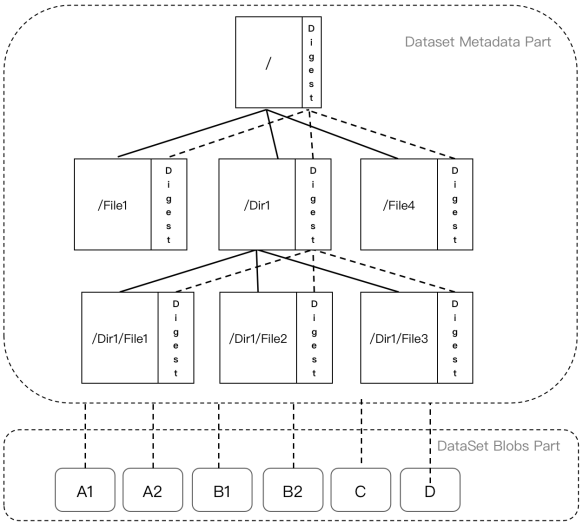

Figure 11: Structure of AI computing network data set

Multi-version dataset structure: The shared data block part then separates the original files of the dataset by block and stitches them together, and is responsible for storing the actual byte contents of the current version of the dataset. Such a separated data structure can improve system storage efficiency by storing high-frequency accessed metadata in memory and high-speed storage devices, and store shared data block files, which may occupy huge storage space, in less costly persistent storage.

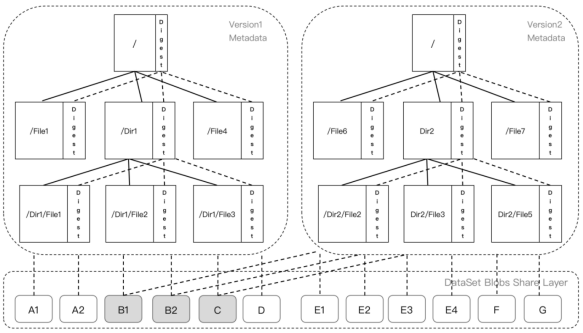

Figure 12: Structure of multi-version data set of AI Computing Network

6.2.3 Architecture

The platform is a unified data storage and management system that supports fast access to heterogeneous wise computing centers, aiming to provide global unified storage management and efficient data scheduling solutions for wise computing networks. The platform adopts micro-service architecture and consists of four parts: system management module, dataset management module, unified agent service module, and unified storage application interface module.

6.3 One-stop platform for debugging and training deployment of AI computing centers

One-stop platform for debugging and training deployment of AI computing centers faces to single-center computing power users and undertakes computing power orders from user. It provides resource management and usage functions for data, algorithms, mirrors, models and computing power to facilitate the one-stop construction of AI task computing environment for the users of AI Computing Center. At the same time, it provides resource management, computing task management and monitoring in a single center to facilitate the operation and analysis of a single center by managers.

6.3.1 Characteristics

One-stop development: Provide users with one-stop functions of debugging, training and deployment for AI computing scenarios, and open up the whole chain of AI computing through modules of data management, model development and model training. Its management is convenient and it provides platform managers with a one-stop resource management platform, which greatly reduces the management costs of platform managers through

visualization tools such as resource allocation, monitoring, and authority control.

Superior performance: Provide high performance distributed computing experience, ensure smooth operation of each environment through multi-faceted optimization, and further improve model training efficiency through resource scheduling optimization and distributed computing optimization.

Good compatibility: the platform supports heterogeneous hardware and heterogeneous networks, such as GPU, NPU, FPGA, IB network, Ethernet, etc., to meet the needs of different hardware cluster deployment. It also support a variety of deep learning frameworks, such as TensorFlow, Pytorch, PaddlePaddle, etc., and can also support the addition of new frameworks by way of custom images.

6.3.2 User functions

Image panel: Container images are used to package applications and their dependent operating system environments, providing an environment for algorithms to run. Mirror management module includes user mirror management and platform administrator preset mirror management. The main functions of this module include: get mirror list, create/query/update/delete mirrors, get mirror list version list, create/query/update/delete mirror versions, share mirror versions to Wise Computing Network data mart, cancel sharing.

Algorithm panel: Users can upload their code files to the platform and then create an AI debugging environment to modify the code. After the algorithm code is successfully debugged and saved, the corresponding AI training environment can be launched with one click to output model results and training process logs. The main functions of this module include: get algorithm list, create/query/update/delete algorithm, get algorithm version list, create/query/update/delete/download algorithm version, share algorithm version to Wise Computing Network Data Bazaar, and cancel sharing.

Data Panel: The datasets in Single Wise Computing Center can be categorized into My Datasets, Public Datasets and Preset Datasets. My dataset is the user's own dataset, public dataset is the dataset shared by the user, and pre-set dataset is the dataset uploaded by the administrator. All the datasets in the single center can be used as the data source of Wise Computing Network to access the unified storage and management platform of Wise Computing Network data. The main functions of this module include: get dataset list, create/query/update/delete/dataset, get dataset version list, create/query/update/delete/download dataset version, share dataset version to Wise Computing Network Data Bazaar, and delete sharing.

Debugging jobs: The platform provides an online programming environment for debugging, running and saving algorithms to support the subsequent creation of training jobs. The debugging module supports various online programming environments (e.g. JupterLab, VsCode) and provides management functions to create, open, start, save, stop and delete debugging jobs.

Training jobs: This module supports users to create AI distributed training jobs and non-distributed training jobs; and is compatible with multiple deep learning computing frameworks (such as TensorFlow, PaddlePaddle, MindSpore, PyTorch, etc.), select the corresponding algorithm and OS image to start AI training jobs; supports "training-model". One-stop development, automatically save the model after training, easy to deploy and call.

Model Panel: Models are the results generated by algorithm training, and through management can help users archive training results and provide model data for subsequent model validation, deployment and evaluation. The main functions of this module include: get model list, create/update/delete models, get model version list, create/query/update/delete/download model versions, share models to Wise Computing Network data mart, and cancel sharing.

Inference Service: This module supports one-click deployment of models as online inference service, and can be configured to generate service external API documents, providing an inference service platform for users to carry out large model business. one can use the module to query the service name, model name, model version, service description, creation time, running time, and status. The inference service adopts elastic scaling technology architecture, when no one visits at night, the number of copies of the inference service is reduced to 0 at the minimum, which reduces resource occupation and saves the cost of users, making the platform inference computational card resources to be shared and utilized most efficiently.

6.3.3 Management functions

Multi-tenant management: it support multi-tenant resource isolation, to meet the different resource needs of different teams, divide different workspaces and match the corresponding training resources, making it more convenient for Team Leader to manage training resources.

Platorm monitor: Objects monitored by the platform are wise computing center cluster monitoring and training task monitoring.Prometheus aggregates monitoring data provided by each node's NodeExporter into its time-series database. The Grafana web service regularly requests metrics from Prometheus, dynamically displaying graphs of metrics data for the cluster and for the tasks the user is running.

Resource management: Platform resource management objects can be divided into server node list, system resource objects, custom resource objects, resource package objects, and resource pool objects.

Machine time management: The platform uses machine time as the unit for billing. When users create an AI debugging environment or training environment, the corresponding machine time will be deducted. The platform sets the unit price of n Utility coins/machine hour according to the price of the resource package. The billing rules for AI jobs are listed as follows:

1. Machine time = Sub-task 1 machine time + Subtask 2 machine time + ... + sub-task n machine hours

2. Sub-task machine time = copy 1 machine time + copy 2 machine time + ... + copy n machine hours

3. Task copy machine time = resource package usage weight * (task copy run end time - task copy run start time)

Preset data management: Preset data set is a data set uploaded by the administrator of the wise computing center, generally a public data set, providing the initial data source for users within a single wise computing center and the wise computing network.

Pre-set algorithm management: Pre-set algorithm is the algorithm code created and uploaded by the administrator of AI computing center, and users in AI Computing Center can query, use, copy, download, and migrate the pre-set algorithm.

Pre-set image management: Pre-set images are container images created by the administrator of the Smart Computing Center, and users within the AI Computing Center can query, use, copy, download, and migrate pre-set images.

Debugging job management: The administrator of the AI Computing Center can use this module to operate and maintain the debugging jobs of users in the AI Computing Center and assist them in troubleshooting debugging job problems. It can query individual job details, view job monitoring information of each dimension, view job logs and force stop operations. Due to the platform storage and calculation separation architecture, the platform administrator cannot view the original input and output data such as user datasets, algorithms and models.

Training job management: The administrator of the AI Computing Center can use this module to operate and maintain the training jobs of users in the AI Computing Center and assist them in troubleshooting training job problems. It can query individual job details, view job monitoring information in each dimension, view job logs and force stop operations. Due to the platform storage and calculation separation architecture, the platform administrator cannot view the original input and output data such as user datasets, algorithms and models.

Inference service management: The administrator of the AI computing center can operate and maintain all the inference services in the AI Computing Center through this module, and the operation and maintenance functions include: querying the service list, viewing the service details and forcing to stop.

6.4 AI computing network data market

The data in the field of artificial intelligence includes algorithm code, container images, annotated datasets, various videos, NLP(Natural Language Processing) natural language large models, etc. The main function of the smart computing network data market is to provide a platform for data trading between buyers and sellers of AI computing services.

6.4.1 Functions

Data publishing: Data providers can publish their data on the AI computing network data market, describing, labeling and pricing the data.

Data purchase: Data buyers can purchase data displayed on the Wise Computing Network data mart. Before purchase, you can ask for quotations and bargains, and after the transaction is confirmed

and settled, you can get access to the data and use and migrate the data on the AI computing network.

Data security guarantee: the AI computing network data market uses data fingerprinting, data encryption, data backup and other measures to guarantee the security and integrity of data.

Data Transaction Supervision: The AI computing network data market uses smart contract technology to supervise and verify data transactions to ensure fair, transparent and traceable transactions.

Data sharing and circulation: The AI computing network data market also supports free data sharing and circulation. Data providers can share their data to other AI computing service buyers for free, thus increasing data utilization and value.

6.4.2 Architecture

Data Management Module: This module is used for data storage, management, search and browsing, including data classification, tagging, description and other functions, this module is architecturally used as the application layer of the AI computational network data unified storage and management platform. The underlying functions such as data unique id generation, data storage media, data permission acquisition, etc. are all realized by the AI computational network data unified storage and management platform.

Data Trading Module: This module uses Utility coins to price, purchase, pay, confirm and supervise data transactions. Module functions include order management, payment interface, data quality evaluation, smart contract implementation and other functions.

Data Security Module: This module is used for data encryption, backup, recovery and supervision, including data encryption algorithms, backup policies, security detection and other functions. This module is also used as the application layer of the AI computational network data unified storage and management platform in the architecture, and the AI computational network data unified storage and management platform provides the underlying security technology functions.

Data sharing module: This module is used for free data sharing, permission acquisition, data circulation and other functions.

7 Performance and Availability

Focusing on performance and usability, Utility network aims to provide users with an efficient, scalable and secure blockchain infrastructure. The following subsections will discuss in detail the consensus speed, On-chain block bursting state, k8s technology, resistance to witch and proxy attacks, ranking mechanism, and explain the advantages of the Utility network in each area.

7.1 Consensus speed

The Utility network uses an advanced asynchronous consensus algorithm that improves consensus speed and reduces confirmation latency. This allows the Utility network to have higher throughput and be able to process a large number of transactions and computational tasks in a short period of time. The consensus speed of the Utility network is attributed to its advanced consensus algorithm and multi-layer validation mechanism. The following details the various stages of consensus speed.

Firstly, random numbers are generated and verified via VRF (Verifiable Random Function) and the fair and unpredictable selection of pre-burst block miners is ensured at a high level by round, providing a solid foundation for consensus, a process that is typically completed in less than 10 seconds.

Next, within 2-3 seconds, a pre-burst block miner generates a new block and commits it to the network. This process ensures efficient performance of the Utility network.

Immediately, within 5-10 seconds, the new block is broadcast to the entire network, allowing other miner nodes to know the existence of the new block. This helps to improve the transparency and consistency of the network.

After the new block is broadcast, other miner nodes will verify the new block within 1-2 seconds to ensure its legitimacy and correctness. This is a key aspect of the security of the Utility network.

Finally, within 30 seconds of the whole consensus process, a new block is created and added to the blockchain of the Utility network. This demonstrates that the Utility network has a fast consensus speed and is able to handle a large number of transactions and computational tasks in a short period of time.

7.2 On-chain block bursting state

Utility network achieves double verification of miners through on-chain state bursting blocks, ensuring miners' computing environment and computational availability, blockchain security and data consistency. By using the difficulty adjustment algorithm and VRF technology, the Utility network is able to achieve fair miner competition and stable updates of the on-chain state. This mechanism consists of two components: proof of availability of computational tasks (CAT) and Proof Of Computation Integrity (POCI).

Computational task availability proof (CAT) requires miners to prove the correctness and security of their execution environment before generating new blocks. In this way, only miners with a complete computing environment can participate in block bursting, thus preventing malicious miners from submitting forged or tampered blocks.

POCI, on the other hand, elects the right miners to generate new blocks through on-chain random numbers and computation competition. This mechanism ensures the availability of miners' computational power and a fair competition in the network.

With these two checksum mechanisms, the Utility network avoids a large amount of invalid computation for fighting the Byzantine General problem, thus freeing up 99% of computing power for meeting the Utility-based service system. This allows the Utility network to achieve large-scale applications in distributed training and inference computing scenarios, providing users with efficient, secure, and reliable computing services.

7.3 kubernetes Operational Extensions and End-Side Standard Component Library USL (Utiility Standard Library)

Utility network takes full advantage of Kubernetes [29] to support large-scale operational scaling and help developers easily build and deploy decentralized applications by providing an end-side standard component library to lower the development barrier. This provides a highly available, easy to deploy and manage solution for the miner side. With Utility Standard Library (USL), Utility network enables rapid integration of end-side standard components [32], making it easy for miners with verifiable computational chips to launch available service clusters efficiently.

USL includes a series of pre-built end-side standard components that work seamlessly with Kubernetes, simplifying the deployment and management process. This means that miners do not need to perform complex configuration and debugging, and can deploy clusters of Utility network services to their own hardware devices with a one-click clone operation.

This feature of the Utility network highlights the focus on miner-side availability, allowing any miner with a verifiable computational chip to quickly start and join the Utility ecosystem. This concept of "Clone is availability, availability is excellent." provides miners with great convenience, lowering the threshold of participation, and further promotes the popularity and development of the Utility network.

7.4 Resistance to witch attacks and agent attacks

A witch attack, also known as a "fake attack" or "forgery attack", is when an attacker forges or impersonates another user or system entity in order to gain unauthorized access or perform other malicious acts. For example, in e-commerce, a witch attacker can forge orders or transactions to gain unauthorized access to goods or services [33].

A proxy attack is the attack that an attacker uses a proxy or man-in-the-middle to deceive a system or user. The attacker uses a proxy to intercept, tamper with, or replay communication data to gain unauthorized access or for other malicious purposes. For example, in a network environment, an attacker tricks a router, gateway, or other intermediate node to intercept, tamper, or replay communication data to obtain sensitive information or perform other malicious acts [34].

Utility network employs multiple security mechanisms, such as strengthening authentication, ensuring multiple rounds of encrypted communication, controlling access, etc. to effectively defend against witch attacks and proxy attacks. At the same time, by limiting the rights and behaviors of malicious miners and imposing severe penalties and sanctions, Utility network ensures the security and stability of the entire network.

7.5 Ranking mechanism and Foundation index miner service

The Utility network introduces the ranking mechanism to rate and rank the services provided by

miners. The ranking mechanism is a mechanism to rank and display relevant results according to certain algorithms and rules in applications such as search engines and e-commerce. Its purpose is to improve the user experience so that users can find the desired information or goods more quickly [35, 36].

Ranking mechanisms typically consider a variety of factors, such as keyword match, page quality, user behavior, social trust, etc., to determine the weight and ranking of each result. The specific algorithms and rules vary by application and can be based on the implementation of technologies such as machine learning, collaborative filtering, and rule engines.

The Foundation Index miner service [37] is an indexing service based on blockchain technology and is provided by the Foundation Index team. This service mainly provides indexing services for decentralized applications and helps developers to implement data queries and interactions more easily. The indexing provided by this service includes contract-based events, on-chain data-based queries, etc., which can be used to support various application scenarios, such as decentralized exchanges, DeFi applications, etc.

In summary, the Ranking mechanism ensures the good service provided by miners and the user's multi-way choice of their search, while the Foundation maintains the index of miner services provided so that users can choose the right miner according to their needs and trustworthiness. These two mechanisms aid to improve the quality of miners' services and users' user-friendly experience, and both parties promote healthy competition among miners within the Utility network, and further optimize the overall performance and availability of the Utility network.

8 Future-oriented possibilities

As an innovative blockchain technology, Utility network innovatively integrates cryptography algorithm, Internet technology, data storage technology and other technologies. It has a very wide and broadened development prospect and potential. In the future, Utility network is expected to achieve more breakthrough technical progress and provide more services with higher quality to users.

8.1 More heterogeneous chips Memory chip/instruction support

As computer technology evolves contiguously, more and more heterogeneous chips have emerged with their own unique advantages and features. In the future, Utility network will continue to upgrade or actively expand its support for these heterogeneous chips, such as memory chips and more instruction sets. This can further improve the versatility and adaptability of Utility network, enabling it to meet more different scenarios and user needs.

Utility network is committed to building a comprehensive, efficient and secure network of computing resources to meet the demands of hyperscale computing power. To achieve this goal, Utility network will support the addition of more verifiable chips, compute storage, and network devices with security engines through the development of the Utility Proposal (UP). With the support of this protocol, Utility network will drive the development of self-configuration in the following three areas.

First of all, Utility network will actively promote the integration and application of various heterogeneous chips to take advantage of their respective computing performance. These heterogeneous chips include the CPU, GPU, TPU, etc., which is familiar to us. These chips have their own unique performance advantages and can fully meet the computing needs in different scenarios. In addition, Utility network will explore in-depth cooperation with various storage devices to achieve efficient management of data storage to match computing resources. This includes support for traditional hard drives, solid state drives, and new storage technologies aimed at improving the speed and reliability of data storage. It is worth mentioning that Utility network will also focus on the development of network devices to enable high-speed transmission and distributed scheduling of computing resources. All of this will help to build a highly optimized network of computing resources to provide users with more flexible and efficient computing services.

Through the above measures, Utility network will give full play to the advantages of storage and computational parallelism and the realization of incentive integration, providing a solid foundation for the all-human network born from super-scale computing. Utility Network will continue to explore and innovate to promote the development of computing resource network and contribute great

power to the scientific and technological progress of mankind.

8.2 Privatized computation services

Utility network will also explore the possibility of a privatized computing service. Such a service will allow enterprises and individuals to utilize the computing resources provided by Utility network while safeguarding the security and privacy of data. Through the privatized computation service, users can perform computation tasks in their own dedicated environment while avoiding data leakage and unauthorized access. At the same time, Utility network will provide comprehensive technical support and management tools to help users easily realize the deployment and operation and maintenance of privatized computation services.

Utility's Private Computing Service provides users with a more secure and efficient solution for computing resources. With privatized computation service, users can achieve large model deployment, private large-model deployment, high performance custom scaling, model fine-tuning and solve sensitive data problems under the premise of data security.

8.2.1 High-performance customized scaling

Private computation services allow users to flexibly adjust computing resources according to their needs, thus achieving optimization of high-performance computation. Users can quickly scale up or down computing resources according to task size and complexity to ensure a balance of computational efficiency and cost.

8.2.2 Large-model deployment and private large-model deployment

Utility network supports users to deploy large AI models in a private environment, ensuring that data is secure and not leaked to centralized cloud services. This means that users can run and manage AI models independently in their own computing environments to meet the computing needs of specific scenarios.

8.2.3 Model fine-tuning

The Private Computing Service supports fine-tuning of deployed AI models to improve their performance in specific tasks and scenarios. Users can tune their models for higher accuracy and performance based on their own datasets and requirements.

8.2.4 Resolving sensitive data issues

Utility network's privatized computation service offers significant advantages when dealing with sensitive data. Since the data only flows in private models and inference tasks, users do not need to worry about data leakage to third parties, thus ensuring data privacy and security.

Overall, Utility network's private computing services provide users with a solution that is both secure and efficient in terms of computing resources. With the four aforementioned components of high-performance custom scaling, large model deployment, model fine-tuning, and solving sensitive data problems, Utility network can also provide users with a powerful and flexible way to manage computing resources.

Looking towards to future, the development of Utility network will focus on the following areas: supporting more types of heterogeneous chips to expand the scope of Utility network's applicability; exploring privatized computation services to provide more secure and reliable computing resources; continuously optimizing network performance and availability to provide users with more efficient services; deeply researching and applying blockchain technology to promote the combination and innovation of Utility network with other fields. Utility network will continue to uphold the spirit of innovation, providing users with quality services, promoting the development and application of blockchain technology.

9 Summary

Utility Network is a decentralized computing service platform built on blockchain technology, which aims to solve the limitations of existing centralized computing resources and provide users with more efficient, secure and scalable computing services. The main technical features of the Utility network as described in details above include:

Efficient consensus mechanism: Proof Of Computation Integrity (POCI) breaks the limits of previous model of computational power consumption, and achieves fast consensus and block generation through chip ownership and VRF for random number generation interval and verification, combined with multi-signature verification and round-based pre-burst block election. Meanwhile, RPOI (Rent Proof Of Instructions) also ensures the effective rental and efficient use of computational power

to promote the generation of distributed computational networks.

On-chain block bursting state: the double verification of block bursters is accomplished by on-chain computation through Computing Availability over Time(CAT) and POCI, avoiding malicious access and fairness and competition in electing block bursters, ensuring effective computational resource allocation while reducing the impact of the Byzantine General problem.

Merkle State Machine off-chain data verification: Merkle Tree is a data storage structure used in the Utility network to relieve the pressure on the on-chain data, transferring off-chain data to the Utility network through Rollup and activating the Merkle State Machine (MSM) to store the hash values of these data or the root node of the Merkle Tree in an orderly manner on the blockchain while enabling fast verification. This technology reduces the storage pressure on the blockchain network while ensuring the security and integrity of off-chain data.

Virtualization and isolated execution environment: based on Arm framework driver and video memory virtualization support, the TPU chip is equipped with flexible sharing of computing tasks, improving the overall computing efficiency. The execution environment is combined with container technology to enable a highly isolated execution environment to ensure the security of computing services.

Multiple execution VM support: including Solidity VM, Kubernetes integration, WASM integration, and PyTorch outreach computing support, building a feature-rich and compatible computing platform.

Utility AI computational network: the Utility AI platform layer is mainly composed of four sub-platforms: a one-stop platform for debugging and training deployment of four AI centers, a unified scheduling and management platform for AI network tasks, a unified storage and management platform for AI network data, and an AI network data market. The goal is to build an innovative AI computing infrastructure with easy access to computing resources, efficient and unified scheduling of tasks, convenient and inexpensive invocation by users, and a sustainable development operation model and mechanism.

The economic model of utility network: it mainly concerns about the token cycle and computing ecology, including mining mechanism, token reduction, Gas and Burn Fee, machine time rental order and inference deployment order. Through this economic model, the Utility network achieves fair incentive allocation and rational resource utilization.

In terms of future-oriented possibilities, Utility network supports more security engines by developing UP (Utility Proposal) to verify heterogeneous chips, storage chips and instruction support, computational storage, instruction sets and network devices to build an incentive network that takes full advantage of both storage and computation; and to realize privatized computation services so as to provide users with more diverse computing services. All the technologies, functions and expectation surrounding Utility network are concisely summarized in the following conclusion figure.

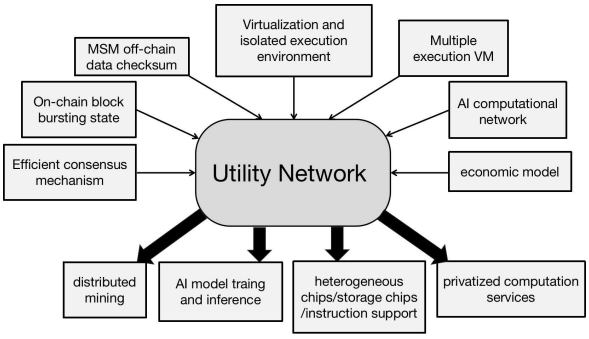

Figure 13: summary of Utility newtork's technical components, features and future outlook

In summary, the Utility network provides a decentralized, efficient and secure computing service platform for users through an efficient consensus mechanism, a secure computing environment, rich execution VM support and a flexible economic model. The development of Utility network will greatly promote the development of distributed computing and make an important contribution to the technological progress of mankind.

Reference

[1] Benet, J. (2014). IPFS - Content Addressed, Versioned, P2P File System. [arXiv:1407.3561v1]
[2] Protocol Labs. (2021). Filecoin: A Decentralized Storage Network. Filecoin Whitepaper. Retrieved from https://filecoin.io/filecoin.pdf
[3] Nakamoto, S. (2008). Bitcoin: A Peer-to-Peer Electronic Cash System. Retrieved from https://bitcoin.org/bitcoin.pdf
[4] Buterin, V., et al. (2014). A Next-Generation

Smart Contract and Decentralized Application Platform. Ethereum Whitepaper. Retrieved from https://ethereum.org/whitepaper

[5] Wood, G. (2014). Ethereum: A Secure Decentralised Generalised Transaction Ledger. Ethereum Yellow Paper. Retrieved from https://ethereum.github.io/yellowpaper/paper.pdf

[6] Boneh, D., et al. (2013). Public Key Cryptography for the Financial Services Industry: The Elliptic Curve Digital Signature Algorithm (ECDSA). ANSI X9.62-2005. Retrieved from https://www.secg.org/sec2-v2.pdf

[7] Merkle, R. C. (1987). A Digital Signature Based on a Conventional Encryption Function. Advances in Cryptology – CRYPTO '87. Lecture Notes in Computer Science, vol 293. Springer, Berlin, Heidelberg. https://doi.org/10.1007/3-540-48184-2-32

[8] Lamport, L. (1981). Password Authentication with Insecure Communication. Communications of the ACM, 24(11), 770-772. https://doi.org/10.1145/358790.358797

[9] Intel Corporation. (2016). Intel® Software Guard Extensions (Intel® SGX). Retrieved from https://software.intel.com/en-us/sgx

[10] AMD. (2013). AMD64 Architecture Programmer's Manual Volume 2: System Programming. Retrieved from https://www.amd.com/system/files/TechDocs/24593.pdf

[11] ARM Limited. (2018). ARM Architecture Reference Manual ARMv8, for ARMv8-A architecture profile. Retrieved from https://developer.arm.com/documentation/ddi0487/latest/

[12] Boneh, D., Gentry, C., & Waters, B. (2005). Collusion Resistant Broadcast Encryption With Short Ciphertexts and Private Keys. Advances in Cryptology - CRYPTO 2005, 258-275. https://doi.org/10.1007/11535218]_16

[13] Parno, B., et al. (2013). Pinocchio: Nearly Practical Verifiable Computation. 2013 IEEE Symposium on Security and Privacy. https://doi.org/10.1109/SP.2013.44

[14] Dwork, C., & Naor, M. (1993). Pricing via Processing or Combatting Junk Mail. Advances in Cryptology – CRYPTO '92. Lecture Notes in Computer Science, vol 740. Springer, Berlin, Heidelberg. https://doi.org/10.1007/3-540-48071-4_29

[15] Ben-Sasson, E., et al. (2014). Zerocash: Decentralized Anonymous Payments from Bitcoin. 2014 IEEE Symposium on Security and Privacy. https://doi.org/10.1109/SP.2014.36

[16] Szabo, N. (1997). Formalizingand Securing Relationships on Public Networks. First Monday, 2(9). https://doi.org/10.5210/fm.v2i9.548

[17] Micali, S., Rabin, M., & Vadhan, S. (1999). Verifiable Random Functions. 40th Annual Symposium on Foundations of Computer Science, 120-130. https://doi.org/10.1109/SFFCS.1999.814594

[18] Benet, J., & Greco, N. (2017). Proof of Replication. Filecoin Research. Retrieved from https://filecoin.io/proof-of-replication.pdf

[19] Ben-Or, M., et al. (1988). Simple Efficient Asynchronous Byzantine Agreement with Optimal Resilience. Proceedings of the first annual ACM symposium on Principles of Distributed Computing, 42-52. https://doi.org/10.1145/62546.62550

[20] Luu, L., et al. (2016). A Secure Sharding Protocol for Open Blockchains. Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security, 17-30. https://doi.org/10.1145/2976749.2978389

[21] Kalodner, H., et al. (2018). An Empirical Study of Namecoin and Lessons for Decentralized Namespace Design. WEIS 2015: Workshop on the Economics of Information Security. Retrieved from https://weis2015.econinfosec.org/wp-content/uploads/sites/5/2015/05/WEIS 2015 submission 72.pdf

[22] Back, A. (2002). Hashcash - A Denial of Service Counter-Measure. Retrieved from http://www.hashcash.org/papers/hashcash.pdf

[23] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., ... & Polosukhin, I. (2017). Attention is all you need. In Advances in neural information processing systems (pp. 5998-6008).

[24] Daian, P., et al. (2020). Flash Boys 2.0: Frontrunning, Transaction Reordering, and Consensus Instability in Decentralized Exchanges. 2020 IEEE Symposium on Security and Privacy. https://doi.org/10.1109/SP40000.2020.00047

[25] Eyal, I., & Sirer, E. G. (2018). Majority is not Enough: Bitcoin Mining is Vulnerable. International Conference on Financial Cryptography and Data Security, 436-454. https://doi.org/10.1007/978-3-662-45472-5_28

[26] Ren, X., & Yu, H. (2020). Blockchain-Based Intelligent Transportation Systems: A Comprehensive Review. IEEE Transactions on Intelligent Transportation Systems, 21(5), 18

[27] Chen, Y., Zhang, Y., Xu, Z., & Huang, J. (2019). Secure execution of

blockchain smart contracts based on virtual machine. IEEE Access, 7, 61933-61943. https://doi.org/10.1109/ACCESS.2019.2918648

[28] Chia, M. P., Gan, C. Y., & Low, K. L. (2018). Securing blockchain execution environment through isolation and hardware-enhanced attestation. IEEE Transactions on Dependable and Secure Computing, 15(4), 570-583. https://doi.org/10.1109/TDSC.2016.2615552

[29] Sun, Z., Cui, Y., Zhang, Y., & Yu, Y. (2020). Containerized deployment of Ethereum smart contract based on Kubernetes. In Proceedings of the 2020 IEEE International Conference on Smart Internet of Things (SmartIoT) (pp. 182-187). https://doi.org/10.1109/SmartIoT49753.2020.00039

[30] Tian, Y., Han, G., Chen, S., Chen, Y., Wu, D., & Wei, Y. (2021). A WASM-based execution model for blockchain smart contracts. In Proceedings of the 2021 IEEE International Conference on Blockchain (Blockchain) (pp. 166-173). https://doi.org/10.1109/Blockchain50986.2021.00028

[31] Zhou, Z., Yan, X., Guan, S., Zhang, J., & Zhu, X. (2021). PyTorch on the blockchain: Integrating deep learning and smart contracts. IEEE Transactions on Services Computing, 14(1), 144-157. https://doi.org/10.1109/TSC.2019.2935417

[32] Chen, X., Chen, G., Wu, H., & Wang, Y. (2018). Design and implementation of IoT platform based on USL. In Proceedings of the 2018 5th International Conference on Systems and Informatics (pp. 558-562). https://doi.org/10.1109/ICSAI.2018.8574167

[33] Diffie, W., & Hellman, M. (1976). On the security of public key protocols. IEEE Transactions on Information Theory, 22(6), 644-654. https://doi.org/10.1109/TIT.1976.1055638

[34] Maurer, U. (1993). Secure communication over insecure channels. Advances in Cryptology — EUROCRYPT'93, 1-14. https://doi.org/10.1007/3-540-48285-7_1

[35] Yang, J., Ma, Y., Xu, X., & Wang, X. (2020). Research on a blockchain-based ranking mechanism in academic community. In Proceedings of the 2020 International Conference on Intelligent Sustainable Systems (ISS) (pp. 55-60). https://doi.org/10.1109/ISS49245.2020.9240599

[36] Wu, J., & Xie, X. (2021). Design and implementation of search engine ranking algorithm based on blockchain. In Proceedings of the 2021 IEEE 4th International Conference on Information Systems and Computer Aided Education (ICISCAE) (pp. 556-560). https://doi.org/10.1109/ICISCAE53270.2021.00098

[37] Li, X., Li, C., Li, W., Li, Y., & Li, G. (2020). Research on blockchain technology based on the Foundation index miner service. In Proceedings of the 2020 3rd International Conference on Computer Communication and Informatics (ICCCI) (pp. 1-6). https://doi.org/10.1109/CCCI48366.2020.9117489