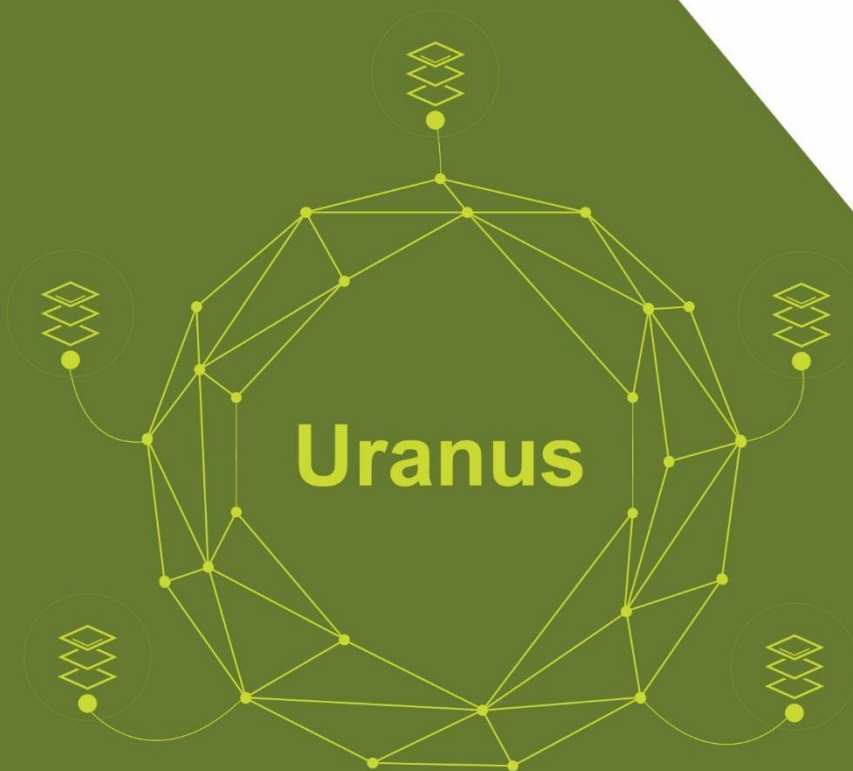


Technical White Paper

Uranus

V2.0



Uranus ,the primordial god of sky, was a symbol of hope and future, delivering light and power to every corner in the world.

Table of Contents

1	Executive Summary.....	2
2	Mission	4
3	Pain Points.....	7
	3.1 Problems of Conventional Public Clouds.....	7
	3.2 Redundancy of Ubiquitously Existing Computing Power.....	9
	3.3 The Sharing of Ubiquitously Existing Computing Power Faces Huge Technical and Business Challenges	10
4	Solution	12
5	Technology	15
	5.1 Architecture of the Uranus System.....	15
	5.2 Uranus Chain Technology	17
	5.3 Computing Power Container Engine (CPCE).....	28
6	Application Scenarios and Ecology System.....	42
	6.1 Application Scenarios	42
	6.2 Business Application Examples	44
	6.3 Container Image and Application Ecology System.....	46
7	Development Roadmap	49
8	Team Members	50
	8.1 Executive Team.....	50
	8.2 Technical Advisers	51
	8.3 Product Development	54
	8.4 Advisory Board.....	58

1 Executive Summary

Uranus is the Uber and Airbnb platform for the computing power resource space, that goes beyond conventional public clouds. To address the current pain points of public clouds, such as its incompatibility for distributed computing, vulnerability to theft of data assets, high tariffs, lack of flexibility and difficult in data migration, Uranus seamlessly links and integrates redundant computing power in the world through establishment of an expandable public blockchain and a distributed container technology and provides efficient, cost- effective and decentralized computing service for users.

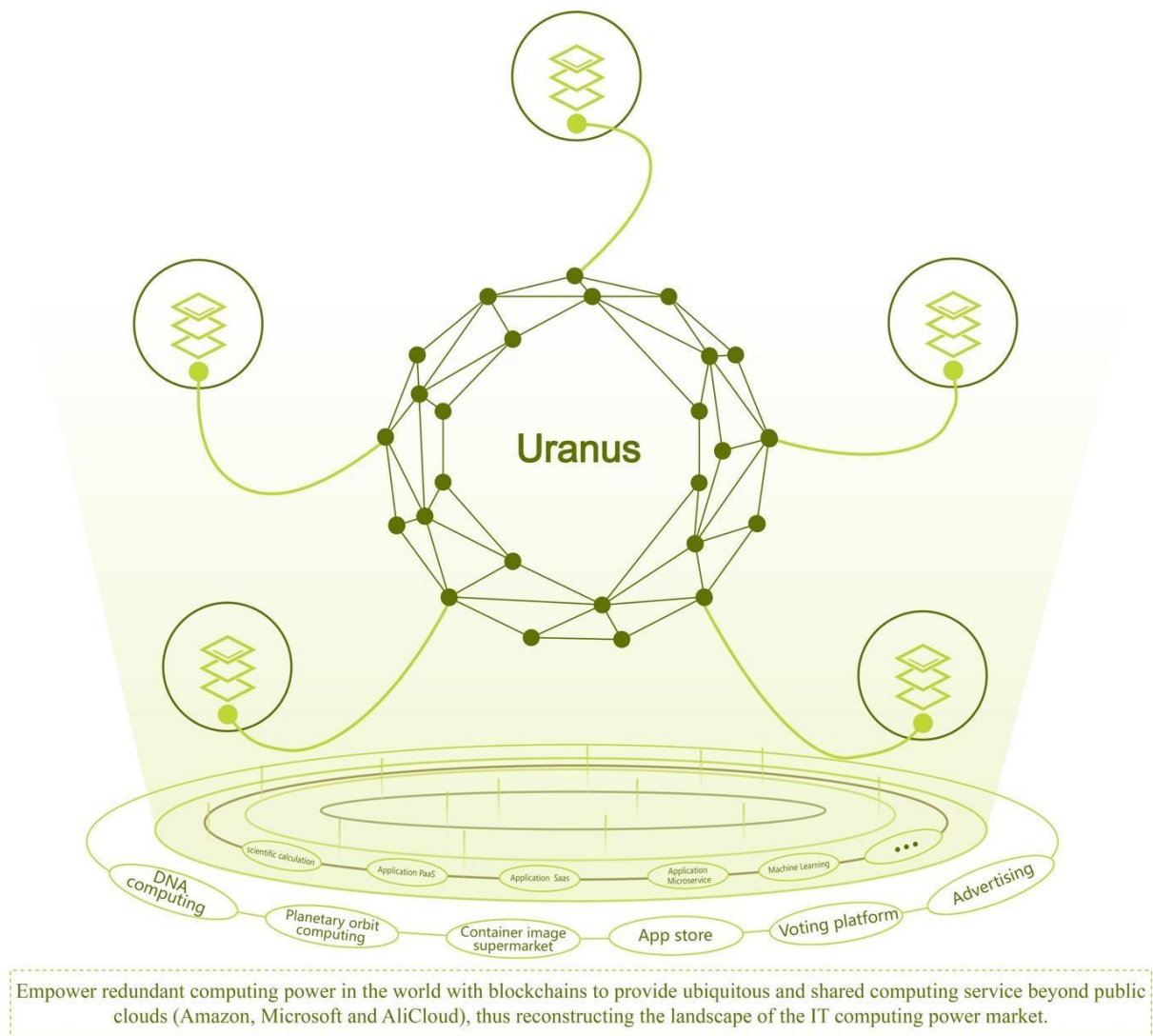


Figure 1 Vision of Uranus

Uranus will be a globalized computing resource platform with thousands upon thousands of resource contributors and users which completely changes the landscape of conventional centralized cloud computing business and resource structure. Its computing power resource platform may be used in large computing projects based on

distributed

computing, such as protein structure computing in Volunteer Projects, computing of new planetary orbits and analysis of weather data. It may also be used in many projects based on edge computing, such as search, voting, games, advertising and IoT node management. This platform may also serve as an underlying service platform for other public blockchains, for example, the provision of one-click deployment of various chains and applications. Particularly, on this new container architecture for distributed computing, container images of supermarkets and application stores may be built, thus constructing an application ecosphere based on distributed computing and edge computing can create huge social and economic values.

The executive team of the Uranus project have solid theoretical foundation and have an average of 20 years' practical experience in this field. They have successfully established enterprises whose products have entered some of the top 20 Chinese corporations. Under the leadership of the executive team, the technology development team of the Uranus project has completed the development and commercial delivery of many major projects over the years. The Advisory Board of the Uranus project consists of international preminent technical experts in their respective fields like the crypto economy, and media.

2 Mission

Empowering redundant computing power all over the world with blockchain technology to provide ubiquitous and shared computing service beyond public clouds (Amazon, Microsoft and AliCloud), thus revolutionizing the landscape of the IT computing power market.

Use containers to consolidate various applications based on distributed computing to build an ecosphere using a public blockchain.

As Mark Waiser, a proponent of ubiquitous computing, states: Technology should create calm, that which informs but doesn't demand our focus or attention.

Globally, resource sharing at present has become an important aspect of technological progress and social development. Carpooling, housing sharing, and even bicycle sharing have become a reality. Next, people will naturally think of sharing computer resources. Although cloud computing that shares computing and storage resources as a service is ubiquitous, large public cloud computing platforms such as Amazon, Microsoft, and Ali, and tens of thousands of private cloud systems that have emerged in the global IT market are not able to support distributed node-based computing, and the physical resources provided by them do not include a large number of personal PCs, business PCs, data centers of small and medium-sized enterprises, servers, workstations, thin terminals, other embedded terminals, and billions of mobile terminals in the market. The redundancy and waste of computing power are measured in a few hundreds of billions of USD. Moreover, the monopoly in the public cloud market has become a barrier to technological progress, cost reduction and efficiency improvement. To this end, the use of redundant computing power in the society (as shown in the figure below) will reconstruct the world's computing market and pattern, thereby promoting the development of the IT industry. The Ubiquitous Computers Pool technology emerged at the right moment.

In the 1980s, American scientist Mark Waiser proposed the concept of Ubiquitous Computing. He envisioned embedding computers in various sizes into everything, and then letting computers serve the people quietly through wireless communications. Ubiquitous computing complies with the following principles:

- ★ The purpose of a computer is to help you do something else;
- ★ The best computer is a quiet, invisible servant;
- ★ The more you can do by intuition, the smarter you are; the computer should extend your unconscious;

★ Technology should create calm.

The implementation of a good idea generally requires technological progress and commercial interests to be driven forward. For example, the Internet of Things (IoT) is the most typical implementation of ubiquitous computing.

The Uranus project has a close relationship with ubiquitous computing. It should be noted that Ubiquitous Computing and Ubiquitous Computers have slight differences in definitions, but their commercial goals are not similar. What Ubiquitous Computing wants to achieve is the ability to compute anywhere and everywhere while Ubiquitous Computers seek to free up existing redundant computing power. Their commonality as pointed out by Mark Waiser when designing Calm Technology is that it should inform but does not demand our focus or attention.

The commercial interests that drive the sharing of ubiquitously existing computing power are the huge amount of redundant computing power that exists in the world. The technological progress that drives the sharing of ubiquitously existing computing power is the rapid development of blockchain, cloud native and microservice technology which has made the sharing of redundant ubiquitously existing computing power possible.

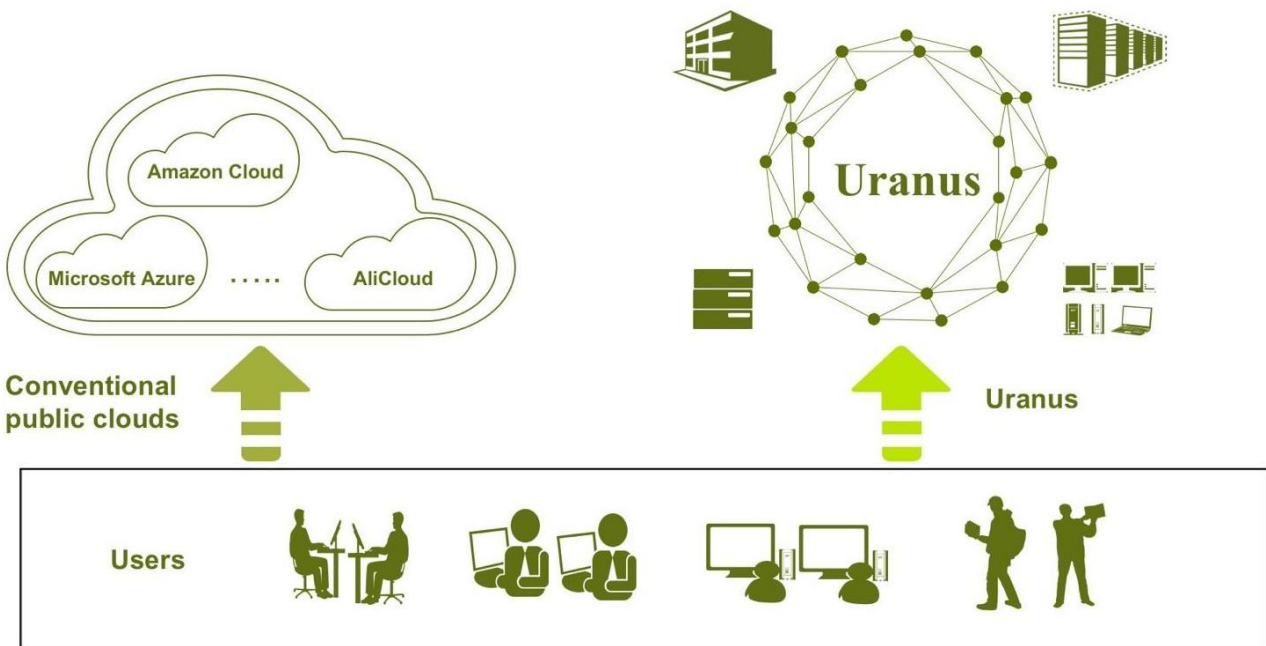


Figure 2 Mission of Uranus

In summary, the Uranus system not only builds a public cloud based service using redundant computing power available but it also serves as an IaaS established on the basis of blockchain. The IaaS supports any future blockchain based PaaS and SaaS. Our platform can also be known as a BlockStack which includes blockchain application stores.

3 Pain Points

3.1 Problems of Conventional Public Clouds

With the rapid development of hotspot technologies such as cloud computing, big data, IoT, smart city and artificial intelligence, the global public cloud market has seen rapid growth. According to the Gartner research report, the global public cloud market has grown from 68.3 billion dollars in 2010 to 307.4 billion dollars in 2017, with the compound annual growth rate maintained above 17%, and the average growth rate is expected to remain above 15% from 2018 to 2020, likely reaching 383.4 billion dollars by 2020. Nevertheless, in 2017, the global public cloud computing market accounted for only 7% of the total global IT expenditures. Even after five years, the share of public clouds in the world's trillion-dollar IT spending market will be still less than 12%. It is because public cloud is facing unprecedented challenges while it is growing at a high speed.

Data Security Risk

The storage of core business data on the public cloud for enterprise users is tantamount to handing over the keys to the company to other parties for safekeeping. This is the most worrying security risk for the users of cloud computing as it is a major source of stolen user data. Here is Gartner's statistics on public cloud data security issues for Top500 companies over the past few years:

Figure 3 Leak incidents of existing public cloud providers

User name	Public cloud provider	Details of leakage
Arby's	Oracle Cloud	Hackers stole consumer credit card data at Arby'sRestaurant Group (ARG)
The Pentagon	AWS	AWS S3 configuration error, resulting in accidental disclosure of the information of 1.8 billion citizens
Uber	AWS	Data leakage incident of 57 million accounts and the names and driving license numbers of 600,000 American drivers. GitHub obtained the accounts and passwords of Uber engineers in AWS
World Wrestling Federation	AWS	S3 configuration problems
Qudian	Ali Cloud	Data leakage of millions of students
Equifax	AWS	Data leakage of about 143 million users –financial institutions
Accenture	AWS	Enormous password data was stored in four cloud servers without encryption protection, causing leakage of highly sensitive passwords and keys.

The current public cloud platforms may lead to common security risks in the following situations:

- User data is lost due to hacking and security breaches;
- Data is not encrypted or quarantined during transmission, resulting in information disclosure;
- User data is not systematically backed up for disaster recovery when it is stored on the cloud;
- Whether users can safely and seamlessly migrate data from the public cloud.

Expensive and Unfriendly Billing Model

The existing public cloud platforms mainly employs annual/monthly and/or volume-based billing model. The former can realize centralized management of virtual machines, increase virtual machine density and resource utilization, and improve profitability. However, whether users actually use the services or not, suppliers must provide 365*24 uptime in accordance with the contract while keeping data center costs high. For users, the annual/monthly model might look economical, but in fact is not. The minimum billing units for CPU, memory, disk, and network are all on the monthly, quarterly, or yearly basis. High-quality IDCs often require users to prepay in a lump sum. This method requires a large amount of initial investment for users and lacks flexibility in adjusting according to the actual needs of the users.

Flexible volume-based billing shall be one of the core features of public cloud, but the extensive pricing models daunt users and are miserable for service providers.

Unfit for Distributed Applications

The digital revolution has caused sweeping changes to how traditional industries operate. The innovations in education, healthcare, transportation and public services are changing with each passing day with almost all data storage and calculation moving to the cloud and accessible everywhere through the internet. While cloud computing might be a core support technology of digitization, it may not be the only solution to digitization for various industries, and in particular it cannot satisfy distributed applications with real-time and local computing needs.

In the conventional cloud computing model, people may click a button and wait for the completion of centralized background operations before being given the results. This model can satisfy many of the applications for cloud computing, but some applications require real-time response in milliseconds. Consequently, use of the existing cloud computing model is not feasible to power real-time computing requirements if transmission of data to the remote cloud terminals is carried out through a laggy and

jittery network. It will be more suitable if the network for computing, storage, application and data is integrated and situated closer to the devices or data sources where the nearest edge computing is used. In addition to the requirement for real-time response, a lot of distributed computing also needs to be done remotely at the edge.

In the long run, some applications are able to use centralized cloud computing while other applications are able to use distributed computing to the edge. These two solutions will be employed according to their respective advantages whilst having the opportunity to collaborate and complement each other.

3.2 Redundancy of Ubiquitously Existing Computing Power

Data from IDC shows that ubiquitous computing terminals (personal PCs, commercial office terminals, traditional physical servers, and data center resources) are generally underutilized, as shown in the following figure:

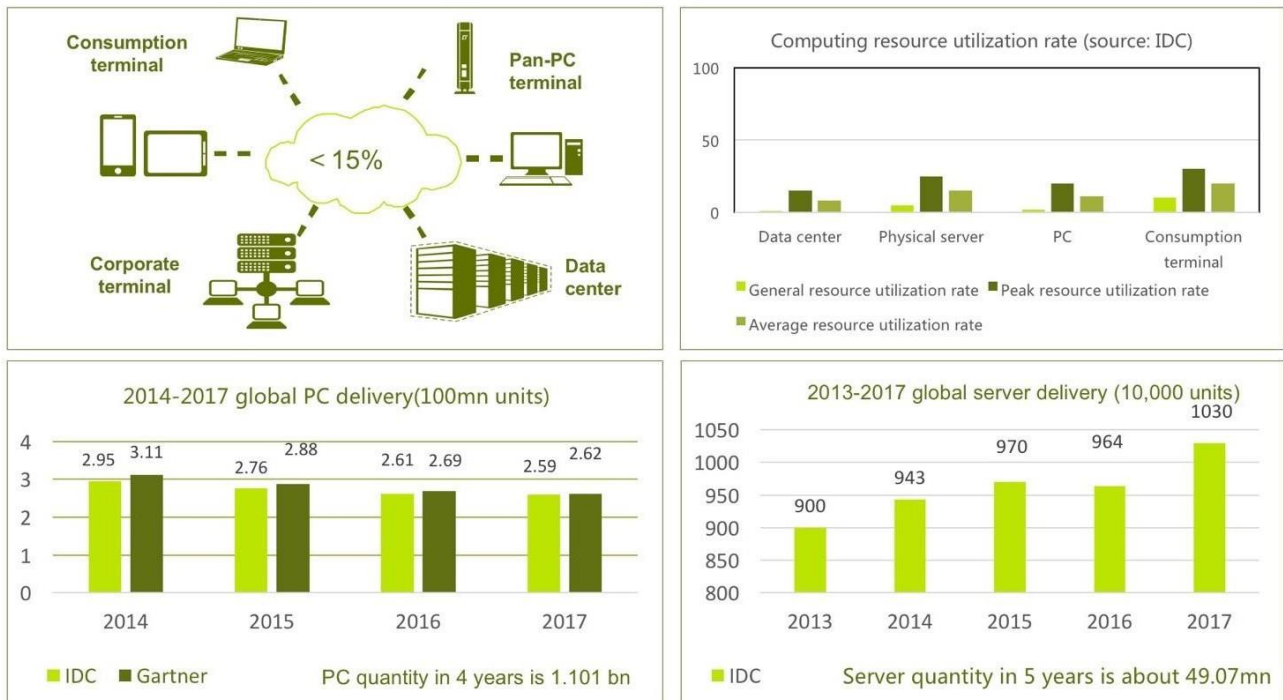


Figure 4 Statistics on sources and volume of global computing power

- ★ The utilization rate of ubiquitous PC computing power resources is less than 15% on average and the redundancy in computing power is severe;
- ★ The current average utilization rate of computing power resources in major public clouds, private clouds, telecommunication operators, and IDC service providers is also lower than 10%-15%, and a large amount of computing resources are idle;
- ★ Assuming that the value of a PC is USD 400, then the total computing power of global PCs will have a value of USD 460 billion;
- ★ Assuming the value of a server is USD 8,000, then the total computing power of global servers will have a value of USD 800 billion.

To be specific:

The purchase volume of commercial PCs is about 262 million units a year worldwide. If we suppose the value of a PC is USD 400, the computing resources worth 560 billion* (0.85) = \$476 billion are idle. Considering the 5-year lifecycle of PCs, at least USD 2380 billion of computing resources are idle. If servers, workstations and personal PCs are considered, the idle computing resources in the worldwide market will exceed USD 4 trillion.

Therefore, the market for sharing of ubiquitous redundant computing power is huge. Even if part of it is utilized, the direct and indirect economic value will be on a scale of a few hundred billion dollars. At the same time, it will promote the change of market structure and technological progress in the world.

3.3 The Sharing of Ubiquitously Existing Computing Power Faces Huge Technical and Business Challenges

In the past two years, excellent players in the US market have attempted to use Docker and Kubernetes (k8s) to integrate redundant data center resources to form an “alliance” type public cloud platform beyond Amazon Cloud to break the monopoly of the American public cloud market. This can be considered as a subset of sharing existing ubiquitous computing power in the field of servers and data centers, but this attempt faces enormous challenge in both technological and business applications.

First of all, contributors of redundant server resources come from different non-affiliated organizations which makes collaboration and cooperation much more difficult. Secondly, such a resource sharing system must have an efficient, trustworthy, secure and real-time transaction mechanism (i.e., Calm Technology) to minimize the cost of sharing and achieve the purpose of effective use of redundant resources. Lastly, because the physical layer and the connection layer of each terminal vary greatly. In order to achieve the sharing of computing power, it is necessary to use popular applications on every terminal that participates in the sharing of computing power, and at the same time to effectively maintain and manage the problems encountered during its use.

Prior to the application of blockchain technology, the industry could not solve the first and second problems above. Even though the third problem could be solved by “cloud native” and “microservice” solutions, the technical considerations in the aspects of “cloud native” and “microservice” solutions are based on a relatively consistent central computer framework which is not able to resolve the interaction between two channels. The need to realize value delivery based on blockchain and the interaction between service delivery channels is also one of the core issues that need to be solved in the future.

4 Solution

The technical team of Uranus hails from two sectors; the computing resource pool technology sector and the blockchain technology sector. The experts from the computing resource pool technology sector have at least 10 years' experience in the development of open source software, cloud computing, containers, virtualization and terminal systems. Their products have been successfully commercialized and is being used by the top 20 Chinese corporations, military and government. The experts from the blockchain technology sector have more than 6 years' practical experience in the development and successful application of consensus algorithm, proof of workload and development of smart contracts developed by public blockchain. As shown in Figure 5, Uranus is a platform built by the experts from the two teams by integrating the technologies in these two sectors to achieve sharing of existing ubiquitous computing power.

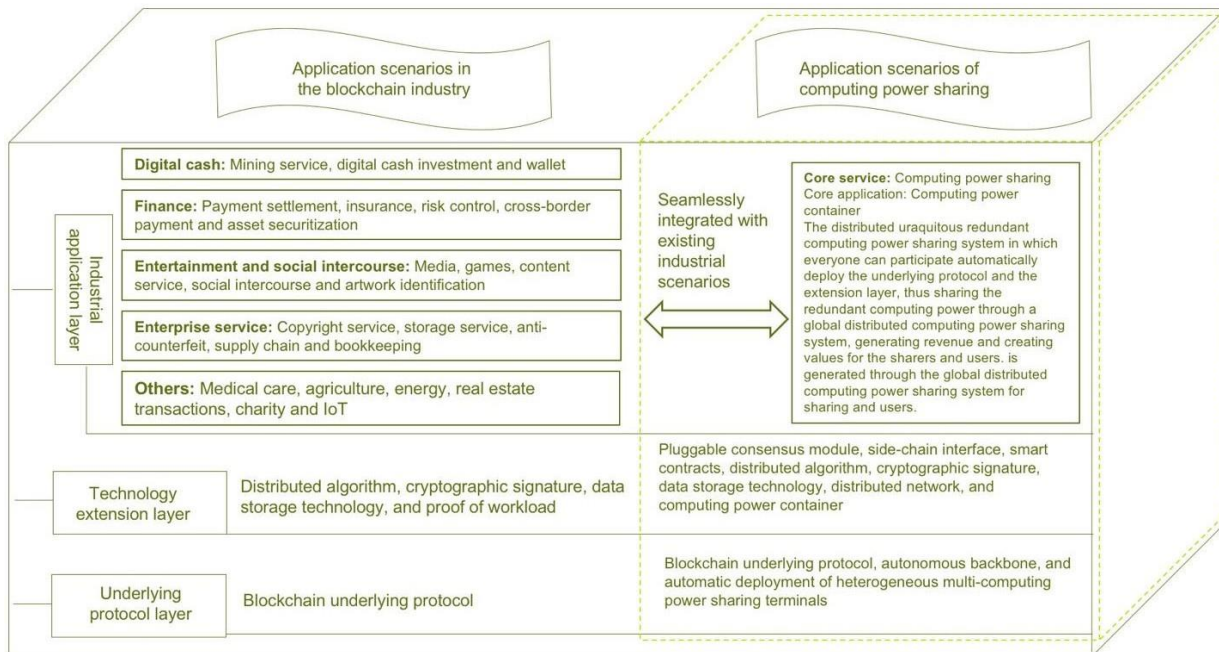


Figure 5 Solution for Uranus platform

The elements of the Uranus platform will be described briefly.:

Unit of Measurement of Standardized Computing Power

Computing power is a complex and multidimensional resource related to use and application. Therefore, standardizing the unit of computing power is a basis for sharing of existing ubiquitous computing power.

Blockchain Connects the Terminals with Redundant Computing Power to Complete Value Delivery

Blockchain can provide efficient, secure, and reliable value delivery between thousands upon thousands upon thousands of terminals that contribute their redundant computing power. However, in order to apply the blockchain to the sharing of existing ubiquitous computing power, we need to build a customized public blockchain with corresponding functions.

Build a Service Delivery Channel for Terminals with Redundant Computing Power

A blockchain cannot deliver services. For this reason, it is necessary to build an additional channel that can deliver applications and computing results.

Implement the Interworking and Interaction Between Two Channels

Real-time interaction are needed between two channels. The current public blockchain does not support such interworking and interaction. For this purpose, a new public blockchain needs to be created to add new functions and utilize real-time interaction between the two channels.

Establish a Smart Contract with Rich Content

The extended smart contract not only stipulates the content of the transactions between the contributors and users of computing power, it also stipulates the troubleshooting processes for various possible problems that might be encountered whilst using the smart contract. This functionality guarantees that the extended smart contract will be completed automatically without manual intervention.

Build Computing Power Containers

The computing power container is a basic configuration that ensures the applications supported by each platform can be implemented successfully after standard computing power units are determined. The core role of this basic configuration is to ensure successful execution and return of results on all the terminals participating in the sharing of computing power.

Transform the Existing Distributed Cloud Native Architecture

The cloud native architecture of the existing distributed system is mainly used on centralized and relatively consistent physical terminals. In the ubiquitous terminal environment, this consistency is greatly reduced. To this end, transforming the existing cloud native architecture is an important link to solve the sharing of existing ubiquitous

computing power.

Establish Perfect Application Stores

The application stores all have validated applications, guaranteeing the feasibility of the applications and can also dynamically balance the load of the terminal system through load balancing technology.

Own a Mining Pool to Carry Values

The tokens of this system are generated by the volume of shared computing power and the digital function. It stores some of the tokens in its own mining pool that are available to buyers. The tokens obtained by buyers can be used to exchange equivalent computing power.

5 Technology

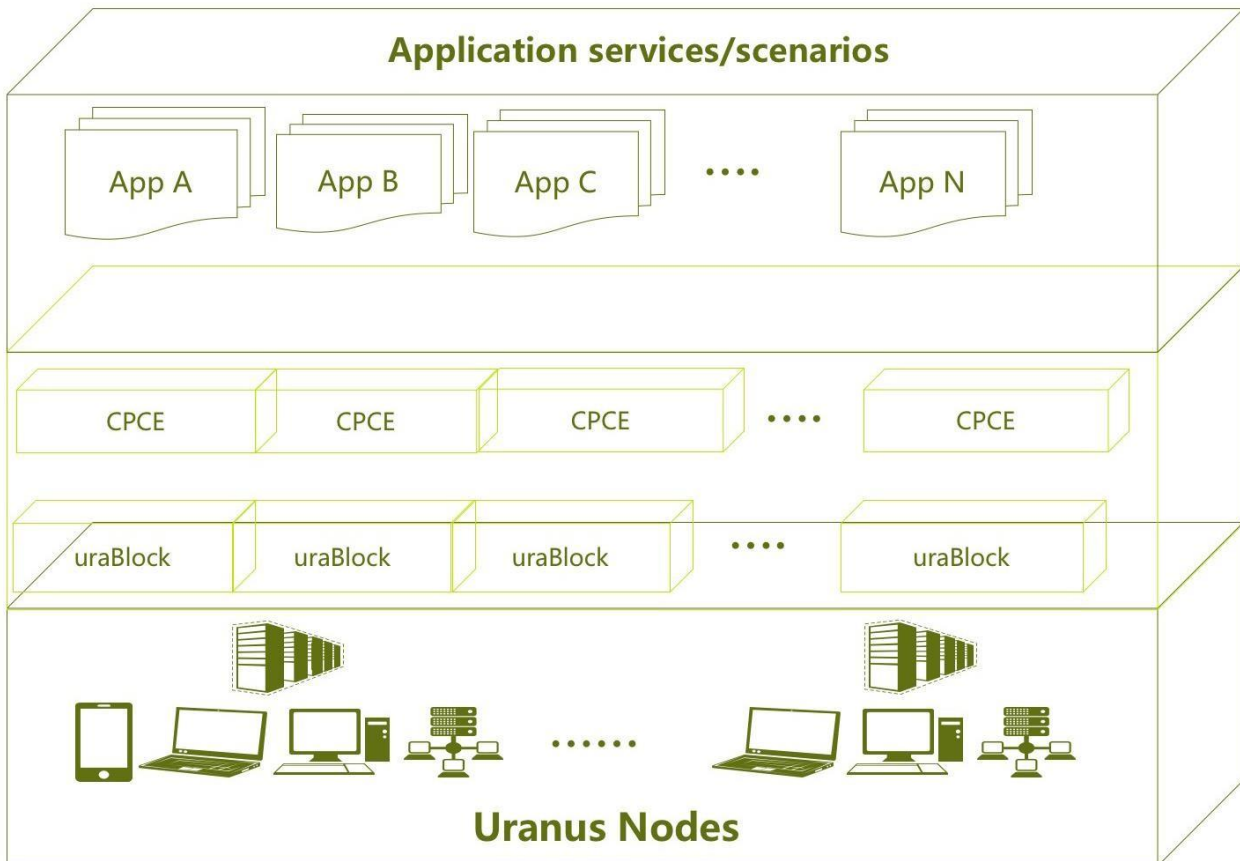
The Uranus system aims to provide a decentralized blockchain-based platform for the sharing of existing ubiquitous computing power that is flexible, on a pay-as-you-go basis and supports user self-service. The design goal pursued by the Uranus platform is:

- ★ System autonomy, no vendor lockout
- ★ User experience is supreme
- ★ Flexible and pay-as-you-go
- ★ High throughput and scalability
- ★ Full support for cloud native

5.1 Architecture of the Uranus System

The Uranus system is a decentralized platform for the sharing of existing ubiquitous existing computing power based on blockchain and cloud native/microservice technology. It provides a fairer and more cost-effective resource trading market for all resource contributors, resource users, and participants of its ecosystem. It will gradually reconstruct the existing global landscape of cloud computing currently dominated by Amazon, Microsoft, Ali and other companies, and promote the rapid development of the next-generation decentralized edge computing and fog computing. Based on the concept of integrating blockchain and computing power sharing, it has established a new and innovative intelligent resource delivery system and value delivery system, as well as communication and collaboration between the two systems.

Figure 6 Schematic diagram of the architecture of the Uranus



Uranus mainly comprises a service delivery system and a value delivery system in terms of its architecture:

Service delivery layer

- ★ Ubiquitous cloud infrastructure, and standardized container metering
- ★ Use on-demand, automatic settlement
- ★ Blockchain-based, decentralized computing resource pool
- ★ High throughput, high availability, and low cost to meet the needs of modern business
- ★ Container-packaged, dynamically managed cloud native applications

Value delivery layer (Uranus chain):

- ★ Basic chain
- ★ uraTiller module
- ★ Enhanced consensus module

The core technologies of Uranus center on innovative blockchain, smart contract, contribution validation mechanism technology for computing power sharing and computing power engine management scheduling, computing power container Worker, Agent, image, and high-speed P2P network distribution technology.

5.2 Uranus Chain Technology

Executive Summary

Blockchain is a technical system that stores data in a block structure that is maintained by multiple parties using cryptographic technology to ensure data transmission, access and storage. Its decentralized, trust-free and tamper-proof nature make it a first choice for digital asset ledger in the early days. Today, the basic chain has evolved to meet the high-throughput, high-performance demand at a commercial level while guaranteeing decentralization and trust intermediary. At the same time, vertical industry innovations based on the basic chain have mushroomed and the industry has entered an unprecedented period of rapid development.

System Architecture

The Uranus system is a technical platform for the sharing of existing ubiquitous computing power based on blockchain technology and the increasingly popular cloud native and microservice technology. It builds upon a decentralized IaaS & PaaS lightweight operating platform, which provides a broader and more cost-effective resource trading market for all resource contributors and users. This will gradually reshape the current entire cloud computing world, and promote the rapid development of the next-generation distributed edge computing and fog computing where blockchain technology becomes the cornerstone for the construction of value mutual trust and value delivery networks.

The Uranus system is mainly composed of core components such as uraBlock, Uranus management engine, uraTiler, and the access gateway. The figure below describes the logical architecture of a complete node in the Uranus blockchain network

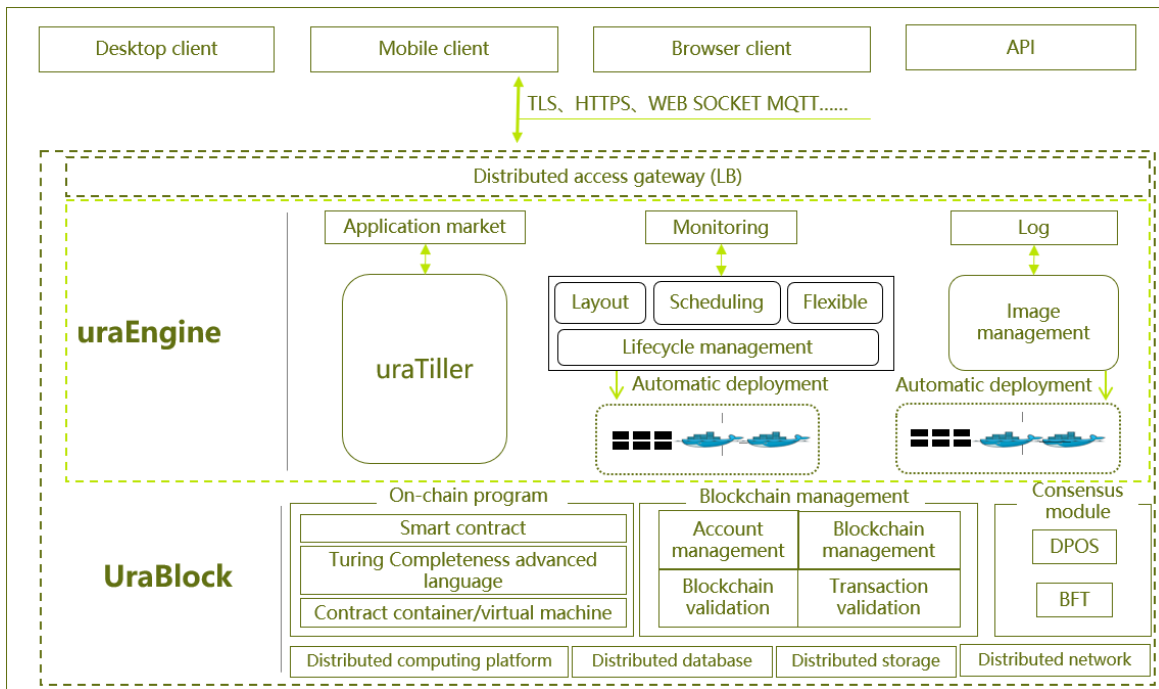


Figure 7 Architecture of the Uranus system

Uranus Chain

The storage layer is responsible for storing all the data on the chain. The data on the chain is stored in the block measuring in unit of pieces and in a Merkle-tree structure. In addition to the traditional transaction data, it also stores the relevant data in computing power containers. The node will write the computing power container information added at each block time into the block which mainly includes the following information: container address, the address of the node to which the container is bound, and the resource weight values of the container (CPU, memory, disk resources, etc. are obtained through calculations by an optimized algorithm). If the container is already serving externally, the corresponding contract ID will be marked, too.

The network layer is responsible for the interaction between the blockchain data and because the statistics and management of the computing power nodes are needed, the data channel and management channel for Computing Power Container Engine (CPCE) are added.

The protocol layer contains a blockchain consensus algorithm. The Uranus chain will use DPOS+BFT technology to build a new incentive mechanism. At the same time, in order to motivate service providers to come up with more and better computing power containers to join the network, we have also added a Proof-of-Online time and Proof-of-Container mechanism to provide a better incentive mechanism.

The extension layer is the implements the main business logic. The operation of smart

contract, the support of DApp applications, the management and scheduling of containers, etc., are all carried out on this layer.

Uranus Chain Consensus Algorithm

The Uranus chain consensus module innovated a DPOS+BFT hybrid consensus algorithm based on DPOS (Delegated Proof of Stake) and with reference to the Tendermint algorithm with Byzantine fault tolerance. Under the DPOS mechanism, the influence of each stakeholder is directly proportional to its interests, and no stakeholders are excluded from attaining influence. Other consensus systems in the market exclude the participation of the vast majority of stakeholders because there are many different ways to replace stakeholders. The popular ones use invitation-only and bid-ranking mechanism. Some systems technically allow the participation of all stakeholders, but they are mostly ignored by the major stakeholders who produce large number of blocks. Only the DPOS mechanism can ensure that the blocks are evenly distributed to all stakeholders, and is economically viable for all stakeholders.

Tendermint is a high-performance consensus algorithm designed exclusively for blockchain. Based on the assumption of a semi-synchronous network, Tendermint can tolerate $1/3$ Byzantine nodes. The consensus nodes exchange voting information for each block through the peer-to-peer consensus message exchange protocol, rapidly forming a majority consensus. The voting result eventually will be recorded on the blocks.

The method we use is to randomly sequence the delegate lists on the current round (assuring that the sequence of the delegates on each round is different and the sequence of the delegates on the next round cannot be predicted), and then allow every delegate to create a block in a round-robin sequence. DPOS is currently a recognized and highly efficient consensus mechanism on the blockchain. The main drawback of this algorithm is that if a delegate node is betrayed, it may broadcast multiple inconsistent blocks. These blocks may include Double Spending, resulting in the entire network being forked. In the voting mechanism of DPOS, most users do not vote actively, and it is difficult to discover attacks on the network in time. In addition, because the DPOS mechanism lacks an economic punishment mechanism against attacks, it also results in it being cheaper to attack the network.

To solve this problem, we took reference from Tendermint's algorithm. Tendermint requires the validator to stake certain tokens. If an attack is confirmed, the deposit will be forfeited from the validator. Before Tendermint conducts the next round of voting, the validator will wait for a short time to receive a complete proposal block from the proposer. This dependence on timeout makes Tendermint a weak synchronization protocol rather than an asynchronous protocol. However, the rest of the protocol is asynchronous, and the validator will proceed with the next step only when more than $2/3$ of the validator set has been received.

Based on the above analysis, Uranus system DPOS+BFT consensus protocol workflow:

1) Validator registration

During registration, the validator needs to provide a deposit V_b , which must be greater than a minimum deposit parameter B_Min set by the system. At any time, the registrant can withdraw from the registration, but the deposit will have a lock-in period, which can generally be set to a month.

2) A stakeholder lends the deposit to the validator

Any stakeholder may lend his own tokens to a validator as a deposit. Assuming the deposit that the stakeholder lends the validator is L_b , then the total deposit held by the validator $V_t = V_b + L_b$

3) Stakeholders elect validators by voting

The stakeholders may vote to choose the validators. Suppose that the total number of tokens of voters is S_t and the validator's ranking weight is $W_t = S_t * V_t$. If the validator does not receive any votes, its weight will be $W_t = V_t$.

4) Election for qualified validators

Registered verification nodes have to maintain a table ranking the top 48 validators by weight (dynamic validation) and may participate in the competitive block generation. The validators in the table are called qualified validators.

5) Package Transaction of qualified validators

The candidate block header hash value H_n is generated using the parent block header hash value H_p , the current transaction Merkle root H_m , the timestamp T and a random number nonce, which is broadcasted as a proposed transaction. Each qualified validator can only issue one proposed transaction per round. The validating node that repeatedly sends a proposed transaction on the same round will be considered as an attack and the deposit will be forfeited, if it is reported. Unless the system does not reach a consensus within a specific period of time (consensus timeout parameters in the system), the qualified validator may not change the nonce to regenerate a new block hash value and propose a new transaction.

6) Confirmation of transaction information

After receiving the proposed transaction, each qualified validation node first checks whether the node that initiated the proposed transaction is in the list of qualified validators. If not, it will ignore this proposal. If it is a proposal initiated by a qualified validator, other validators will validate the transactions in this block. If there is any illegal or double-spend transaction in the block, the proposal will be ignored. Finally,

the validator will use H20, the last 20 digits of the hash value of the candidate block in the proposal to calculate the voting weight value $Vw = Wt/H20$, vote on the proposed transaction corresponding to the current maximum Vw , and concurrently sends the VoteBlock message. After the original proposer receives the VoteBlock messages from more than 2/3 validators, it will initiate a CommitRequest message. Each validation node will validate whether the VoteBlock messages corresponding to the CommitRequest message exceeds the 2/3 vote. If the vote exceeds 2/3, the CommitRequest will be carried out and the Commit message will be broadcast. Each qualified validation node will broadcast the NewHeight transactions after receiving the Commitrequest. After receiving a NewHeight transaction, each node will confirm the transaction and update their respective local blockchains. The NewHeight transaction also allows a new block to be confirmed and ensures that it cannot be forked.

The following three attributes are needed to measure the blockchain consensus algorithm:

- 1) Agreement: Agreement means that all the honest nodes consistently comes to a consensus.
- 2) Validity: Validity means that all transactions in the blockchain come from honest nodes.
- 3) Activity: Means that the consensus algorithm of the blockchain can always form a consistent decision without deadlock.

The Fish-Lynch-Patterson (FLP) Law proves that it is impossible to judge whether a process has crashed or message delayed in an asynchronous communication network environment. As long as there is an unreliable procedure, no consensus can be reached on all successful processes.

In an actual situation, it is generally necessary to make some assumptions in order to avoid the non-consensus limitation of FLP.

The Uranus consensus algorithm is assumed to exist in a weakly synchronized network environment. Among any selected 48 qualified validators, less than 1/3 are Byzantine nodes. We set out below the proof of agreement, validity and activity of the Uranus consensus algorithm under this assumption.

1) Proof of agreement

In any round of the consensus process, the Uranus blocks are generated by 48 qualified validators. The final confirmation of a block requires the votes of more than 2/3 of the 48 validators, including the broadcast of more than 2/3 of the VoteBlock and Commitrequest messages. If there is a fork, that means at least 1/3 or more of the honest validators have voted for two or more blocks for the same round (because in the most extreme case, it is assumed that 1/3 of the Byzantine nodes have voted for two nodes). It runs counter to the assumption that the honest nodes must execute the voting strictly

according to the agreement, so forks won't appear.

2) Proof of validity

Assuming that there is a block generated by a Byzantine validator in the Uranus blockchain, and the block contains one or more illegal and/or double-spend transaction, then at least $2/3$ of the validators have voted on this block in the VoteBlock and Commit stage. Assuming that $1/3$ of the Byzantine validators have voted on this block, another $1/3$ of the votes must come from non-Byzantine validators. Since we assume that non-Byzantine validators must vote according to the protocol rules, there is a contradiction, thereby ensuring that that Uranus's blockchain does not include illegal transactions.

3) Proof of activity

Since each new block requires $2/3$ of the votes for confirmation, there may be some extreme cases where the network is delayed, or less than $2/3$ of honest validators are online and consensus may not appear on certain consensus rounds. The system will set a time limit for each round of consensus. After the timeout, each validator will re-initiate a new block Proposal. If there are repeated situations in which the system cannot reach a consensus, the community needs to initiate a new round of voting to remove the offline validators and reselect validators.

Although Uranus cannot fully guarantee activity at all times, it accords to FLP's non-consensus law, and at the same time avoiding a failure in consensus through Uranus's governance mechanism.

The DPOS+BFT consensus algorithm of Uranus is an innovation based on the DPOS and Tendermint consensus algorithms of BitShares/EOS. It not only avoids the weak points of excessive concentration of validators in BitShares and EOS (101 validators in BitSares and 21 validators in EOS generate blocks in turn) and lack of measures for economic sanctions against illegal validators. Meanwhile it also avoids the unfairness in the Tendermint consensus algorithm. That is, the more the interests there are, the more likely the blocks are generated. The consensus algorithm of Uranus increases the randomness to random generation of hash values similar to Bitcoin, and reduces the probability of illegal transactions carried out by validators.

uraTiller module

The uraTiller module contains the Tiller virtual machine, Tiller native smart contracts, and interfaces to engage external data which logically links on-chain and off-chain data. uraTiller has a nested virtual machine model that realizes seamless integration between the Tiller virtual machine and the underlying virtual machine to ensure data security and efficient execution of complex business logic.

Because most of the components and modules of the Uranus system have realized the microservice and containerization technology, the communication among all microservices uses a message bus. The interaction between off-chain data and other modules may realize functional expansion by subscribing to the messages on the bus.

The uraTiller microservice module communicates with the Uranus engine and delivers business-related information to the Uranus engine. If the user's business information meets the default scheduling and allocation parameters of the system, the computing power container will execute it automatically. Otherwise, the system will guide the user to complete a transaction with a third-party supplier, establish a smart contract, and execute the user's special requirements and deployments.

Uranus Computing Power Engine

The Uranus computing power engine module is an infrastructure module used for large-scale scheduling, automatic deployment, extension and management of containers. The Uranus computing power engine module draws on the design concept of Borg and Kubernetes. It can:

- ★ Automatically deploy and reproduce containers
- ★ Flexible Scale-out
- ★ Manage containers across machines and organize better load balancing of containers
- ★ Dynamically upgrade application containers
- ★ Self-repairing capability

Uranus Smart Contract

Smart contract is a contract defined in digital form with the terms of the agreement between parties set out therein and must be executed by them. The smart contract system in the Uranus project are mainly used to achieve conditional limitation and matching of users and computing resources, definition of fault tolerance conditions, and event triggering.

On the Uranus platform, two types of smart contracts are deployed. One type is the explanatory smart contract similar to Ethereum, and the other type is the native smart contract in the uraTiller module. The native smart contract of uraTiller can access a variety of data sources beyond the blockchain, such as APIs, file systems, URLs, databases, other data sources, as well as the business logic completely irrelevant with chain. Explanatory smart contracts are only used to process user transaction information.

In the Uranus system, Oracle runs as a native smart contract. In this way, the contract can be directly converted into a machine code and executed, greatly improving the execution efficiency of the contract and satisfying the complex business logic scenarios in the cloud environment.

5.2.1 PoC (Proof of Contribution) Mechanism

Uranus uses a PoC algorithm to schedule the computing power containers entering the chain. Machines with a higher contribution value are more likely to be scheduled and

allocated to users in priority, thus earning more tokens.

Contribution Value Calculation Method:

Total contribution value of containers = basic contribution value + service contribution value + chain contribution value

Basic Contribution Value:

The basic contribution value is obtained through multi-dimensional scoring and statistics of the container's CPU computing power, uplink bandwidth, sharable storage space, actual available memory, effective online duration and other information.

CPU computing power is scored based on CPU logic frequency by gradient. The memory is scored based on the current effective remaining memory by gradient. The uplink bandwidth is scored based on the measured value by gradient. The available shared memory space is scored based on measured value by gradient.

Basic contribution value =

CPU gradient value * CPU coefficient + memory gradient * memory coefficient + bandwidth gradient * bandwidth coefficient + storage gradient * storage coefficient

The value of each coefficient (CPU coefficient, memory coefficient, etc.) is comprehensively considered by analyzing the current total volume of this resource in the network and the current usage amount of this resource in the network, which is analyzed and obtained by using Poisson distribution queuing algorithm. For the introduction to the algorithm section, please refer to: 4.2.2 - Mechanism for automatic adjustment of system resource allocation.

Service Contribution Value:

Service contribution value obtains a comprehensive score through statistical analysis of the container's past transaction records and user's feedback.

Chain Contribution Value:

Chain contribution value is obtained when a container's account makes a contribution to the chain. For example, the account actively participates in the voting of the entire network and maintains the stability of the chain's network.

The total contribution value of a single container is accumulated with the increase of its online presence. The absolute value is halved after passing through T blocks. The scheduling is realized through partition scheduling in the round robin algorithm. The secondary algorithm is similar to the system thread queue scheduling of the modern operating systems (Windows and Linux both use this algorithm and its modifications).

The advantage of this algorithm is that it allows better container resources to provide more opportunities for execution and a certain guarantee for the scheduling of containers with relatively low performance, so long as the time period is of sufficient length, it may still have a chance of being scheduled.

Due to space constraints, this section does not discuss the partition scheduling in the round robin algorithm. Interested readers may refer to related books.

5.2.2 Mechanism for Automatic Adjustment of System Resource Allocation

As user's demand is random and the resources in the system are relatively fixed for a specific period of time, a mechanism for automatic adjustment of system resource allocation based on the principle of negative feedback is needed in order to ensure the stable operation of the system. A resource allocation weight factor σ (Sigma) is used.

$$\text{Container score} = \text{cpu metering}^{\times\sigma_c} + \text{disk metering}^{\times\sigma_s} + \text{bandwidth metering}^{\times\sigma_b}$$

Where: σ_c , σ_s and σ_b are the weight factors of cpu, disk (storage) and bandwidth.

These factors are a real-time reflection of the extent to which system resources are being used. When the system is small, these factors are easily calculated. However, when the system exceeds a certain scale, the calculation of these factors not only consumes a large amount of resources, the actual values also cannot be calculated in real time. For this reason, the maximum likelihood parameter estimation method in probability and mathematical statistics is needed. These factors can be calculated relatively accurately with a small sample size.

Assuming that the unit measurement demand of CPU, the unit measurement demand of disks space and the unit measurement of bandwidth demand that appear in unit time are

X_c , X_s and X_b respectively, these

three unit measurement are all random variables that comply with Poisson Distribution when system resources are relatively stable. It is a very classic mathematical model of the queuing theory. After the system has been established and the sample size is large enough, a statistical test needs to be carried out on this assumption.

Below we will briefly introduce the Poisson Distribution to deduce the estimation methods

of these three factors σ_c , σ_s and σ_b .

It is assumed that x complies with the Poisson Distribution. As the value of X here is an integer, X complies with discrete Poisson Distribution, and the probability of $x=k$ ($k=0, 1, 2, 3, \dots, \infty$) is:

$$P(x=k) = \frac{\lambda^k}{k!} e^{-\lambda}, k=0, 1, 2, 3, \dots, \infty$$

Here, λ is an interesting parameter. It is both the expected value of the Poisson Distribution function and the variance of the Poisson Distribution function. The expected value of a random variable is the “maximum possible value” of a random variable, and the variance is the size of the spread range. The expected value and variance of the Poisson Distribution function can be calculated by the following formula (omitting the derivation process)

$$E(X) = \sum_{k=0}^{\infty} \left(\frac{\lambda^k}{k!} e^{-\lambda} \right) \cdot k = \lambda$$

Expected value

Variance

$$D(x) = E(x - \lambda)^2 = \lambda$$

In the above calculation process, it is assumed that λ is a known constant, but in actual application, it is an unknown parameter. Its value is estimated by using the maximum likelihood parameter estimation method in mathematical statistics and some samples. Below we will introduce the maximum likelihood estimation (omitting the derivation process)

Assuming that there is a set of sample observations from the population, the maximum likelihood function is

$$L(x_1, x_2, \dots, x_n, \lambda) = \prod_{i=1}^n \frac{\lambda^{x_i}}{x_i!} e^{-\lambda} = e^{-n\lambda} \prod_{i=1}^n \frac{\lambda^{x_i}}{x_i!}$$

$$\hat{\lambda} = \bar{x}$$

i.e., maximum likelihood estimator of λ

The above is just a simple theoretical explanation. In practical applications, whether the three factors mentioned above is in line with the Poisson Distribution needs to be tested. The sample size and time period of the test shall be calculated accurately. The maximum likelihood estimator is based on the sample size. In this sample size, the error

of the maximum likelihood estimator needs to be further calculated. It may be necessary to have an empirical correction factor in addition to the theoretical calculation. Whether this correction factor converges or not and whether the negative feedback adjustment mechanism is satisfied or not needs more detailed mathematical derivation. They are not described in this document.

5.3 Computing Power Container Engine (CPCE)

The distributed nodes of the Uranus system are an integral part of the global Uranus resource pool. They constitute the cornerstones of a distributed computing power resource pool that is able to compete with giant vendors of centralized cloud service (Amazon, Microsoft, Ali, etc.) and are also inadvertent suppliers of ubiquitous computing power. The core components of Uranus include computing power nodes, computing power containers, container Worker and Agent, and peripheral distributed auxiliary systems.

Uranus CPCE (Computing Power Container Engine) also runs on the nodes that make up the Uranus blockchain. The Uranus CPCE provides an application-oriented container cluster deployment and management system. The goal of Uranus CPCE is to eliminate the burden of scheduling the physical/virtual computing network, storage infrastructure and to allow the application operators and developers to fully focus on container-centered automation for self-service operation. Uranus CPCE also provides a stable and compatible foundation for building customized workflows and more advanced automated tasks. Uranus CPCE has perfect cluster management capabilities, including a multi-level security protection and access mechanism, a multi-tenant application support capability, a transparent service registration, service/failure discovery mechanism, a built-in load balancer, a self-repair capability, rolling system upgrade and online capacity expansion, a scalable resource auto-scheduling mechanism, and a multi-granular resource quota management capability. Uranus CPCE also provides perfect management tools covering development, deployment testing, operation and maintenance monitoring.

5.3.1 Computing Power Container Engine Architecture

The Uranus CPCE management engine draws on Borg's design concepts, such as Pod, Service, Labels, and single-Pod single IP. The overall architecture of the Uranus engine is very similar to that of Google Borg and Kubernetes, as shown in the following figure:

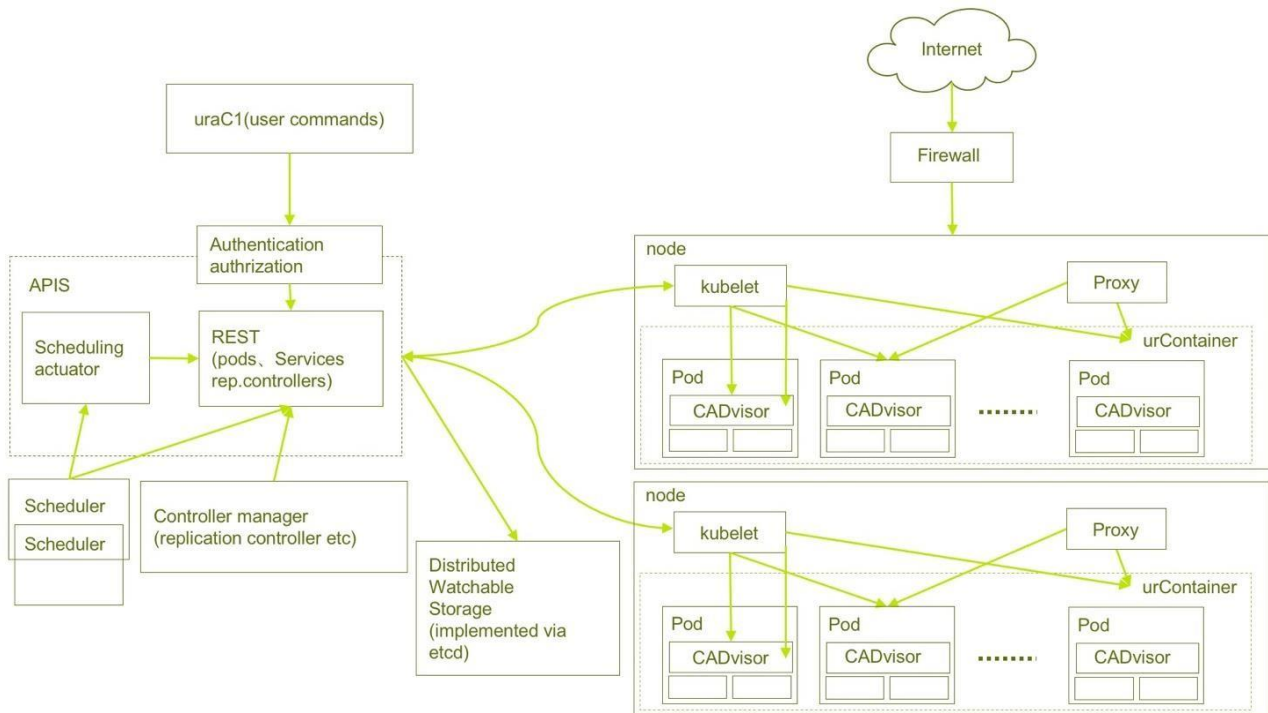


Figure 8 Uranus CPCE architecture

The Uranus CPCE engine comprises the following core components:

- ★ etcd saves the state of the entire cluster;
- ★ Apiserver provides a unique entry for resource operation and the mechanisms for authentication, authorization, access control, API registration and discovery, etc.;
- ★ The control manager is responsible for maintaining the status of the cluster, such as fault detection, automatic expansion, and rolling updates;
- ★ The scheduler is responsible for resource scheduling, and schedules PoD to the corresponding machines according to the predetermined scheduling strategy;
- ★ kubelet is responsible for maintaining the container's life cycle, and is also responsible for the management of Volume (CVI) and network (CNI);
- ★ Container runtime is responsible for image management and the operation of the Pod and containers (CRI);
- ★ kube-proxy is responsible for providing service discovery and load balancing within the cluster.

In addition to the core components, there are some recommended Add-ons:

- ★ kube-dns is responsible for providing DNS services for the entire cluster;
- ★ Ingress Controller provides external network access for services;
- ★ Heapster provides resource monitoring;

- ★ Dashboard provides GUI;
- ★ Federation provides clusters across the availability zones.

5.3.1.1. Uranus Resource Control Management

Through the addition of a separate Resource Controller, the Watch mechanism adopts a fully asynchronous non-blocking event-driven model. In the case of insufficient resources, an asynchronous resource request will be initiated to automatically handle the subsequent processes. Once the resources are ready, rescheduling will be triggered at once. During the request, the intermediate layer also prepares the resource pool in advance, initializes the Worker and prepares the installation steps in the container image in advance. The creation process is as follows:

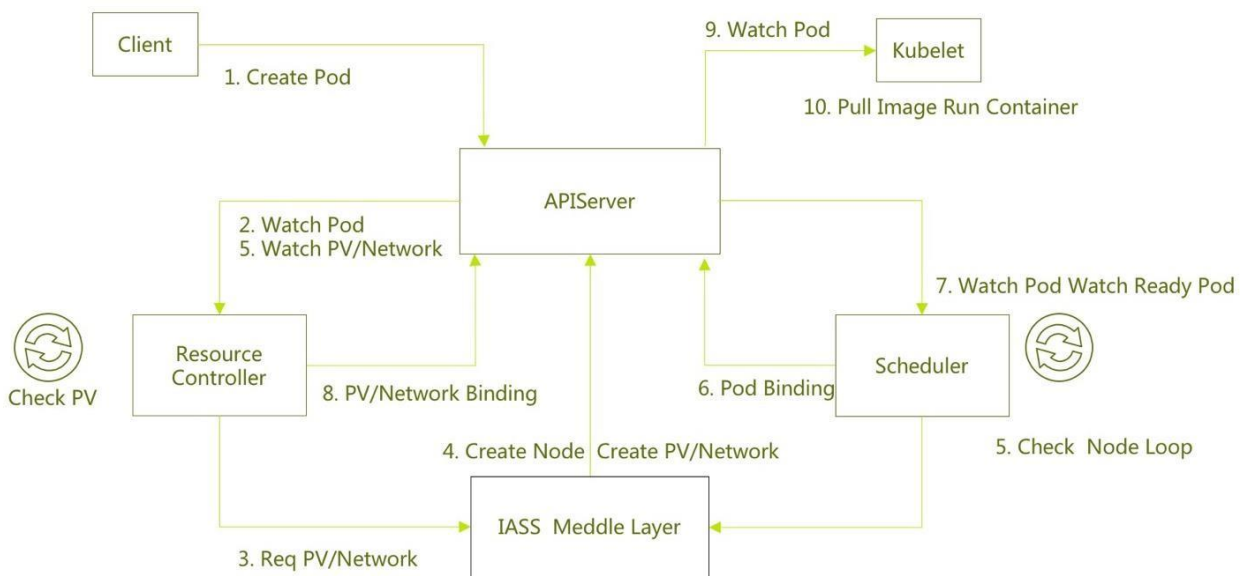


Figure 9 Uranus resource control management

Resource Controller manages all the resource-related underlying processes. The specific Pod creation steps are as follows:

- ★ The client requests the api server to create a Pod;
- ★ The Resource Controller watches a new Pod and checks whether PV and Network have been created. At the same time, should the scheduler find that the new Pod is not scheduled, it will try and schedule it;
- ★ Because the resources were not prepared in advance, the Resource Controller will not be able to find the PV initially and the Network that matched the Pod. It will then request the IAAS layer to create storage and network resources.

Scheduler will also requests the IAAS layer to create a container resource of the corresponding specifications because it is not able to find an available node. In this case, the Pod will not reenter the scheduling queue any more. The rescheduling will

be conducted only after everything is ready.

- ★ Generally, the IAAS layer creates resources slowly, and only provides asynchronous interfaces. PV, Network, and Node resources will be registered through the apiserver immediately after the underlying resources are ready.
- ★ When the ResourceController finds that both PV and Network are ready, it will bind them to Pod. After the Node applied for by Pod is registered, and PV and Network are bound, Pod will be set in a ResourceReady state.
- ★ When the Scheduler once again notes that the Pod is in a ResourceReady state, it will retrigger the scheduling process;
- ★ Successful Pod scheduling is bound to the dynamic creation of Node;
- ★ When the kubelet corresponding to Node watches that the newly scheduled Pod has not started yet, it will pull the image first and then start the container.

5.3.1.2. Multi-Region Scale Computing Power Container Management

When the scale of the computing power containers increases in size, applications and services will be numerous and deployment will be extensive. Basically, it will cover all the regions in the world. Moreover, the release is frequent and the iterative cycle is short. In this case, the deployment of the computing power container management system across multiple regions will reduce the management, operation and maintenance costs of each computing power pool.

In the management of computing power containers, cluster scheduling-related API is added to the Uranus engine to optimize the status management of sub-clusters, and allow better performance in the distributed federal coordination processing tasks, as shown in the following figure:

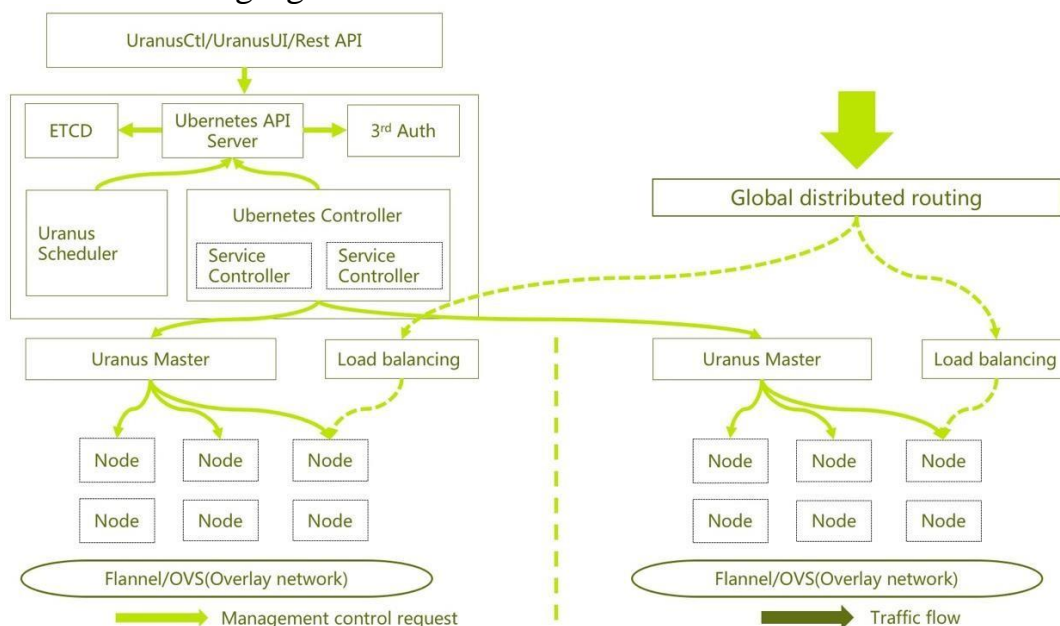


Figure 10 Uranus multi-region scale computing power management

In the original architecture, a Node manages a Worker node. Here, a Node manages multiple clusters. The Cluster Controller will maintain the health status of every cluster and monitor the load status, while the Service Controller will find cross-cluster services.

We further enhanced List-Watch which improves asynchronous message delivery in distributed systems and greatly improves key performance indicators such as system performance and data agreement.

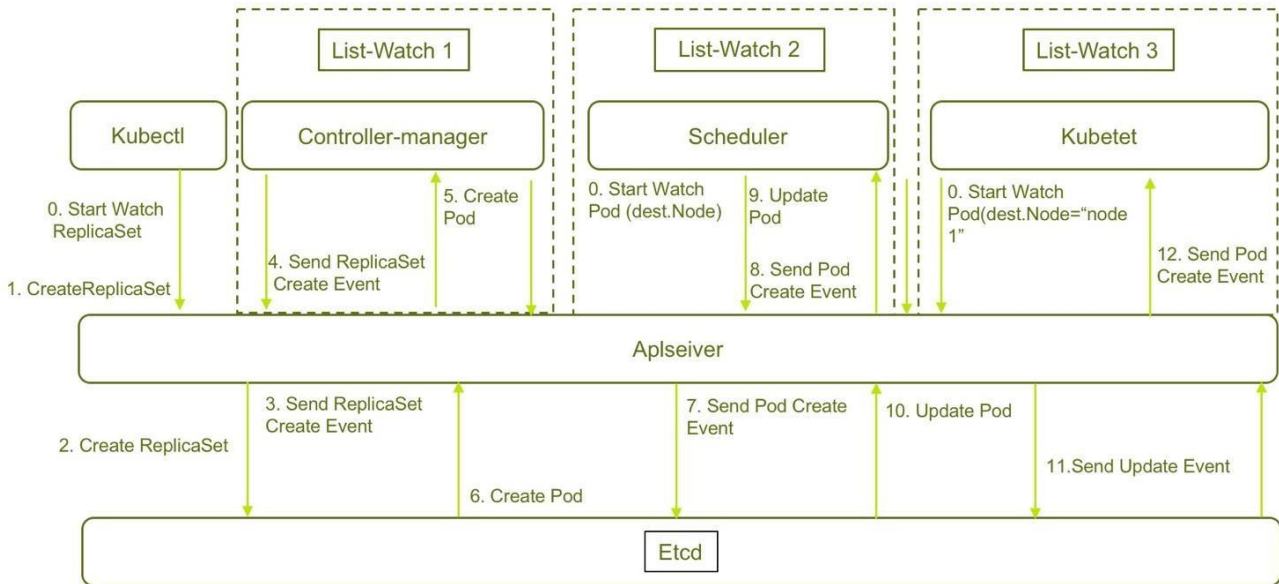


Figure 11 Uranus Controller cross-node scheduling management

In general, the cluster management architectures are similar across all mechanism. They all have Controllers, Schedulers, and Agents. Here, List-Watch mechanism is used to realize the decoupling of interactions among components and embodies the design concept of Everything talk to API. It is similar to the message bus in SOA, but its difference from the message bus is the unified processing of data storage and event notification. While receiving a new event, the cluster may get this new data, including whether the data change is additional or just an update of the object.

From the above process, it can be seen that each component achieves natural decoupling through List-Watch of different objects (even if it is a same object, the state is different). Each component handles different phases of an application life cycle, and ensures the sequence of processing flow among multiple components in an asynchronous distributed system. At the same time, List-Watch only monitors events, so every time the Watch creates a pod, for example, it acquires incremental information. The number of interactions acquired is very small. There is a possibility of loss when a message/event is delivered via the message bus. This requires a fault-tolerant mechanism to guarantee the eventual agreement. The engine solves this issue by a producing a List. For example, the ReplicaSet Controller will periodically provide a full-amount List for ReplicaSet. The acquired data is the application data that the current system needs. It will check the current running status of each application, carry out the corresponding processing, delete the redundancies and make up the insufficient.

5.3.1.3. Computing Power Container Service Discovery and Deployment

The standard plug-in in the Uranus engine: kube-dns may provide DNS service inside the cluster and accesses Uranus engine service through the resolution of service names by DNS. The Uranus engine service is made up of a set of PODs, which are containerized applications. These PODs previously used load balancers. If there is a Worker cluster which has an internal service called Mysql composed of a group of Mysql PODs, other applications in this cluster can access this Mysql service via DNS, and the PODs can resolve services across clusters transparently.

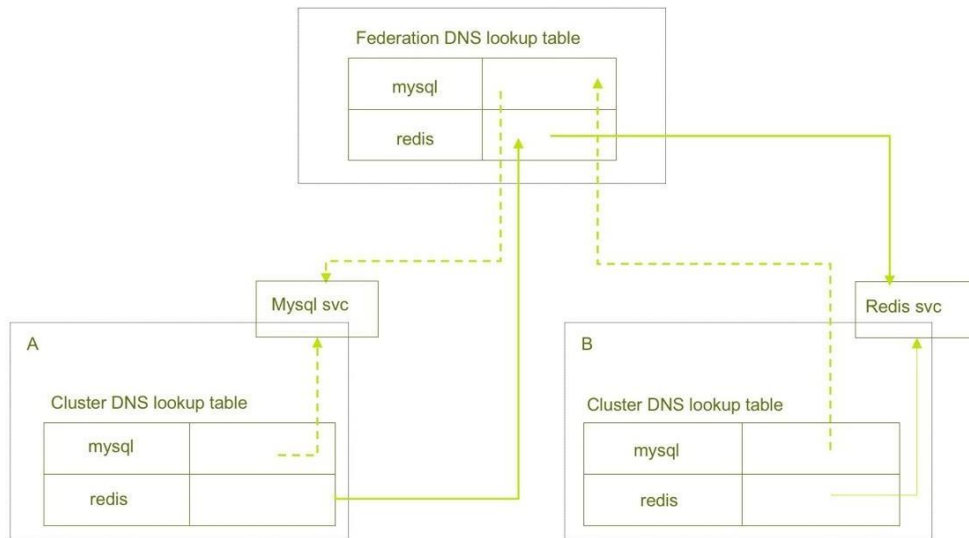


Figure 12 Uranus computing power container discovery and deployment

Deployment example:

Through the automatic scheduling of a Node, the computing power container is allocated to four different Worker machines through HAProxy.

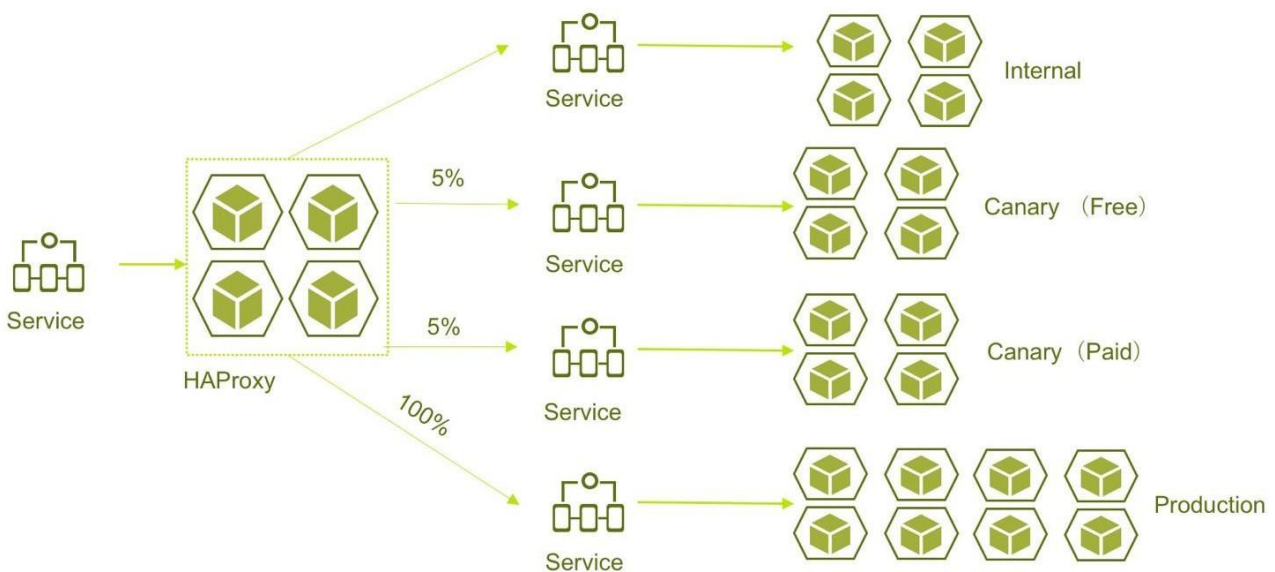


Figure 13 Single-node computing power container scheduling management

Two internal Pods, four free Pods, four paid Pods, and eight production Pods are allocated to each microservice, plus four HAProxy Pods.

A typical service yaml file is as follows:

```
apiVersion: v1
kind: Service
metadata:
  name: account-service-production
  labels:
app: account-service- production
  tier: service
  lb: private
spec:
  ports:
    - port: 8080
      name: http
      targetPort: 8080
      protocol: TCP
  selector:
    app: account-service
    tier: service
    track: production
```

The API gateway in the system will allocate an X-Traffic-Group identifier to each user request header. According to this identifier, HAProxy will route user requests to different deployment environments. When the scale of production is expanded, monitoring sometimes cannot effectively track the status of each deployment and the excess or inadequacy of the number of the copies of each pod. The system will automatically monitor and manage various deployment status of the operation. In addition, the automatic health check tool will check whether the HAProxy related to this service is healthy or not and check how many copies each Pod contains. It will also check that the four deployments behind the HAProxy (internal, free, paid, production) have at least two valid endpoints. If any error occurs, the system will automatically send an alarm notice via email.

5.3.1.4. Affinity and AntiAffinity in Computing Power Container Scheduling

Due to the characteristics of the distributed peer-to-peer system, the Uranus engine enhances Pod scheduling, involving multiple-scheduler configuration changes, node Affinity /AntiAffinity characteristics, Pod Affinity/AntiAffinity characteristics, stain and tolerance characteristics, and node problem reporting characteristics.

Affinity and AntiAffinity are runtime scheduling policies, mainly including nodeAffinity (HostAffinity), PodAffinity and PodAntiAffinity.

- ★ **NodeAffinity:** It is used to specify which node a Pod can be deployed on or which nodes it cannot be deployed on, and solve the problems of Pod and host.
- ★ **PodAffinity:** It is used to specify which pods the Pod can be deployed on together under a same topological structure.
- ★ **PodAntiAffinity:** PodAntiAffinity: It is used to specify which pods the Pod cannot be deployed together under a same topological structure and to solve the relationship between Pod and Pod together with PodAffinity.

There are three rules that can be set when these three types of Affinity is used:

- ★ **RequiredSchedulingRequiredExecution:** This rule indicates that the corresponding Affinity rule must be satisfied during the first scheduling. If there isn't any node that meets the requirements, no scheduling will be conducted. If the corresponding Affinity rule is no longer satisfied during the operation of the Pod, rescheduling will be conducted.
- ★ **RequiredSchedulingIgnoredExecution:** This rule indicates that the corresponding Affinity rule must be satisfied during the first scheduling. If there isn't any node that meets the requirements, no scheduling will be conducted. In the subsequent operation of the Pod, whether these rules are satisfied or not will not be checked.
- ★ **PreferedSchedulingIgnoredExecution:** This rule indicates that the corresponding Affinity rule should be satisfied as much as possible during the first scheduling. If there isn't any node that meets the requirements, scheduling will be conducted, too. In the subsequent operation of the Pod, such check will not be conducted.

During operation of computing power container across multiple physical domains, containerization and microservices may encounter problems with Affinity and AntiAffinity. Originally, multiple components may be installed on one virtual machine where there will be communication between processes. However, the containers are often split directly based on processes during the containerized split. For example, business processes are put in a container, monitoring log processing or local data is put in another container, which have independent lifecycles. In this case, if they are distributed to two farther points in the network, the request will be forwarded many times and the performance will be poor. Therefore, the deployment of Affinity will enhance the network capacity and reduce the loss network uptime through seeking the

nearest routing for communication. AntiAffinity is used mainly for its reliability to maximally disperse instances so that when a node is failed, only one instance or one tenth of the instances are affected.

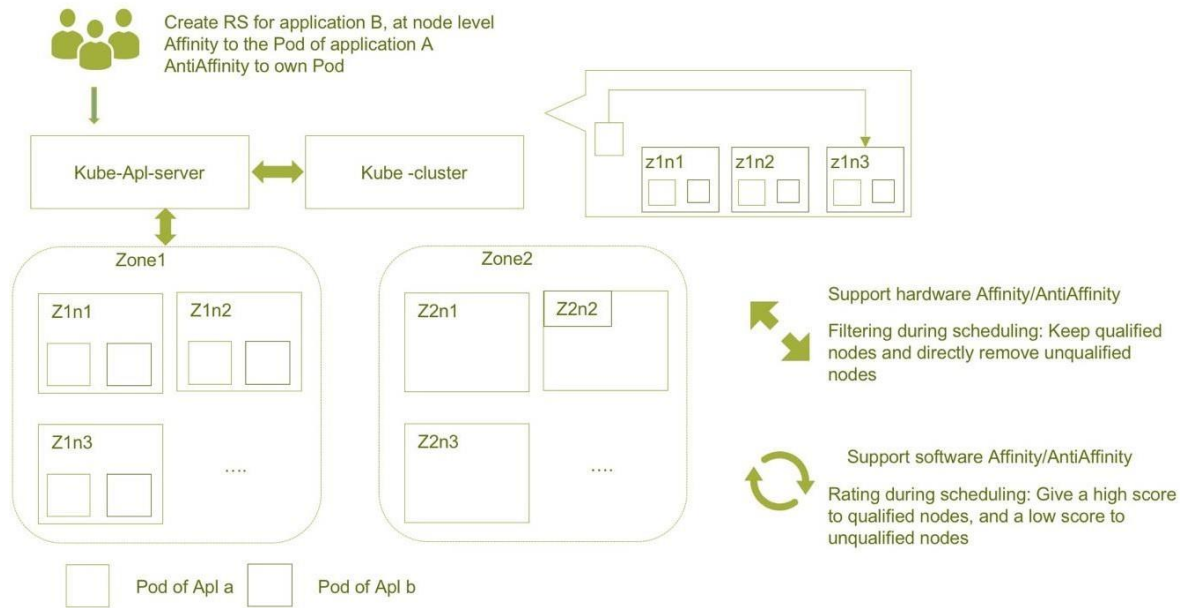


Figure 14 Affinity strategy of uraContainer

In the support of Affinity and AntiAffinity, both rigid and flexible supports (as shown in the figure) will be deployed as the sole use of rigid supports under improper configuration may easily lead to the failure of application deployment. In essence, there are two algorithms with realization logics that are quite similar. The hard algorithm filters out all the unacceptable nodes to ensure that the remaining nodes meet the conditions. In the flexible algorithm, these harsh requirements are not needed, and only the best effort is required. If the conditions are not met, it is hoped that the application is successfully scheduled and deployed. Here, we adopt sequencing by score. The ones with a high degree of compliance with Affinity and AntiAffinity will be given a high score and vice versa. Workers are then selected in a downward order of scores.

5.3.1.5. Workload Migration and Fault Recovery of Computing Power Containers

We use uraContainer to realize the pooling of distributed computing resources and the standardized measurement of virtual computing resources. In the space of a computing power container, various enterprise-level mainstream applications can be widely supported. When a single computing power container on a single pod triggers the standard health threshold of computing power, Uranus will automatically help it to migrate from the current Pod or the optimal Pod that complies with the Affinity algorithm to the target healthy Pod, ensuring that business applications always can provide efficient and stable service capabilities. At the same time, in a complex distributed network, the failure of physical hardware and interruption of the network

makes the interruption of any Uranus node unavoidable. However, the built-in self-healing mechanism for workload failures in the Uranus system will automatically help the user to select a new optimum target computing power container from the Uranus computing power pool in the shortest possible time in order to restore the business and ensure execution of tasks. The following figure shows an example for workload migration and fault recovery of the uraContainer:

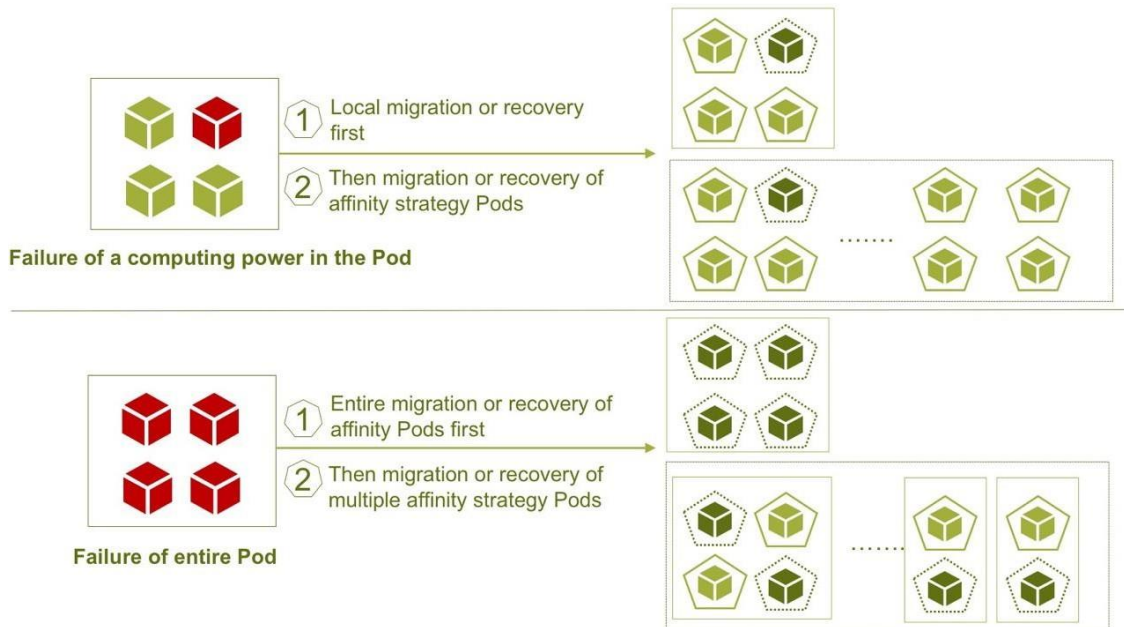


Figure 15 Workload migration and fault recovery of uraContainer

5.3.2 Computing Power Container uraContainer

Although container technology has many advantages, some outstanding basic security problems need to be solved. For example, during operation, the containers can hardly guarantee complete isolation between them. Based on Pouch technology and Kata Containers, we have created a more powerful uraContainer that is suitable for a ubiquitous computing environment.

We know that Pouch is a lightweight container technology. It features high speed, high efficiency, high portability and low resource consumption. It helped Alibaba achieve faster internal business delivery, and raised the utilization rate of physical resources of the data center on a super large scale at the same time, also. In addition, Hyper and Intel Clear Linux are also exploring the combination of VM and Container technologies to enhance isolation between containers while minimizing performance loss. They have jointly launched Kata Containers.

Our enhanced uraContainer is as shown below:

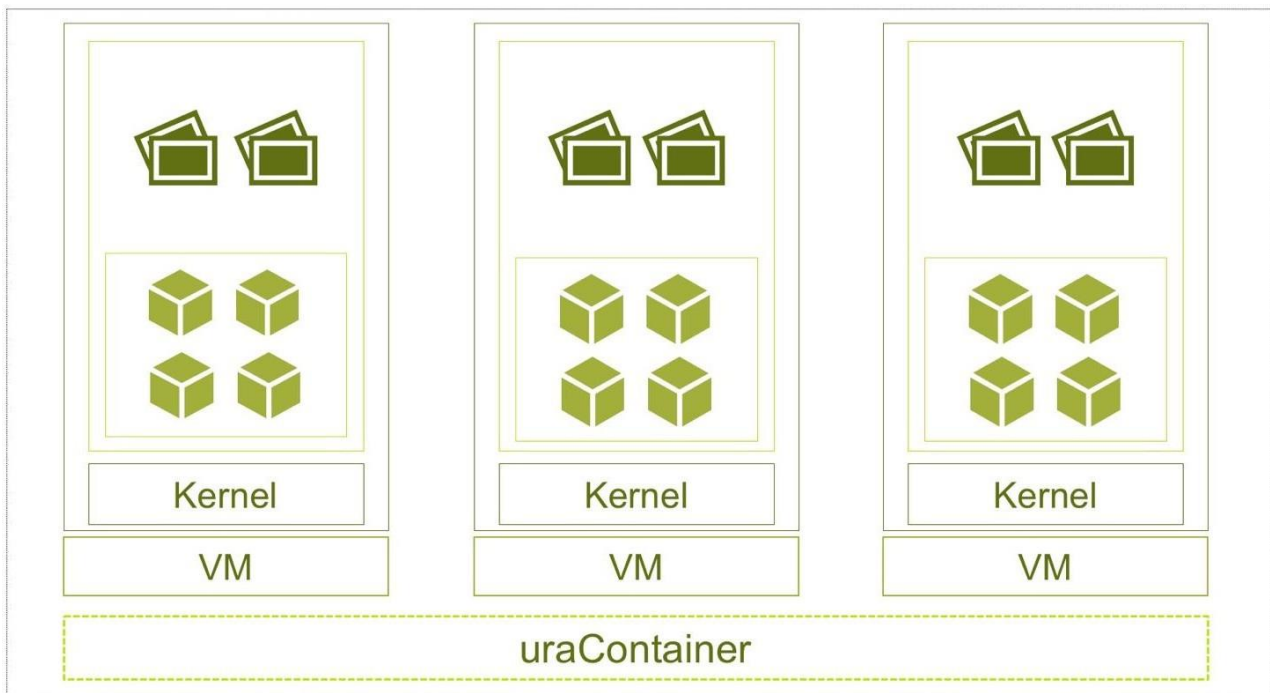


Figure 16 Architecture of the uraContainer system

Technically, the essence of uraContainer is a container running on a hypervisor. Based on the customized kernel, it adds SR-IOV, multi-queue management and other features. uraContainer allows users to achieve both the security of a virtual machine and the high efficiency and manageability of container technology. uraContainer can mask hardware differences, achieve strong isolation, is compatible with the OCI specification and Kubernetes container operation standards, lightweight, and with fast startup capability while supporting standard image formats.

5.3.3 Worker and Agent Design

In the Uranus architecture, the Uranus engine will contact the Worker host, deploy containers, manage images and schedule tasks. Worker hosts are run by Pods, either physical or virtual. In order to manage Pods, each Worker node should at least run uraContainer, agent and proxy services.

In the Uranus engine architecture, Pods must be used to manage containers. Each Pod may contain one or more container. Pod is a set of closely related containers. They share PID, IPC, Network and UTS namespace. They are the basic units for the scheduling of the Uranus engine. Pod is designed to support multiple containers to share network and file systems in a Pod. Services can be accomplished through a simple and efficient way of inter-process communication and file sharing.

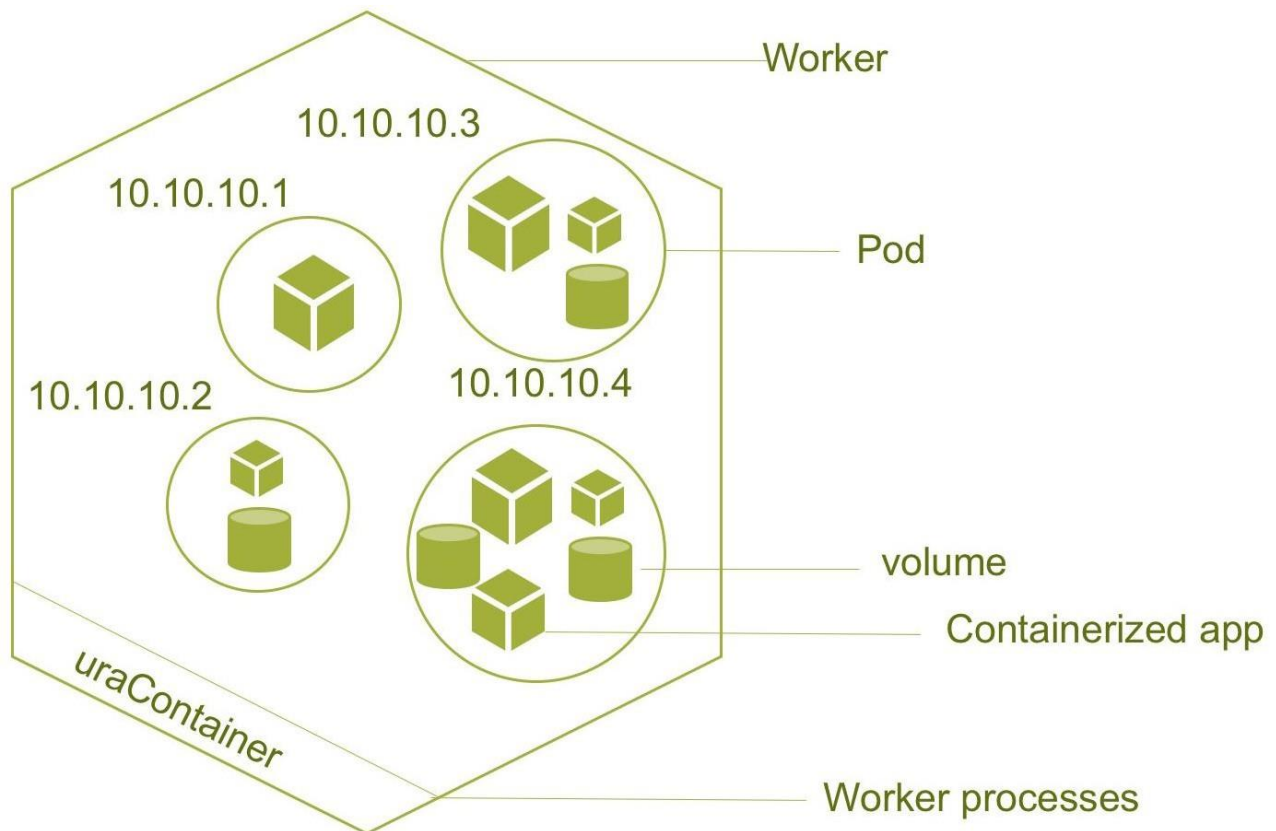


Figure 17 uraContainer Work and Agent architecture

5.3.4 Image and P2P Network Distribution

Although container technology has brought many benefits, the inefficient distribution of container images has posed a great challenge to us. We know that in a distributed computing environment, the cloud native applications of users will grow explosively. Under tens of thousands of deployment scenarios, the sharing of files and container images will still be unable to meet actual requirements of applications. Fortunately, based on the above scenarios, we have developed Beefly, a large-scale distribution tool. Beefly is a universal file distribution system based on CDN and intelligent P2SP technology. This solves the problem of time consuming distribution, low success rate and wastage of bandwidth when carrying out large-scale file distribution scenarios when combined with the IPFS technology, greatly improving the ability to publish and deploy container images.

Beefly combines various innovative technologies such as smart compression and intelligent flow control to solve various document distribution issues when encountering large-scale file downloading and cross-network isolation. This will improve data preheating and other service capabilities, as shown in the figure below:

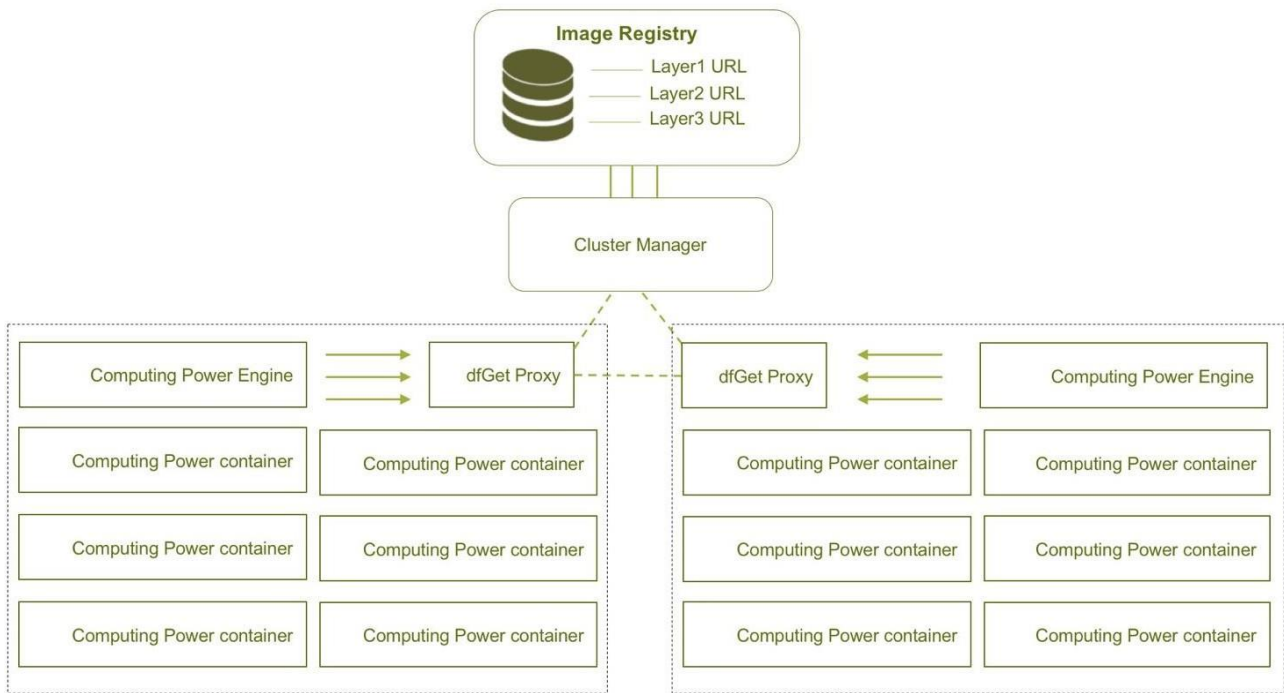


Figure 18 uraContainer image and P2P distribution

At the same time, Beefly has done some work in safe transmission. When sensitive files (such as key files or account data files) are being downloaded, the security of the transmission must be effectively guaranteed. In this regard, Beefly primarily does two jobs: supports the header data carrying HTTP to satisfy the file sources that need to be authenticated through the header and uses a symmetric encryption algorithm to encrypt the file content during transmission.

6 Application Scenarios and Ecology System

6.1 Application Scenarios

The Uranus platform is able to perform some large projects based on distributed computing, such as the famous Volunteer Project. In addition, because it can automatically complete computing tasks in real time, it is more suitable for small programs, and distributed users with ad-hoc demand at multiple nodes. Once a user accesses the Uranus system, no further commercial contract is required. The user only needs to transact directly with the computing power contributors when a demand is generated. The transaction is completed automatically. The demander only needs to download the application to be run (if it has been downloaded, no need to download it again) and issue a running command. The system will automatically search the nodes that can contribute the corresponding computing power at this moment and run the application after the initial transaction. After the operation is completed, it will return the result and completed settlement automatically. That is in the light of Mark Waiser's spirit of Calm Technology.

The Uranus project empowers computing services through blockchain so that the resources can be used for specific purposes, and execute related tasks. Here is a list of market scenarios:

Distributed implementation of general-purpose computing tasks

- ★ DNA computing
- ★ Computing of planetary orbits
- ★ Analysis of weather data

Projects based on edge computing

- ★ IoT node management and deployment
- ★ Voting platform
- ★ Advertising
- ★ Games
- ★ Data acquisition

Application Ecology (Block as a Service)

- ★ One-click deployment
- ★ Container image supermarket and App store

Distributed implementation of general-purpose computing tasks

Scientists from America previously organized a project called “Volunteer Computing.” They were trying to break down large-scale general-purpose computing tasks, such as protein structure analysis, planetary orbit computing, and large-scale weather data analysis, into a large number of tasks on idle private computing terminals through the distributed computing structure. However, due to the lack of an effective value delivery mechanism, these projects started piling up. Based on the public blockchain, the Uranus project offers a good solution to the value delivery mechanism and is able to achieve the distributed implementation of general-purpose computing tasks.

Projects Based on Edge Computing

The conventional cloud computing model can issue computing instructions and give response to results based on centralized operations. This model can satisfy many of the application scenarios, but for other applications that need real-time response within milliseconds, it cannot fulfil the application requirements. This is especially so if the existing cloud computing model is used to transmit data to cloud terminals thousands of miles away via a lagging and unstable network. If the network, computing, storage, application and data are close to the devices or the data sources are integrated, then the nearest edge computing is used and the response result is returned more quickly. This method is more suitable for these applications that require real-time computing, as shown in the figure below:

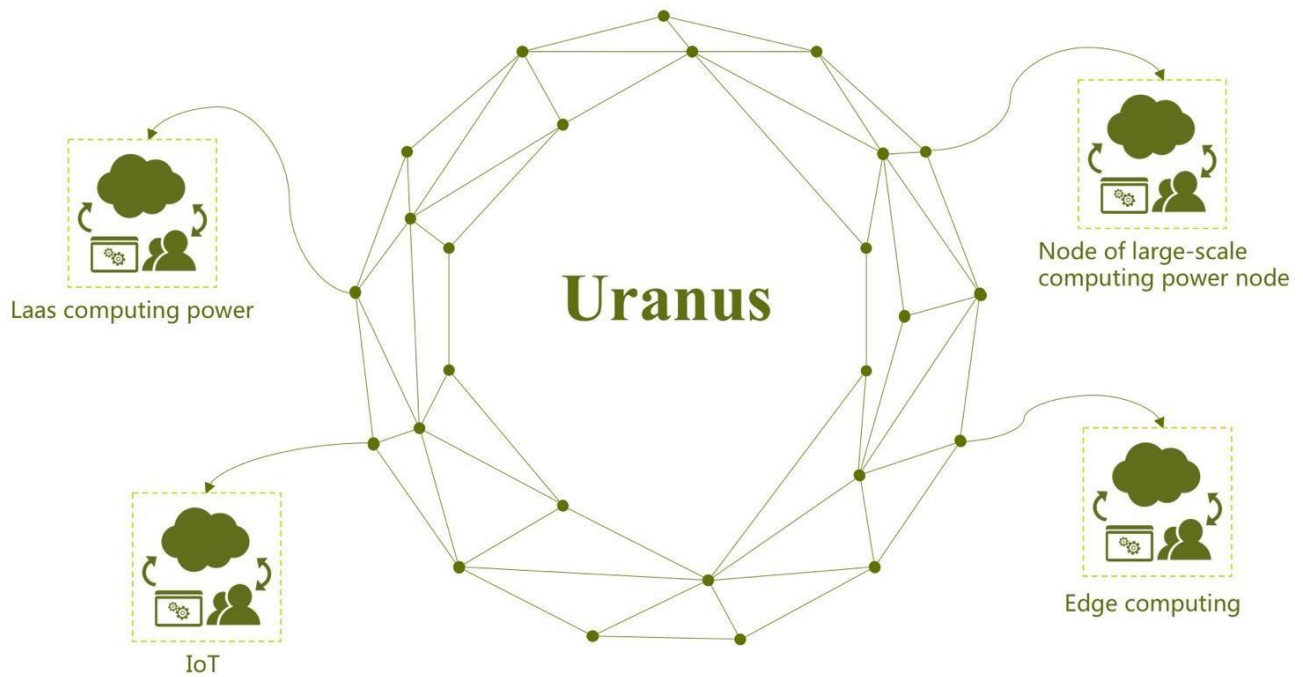


Figure 19 Application scenarios in the category of edge computing

In addition, some computing based on distributed nodes, such as IoT node management, voting platform, games, and advertising, needs to be completed through the architecture of distributed computing and edge computing and not the conventional public cloud based on a centralized structure.

Application Ecology

The Uranus system actually provides a Block as a Service (Baas). It provides one-click deployment for various chains and applications. On this distributed container architecture based on a public blockchain, there are container image and application stores provided, thus building an ecosystem that has thousands of computing contributors and users. Using the public blockchain and distributed computing enables application developers and application users to cooperate with one another.

6.2 Business Application Examples

To explain how the Uranus platform meets the needs of specific commercial applications, here are a few examples for network data acquisition, voting, games, and IoT terminals.

Network Data Acquisition

Data is continuously generated on the Internet, and it is an important source of business, technologies and technology innovation. Data on the Internet is automatically acquired (grabbed), processed and used to prevent potential security risks, thus improving user experience and enhancing business value. The most popular method is to develop an automated program that requests data from web servers (usually HTML forms or other web files are used), then resolves the data and extracts the needed information. The most common problems encountered during network data acquisition are: forms are rejected by the server after submission, and the IP address is blocked by the website and inaccessible. Although we can temporarily solve the problem of network data acquisition through proxy servers or variable IP technologies, we still have to face the issue of achieving mass commercialized and sustainable data acquisition from multiple automated network nodes. Uranus network links many devices from all over the world. Owing to its breadth and comprehensiveness of resources, Uranus is the best carrier for global ubiquitous network data acquisition.

Voting Platform

With the rapid development of Internet, online voting has been widely used in various businesses. To ensure the fairness of voting and eliminate paid posters, paid votes and other illegal activities, the voting systems usually adopt using technologies such as random ID, multiple verification codes, limited IP address segment, and limited voting start time..Some teams with strong overall strength, for example with more than 10,000 voting members and own software development technicians, often develop their own voting platforms so that the operators can publish tasks on the platforms and the voters can receive tasks on the platforms. This mode saves a lot of manpower and material resources, but the threshold of the independent IP and voting terminal of each user required by the voting platforms cannot be exceeded.

The Uranus network links many computing devices around the world. Every computing node terminal has an independent IP and is used as a qualified computing terminal node. The Uranus computing node is an ideal carrier for this application scenario. The platform resources are flexible, economical, safe and only requires payment upon usage.

Games

Data shows that the revenue of the global game market reached USD 108.9 billion in 2017, including USD 27.5 billion from China region, and the number of digitalgame

users exceeded 600 million, representing an increase of 17.7% year-on-year.

Behind the rapid growth of the gaming industry are hundreds of millions of users and thousands of gaming server nodes. The ability to support the operation of computing nodes for large-scale games is the key to this growth. In addition, the ability to rapidly expand, meet the demand for computing power and rationally utilizing the computing nodes at off-peak hours will become the key factor to game operators profitability .

IoT Terminals

Market research firm IDC predicts that by 2020, there will be more than 50 billion terminals and devices hooked to the Internet. In the future, more than 50% of the data needs to be analyzed, processed, and stored on the edge of the network.

For example, if you buy a new smart door lock, you need to rely on a home gateway for biometric processing, and a harmful gas detector that is processed locally on the home gateway.

The core capability of the public cloud is the computing power based on data centers, but in the era of the “Internet of Everything”, this capability needs to be seamlessly extended to the edge computing that is closer to terminals, forming center and edge synergistic computing.

6.3 Container Image and Application Ecology System

The Uranus platform is establishing an expanded public blockchain and optimized container scheduling technology to seamlessly link and integrate redundant computing power and provide efficient, cost-effective and decentralized computing services for numerous computing contributors and users. We are building a computing power sharing ecosystem that is owned, managed and shared by the above players, thus reshaping the pattern of the IT computing power market and the production relationship between resource providers and users. Here is an example of the container image market ecology and the application market ecology.



Figure 20 Composition of the computing power sharing ecosystem

Container Image Ecology

There are numerous resource contributors, resource users and platform application developers on the Uranus platform. With the deployment of numerous computing nodes, the joining of platform contributors and users, and the continuous innovation of application developers, the world's largest container image market with the most comprehensive coverage of industries and technologies is automatically formed. Based on this ecology, the platform resource contributors may not only contribute their redundant computing power but also upload their users' images to the platform by a single click to harvest gains. The users are freed from the traditional complex, inefficient application environment deployment and configuration. Instead, they can better focus on business innovation. The platform developers also benefit from multiple parties. A benign, sustainable, and automatically evolving container image market continues to provide efficient support for social operations and reduce overall costs, creating value for the contributors, users, developers, and other parties participating in the platform ecosystem.

Ecology of the Application Market

With the rapid development of blockchain technology and the deepening of the applications of the computing power sharing platform, thousands of resource contributors and computing users are able to access distributed resources. A larger market size also attracts more application developers, leading to more innovative platform applications uploaded to the platform. The intensive effects of platform ecosystem participants greatly increases the application value of the platform and the benign development of platform ecosystem, resulting in geometric

growth of the applications, users and transactions. The platform contributors and users also benefit from an efficient, cost-effective and decentralized ecosystem. More developers are attracted to migrate to this platform, thanks to the open protocols, clear beneficial distribution mechanism, large base of accessible users, and extremely low cost of computing and bandwidth resources.

7 Development Roadmap

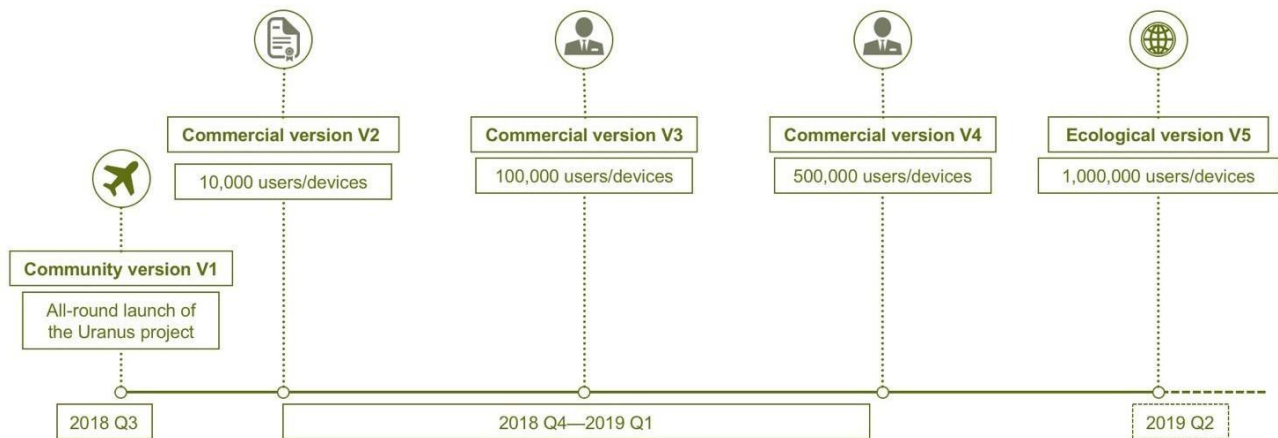


Figure 21 Uranus development roadmap

Phase 1: Community Version (V1)

2018/Q3: Completed close beta test and released a community version; users can register their own computing devices and use a small number of verified applications.

Phase 2-Phase 4: Commercial Version (V2, V3, V4)

2018/Q4-2019/Q1: Released three commercial versions V2, V3 and V4. In the corresponding development phase, the number of platform users rapidly increased to the orders of 10,000, 100,000 and 500,000. Following the continuous increase of the commercial level of the computing resource pool, the covered industrial and application scenarios are further enriched and the resource contributors obtained more revenue.

Phase 5: Ecological Version (V5)

2019/Q2: Based on the scale of the platform contributors and users, the value of the platform is becoming increasingly prominent. In this phase, an ecological strategy will be fully launched to allow more industries and more collaborators to participate in the application creation and contribution, gain benefits, promote the automatic operation and sustainable development of the platform and form a win-win ecosystem.

8 Team Members

8.1 Executive Team Chief

Executive James Jiang

- ★ Master of Mathematics, Texas State University / Master of Operational Research, Nanjing University of Science and Technology;
- ★ Senior Researcher at Bell Northern Research, US, owning six US invention patents
- ★ Vice chair of the U.S. National Communications Standards Organization TR45.5.2;
- ★ Established ZTE's subsidiary Shenzhen ZTE Integrated Telecom Ltd./ Shenzhen ZTE Mobile Telecom Co. Ltd and serving as General Manager and Board Chairman;
- ★ Vice President of a subsidiary of VIA Group, and Assistant to the President of the Group;
- ★ Established an enterprise-level desktop cloud company, leading the Chinese market.



Chief Architect Halley Han

- ★ Master of Automation, China Agricultural University;
- ★ Expert at virtualization and cloud computing, Linux kernel engineer and expert at open-source technology;
- ★ Expert at distributed virtual computing security delivery and audio and video algorithms, serving clients including the United States Department of Defense, Federal Ministry of Defence (Germany), the Turkish Navy and Citibank.
- ★ Co-founder & CTO of Beijing Aner Kechuang Information Technology Co., Ltd.;
- ★ Linux leader and architect of Wyse Technology LLC;
- ★ Chief System Architect of ISoft Infrastructure Software Co., Ltd.;
- ★ Established an enterprise-level desktop cloud company and led the technical architecture and product development.



Assistant to the President and Chief Operating Officer **Kevin Wang**

- ★ Deputy Manager, CSS Department, Taiyuan Coca-Cola Beverage Co., Ltd.
- ★ Alibaba B2B Business Unit for the sale of the Iron Army
- ★ Midea's domestic marketing company e-commerce business division (Jingdong & Tmall)
- ★ Yumi Network National Operations Director
- ★ COO of New Foods Division of New Hope Liuhe Co., Ltd.
- ★ Xinyuan China's Beijing Love Relay Technology Development Co., Ltd. Operation VP
- ★ Co-founder and COO of Beijing Xiaolu Express Consulting Co., Ltd.
- ★ Chain Smart Technology (Shanghai) Co., Ltd. Operation VP



8.2 Technical Advisers

Chief Container Expert **Dr. Liang Sheng**

- ★ Master of Computer Science, the University of Science and Technology of China;
- ★ Ph.D. in Computer Science, Yale University;
- ★ CTO of Citrix CloudStack in 2011-2014;
- ★ Founder & CEO of CloudStack, acquired by Citrix in July 2011;
- ★ SEVEN Networks VPE;
- ★ Software Engineering Director of OPENWARE;
- ★ Founder and CTO of Teros Networks;
- ★ Staff Engineer of Sun Microsystems Inc and a main contributor to java virtual machines.



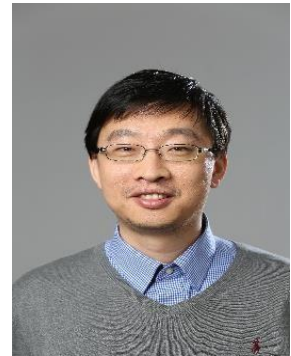
Chief Scientist Liren Chen

- ★ Master of Computer Science, Carnegie Mellon University / Master of Computer Science, Tsinghua University;
- ★ Co-founder and CTO of Hubat, researched and developed a search engine for the entire network;
- ★ Worked for Google for seven years, mainly engaged in the R&D of search in Chinese, Japanese and Korean languages, big data and large-scale systems, academic and legal search (Google Analytics, Google+, SRE, etc.);
- ★ The startup company vivisimo was acquired by IBM in 2011;
- ★ The Principal of 360 Mobile Search, and CTO of Pangu Search;
- ★ The startup company yunyun.com was acquired by Sina in 2013;
- ★ The startup company Passion Technology was acquired by Meituan in 2016;
- ★ Co-founder & CTO of an AI and blockchain company.



Chief Large Scale Resource Platform Management Advisor Dr. Yang Haiming

- ★ Master of Electronic Engineering, Tsinghua University;
- ★ Ph.D. in Electrical Engineering, Rensselaer Polytechnic Institute (RPI), United States;
- ★ Worked for Cisco for five years, and was responsible for the R&D and O&M of the unified resource management platform for Cisco Global Data Center;
- ★ Chief Architect in Cloud Computing and OpenStack Technology, IBM Greater China. Was responsible for the R&D of IBM's OpenStack technology in China, IBM cloud platform IaaS, and the implementation of end-to-end technology integrated with business;
- ★ Chief Architect of CTO System of Jingdong Group. Was responsible for the technical strategies of Jingdong Group, R&D of blockchain underlying technology, and technology integration.



Chief Financial Technology Advisor Shao Zhou

- ★ Certified software architect and information security and risk management expert of TOGAF;
- ★ A pioneer of IoT technology;
- ★ An explorer of blockchain technology and its commercial application;
- ★ An active organizer and contributor of open-source technology communities;
- ★ Chief Financial Technology Architect of AIIB Headquarters;
- ★ Have served many Fortune 500 companies, and has more than 10 years of experience in the international financial technology industry as well as rich experience in building and managing efficient development O&M teams involving investment, asset management and insurance sectors. Has also led and developed several large trans-regional cross-business line complex projects.



8.3 Product Development

CPO Li Fei

- ★ Founder of China Virtualization Technology Forum;
- ★ Chief Editor of “China Desktop Cloud Standardization White Paper”;
- ★ Technical Manager of Realtor Information & Technology Co., Ltd. /Beijing Gintel Technology Co., Ltd. in North China;
- ★ Virtual Product Line Product Manager of ISoft Infrastructure Software Co., Ltd.;
- ★ CPO of an enterprise-level cloud computing/desktop cloud company.



Software Architecture and Algorithm Expert Li Mingbo

- ★ Master of Electronic Information Engineering, Communication University of China;
- ★ Manager for Application Virtualization Software Architecture, ISoft Infrastructure Software Co., Ltd.;
- ★ Manager for development of remote display protocol algorithms for 10 years;
- ★ Expert at remote protocol peripheral redirectioning.



P2P Transmission Technology Expert Xing Zhen

- ★ Bachelor of Computer Science, Jilin University;
- ★ P2P transmission technology product manager for 8 years;
- ★ Product manager of enterprise-level cloud computing/distributed computing desktop cloud platform;
- ★ Product manager of enterprise-level cloud computing/file P2P sharing and backup.



Software Architecture and System Architecture Expert Wang Shishuang

- ★ Bachelor of Computer Science, Beijing Jiaotong University;
- ★ Dell-wyse engineer, responsible for Linux OS architecture and device driver;
- ★ Senior development manager of Linux system architecture and virtualization algorithms for 15 years.



Blockchain Expert Chen Yaowen

- ★ Expert at machine learning;
- ★ Focus on industrial scenarios for deep learning of NLP;
- ★ Focus on blockchain+AI;
- ★ Deep learning of Chinese participle project kcws (GitHub 1700 star).



Senior Expert at Blockchain Zhang Cheng

- ★ Master of the Institute of Computing Technology, Chinese Academy of Sciences, 10 years' experience in engineering architecture;
- ★ Expert at large-scale search engine technology;
- ★ Expert at big data;
- ★ Senior expert at blockchain, intensively studying mainstream public blockchains and the economic and ecological planning of blockchains.



Cryptologist Dr. Tang Yi

- ★ Dr. & Professor of Sun Yat-Sen University, Guangzhou;
- ★ Presided over/participated in the completion of a number of national, provincial and ministerial level scientific research projects and won the second prize for progress in cryptologic technology;
- ★ Published multiple cryptology-related papers;
- ★ Visited Hong Kong Baptist University & Department of Computer Science, University of North Carolina, US on behalf of Sun Yat-sen University.



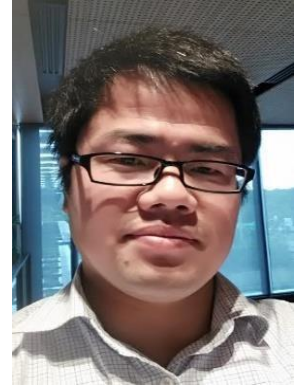
Distributed System/Grid Computing Expert Chen Chao

- ★ Master of Computer Science, Jiangnan University;
- ★ 15 years' experience in enterprise-level system architecture consulting, design and development, an Augmentum system architect, leading the design and deployment of Motorola's American 911 system, and the integration of gas dispatching SOA system of China Resources;
- ★ Senior R&D Manager of Cisco China, completing the development and definition of Edge300 series, Fanuc ZeroDowntime project, and the development and definition of Lora IoT gateway;
- ★ Co-founder of Yilujia Network Technology Co., Ltd;
- ★ Youaqu, E+ and Jinwangyigou system architecture and team management;
- ★ Founder of iweihi smart platform.



Distributed System/Grid Computing Expert Yue Yongqiang

- ★ Master of Mathematics, Dalian University of Technology;
- ★ Manager for the R&D of Windows system architecture and applications for 10 years;
- ★ Six years' experience in the architecture and development of ocloud computing application platform;
- ★ Eight years' experience in the management and development of ubiquitous terminal devices, and the management of more than 10,000 concurrent terminals;
- ★ App R&D Manager of Beijing Founder Electronics Co., Ltd.



Distributed System/Grid Computing Expert Ma Xiliang

- ★ Bachelor of Computer Science, Hebei University of Engineering;
- ★ Manager for the R&D of Linux cloud computing platform system architecture for eight years;
- ★ Six year of experience in front-end development of cloud computing platforms;
- ★ The developed products have been successfully applied in more than 1000 enterprise-level users.



Expert at Distributed System/ Blockchain Expert Andrew Dong

- ★ A development engineer of Amazon, he was responsible for rear-end authentication and authorization service of Amazon.cn.
- ★ A core development staff of IBM Labs WebSphere Commerce that was released in WebSphere Commerce 7.0;
- ★ A researcher of distributed consensus system, blockchain and distributed system of the University of Sydney, and a member of a distributed system research team ranking top in Australia; published patent applications relating to blockchain consensus protocol based on GPU common resources on IEEE Cloudcom;
- ★ Chief Blockchain Architect of Fujitus Australia, successfully developed and delivered multiple banking and financial projects.

8.4 Advisory Board

Advisor Michael Meng

- ★ Senior Vice President of Xinyuan Technology Group and Partner & CMO of RChaintech;
- ★ A senior in the chain and currency circles and an early investor in the currency circles;
- ★ Former Founder & CEO of Turingcat;
- ★ Has 15 years of experience in IBM as the general manager of a branch company and a general manager for smart city.



Advisor Wang Zishang

- ★ Author of “Cloud Management 2.0”;
- ★ Council member of China Artificial Intelligence Industry Alliance (AIIA);
- ★ Founder & CEO of Topfun Media (835872), which has hosted TOP50 Golden Apple Award and TFC Global Mobile Games & Intelligent Entertainment Exhibition for 13 years in a row. Participated in multiple famous blockchain projects, such as TOPC, CSDN and POT;
- ★ “Top 10 Women Entrepreneurs of Science and Technology” of China Women’s Development Foundation’s and “Her Entrepreneurship Mentor”, 2017 China Excellence Award for Scientific Management.



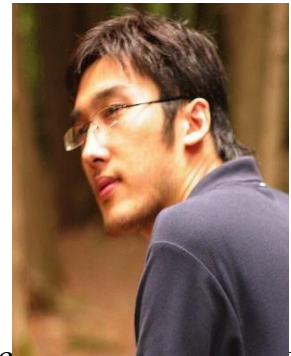
Advisor Song Xiaonan

- ★ Expert at Unix and distributed system and grid computing;
- ★ Has 10 years of experience in the R&D and marketing of cloud computing, having in-depth study on operating systems, middlewares and cloud platform and practice in large projects;
- ★ Worked in the cloud computing department of Huawei, responsible for product planning, business insight, and market promotion and technical cooperation in EMEA region;
- ★ Chief cloud computing architect of Primeton.



Advisor Zhao Peng

- ★ Hyper CEO;
- ★ A renowned expert in the cloud computing open sourcefield;
- ★ Hyperledger angel investor;
- ★ Ever served JP Morgan Chase, Quantum Fund, HSBC, Lehman Brothers, MUFJ and many other international financial institutions;
- ★ Possesses rich experience in the combination of blockchain and financial technology.



Advisor Lv Xinhao

- ★ Co-founder of iotchain;
- ★ Initiated a blockchain team in Xiaomi at the end of 2016 to solve the problem of data island;
- ★ Participated in the translation of mastering bitcoin in 2014, which is a very good technical document about blockchain in Chinese.



Copyright Notice

LEGAL DISCLAMIMER: The information and any attachments contained in this document is all copyright and intellectual property of Uranus Foundation reserved. Without prior written permission of any copy, reproduce, propagation or storage will be prohibited.

Disclaimer

Uranus Foundation reserves the right to interpret and modify this document. Any amendments, updates or interpretations of this document will be announced on the website of Uranus Foundation (<http://www.uranus.io>).

Uranus

