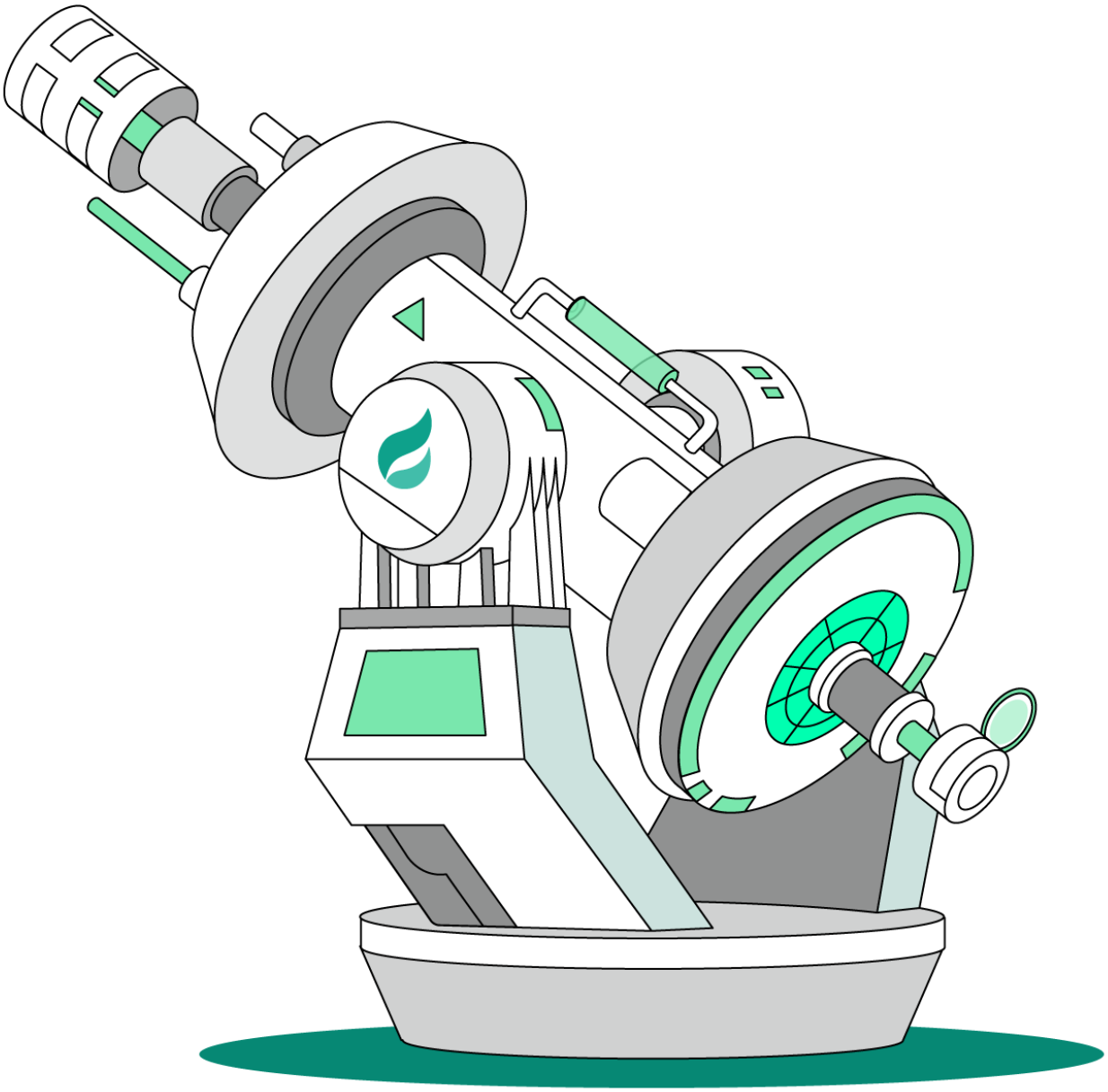
Huobi Ventures

Technical Market Analysis

Middleware

By Jinbin Xie · **Huobi Ventures**
jinbinxie@huobi.com



1. How to define middleware for the Web3 domain

1.1 Middleware in the traditional sector

In the traditional Internet sector, the middleware business is mature and has spawned a large number of niche tracks. The business scenarios it serves are such that, with the current development of the industry, the number of technical practitioners has increased, but the technical level of the practitioners among them varies. For some business scenarios that require complex technical implementation, or numerous complex technology stacks, this poses a huge challenge to technical staff.

The differences arising from different technologies can be addressed with a unified middleware product.

The core objective is to smoothen out the huge challenges posed by the development process, for both people and applications.

1.2 Middleware-oriented user scenarios

New application development

Middleware supports modern, common runtime environments for a variety of use cases. Developers and architects can work flexibly across platforms using the following basic runtimes, frameworks and programming languages. In addition, middleware can offer common features such as web servers, single sign-on (SSO), message passing and in-memory caching.

Optimizing existing applications

Middleware helps developers convert traditional monolithic applications into cloud-native applications, giving valuable tools a new lease of life with higher performance and greater portability.

Fully integrated

Middleware integration tools connect key internal and external systems. Integration features such as conversion, connectivity, composition and enterprise messaging are combined with SSO authentication to make it easier for developers to extend functionality between applications.

Application Programming Interface (API)

Many middleware services are accessed through APIs, a collection of tools, definitions and protocols that allow applications to communicate with each other. APIs make it possible to connect completely different products and services through a common layer.

Data flow

APIs are a method of sharing data between applications, with another being asynchronous data streaming. This involves replicating data sets in intermediate stores where data can be shared between multiple applications.

Intelligent Business Automation

Middleware can help developers, architects, IT and business executives automate manual decisions, to improve resource management and overall efficiency.

1.3 Analysis of middleware business forms

The earlier overview of different user scenarios gives us a clear idea of how middleware is presented in different scenarios.

Here, we will select typical application cases in different user scenarios to help readers better understand the usefulness of middleware.

New Application Development

With different operating systems and programming languages, the average developer needs to spend a lot of time researching the technical documentation of different systems or operating environments. Different APIs or technical implementations can require developers to spend considerable time on research.

It is even more costly to recruit a large number of developers and develop projects to be compatible with multiple environments and platforms. For example, Google has developed the Flutter cross-platform application development framework for developing applications for Android, iOS, Windows, macOS, Linux Desktop, and Google Fuchsia. This allows developers to quickly develop applications without having to understand the differences in the underlying different platforms. The number of staff hires and development costs are reduced.

Optimisation of existing applications

With the development of cloud technology, high-performance algorithms or software are put on the cloud in a modular way, so that developers can integrate them into their own applications according to their needs and improve the performance of their own applications at a lower cost.

Full Integration

Developers can automate business collaboration between users by accessing the SDKs. These are mainly in the form of development SDKs such as Facebook Open Platform and Slack Development Platform.

Application Programming Interface (API)

Most middleware exists in the form of APIs, and different APIs can be combined to create new application forms.

Data Flow

Some BI analysis business scenarios require a common data warehouse to support the data needs of different reports.

Intelligent Business Automation

Examples include some Jenkins (to automate code test deployment), and some low-code development platforms.

.

1.4 Defining the middleware presentation for the blockchain industry

With the rapid development of the blockchain industry, various public chains have emerged, with different programming languages, technical standards and communication data standards. How can we build a full node service for a public chain? How do we query all the block data of a public chain? How do we communicate with multiple public chains?

In the blockchain industry, traditional user scenarios will also emerge, but the middleware position will become increasingly prominent for a decentralised application architecture system. This is because decentralised application architecture leads to fragmented product forms which result in limited attention spans, whether for developers or users. The scattered product forms will be aggregated in some reasonable way to make it easier for developers and users to use. There will be no need for dapps to be tied to each other, but also for middleware to be tied in between.

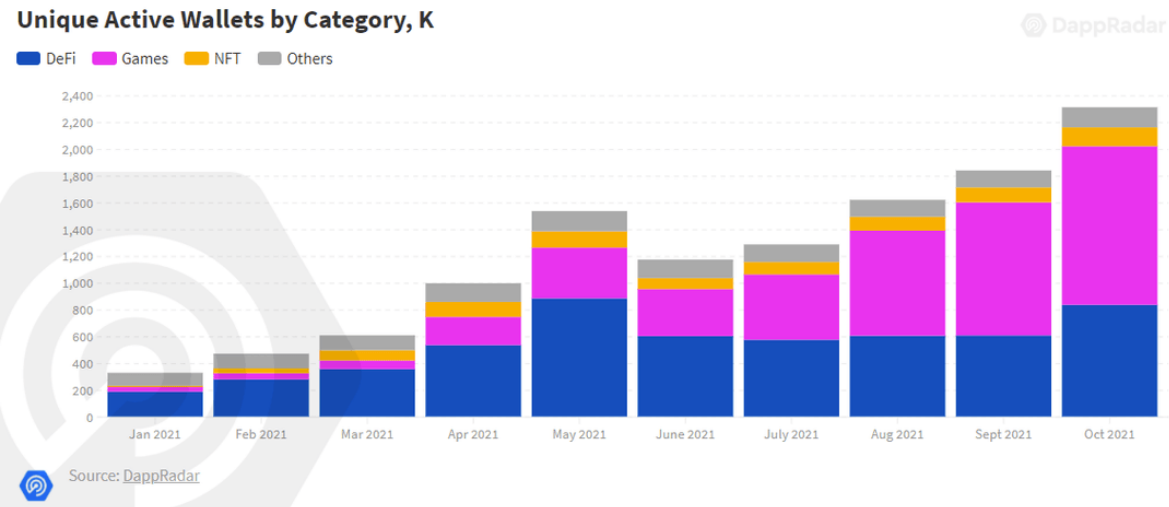
2. Current market context

2.1 Increasing growth of dapp

Quote (https://dappradar.com/blog/dapp-industry-overview-october-2021-on-nfts-defi-and-games)

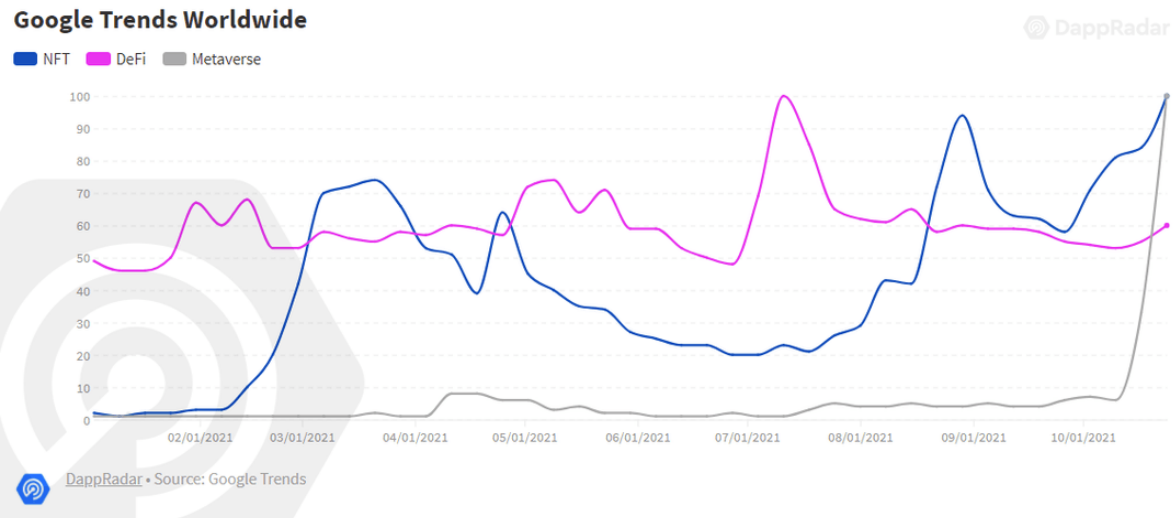
Let us first look at a few pieces of data.

1. Two million unique active addresses and Dapp interactions per day



2. The NFT space keeps up the pace after an impressive summer with $4.2 billion in trading volume in October alone, a 2% increase month-on-month.

3. Google trends direction



From the above data, we can see that in the context of the bull market last year and with the NFT product out of the loop, a large number of startup teams have gone into developing Dapps. The number of Dapp users has also increased, both in terms of Google search trends and the number of unique active addresses interacting with Dapps on a daily basis. There is growth in both Dapp and the user base, crypto penetration and both ends of the bilateral market, with some market conflicts beginning to emerge.

3. Main contradictions in the market

How did the main market contradiction arise? This started when some of the limitations of blockchain meant it could not meet the needs of users, starting from the decentralization of Dapp. In the following paragraphs, we will start from the perspective of the limitations of Dapp and the needs of users, and examine in increasing depth.

1. *The first problem that ordinary users encounter when using a Dapp is that they have to buy ETH to pay for the Gas required to interact with the Dapp. At the user level, whether it is buying ETH through fiat currency or installing a wallet plugin to create a wallet address, there is a certain learning cost. There is also a risk of losing assets due to improper operations.*

2. *With hundreds of Dapps now available, users are faced with so many options, which inevitably increases the cost of learning. The aggregation of so many Dapp contract APIs is one of the main conflicts.*

3. *Not only are users faced with the fear of choice when it comes to Dapp, developers have the headache of different technical standards under various public chain platforms, standards for RPC services, and standards for data on different chains. The fundamental reason for the contradictions in this market is that there are numerous different standards, which translate to huge workloads.*

4. *Most of the current Solidity smart contract public chains have many limitations. On-chain storage is expensive, execution cannot be automated, complex calculations cannot be done, smart contracts on different public chains cannot communicate with each other, and individual contracts cannot be automated even as developers are challenged by more complex business scenarios.*

5. *Because the Dapp application architecture is decentralised, the connection with the user is broken and there is no effective messaging channel. This poses a huge challenge for the daily marketing operations and secondary marketing of the Dapp.*

6. *The aforementioned market contradictions are just the tip of the iceberg. The line of thinking is as follows: identify differences and smoothen them out; identify flaws and rectify them; and aggregate everything that can achieve a network effect.*

4. How to solve

4.1 Defining scenarios with potential network effects

Drilling down into middleware-specific scenarios, we observe that some technology tools that clearly address specific needs are adopted by many but ultimately fail to deliver value. The value of crypto projects comes from the network value they generate once they are adopted, rather than just being useful. We cite NFX's research on network effects, which found that the adoption of the following network effect models for middleware would add significant value.

Why is middleware only valuable when it creates a network effect?

Introduced by computer networking pioneer Robert Metcalfe in the 1980s, Metcalfe's Law defines the value of a network as a mathematical function based on the number of devices (fax machines, telephones, etc.) it connects to, derived from his experience selling Ethereum. Metcalfe's Law is a central pillar in the study of network effects, and specifically states that "the value of a system of compatible communication devices increases as the square of the number of devices in it". Simply put, every time a user joins an application, the value of the application increases to n^2; if the number of nodes in a network doubles from 100 to 200, its value does not double, but becomes quadruple.

Take the example of Log4J, the most adopted log management framework in the Java open source community. It was only with the recent exposure of a vulnerability that could be remotely executed that people realized such a widely adopted development framework was left to the hard work of volunteers, but with no revenue. Despite a high adoption rate, commercialization is exceptionally difficult. The three traditional Internet business models of advertising, e-commerce and gaming are difficult to replicate in the middleware industry. If a paid model is adopted, it will have an impact on its adoption rate. What's more, each Log4J adoption unit is separate from each other and cannot achieve value delivery and value retention. Just like in the ant colony algorithm where each node leaves pheromones behind, each node is connected to each other, and in the end the entire network value is reflected in finding the optimal path. The above illustrates the difference between a pure middleware tool and middleware with network effects.

Protocol: This refers to a standard for information transmission or data processing, such as the Ethereum grid which is highly defensive and has the second most common type of grid effect. Most protocols derive their value from grid effects, and their use is almost non-existent when the threshold is breached. They are often embedded in products that use the protocol.

System or software platform (Platform): Examples include Windows, iOS, and Android, with attributes such as bilateral stakeholders (users and developers), and positive indirect grid effects like marketplaces. The product and sales approach is more important than online marketplaces; multi-tenancy is also a challenge for platforms.

Data networks: Examples include Waze and Yelp! Data is the core of the value of a product; more usage requires more useful data to be generated and collected. The effect is usually achieved after a certain threshold of data volume is reached – different from the scale effect of data.

Tech Performance Network: A product is said to have a Tech Performance Network effect when its technical performance can **directly** increase the number of users. In a Tech Performance network, the more users or endpoints are involved, the more the technology can be used, for example to make a product or service faster, cheaper or easier to use. Examples include Bittorrent and Skype.

4.2 Analysis of specific application scenarios

The Protocol network	System or software platform (Platform)	Data grid	Tech Performance grid
cMix grid communication confusion	Bico	Chainlink	Message notification system
Peer to peer file system	Modular implementation layer	The Graph	High Performance Communication Services Middleware
Cross-chain communication	NFT SDK	Decentralized database	Distributed Computing
Decentralized single sign-on system	Programming languages across smart contract platforms	High performance data analysis: Dune	Document Retrieval Predictor
State machine communication across chains	Off-chain Computing Extension Protocol	Data Collection Warehouse	Authentication Node Hosting
Libp2p/IPLD	Aggregation of computing tasks under different chains	Data Lake	zkSnark accelerates SaaS services

Infura api	LowCode's Dapp development platform	Data auction transactions	
Data search criteria for different chains	DWeb Application Development Platform		
Interactive communication between different Layers			
Aggregation of different storage protocols			

1. Protocol networks

a) cMix Communication Obfuscation Network (Representative: Nym): Compared to P2P communication, which is now basically plaintext transmission, cMix technology exists as middleware and developers only need to access it as a socket protocol to fully implement obfuscated encryption. For the average developer who is unlikely to have a significant background in cryptography, it is difficult to put a shell of encrypted communication on their application. cMix technology, which hides the IP of the sender, the time of sending, ensures truly encrypted communication by randomly disrupting the sending queue and encrypting in onion-like layers. With such a complex technology, developers only need access in the form of a socket protocol to enable application integration of encrypted communications. It reflects the market positioning of middleware and helps developers to lower the threshold of technology development. At the same time, it is a new presentation of information transfer, the more developers adopt it, the stronger the network effect and the higher the barriers to exclusivity.

b) Peer-to-peer file system: The most public chain information exchange is blockchain ledger information, but for multimedia files, these public chains are difficult to achieve exchange. The storage resources on the chain is expensive, and the protocol scalability is limited. A decentralized file system is needed as a middleware product to help developers better develop the complex application scenarios they need. A decentralized file system is, broadly speaking, a presentation of a multitude of information aggregated and then distributed twice. The more people adopt it, the higher the barrier. The first decentralized file system that comes to mind is IPFS. The establishment of a network effect will also affect the user's perception of the brand.

c) Cross-chain communication: In the era of multiple chains, each public chain is an island to each other, and as described in the book "Small Island Economics", commodity exchange and currency

exchange are bound to happen between different islands. Similarly, speculators will pursue token swaps between different chains and arbitrage swaps between different chains. Faced with user demand, it also takes a lot of development work for developers to develop a cross-chain system. Therefore, like how deBridge developed an SDK, ordinary developers only need to access the SDK to complete the development of cross-chain applications. This application scenario also validates the problem of how to smoothen out the different levels of developers, and how to develop quickly when faced with development challenges.

d) In a decentralized single sign-on system, the user login scenario is the only channel through which the Dapp can know the user's identity. In the original scenario, for example, sign in with eth had to rely on a centralized backend or the browser's local storage to cache the user information for the Dapp to identify the user. The various technical solutions vary widely. A unified identity login system is needed, where the developer cannot think too much i.e. access to the user's identity role. The more this decentralized single sign-on system is adopted, the stronger its network effect will be. It is the equivalent of a fully integrated business direction in traditional middleware.

e) State machine cross-chain communication: Different public chains have maintained their own state machine, regardless of account state and contract state, using Merck tree maintenance, and to ensure that it is not tampered with. Challenges include how to avoid man-in-the-middle attack, cross-chain communication to ensure that the message forwarding is not tampered with, the synchronization of each chain's state machine, and strong consistency to avoid the risk of asset forgery casting.

f) Libp2p/IPLD: P2P communication is the infrastructure of all public chains, guaranteeing data synchronization and information exchange between different nodes. However, there are many challenges in the development process, such as link finding, KAD path calculation and other technical difficulties. Protocol Labs encapsulates the P2P technology into a single module, which public chain developers only need to integrate. There is no need to spend a lot of time on technical development and exploration. Moreover, there are development teams trying to apply Libp2p in cross-chain scenarios, where multiple public chains integrated with Libp2p are interconnected.

g) Infura api: The equivalent of an application programming interface (API) is required for orientation when a wallet developer integrates with a public chain, or when aDapp developer needs to communicate with the chain. However, each interface definition standard is different and developers need to spend a lot of time to study each RPC technical document to ensure that they can communicate with the public chain RPC service properly. With the emergence of Infura api, the communication standard of each public chain is aggregated through a platform, and it is

only necessary to call the API interface of the platform to achieve communication with each public chain. Such an API aggregation platform really solves the pain of developers facing a wide range of interface standards. The more it is adopted, the stronger its network effect. This has been proven in the market so far.

h) The data query standards of different chains are unified because different public chains have inconsistent data structure definitions for information on the chain. Challenges encountered when developing data on the chain include how to use a standard query statement to query multiple data sources. Some technical teams have tried to use GraphSQL to establish a unified query standard to clean and organize data from different chains, and finally present it as a unified query library middleware. For data scientists, this greatly compresses the previously large amount of data collation work. The more complex the data source, the stronger the exclusive barrier to this query repository middleware.

i) Communication between different Layers: With the massive rise of Layer 2, the CallData capacity of Ethereum smart contracts has increased. Ecological applications gather from multiple chains to multiple Layers, and the Ethereum ecosystem is expanding outwards layer by layer in an onion-like structure. However, messages are passed between different Layers through CallData, and new EIPs can interoperate with contracts on Layer 1 to Layer 2. Of course, there are still challenges in how to effectively interoperate messages, and there must be a middleware role in which to act as a middleman. The same is true for data forwarding and delivery between different Layers. Because the Layer 1 main contract does not need to deploy the complete contract code on one side of each Layer, different Layers only need to deploy sub-logic, and the core logic is still deployed on Layer 1.

j) An aggregation API for different storage protocols: There are more distributed storage protocols on the market today, but they all have different ways of fetching files and different storage methods behind them. Again, a standard API will help developers smoothen out the hurdles, and the more this middleware is adopted, the greater the value of the network effect.

2. System or software platform

a) Bico: Currently there are many smart contract development platforms, and users have to pay for Gas when using them, which is a challenge for both users and developers. The middleware does some pre-work on this: For example, the developer pledges the cost of the Gas required in advance within the agreement to help the customer pay on their behalf. There is compatibility with multiple chains through one set of development standards. Developers develop applications on this middleware, which itself becomes a software system platform, dovetailing both the user and developer ends. The more developers there are on its platform, the stronger its network effect.

b) Modular execution layer: A large number of public chains exist as monolithic applications, and consensus security and execution layer are highly coupled. If there is a blockage in the execution process in the execution layer, it eventually leads to the whole chain going down and consensus security cannot be guaranteed. The different modules are disassembled and decoupled so that more room for expansion can be added. The execution layer can reach a parallel state, where the proof generated by the execution layer in response to a change in the state of the main chain is submitted to the consensus security layer for validation. The modular execution layer can also be designed for generalization and sandboxed for resource segregation to serve different chains. For developers, it is a more efficient platform for new applications.

c) The NFT SDK which focuses on helping developers to make NFT application development faster and simpler, requires only a few lines of code.

d) The current market situation for programming languages across smart contract platforms is that different smart contract platforms have designed their own programming languages, for example StarkNet designed Cario and so on. For a developer learning a new programming language, there is a different learning curve. If there is a programming language that is compatible with different smart contract platforms, it greatly stimulates developer interest. Because the opportunity cost behind learning a new programming language needs to be measured, the developer is left cold in the job market if that smart contract platform falls out of favour. This is why Java developers have been in demand in the job market for decades, because of the high adoption rate. The network effect of a programming language is also very evident once it has been widely adopted.

e) The off-chain computation extension protocol, Solidity Programming Smart Contracts, also has many limitations because of the design of the Gas mechanism, which cannot be designed for overly complex computations because of the huge cost of the Gas it generates. But what should developers do when faced with complex business scenarios? Move complex business logic calculations off-chain, generate zkproof of the execution process, and then have it validated by the consensus security layer. As a middleware for complex business calculations, it greatly expands developers' business design scenarios and opens up their imagination.

f) Aggregation of computing tasks under different chains, followed by the previous one, for large and complex computing operations can be done by aggregating interfaces and then computing operations split, but using MPC in the distribution process to guarantee secrecy.

g) LowCode's Dapp development platform: Contractual vulnerabilities appear frequently in the Dapp development process. This is not just due to the lack of security management, but also the developer's inexperience. LowCode Dapp development platforms are more likely to provide secure code templates for developers without extensive development experience. There are of

course limitations, such as customized development scenarios and the like.

h) DWeb development platform, a complete web application at Web2, requires user identity login system, business logic modules, database and file system. There are proven centralised solutions for all of the above. Fission.codes provides UCAN, WNFS, WNDB and a compute layer to help developers truly decentralized their web applications.

i) Data grid: The essence of the middleware application scenarios listed in the table is to help developers design their own data products faster and easier. The more the application itself is adopted, the more data is accumulated, creating its own exclusive barriers.

j) Technical performance networks: The middleware application scenarios listed in the table are notable for scenarios that are not possible with Solidity applications, helping to lower the technical threshold for development, while their business forms reflect stronger performance as more nodes or roles are added to their networks.