Computecoin Network: The Infrastructure of Web 3.0 and The Metaverse

Computecoin Network Foundation

Abstract

Web 3.0, an evolution of Web 2.0, refers to decentralized applications (dAPP) that run on the blockchain. These are the applications that allow anyone to participate with their personal data well protected and controlled by themselves. However, there are quite a few challenges in the development of Web 3.0 such as accessibility (i.e., less accessible to most users as that in modern web browsers) and scalability (i.e., high cost and long learning curve for using decentralized infrastructure). For instance, although non-fungible token (NFT) is stored on blockchain, the content of most NFTs is still stored in centralized clouds such as AWS or google clouds. This puts a high risk on user's NFT assets, contradicting the nature of Web 3.0. The metaverse, first proposed by Neal Stephenson in 1992, refers to an infinitely vast patchwork of persistent virtual worlds in which people can freely travel, socialize and work. However, metaverse applications and platforms such as Fortnite and Roblox face an enormous challenge: their growth is limited by a finite supply of low-cost and instantaneous computing power from centralized clouds. In summary, building the next-generation applications on the current centralized infrastructure (built since 1990s) has become the bottleneck on the critical path towards our dreamed world.

We have initiated this project, the Computecoin network along with its native token CCN, to resolve this issue. Our objective is to build the next-generation infrastructure for all-purpose applications on Web3 and the metaverse. In other words, we aims to do for web 3.0 and the metaverse what centralized cloud providers did for Web 2.0. The basic idea of our system is to first aggregate decentralized clouds such as Filecoin and data centers around the world (rather than build new infrastructure as AWS did 20 years ago) and then offload computation to a proximity network of the nearby aggregated decentralized clouds to empower end users' data processing tasks such as AR/VR 3D rendering and real-time data storage in a low-cost and instantaneous manner. Computecoin network comprises two layers: PEKKA and the metaverse computing protocol (MCP). PEKKA is an aggregator and scheduler that seamlessly

.

Computecoin is still under development. Please visit our official website to stay up-to-date on our progress, and to view the latest version of this white paper.

integrates decentralized clouds and dynamically offloads computation to a proximity network. PEKKA's capabilities include deploying web3 and metaverse applications to decentralized clouds in a matter of minutes, and providing a unified API for easy data storage and retrieval from any decentralized cloud, like Filecoin or Crust. The MCP is a layer-0.5/layer-1 blockchain featuring an original consensus algorithm, proof of honesty (PoH), which guarantees that the results of outsourced computation in the decentralized cloud network are authentic. In other words, PoH establishes trust in computation tasks outsourced to trustless decentralized clouds, building the foundation for web 3.0 and the metaverse ecosystem.

CONTENTS

I. Introduction

It is widely agreed that Web 3.0 is the key to actualizing a more decentralized and interactive experience in the metaverse. As a result, we usually view Web 3.0 and related technologies as the building blocks for the metaverse. Therefore, in what follows, we focus our discussion on the metavese, the ultimate goal that computecoin targets.

A. Introduction to metaverse

Imagine every activity and experience in your daily life taking place within arm's reach of each other. Imagine seamless transit between each space, each node, you inhabit and the people and things you interact with in them. This vision of pure connectivity serves as the beating heart of the metaverse. The metaverse, as its name suggests, refers to an infinitely vast patchwork of persistent virtual worlds between which people can freely travel. Neal Stephenson is often credited with laying out the first description of the metaverse in his seminal 1992 science fiction novel *Snow Crash*. Since then, dozens of projects – everything from Fortnite and Second Life to CryptoKitties and Decentraland – have nudged humanity closer to the metaverse. When it does take shape, the metaverse will offer its inhabitants an online experience as rich as, and intimately linked with, their lives in the physical realm. Indeed, these bold pioneers will be able to immerse themselves in the metaverse through all manner of devices, including VR headsets and 3D-printed wearables, as well as technological standards and networks like blockchain and 5G. Meanwhile, the metaverse's smooth functioning and capacity to expand boundlessly will depend on a durable base of computing power.

The metaverse's development has taken a bifurcated path. On the one hand, centralized metaverse experiences, like Facebook Horizon and Microsoft Mesh, aim to build standalone worlds whose territory lies entirely within proprietary ecosystems. On the other hand, decentralized projects seek to equip their users with the tools to create, exchange and own digital goods, secure their data, and interact with each other outside the confines of corporate systems. In both cases, though, the metaverse is not a mere platform, game, or social network; it is potentially every online platform, game and social network used by people around the world all bundled together in one landscape of virtual worlds owned by no one user and by every user at the same time.

In our opinion, the metaverse comprises five layers stacked on top of each other. The most elemental layer is infrastructure – the physical technologies that support the metaverse's functioning.

These include technological standards and innovations like 5G and 6G networks, semiconductors, tiny sensors known as MEMS and Internet data centers (IDCs). The protocol layer comes next. Its components are the technologies, like blockchain, distributed computing and edge computing, that ensure the efficient and effective computing power distribution to end users and individuals' sovereignty over their own online data. Human interfaces make up the third layer of the metaverse. These include devices – like smartphones, 3D-printed wearables, biosensors, neural interfaces, and AR/VR – enabled headsets and goggles – that serve as our entry points into what will one day be a collective of persistent online worlds. The creation layer of the metaverse stacks on top of the human interface stratum, and is made up of top-down platforms and environments, like Roblox, Shopify and Wix, designed to give users tools with which to create new things. Finally, the aforementioned experience layer completes the metaverse stack, lending the metaverse's working parts a social, gamified exterior. The components of the experience layer range from non-fungible tokens (NFTs) to e-commerce, e-sports, social media and games.

The sum of these five layers is the metaverse, an agile, persistent, and interconnected patchwork of virtual worlds standing shoulder-to-shoulder in one contiguous universe.

B. Limitations of the metaverse development

Today, the world's most popular online worlds, like Fortnite and Roblox, cannot support the radical accessibility, connectivity and creativity that will define the metaverse of tomorrow. Metaverse platforms face an enormous challenge: Constricted by a limited supply of computing power, they fall short of delivering a true metaverse experience to their users. Although high profile projects – such as Facebook's upcoming Horizon project and Mesh, Microsoft's foray into the world of holoporting and virtual collaboration – have the backing of leading cloud services, the virtual worlds they offer users will still be covered in red tape, highly centralized and lacking in verisimilitude. Metaverse projects in 2021 are mostly low-fidelity and operated in a top-down manner, with corporate developers deciding how many users can share experiences in the same place at the same time and ratcheting-down customizability to smooth out lag problems.

12 million people attended Fortnite's Travis Scott concert in 2020, but the game could only host this enormous audience by bundling players into groups of about 50, called shards, to whom the concert was broadcast simultaneously. That is to say, Fortnite could not manage to gather the concert's 12 million viewers in one place at the same time; rather than a true metaverse experience, users were shown an illusion of synchronicity. Roblox, on the other hand, only

allows 200 players to enter its relatively low-fidelity worlds at a time. And Freefire, a mobile-only game played mostly on low- to mid-range Android devices, caps players at 50 per game in its main battle royale mode. These platforms give their users a glimpse at what the metaverse could be: an infinite canvas on which people can create and trade unique items, explore and build, make friends and brainstorm with colleagues. We designed a solution to make this vision a reality.

C. Our solution: the computecoin network

Computecoin is a self-evolving computer built to serve the metaverse. Our network squarely addresses the shortcomings of today's metaverse platforms by introducing a pipeline of rich, low-cost and instantaneous computing power. This solution will empower the development of metaverse platforms that are high-fidelity, decentralized, capable of handling an unlimited number of users in one virtual space at the same time, and offer users an inexhaustible array of customizable features.

Computecoin comes fully loaded with functionality. The platform will build a robust metaverse ecosystem enabled by the Computecoin's underlying layers: PEKKA and the Metaverse Computing Protocol (MCP). See Fig. 1. PEKKA, the network's power source, first aggregates decentralized clouds, like DFINITY, Filecoin and data centers – allowing users to access and use decentralized computing power in the easiest, most affordable and most accessible way yet. Then, it offloads computation to a proximity network (for instance, a nearby data center) to supplement local compute. Thus, PEKKA fills in gaps in computing power supply to ensure that end-users on mobile devices like iPhones and Oculus headsets can seamlessly immerse themselves in the metaverse. The MCP – the network's guardian – is a layer-1 blockchain featuring the original consensus algorithm proof of honesty (PoH), which guarantees that the results of computation tasks outsourced to proximity networks are authentic. PoH establishes trust in computation tasks outsourced to trustless decentralized clouds, building the foundation for the metaverse ecosystem. Furthermore, our AI algorithms – the network's "DNA" – constantly self-evolve to improve end users' metaverse experience by, for example, finding the best decentralized cloud for each of their compute and storage tasks, optimizing the configuration of a proximity network and the scheduling of computation offloading, and using data collected in our system to enable the system to self-evolve.

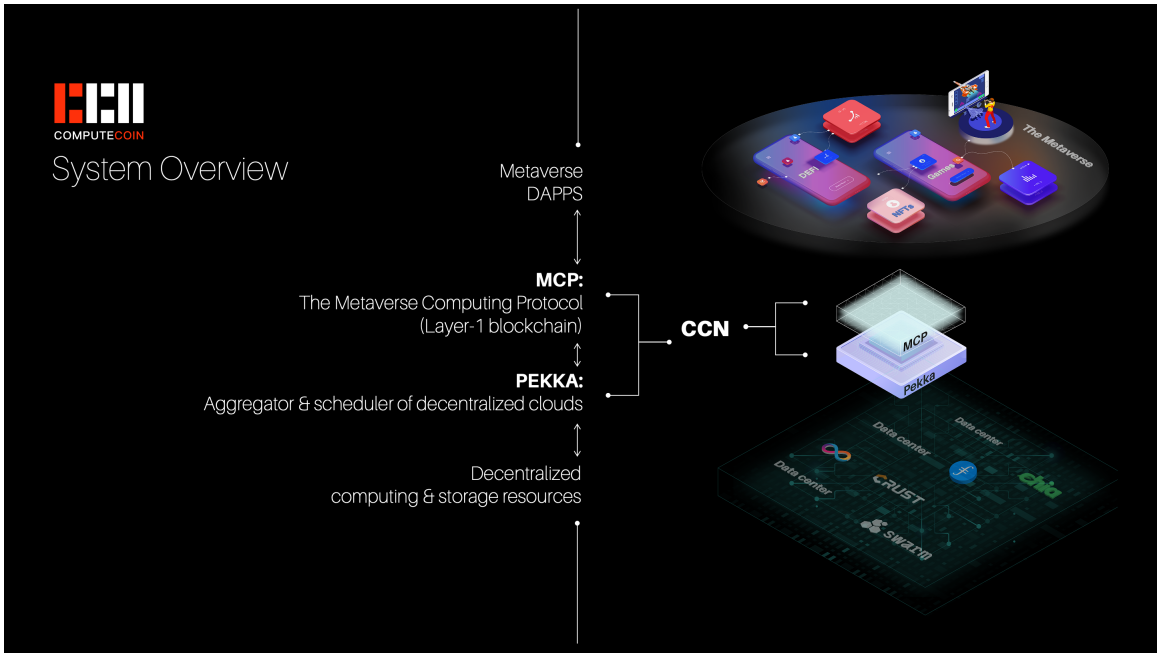

Fig. 1. Computecoin network: a system overview

Computecoin's heritage is one of disruption and innovation. Currently, we hold two U.S. patents, one for decentralized computing with blockchain technology, and one for our novel consensus algorithm "proof of honesty" (PoH). PoH solves a long-standing quandary in the computing field known as the verifiable computing problem. Before, there was no fail-proof way to authenticate the results of outsourced computing tasks. PoH provides an elegant solution to this problem; the algorithm fosters trust in the Computecoin network by enabling users to outsource tasks to computing power providers with the knowledge that the tasks will be faithfully executed. The Computecoin research team has also published a number of scholarly articles on our technologies.

Computecoin Network (CCN) began in 2018 as a university research project. Today, we are a fast-growing platform backed by a team of computer scientists from the world's top universities, IEEE fellows, engineers, and experts at the cutting edge of the blockchain industry.

D. Paper organization

The following sections will offer an in-depth examination of Computecoin network with one section for each layer of the system. The technical derivations and proofs (e.g., the PoH consensus

algorithm) will be published in a separate technical publication.

II. PEKKA

A. Overview

Rich, low-cost and instantaneous computing and/or storage power is essential to power humanity's immersion to the metaverse. To this end, we need to first integrate the world's decentralized clouds to guarantee cloud service coverage for end users. Then we must dynamically configure a proximity network of the integrated decentralized clouds to process each computing task for the end user. PEKKA is designed to achieve these two objectives. That is, PEKKA is an aggregator that integrates decentralized clouds and dynamically configures a proximity network to offload end users' computation. See Fig. 2 for an illustration. In summary, PEKKA's functionalities include the aggregation of decentralized clouds and the configuration and scheduling of a proximity network.

B. Aggregation of decentralized clouds

Rather than building new infrastructure to serve metaverse applications, PEKKA provides an all-in-one solution that seamlessly integrates decentralized clouds, such as Filecoin, Crust, DFINITY and data centers. Generally speaking, developers today must overcome a steep learning curve to deploy metaverse applications to the decentralized cloud. For instance, game developers need to decide which decentralized storage network they should choose to store the game's NFTs. However, this choice is extremely difficult, as networks such as Filecoin, Crust, Swarm and Chia are highly dynamic in terms of price, reliability and coverage. PEKKA will close this gap by providing:

1) a user-friendly and simple client software to integrate decentralized cloud servers, including cryptocurrency mining machines;
2) a simplified deployment process such that developers can deploy metaverse applications to the decentralized cloud in a matter of minutes;
3) unified API for seamless data storage and retrieval from any decentralized cloud such as Filecoin or Crust.

Metaverse applications in the Computecoin network run on two APIs: the *PEKKA API* and *MCP interoperability API*. The PEKKA API is a unified API providing access to all underlying

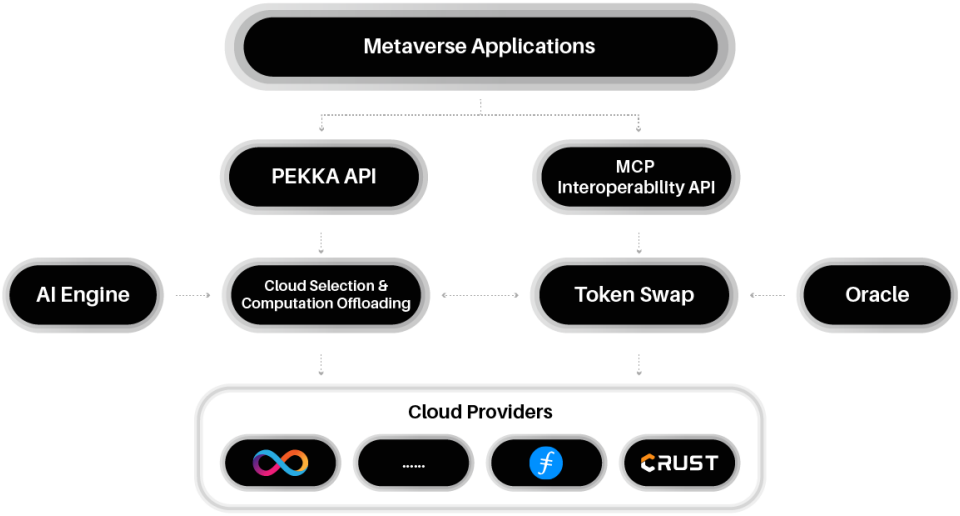

Fig. 2. PEKKA: aggregator and scheduler of decentralized clouds.

resources such as Filecoin, Crust, etc. It runs cloud selection and computation offloading algorithms by matching users' requirements to the availability and price of nearby resources. For example, the API will assign cold data and hot data to different decentralized storage clouds. The AI engine, for example, allows us to maintain the lowest costs and operating fees in the industry. PEKKA provides both file storage APIs and retrieval APIs which will be compatible with pubic cloud solutions.

The MCP interoperability API is designed to allow payments to be collected from CCN for any service in the decentralized cloud. Using MCP interoperability, a token swap is deployed to convert CCN to native tokens for payments to other platforms. The interoperability is intrinsic in the layer-1 solution of MCP, which makes the swap experience seamless and convenient for end users.

Metaverse developers will likely find the aggregation of decentralized clouds very attractive. For example, game developers need cheap and reliable storage for tons of data generated by players' activities. This data typically needs to be archived and stored for a long time for several purposes, mainly to create transparency and auditability for the game's mechanics, payments

and reward systems.

That aggregation of decentralized clouds supplies users with simple and reliable computation and storage solutions at the lowest cost. This is the key to the massive deployment of metaverse applications.

C. Computation offloading to a proximity network

The process of moving all computing tasks to centralized clouds represents a bottleneck for metaverse applications. For instance, Facebook's Oculus VR headsets capture a huge amount of data from users' environment, which the system must process in real time to yield a good gaming experience. If the VR gear must send data to a centralized cloud for processing, the resulting response time would be too long. And the number of VR gear in one area would be strictly limited due to the constraints of network bandwidth and reliability. Thus, processing data in the end user's proximity network would yield shorter response times, more efficient processing and less pressure on the network. Recent work on micro-datacenters (mDCs; small, modular datacenters designed to optimize networked devices' performance via the cloud [11]) has proved the feasibility and benefits of this idea. In addition, authors [5] have demonstrated, on a real proximity network testbed (Fig. 3), that latency can be reduced by up to 88% and that AR users' energy consumption can be brought down up to 93% by offloading computation to a proximity network.

Along these lines, PEKKA's other feature is to first dynamically configure a proximity network at the end user's location, and then offload computation to a proximity network made up of decentralized clouds to make up for its lack of computing resources, like VR gear. In this way, we could meet the need for real-time interactive response by enabling low-latency, one-hop, high-bandwidth access to the cloud. In particular, the end-to-end response time of applications executing within the cloud can be as fast as a few milliseconds, and predictable. If no nearby clouds are available, the application can easily be degraded to a fallback mode that involves a distant cloud. When a nearby cloud is found, the application's full functionality and performance will be immediately restored.

One important aspect of PEKKA's computation offloading is how to utilize and manage offloading process in practice. The end user equipped with a CCN client will perform two functions for computation offloading:

1) **Function 1**: determine what application tasks could be offloaded;

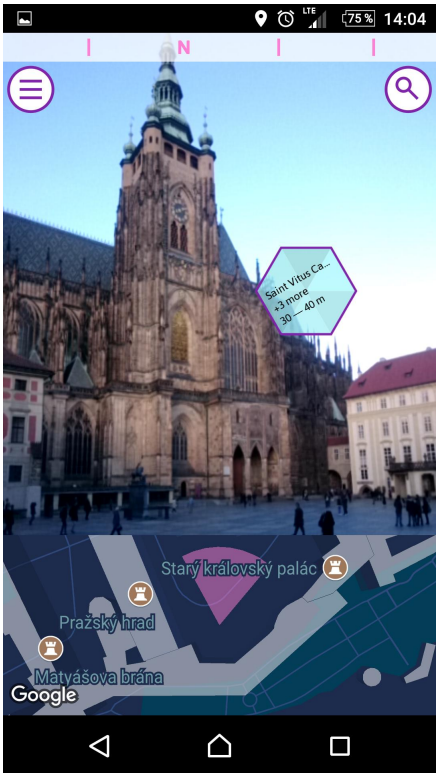

Fig. 3. Percipio: an augmented reality app that allows tourists to discover and render nearby places and structures.

2) **Function 2**: report vital parameters for the offloading scheduler, such as bandwidth, the volume of data to be offloaded and the energy spent by executing the task, and then determine whether to offload or not.

The details of the two functions are presented as follows.

1) Offloading function 1: Given an application task, PEKKA must first check the offloadability of the application. In one case, the application can be divided into a set of parts that all can be offloaded to the proximity network. In the other case, only a subset of parts can be offloaded (whereas other parts, such as real-time processing of user input, cannot). Next, PEKKA needs to identify the dependency of individual parts to be processed. If all parts are independent, they can be offloaded and processed in parallel in the proximity network. However, if processing some parts requires input from other parts (the relationship among individual components can be presented by a direct cyclic graph (DAG) or component dependency graph (CDG)), parts cannot be offloaded in parallel. In this case, we will always offload the application to one or more interdependent cloud servers in the vicinity.

2) Offloading function 2: Given the offloadability of the application task, along with the reported vital parameters for the offloading scheduler, PEKKA needs to determine whether to offload a part or not. On the one hand, for all parts that can be processed in parallel, the decision is easy to make following a simple optimization of resource allocation. That is, assigning a set of servers in proximity to process the parts of the task in parallel. On the other hand, interdepentent parts, such as a face recognition application that consists of a face detector and a classifier, prompt the decision of whether to offload these parts or not. Indeed, just how these parts should be offloaded presents a challenge. To solve this problem, PEKKA first introduces a DAG structure to represent the dependency of the parts, where a vertex represents a part in the application task and a directed edge represents the dependency between parts. A part can start to run only when all of its predecessors are successfully processed. One can assume that the required computation workload (e.g., CPU cycles) and transmission data size for each part in a given DAG are known beforehand. An offloading plan is defined as a sequence of offloading decisions for all the parts in the DAG. See Fig. 4 as an example. Finding an optimal plan that minimizes the cost of latency and the energy consumption of the end user's device is our goal with respect to the scheduling of parts. PEKKA employs deep reinforcement learning (DRL) to search for such an optimal plan by carefully defining DRL elements, such as state space, action space and the reward function.

III. Metaverse Computing Protocol

A. Overview

As discussed in the previous section, PEKKA provides a solution for utilizing rich, low-cost and instantaneous computing power for end users by integrating worldwide decentralized clouds and then offloading computation to a proximity network to fulfill end users' needs. However, these providers of distributed computing power come from all over the world and in most cases do not enjoy a stellar reputation. Guaranteeing that the results of outsourced computing are authentic poses a challenge. In academia, this is a long-standing problem known as "verifiable computing", a fundamental problem for all shared-computing platforms. In the Computecoin network, we propose MCP, a layer-1 blockchain, to resolve this issue from a probability standpoint. Different from other layer-1 blockchains, which aim to improve throughput, enhance security, enable interoperability and more, MCP aims to build a trustable metaverse ecosystem layered on top of trustless decentralized clouds. To this end, we propose a novel consensus algorithm – proof

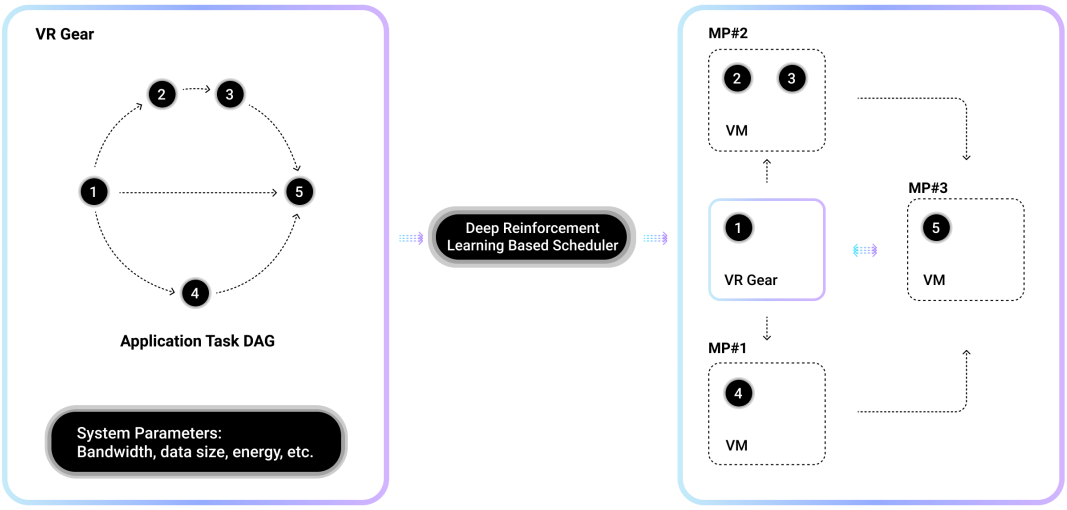

Fig. 4. Computation offloading process for application tasks with dependency. "MP" and "VM"refer to metaverse power provider and virtual machine, respectively. In this example, VR gear identifies 5 tasks with dependency as shown in DAG. The VR gear starts with processing the 1st task. Then MP 2 and MP 1 respectively process task 2&3 and task 4. Once all task 1-4 are successfully processed, MP 3 aggregates the results from MP 1 and MP 2 to process task 5.

of honesty (PoH) – as a solution to the verifiable computing problem, without which malicious nodes could overcome PEKKA and destroy the network by simply sending fake results. In this paper, we refer to PoH as "EntrapNet", which will be presented in the following section.

In addition to PoH, the MCP also has superior performance compared with other layer-1 blockchains. In particular, the MCP uses the DAG structure for its underlying blockchain technology. DAG is known to have the advantages of high concurrency, high throughput, and low transaction fees. We have deeply optimized and improved the traditional DAG technology based on our new vision and on metaverse applications' design requirements. The overall DAG achieves supreme performance by introducing certain rules on how a block selects parent blocks in the DAG structure. The MCP pushes the boundaries of public blockchain technologies specific to the metaverse in the following directions.

1) Higher throughput: in the metaverse, a massive number of transactions must be processed rapidly, which is a fundamental requirement for building the cornerstone of the metaverse. Right now, the MCP has reached 20,000+ TPS (transactions-per-second) in fully sharded deployment.
2) Easy and fast access to resources: heterogenous metaverse devices, that is, those not limited by hardware or software, could be connected to the MCP. For example, some edge devices have a very tiny amount of computing power, but they handle and store important data, and thus, to strengthen their security, they should be a part of the MCP.
3) Verifiable computation: our patented consensus algorithm "proof of honesty (PoH)" guarantees that the results of outsourced computing tasks can be proven authentic, solving a long-standing problem in the field of computer science. This is the foundation of all shared computing platforms. The details of PoH will be presented later in this section.
4) MCP supports EVM (Ethereum Virtual Machine) and is solidity-compatible. That is, it allows users to deploy any solidity-based smart contracts such as ERC20, ERC721, DeFi applications and more. It also offers a comprehensive set of developer libraries and tools for metaverse applications.
5) The MCP's unique structure allows for elegant cross-chain interoperability, which is a necessary for CCN to work as an aggregator of resources from other blockchain projects such as DFINITY and Filecoin. For example, a metaverse gamer could simply pay a Filecoin miner CCN tokens to store his or her NFTs. Neither parties needs to understand how the tokens are swapped.

In addition, the native token CCN serves as an incentive for those CCN system operations that boost the development of a global ecosystem. The CCN token's functionalities include, but are not limited to,

1) The client needs CCN tokens to pay for the computing power he or she rented from a the decentralized cloud provider.
2) Computing power hosted on the CCN platform will be rewarded in CCN tokens even if it is not rented out. That is, the network guarantees earnings 24/7.
3) Both the client and the decentralized cloud provider need CCN tokens to pledge a collateral for their participation.

B. Consensus: Proof of Honesty (PoH)

In this section, we present PoH in detail. The question of outsourcing computational tasks to another party such that the client/user can efficiently verify a result without re-executing the task represents a long-standing dilemma in the field of computer science. On one hand, providers do not necessarily have a strong incentive to ensure correctness. On the other hand, it is difficult for complex and large-scale providers (e.g., cloud servers) to guarantee that tasks are always properly executed due to misconfigurations, randomness in hardware, etc. For nearly a decade, computer scientists have studied this problem, known as *verifiable computing*.

A straightforward solution to this problem is to replicate computations on multiple computing devices [1]–[3]. However, this solution implicitly assumes that failures from these computing devices are uncorrelated. This assumption can be invalid in many cases; for example, cloud servers always have homogeneous hardware and software platforms. Another solution is based on the method of taking a small group of samples and then auditing the responses in these samples. However, if the incorrect outputs do not occur frequently, this solution may not perform well. One can also find other solutions, such as attestation [6] and trusted hardware [4], but these solutions always require a chain of trust and guaranteed correct hardware computation. One groundbreaking approach that uses the traditional technique is to force the provider to provide a *short proof* that can prove the computation's correctness. This proof should be "short and easy-to-check" compared to re-executing the computing task. Many proof-based systems have been developed recently, including Pinocchio [7], TinyRAM [8], Pantry [9] and Buffet [10]. However, immense overhead associated with the cryptographic setup between the client and provider, and the computational cost for the provider to construct such a short proof, makes such solutions impractical [12]. Another novel approach leverages blockchain technology to secure outsourced computation. One example is TrueBit [14], a smart contract that verifies computational results through a trustless economic protocol. However, the 5-to-50 times greater cost of using Truebit and its being restricted to the Ethereum network could limits the contract's mass adoption. A new approach with a similar idea to our work adds some precomputed subtasks to the original data set and verifies them upon task completion. For example, [15] has presented a mechanism to verify outsourced computation for biometric data, which inserts some fake, precomputed items into a biometric data set to detect any misbehavior by lazy servers. However, this method only focuses on specific computation algorithms and data set structures. Furthermore, inserting fake

items into every computation task for every server is inefficient.

In what follows, we introduce a novel verifiable computing protocol, called *EntrapNet*, which borrows the idea from the practice of criminal entrapment. EntrapNet can be viewed as an independent blockchain-based computing verification layer atop of a distributed shared-computing network. Therefore, it can be easily implemented to protect any distributed network that involves outsourced computing.

Next, we review the EntrapNet protocol in detail. We start out with an overview on the system architecture and then discuss the details of each key aspect of the system. First of all, EntrapNet makes the following two basic assumptions:

Assumption 1. *The blockchain network used in the EntrapNet is reliable.*

Here, the reliability of any blockchain network is defined as the ability to effectively record the creation of a digital asset and its transfer in accordance with the intended purpose. Note that how to keep a blockchain network reliable is an independent research topic that is out of the scope of this paper and will be studied separately.

Assumption 2. *All participants in the network are rational in the sense that their actions are intended to maximize individual profits.*

1) Algorithm overview: In criminal law, entrapment refers to the practice whereby a law enforcement officer induces a person to commit a criminal offense that the person would have otherwise been unlikely or unwilling to commit [13]. EntrapNet borrows such an idea, such that a client/user in the network, playing the role of "officer", aims to catch a malicious provider in the network by assigning the provider with a phishing task. Since the outcome of the phishing task is predictable to or known in advance by the officer, the officer will be able to easily detect any computing misconduct. We provide an overview of the EntrapNet mechanism, which will be discussed in two phases. Fig. 5 shows an overall illustration of EntrapNet.

Phase 1 - Preparation: A user who wishes to be an officer must build up a phishing-task repository. This repository contains one or more computing tasks the results of which are known and verified by the network. Additionally, the information of each phishing task in the repository is written to the repository smart contract on the blockchain, which will be activated later for the process of result verification. Note that based on the incentive protocol (see Section III-B6), each user can choose to be an officer or not. Meanwhile, a relatively large deposit is required

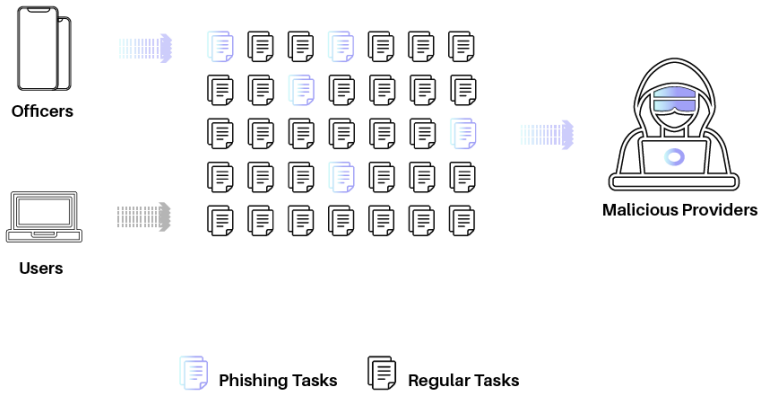

Fig. 5. EntrapNet: the task pool contains both phishing tasks and regular tasks which the malicious node cannot distinguish.

for a provider to offer its idle computing resources, which will be sent to the incentive pool on the blockchain. The incentive pool is another smart contract in charge of storing deposits from providers and rewarding the officers. It is noteworthy that such a deposit is always far greater than the reward received by the provider for correctly completing a computing task. This way, the provider would suffer a big loss if it is found to be faulty.

Phase 2 - Execution: In the second phase, the network randomly schedules a task from the task pool to a provider, in which the task pool contains both phishing tasks and regular tasks submitted to the network in a fixed time slot, e.g., one minute. The information of all the pool tasks are written into the task pool smart contract on the blockchain; the task scheduler will invoke the smart contract that randomly assigns a task to providers. Since EntrapNet ensures that a malicious provider cannot filter out the phishing tasks, it is likely that such a provider will execute a phishing task in its usual, faulty manner. Once the provider feeds back the outcome to the officer and records its hash value on the blockchain, the officer can easily identify the correctness of the results by comparing the outcome with the one in its phishing-task repository.

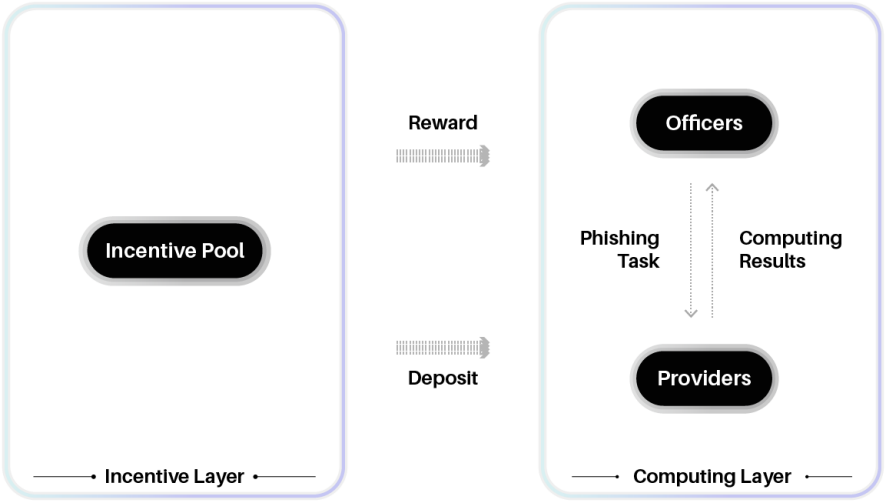

Fig. 6. EntrapNet Incentive Protocol.

If the officer deems the results faulty, he or she will both results on the blockchain via a verification smart contract. The judges, a subset or all of the nodes who run this verification smart contract, will rule on whether the officer is correct (e.g., through voting). If proved to be correct, the officer will be incentivized with a reward from the EntrapNet incentive pool whereas the provider's deposit will be forfeited and contributed to the EntrapNet incentive pool. An illustration of the aforementioned mechanism is shown in Fig. 6. EntrapNet is considered a combination of two functional layers: an incentive layer and a computing layer, both consisting of several smart contracts on a blockchain system.

One can see that EntrapNet could be a very effective means to secure outsourced computing if enforcement (of entrapment) is highly strengthened or the financial punishment for a malicious provider, if caught, is extremely severe. To strengthen enforcement, the network can simply provide a higher reward to officers who discover misconduct. Consequently, more users could be willing to serve as officers and submit phishing tasks more frequently. However, the overhead of the network in which many providers compute tasks whose results are already known and have been recorded on the blockchain could be much greater, resulting in a more secure but much less efficient system. On the other hand, EntrapNet can significantly increase the amount providers

are required to deposit. However, a prohibitively high deposit strongly discourages trustable providers from offering their computing resources. Therefore, it is nontrivial to optimize these hyper-parameters in the EntrapNet system and balance the system's security and efficiency. In Section III-C, we will present a general mathematical framework to optimize the system and provide an asymptotically optimal solution.

The following sections will describe the technical details of each EntrapNet building block, including the phishing-task repository and task scheduler, as well as result verification and incentive protocol. Each is realized by smart contracts and one smart contract can invoke another. To facilitate the reading, we present pseudocodes Algorithm 1 and Algorithm 2 for the aforementioned process in EntrapNet.

Algorithm 1 Phase I - Preparation

1: Along with the request to be an officer, a user submits a task to a set of witness nodes.
2: Each witness executes the task and sends results to the verification smart contract (Section III-B4).
3: **if** All results are matched **then**
4: The user adds this task into its phishing task repository (Section III-B2) and posts a record into the repository smart contract.
5: The request to be an officer is approved.
6: **end if**
7: **if** The results are not matched **then**
8: The request to be an officer is denied.
9: **end if**

2) Phishing-task repository: To become an officer, a user must first prepare a set of phishing tasks, the results of which must in turn be verified by the network. One approach to generating a phishing task (e.g., a Python script) is described as follows. A user first submits a regular computing task [1] to the computing network along with a flag indicating that this task will be used later as a phishing task. If the task scheduler detects such a flag (see Section III-B3), this

[1]In this paper, by submitting a task to the network, we mean that a user is submitting a task request to the blockchain network for a scheduler to schedule this task. Once scheduled, the task script (e.g., Python codes) will be transferred to the provider in a peer-to-peer manner.

Algorithm 2 Phase II- Execution

A provider sends deposit D to the incentive pool.
The task scheduler (Section III-B3) uniform randomly assigns a task to the provider.
The provider executes the task and sends the result to the officer.
if the result is identified to be correct **then**
    The provider receives the reward.
    Incentive pool sends back the deposit D to the provider when the hosting period ends.
end if
if the result is identified to be faulty **then**
    The officer initiates an appeal to the judges (Section III-B5).
    The judges run the verification smart contract (Section III-B4)
    if the appeal is valid **then**
        The officer is rewarded from the incentive pool
        Deposit D of the provider is forfeited and contributed to the incentive pool.
    end if
    if the appeal is not valid **then**
        The provider receives the reward.
        Incentive pool sends back the deposit D to the provider when the hosting period ends.
    end if
end if

task will be executed by a set of witnesses in the network.[2] These witnesses then send results to the verification smart contract (see Section III-B4 to verify the results) and the smart contract will confirm the witnesses' identity. If the witness or witnesses' identity is correct and these results are proved (by the smart contract) to be matched, one can claim that the result has been verified by the network. At this moment, information about the phishing task will be put into the repository smart contract, and its result will be recorded in the verification smart contract. The user will receive a proof associated with the task script and the corresponding results from

[2]A witness is defined as a highly reputable user with a real-world identity, such as a bank, conglomerate or big cloud server like Google Cloud.

the network. For instance, this proof can be an output of the verification smart contract when the computing results are verified. Also, no phishing task will be assigned to the same identity twice, in case this identity in the network becomes familiar with this phishing task.

The user can now add this task to his or her repository and posts a record into the repository smart contract on the blockchain, indicating that this task in his or her repository has been verified as a phishing task. In EntrapNet, such a record is an abstract (e.g. a hash via SHA-1) generated from a block containing the following attributes:

1) task script;
2) computing results;
3) network verification proof;
4) a user-specific key (e.g., a private key or a field like Bitcoin nonce).

Note that a phishing task in the officer's repository needs to be removed if this task has been used to successfully catch a malicious provider. This is because, otherwise, this provider can announce this task script on the blockchain, making this phishing task known to every user in the network.

3) Task scheduler: There are many well-studied compromise-based scheduling algorithms in the literature, such as proportional fair scheduler widely used in the communication systems. In EntrapNet, however, we suppose that malicious providers are capable of processing any information to avoid being trapped. In other words, any compromise-based design in scheduling will be utilized by malicious providers to lower the probability of their being caught. Therefore, EntrapNet employs a uniformly random task scheduler. That is, a submitted task is assigned to all providers with equal probability. Another benefit of doing so is that the computation needed for scheduling is minimized. This indicates that such a task scheduler can be easily implemented in a distributed fashion, say, executed by smart contracts.

Remark 1. *In practice, the smart contract of scheduling can be executed by one node (rather than all nodes in the network) that is dynamically selected in every scheduling interval. For instance, this node can be a witness node chosen according to the rule of round robin among all witness nodes. Another example is that the node is chosen according to the rule of proof of stake (PoS) [18] among the witness nodes. A witness who possesses p fraction of the total amount of coins among all witnesses becomes eligible to schedule the next task with probability p.*

The Task Scheduler in EntrapNet can obtain a small transaction fee from each scheduled task. In practice, as the number of task submissions from individuals follows Poisson process over time, it is likely to have no task submitted in a given time slot T, i.e., an empty queue for scheduling. We make the following assumption to facilitate system analysis and optimization (which will be seen in Section III-C).

Assumption 3. *In every time slot T there exists at least one task in the queue for scheduling.*

In other words, EntrapNet always imposes one computing task in the queue in every time slot T. To achieve this goal, for example, the witnesses in the network can be required to submit a task in every time slot T in a round-robin manner.

4) Result verification: EntrapNet has two result verification processes. One occurs in constructing the phishing-task repository and the other in verifying the outsourced computation.

In the phase of constructing the phishing-task repository, N witnesses execute the same computing task and output results, vector Y_i (the outcome from the i-th witness with $i = 1, 2, \cdots, N$). Considering the randomness inside the task script and hardware inaccuracy of the computing device, the results from multiple executions are defined to be aligned if any pair of the normalized results lie within a margin, Δ_{val}, under the Euclidean norm, i.e.,

$$\left\| \frac{Y_i}{||Y_i||} - \frac{Y_j}{||Y_j||} \right\| \leq \Delta_{val},$$

for $i \neq j$. This metric can be easily generalized to weighted norms (e.g., infinity norm) or other notions of distance. If all results are proved to be aligned, the mean value

$$\bar{Y} = \frac{1}{N} \sum_{i=1}^{N} Y_i,$$

along with the task script, the network verification proof and a user-specific key, will be used to generate a hash-based abstract and recorded on the blockchain. This indicates that this phishing task has been verified by the network and can be used later for entrapment.

Another verification process effectively ensures that the phishing-task results Y_f from the provider are correct. This occurs when a phishing task is complete by a provider and its results are fed back to the officer. The officer then checks the following:

$$\frac{||Y_f - \bar{Y}||}{||\bar{Y}||} \leq \Delta_{ver}. \tag{1}$$

If the above inequality is violated, the officer will open a case to claim misconduct by the provider. Note that in applications the margins Δ_{val} and Δ_{ver} need to be carefully tuned

(determined beforehand by the network). Additionally, the aforementioned verification can be performed on a dedicated part of the result vector rather than on the whole vector Y_i when Y_i has large dimensions.

Note that the question of how this system can handle tasks (such as machine learning tasks) as the output may be non-deterministic and/or discrete (e.g., a classification task) will be a topic for future research.

5) Judgement: If a computing outcome is identified to be faulty, the officer will initiate an appeal by submitting necessary information to the verification smart contract. This information, as an input to the smart contract, includes the provider's results Y_f and the four pieces of phishing-task information defined in Section III-B2. That is,

1) task script;
2) computing results;
3) network verification proof;
4) a user-specific key (e.g., a private key or a field like Bitcoin nonce).

Then the judges, a subset or all of the nodes in the network, will run this verification smart contract and hence rule on whether the officer's appeal is valid. The verification smart contract will execute the following in order.

1) *phishing-task verification*: Using the phishing-task information submitted by the officer, the contract will first confirm if this phishing task is a network-verified task. In particular, the contract will generate the abstract from the phishing-task information and compare it with the one stored on the blockchain which was submitted by the officer in the phase of building his phishing-task repository (Section III-B2);
2) *Result verification*: The contract will verify the results Y_f according to (1);
3) *Incentive allocation*: If the provider's result Y_f is proven incorrect, the contract will reward the officer from the incentive pool, and have the provider's deposit forfeited and contributed to the incentive pool.

6) Incentive protocol: In this section, we discuss the EntrapNet incentive protocol which could effectively discourage provider misconduct. The proposed incentive protocol is controlled by a smart contract on the blockchain, and consists of three financial components, which are presented in order as follows. Due to the anonymity of all participants, we need to consider Sybil attacks in our incentive protocol, in which the attacker subverts the network by creating a

large number of pseudonymous identities. In our paper, we assume that an attacker may create as many identities as he or she likes. The EntrapNet resists such attacks via its incentive protocol.

1) *Cost of generating phishing tasks:* When outsourcing a computing task, a user must fairly compensate the provider for computing such a task. This compensation should be at least higher than the cost to the provider for doing this computation job. As described in Section III-B2, to generate a phishing task, a task is always assigned to a set of reputable and trustable users (a.k.a. witnesses) rather than assigned, like a regular task, to only one provider in the network. Since there exist only a few witnesses in the network, if the compensation for computing a phishing task is low, a Sybil attack by consistently submitting a large number of phishing tasks in a period from a number of anonymous participants will easily occupy all witnesses' computing resources, leading to a situation in which the witnesses have no capacity to execute their normal jobs, e.g., verifying transactions on the blockchain as used in many directed acyclic graph (DAG)-based blockchain protocols [16]. As a result, EntrapNet imposes a high cost for generating a phishing task. That is, the user needs to pay much more for a phishing task that requires verification from the network than a regular task. Note, however, that if the reward for reporting a malicious provider is much higher than the cost of preparing a phishing task, the willingness of generating phishing tasks from users should remain stable.
2) *Deposit of Officer and Provider:* EntrapNet requires deposits from providers to discourage them from engaging in misconduct, and from officers to thwart Sybil attacks. Specifically, if a provider's misconduct is detected by the officer, the deposit D from the provider will be forfeited, as a penalty, and contributed to the EntrapNet's incentive pool, which is owned and maintained by the EntrapNet. The payout will be taken from the incentive pool as a reward to the officer. To balance the incentive pool and guarantee the EntrapNet's effectiveness on security, an adaptive mechanism on deposit D is imperative. At this point, the design could be quite different case by case. Nevertheless, we will provide guidance in the next section.

   On the other hand, in order to avoid a Sybil attack, a small deposit $\tilde{D}$ from the officer who reports misconduct is required. This small deposit, however, must be enough to pay for the cost of the judges (e.g., in Ethereum, at least the gas cost for running the verification smart contract) to perform the result verification. We assume $\tilde{D} << D$.

3) *Rewards for reporting misconduct:* First of all, the reward R to the officer for successfully reporting misconduct must be derived from the target security level of the network, which can be measured by the expected probability for a provider being assigned with a phishing task p. Since the expected probability p will change over time, the reward R is also dynamic and will be periodically recalculated in the system. Then, the reward R_t at time t is determined by

$$R_t = f(p_t), \tag{2}$$

where p_t is the expected probability p at time t, and f is a continuous, invertible and non-decreasing function mapping the security level to a reward. In the long run, the function f can be learned or approximated from the data set (R_t, p_t), where the expected probability p_t can be calculated by using the arrival rate of phishing tasks and regular tasks in the network (details are given in Section III-C), and the data set can be obtained by adjusting reward R in the EntrapNet and observing the real-time throughput of tasks.

Next, we discuss the non-negative property of the incentive pool, a fact that is necessary to guarantee the long-term economic operation of EntrapNet. Assume that in time slot t the expected number of true misconduct reports is n_t. Then the averaged deposit forfeited and contributed to the incentive pool is $L_t = n_t D_t$ where D_t represents the required deposit in time slot t, while the averaged reward taken from the pool to the officers is $G_t = n_t R_t$. EntrapNet balances the incentive pool by maintaining the following conservation law:

$$\lim_{T\to\infty} \left(\sum_{t=1}^{T} L_t - \sum_{t=1}^{T} G_t \right) > 0.$$

Note that this equality does not imply $D_t > R_t$ for $\forall t$. Given a target security p_t (thus R_t in (2)) and n_t reported misconduct, this equality provides basic guidance for dynamically setting the deposit D_t.

C. System optimization

In the preceding section, we have introduced the architecture and key components of EntrapNet. Although there exist several hyper-parameters to tune in applications, the most important one is the rate of submitting phishing tasks for a given arrival rate of the regular tasks. The higher the rate is, the more securely the network performs. However, a higher rate implies that the network-wise overhead of computing the phishing tasks can be overwhelming as computing phishing tasks

are a complete waste of network resources. Therefore, the tradeoff between network security and efficiency needs to be carefully studied. In the Computecoin network, we optimize this tradeoff by providing solutions to the following problem:

For a given metric on network security and efficiency, what is the optimal rate of submitting phishing tasks corresponding to the arrival rate of regular tasks in the network?

The solution is rather technical. We refer those who are interested in reading the technical paper to [20] for further details.

IV. AI Powered Self-evolution

Computecoin network greatly relies on AI technologies to enable self-evolution. The AI algorithms used in the system include, but are not limited to,

1) finding the best decentralized cloud (e.g., Filecoin, Crust, Chia, or nearby data centers) for each of the end user's compute and storage task;
2) auto-configuring a proximity network of decentralized clouds to empower end users' data processing capability;
3) optimizing the schedule of computation offloading to the proximity network.

In our proposal, we will achieve the goal of self-evolution in two phases. In Phase I, we aim to automatically tune the hyperparameter in our AI models, e.g., deep convolutional neural networks. Some of the common optimization strategies can be investigated to achieve this goal, such as grid search, random search, hill climbing and Bayesian optimization. Furthermore, more advanced optimizations such as asynchronous reinforcement learning can be studied along the line. In Phase II, we aim to resolve a long-standing struggling problem in machine learning - what is the best network architecture for a given machine learning problem? That is, one needs to search for entire algorithms from scratch for a specific machine learning task. This is challenging due to the vast and sparse search spaces. The recent ground-breaking research result, "AutoML-zero", from Google sheds some light on this research direction. They use a variant of classic evolutionary methods to search the space of algorithms. In fact, these evolutionary methods have proved very useful in discovering computer programs since the 80s. The Google team further simplified the method to make it suitable for the discovery of learning algorithms.

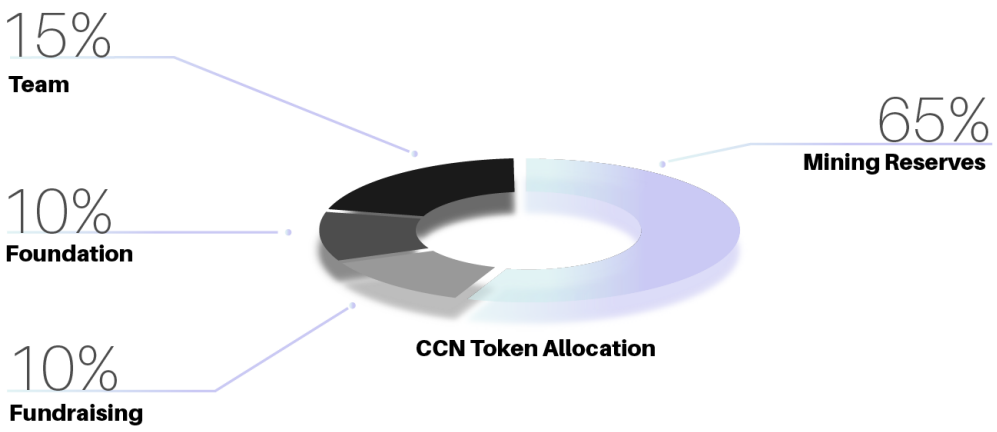

Fig. 7. CCN token allocation

Although this result is far from mature from the perspective of applications, the realization of a self-evolutionary CCN system is convincing to us and the public.

V. Tokenomics

A. CCN token allocation

The total number of CCN tokens is 2 billion, of which 15% is reserved for the founding team, 10% is allocated to early investors, 10% is reserved for the foundation, and the remaining 65% is set aside for miners.

B. CCN stakeholders and their rights

The CCN ecosystem is composed of token holders, miners, PoH verifiers, liquidity providers and developers. Balancing the rights and interests of these stakeholders and ensuring their continuous support of and contribution to CCN is essential to the project's success.

1) *Token holders*: CCN is a utility token which token holders use to purchase PEKKA services, and for community governance and elections. To incentivize CCN holders, we will continuously repurchase and burn CCN tokens with the platform's operating incomes.

The maximum burn amount is 800 million (40% of the total number of CCN). No CCN will be repurchased or burned after this limit has been reached.

2) *Miners*: Miners mainly obtain income from two sources. One source is newly minted tokens. The miners get new tokens in proportion to the aggregated performance and working time of their mining machines. Based on the computing needs of the Computecoin platform, the mining performance of machines will be comprehensively evaluated based on the machine's computing power, memory size, hard disk capacity and bandwidth. The specific performance measurement formula will be adjusted as the project progresses. Any adjustment should be approved by the community for the sake of full transparency. Miners' other source of income is the machine renting cost paid by clients. The price of renting should be determined by the supply and demand in the computing resource market. The Computecoin network will take a certain amount of service fee from each renting bill.
3) *PoH verifier*: Computecoin network is permissionless for PoH verifiers. The phishing tasks provided by PoH verifiers should run on the Computecoin platform transparently. The verifiers should continuously receive rewards from the system. A portion of these rewards will be used to cover the cost of submitting phishing tasks, and the additional part will be in the form of mining income. Verifiers can be promoted to senior verifiers based on their working time and the amount of rewards received. Senior verifiers have the right to select trusted miners.
4) *Trusted miners*: Senior PoH verifiers are eligible to designate trusted miners. Trusted miners will be more likely to attract clients and get multiple times the income of ordinary miners. The stake tokens required from trusted miners are also several times that of ordinary miners. Therefore, if trusted miners engage in misconduct, they will be punished more severely by the platform.
5) *Staking and liquidity providers*: The staking pool is a decentralized finance (DeFi) lending protocol specifically designed for CCN. It provides liquidity mining rewards to depositors and borrowers. There are two types of depositors: miners and ordinary depositors. Miners have machines bound to their stake, while ordinary depositors do not. The liquidity mining rewards are crafted to adjust with the circulation of CCN.
6) *Developers*: In addition to applying for seed funds for their project from the CCN Foundation, developers also receive continuous rewards for users and traffic imported from their apps. Computecoin will keep count of the number of active users on the platform and

reserve a portion of tokens from the mining pool for the apps that contribute to it.

C. Mint CCN tokens

The CCN token minting schedule ensures that miners can actively participate in mining as early birds, while also considering matching token circulation to the development of the ecosystem. In order to encourage miners to list machines with advanced performance, CCN has reserved a portion of the tokens to be released when the platform's capacity reaches a certain scale, in addition to the tokens released according to the half-reduction cycles.

1) *Benchmark minting*: 20% of CCN tokens are to be released according to the total computing power of the CCN network. We use the global cloud computing market revenue as a benchmark to measure computing power. According to IDC statistics, the total revenue of the global cloud computing market reached 300 billion US dollars in 2020. The Computecoin network's goal is to seize a portion of the global cloud computing market, and CCN will gradually be released until the Computecoin network achieves this goal. One example of the release schedule is summarized in the following table. The community will determine the final version of the release schedule at the beginning of the mainnet launch.

Percentage of benchmark computing power	percentage of released CCN tokens
0.08%	5%
0.4%	5%
2%	5%
10%	5%

   The benchmark minting mechanism ensures that the tokens are not excessively released before the Computecoin network has sufficient computing resources. On the other hand, the faster the Computecoin network scales, the faster reserved tokens will be released, and the higher the income of early miners.
2) *Half-reduction minting*: 40% of CCN tokens are released with a half-reduction cycle of 4 years starting from the mainnet launch.
3) *Reserves*: 5% of CCN tokens are set aside to incentivize future mining types. It will be up to the community to decide how these tokens will be issued and which stakeholders should be motivated, but at present this part of the total supply is kept in reserve.

D. Token release plan

To ensure that token holders align their interests with the project's long-term goal, vesting plans are designed for token holders before they may fully claim their tokens.

1) *Founding team*: 48 months linear release
2) *Investors*: private sale tokens to be released linearly within 12 months; public tokens to be released linearly within 8 months.
3) *Foundation*: 36 months linear release
4) *Miners' mining income*: 75% of block rewards earned by miners vest linearly over 180 days while 25% are made immediately available to miner.

E. Mining Pass and staking

The CCN mining pass is your passport to become a node on Computecoin's mainnet. The certificate is essentially a non-fungible-token (NFT) created, auctioned and issued by the Computecoin community. A mining pass indicates what type of metapower the dedicated machine(s) will be providing, as well as the maximum metapower that machine is allowed to contribute to the network.

Auctioning mining passes deflates CCN tokens, making CCN tokens more valuable. 30% of CCN tokens will be burned immediately when a miner purchases a pass after the auction. The remaining 70% will be in CCN staking for 18 months, which is the lifetime of a mining pass. Unreliable computing resources severely reduce the usability of the network, the staking from an unreliable miner will be burned for punishment.

After a mining pass expires, the staked CCN tokens will be returned to the miner. We believe this process will help us build a stronger, more secure, and reliable network by bringing together a group of committed and adventurous miners.

F. Development stage

The basic motivation for designing tokenomics is to have token circulation synchronized with Computecoin's progress and allow for the healthy development of the ecosystem. Based on the design above, the development of CCN's token model will go through the following stages:

1) *Initiation stage*: In this stage, crypto miners and individual computer providers are the main participants in the ecosystem. Crypto mining and service to small and medium-sized enterprises and individuals are the main sources of revenue for the entire platform.

2) *Ramping-up stage*: In this stage, CCN will be listed on major global exchanges and gain sufficient liquidity. The lending business in the staking pool will begin to operate and generate revenues. The entire CCN token economy will become closed-loop.
3) *Development stage*: The Computecoin network gets access to data center computers in this stage, which enables the network to provide high-performance and high-concurrency services and begin to generate strong cash flow. The PoH consensus algorithm will become an important factor supporting the scalability of the entire platform. The first benchmark minting of CCN tokens is expected to happen during this phase.
4) *Breakthrough stage*: In this stage, Computecoin network has many trusted miners, and the operation of the entire platform is greatly improved in terms of efficiency. Investment on research and development gets paid off. Research results and patented technologies are implemented to ensure the leading position of the Computecoin network as a shared cloud computing platform. The second and third benchmark minting of CCN tokens is expected to happen during this phase.
5) *Maturity stage*: In this stage, Computecoin network has many developers and users, and its overall revenue is close to 10% of the global cloud computing revenue. The last benchmark minting of CCN tokens is expected to happen during this phase.

VI. Publications

Books:

1. C. Li and M. K. Qiu, " Reinforcement Learning for Cyber-Physical Systems", CRC Press, 2019.

Academic Papers:

1. C. Li, L. Zhang and S. Fang "EntrapNet: a Blockchain-Based Verification Protocol for Trustless Computing", IEEE Journal Internet of Things, 2021

2. H. Wu, A. Ashikhmin, X. Wang, C. Li, S. Yang and L. Zhang, "Distributed LDPC Coding Scheme for Low Storage Blockchain Systems" IEEE Journal Internet of Things 7 (8), 7054-7071.

Patents:

1. C. Li, L. Zhang and S. Yang, "Methods and Apparatus for Performing Distributed Computing using Blockchain", US16/274,178, Aug. 31st, 2021 (Granted).

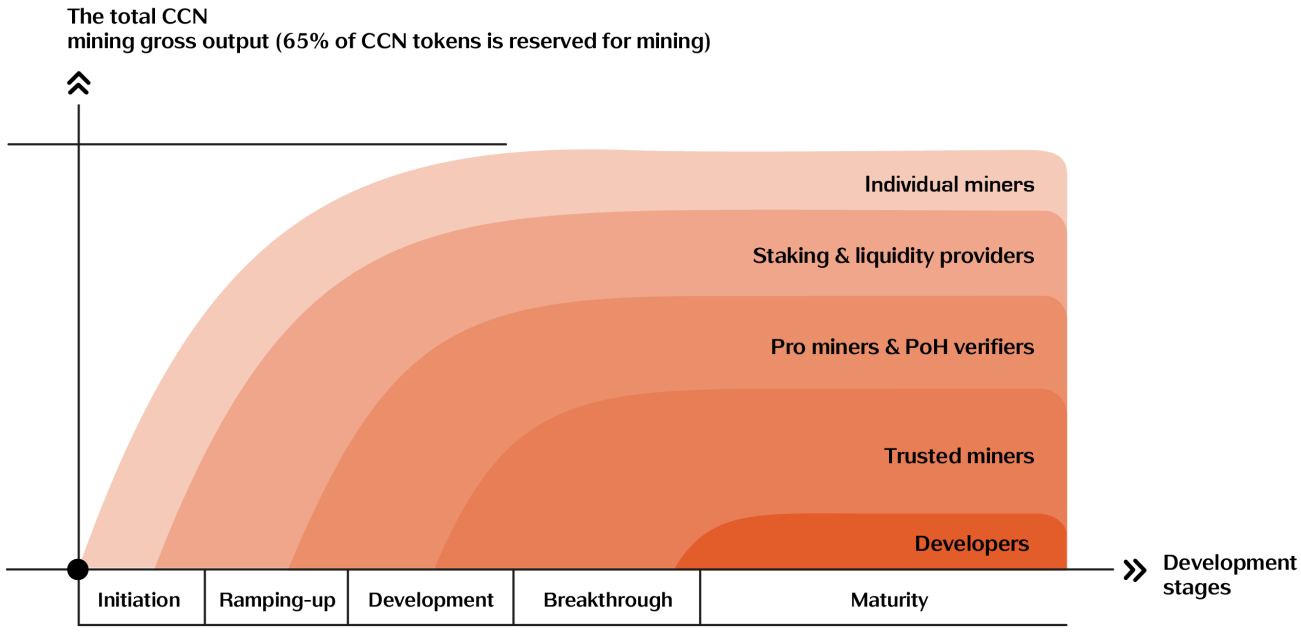

Fig. 8. CCN token mining during different development stages

2. C. Li, L. Zhang and S. Yang. “Methods and Apparatus for Verifying Processing Results and/or Taking Corrective Actions in Response to A Detected Invalid Result”, S.N. 16/370,629, April. 6th, 2021 (Granted).

VII. Conclusion

With the objective to become the cornerstone of the metaverse, Computecoin network is proposed as a self-evolving computer that serves the metaverse. This system contains two layers: PEKKA and the metaverse computing protocol (MCP). PEKKA is designed to integrate worldwide decentralized clouds, the foundation of the metaverse, and offload computation to a proximity network to supplement end users’ local compute. MCP is a layer-1 blockchain featuring the novel PoH consensus algorithm that establishes trust in outsourced computing in trustless decentralized clouds. Additionally, our infused AI algorithms, Computecoin “DNA”, will constantly self-evolve to improve users’ metaverse experiences. Based on its unique system architecture and its employment of cutting-edge technologies such as blockchain, AI and net-

working science, the Computecoin network will naturally establish a revolutionary ecosystem in the metaverse space.

REFERENCES

[1] R. Canetti, B. Riva. and G. Rothblum. "Practical delegation of computation using multiple servers". In *Proceedings of the 18th ACM conference on computer and communications security*, 2011, pp. 445-454.

[2] M. Castro, and B. Liskov. "Practical Byzantine fault tolerance and proactive recovery". *ACM Trans. on Comp. Sys*, vol. 20, no. 4, 398-461, 2002.

[3] D. Malkhi and M. Reiter, "Byzantine quorum systems", *Distributed Computing* vol. 11, no. 4, pp. 203-213, 1998.

[4] A.R.Sadeghi, T. Schneider and M. Winandy, "Token-based cloud computing: Secure outsourcing of data and arbitrary computations with lower latency". In *Proceedings of TRUST*, 2010.

[5] J. Dolezal, Z. Becvar, and T. Zeman, "Performance Evaluation of Computation Offloading from Mobile Device to the Edge of Mobile Network", *IEEE Conference on Standards for Communications and Networking (CSCN)*, 1-7, 2016.

[6] B. Parno, J.M. McCune, and A. Perrig,"Bootstrapping Trust in Modern Computers". Springer Science & Business Media, 2011.

[7] B. Parno and C. Gentry, "Pinocchio: Nearly practical verifiable computation", *IEEE Symposium on Security and Privacy*, Oakland, 2013, pp. 238-252.

[8] E. Ben-Sasson, A. Chiesa, D. Genkin, E. Tromer, and M. Virza, "SNARKs for C: verifying program executions succinctly and in zero knowledge". In *Advances in Cryptology-CRYPTO*, pp. 90-108, 2013.

[9] B. Braun, A. J. Feldman, Z. Ren, S. Setty, A. J. Blumberg, and M. Walfish. "Verifying computations with state". In *Proceedings of SOSP*, Nov. 2013.

[10] R. S. Wahby, S. Setty, Z. Ren, A. J. Blumberg and M. Walfish. "Efficient RAM and control flow in verifiable outsourced computation." In *Proceedings of NDSS*, Feb. 2015.

[11] A. Greenberg *et al*.,"The Cost of a Cloud: Research Problems in Data Center Networks" *ACM Computer Communication Rev.*, vol. 39, no. 1, pp. 68–73, 2008.

[12] M. Walfish and A. J. Blumberg, "Verifying computations without reexecuting them." *Communications of the ACM* 58.2, pp. 74-84, 2015.

[13] Wikipedia, "https://en.wikipedia.org/wiki/Entrapment".

[14] J. Teutsch and C. Reitwiessner, "A scalable verification solution for blockchains", *online*, Mar 2017.

[15] Blanton, Marina, Yihua Zhang, and Keith B. Frikken. "Secure and verifiable outsourcing of large-scale biometric computations." ACM Transactions on Information and System Security (TISSEC) 16.3 (2013): 1-33.

[16] A. Churyumov, "Byteball: A Decentralized System for Storage and Transfer of Value" https://byteball. org/Byteball.pdf, 2016

[17] C. D. Coath, R. C. J. Steele and W. Fred Lunnon, "Statistical bias in isotope ratios", *Journal of Analytical Atomic Spectrometry*, 28 (1), pp. 52-58, 2013.

[18] I. Bentov, A. Gabizon, and A. Mizrahi. "Cryptocurrencies without proof of work." In International Conference on Financial Cryptography and Data Security, pp. 142-157. Springer, Berlin, Heidelberg, 2016.

[19] L. M. Bach, B. Mihaljevic and M. Zagar, "Comparative analysis of blockchain consensus algorithms", 41st International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO), 2018.

[20] C. Li, L. Zhang and S.Fang, "EntrapNet: a Blockchain-Based Verification Protocol for Trustless Computing", IEEE Journal Internet of Things, 2021.