Whitepaper

Version Chiang Mai Rev3

Hyperdust Team
2024

Content

Make AI open again.

AI impacts every intelligent living being on this planet and its development also requires globally human collaboration. But we've seen instances where AI

has been promised as open but turns out to closed, where AI was intended to be used to improve human welfare but is used as monopoly.

Do not trust any alliance on paper, trust no one, trust the code. Build AI with blockchain technologies. Make AI open again.

1. Introduction

The first application categories (financial application) listed in the whitepaper of Ethereum are well implemented and practiced. But the second category (semi-financial applications) are still in progress. Some solutions are presented and implemented but we are still missing self-enforcing bounties innovations for solutions to intensive computational problems. Metaverse and AI are such intensive computational problems which could and should be self-enforcing bounties.

Ethereum's smart contracts are designed primarily for financial applications, which is why they must adhere to strict EVM execution. While this simple execution environment, represented by agents (smart contracts), is Turing complete, it falls short in addressing computationally intensive tasks. Smart contracts can help establish self-enforcing incentive mechanisms to tackle these challenges. However, relying solely on smart contracts as agents is insufficient. Hence, we have introduced a new class of computationally intensive Autonomous Agents, such as 3D Agents and AI Agents.

To cater to computationally intensive applications, solutions have been proposed beyond the expensive smart contract validation. Technologies like off-chain virtual machine execution with on-chain arbitration, known as Layer 2 (L2) technology, have been introduced. Nevertheless, these technologies still fall short of meeting the real-time 3D computing demands of the metaverse. Calculations within 3D Pipelines, such as translations, rotations, Euler transformations, and rendering, are challenging to define using off-chain virtual machine instructions like memory and registers. Similar issues exist in AI training and inference computations. Inspired by the Agatha project's graph-based Pinpoint protocol, we have combined the characteristics of 3D computing to propose a 3D pipeline-based Pinpoint protocol for trustworthy 3D computation. Later, we will demonstrate that these two protocols are equivalent algorithms. AIGC and 3D computing are the primary computational drivers of the metaverse. Until now, they lacked viable decentralized trust mechanisms. Through our innovation, we have achieved decentralized trustworthy computation foundations for computation intensive applications (such as 3D rendering and AI).

2. Related Works

To expand the computational capacity of blockchain, various solutions involving off-chain virtual machine execution and on-chain Pinpoint protocol arbitration have been proposed, such as TrueBit, Arbitrum, YODA, ACE, and Optimism. However, there is still a lack of technical solutions for intensive computational applications such as real-time rendering, AI model training, and inference.

Inspired by the AnyTrust approach of Arbitrum, the Agatha project has introduced a two-stage Pinpoint protocol to achieve trustworthy deep neural network (DNN) inference computations. Agatha utilizes local clients instead of virtual machine execution for off-chain computations. The computation is divided into two layers, namely, operations (OPs) and basic operations (BOPs). Both submitters and verifiers perform calculations using local clients. In cases where verifiers disagree with the results, an arbitration mechanism is initiated, involving submission nodes, verification nodes, and the Pinpoint smart contract. Verification nodes first identify which OP (e.g., the second convolutional layer operation in a deep neural network) is in question, then locate the specific BOP (e.g., a 32-bit floating-point addition operation) within that OP. This BOP and its input are submitted to the smart contract for arbitration. The smart contract compares the on-chain computation results with those of the verification and submission nodes and supports the side with matching results, thereby obtaining a trustworthy outcome.

OpCraft introduces an intelligent contract pattern based on ECS (Entity, Component, System) aiming to establish a fully on-chain game or autonomous world. It achieves some on-chain computations other than asset-related contracts, such as calculations for position changes or player block placement/removal. However, even when OpCraft operates on a Layer 2 chain, its actual feasibility remains uncertain. Moreover, there is a lack of validation and trust mechanisms in rendering and presentation layer computations, making it incapable of meeting the trustworthy computation requirements for AAA-quality XR and AIGC metaverses.

Metaverse is not only about virtual space but also about Artificial General Intelligence (AGI). Artificial Neural Network (ANN) is a fundamental of generative artificial intelligence. But the nature of intelligence is not well studied or not found yet. Since 2004, bio-inspired Aspect-Oriented Artificial Intelligence (AOAI) has been proposed and implemented to solve the MONK problem. This would be the first endeavor to define the nature of intelligence and program software run in light of self-evolve intelligence. With recent research of causal emergence theory, advanced GPU with 100k kernels running parallel and breakthrough of LLM, we believe that AOAI will functions like convergent part of AI with cross-cutting all other generative AI models, e.g. neutral language text, voice, image, actions etc. (this will be discussed in detail in section *)

Some thin protocols are developing to integrate game engines and blockchain, e.g. Emergence Protocol. This is a utility layer to integrate game engines and blockchain for convenience. But as discussed above we need fundamental infrastructure to build financial grade trust virtual space and it is still unavailable.

Render Networks is an early attempt to realize decentralized graphic rendering but it has two drawbacks. First, it only supports off-line rendering for CG or videos rather than real time rendering needed by games and metaverse. And all rendering is done off chain. Only the rendering machine matching, deals and payments are regulated by smart contracts.

3. Hyperdust System

Hyperdust is a protocol designed for creating decentralized compute-intensive applications. It operates under the HyperTrust assumption and enhancing virtual machines with more powerful local clients for calculations based on computation graphs and 3D pipelines using the Pinpoint protocol. It integrates computational capabilities into the blockchain. Unlike rollup scaling systems such as Optimistic Rollup (OP), Arbitrum, or ZK, Hyperdust provides a significantly more powerful execution environment than Ethereum. It enables the implementation of more powerful autonomous agents than Ethereum smart contracts, including features like AAA cloud rendering spaces, AI training and inference, and digital beings, collectively referred to as HAoC (Hyper Agents on Chain). Hyperdust builds upon existing L2 chains but adds the HyperTrust consensus algorithm, integrating Graph Virtual Machine with (GVM) Ethereum Virtual Machine (EVM) to achieve enhanced computational capabilities while maintaining security similar to the AnyTrust protocol.

3.1. Architecture

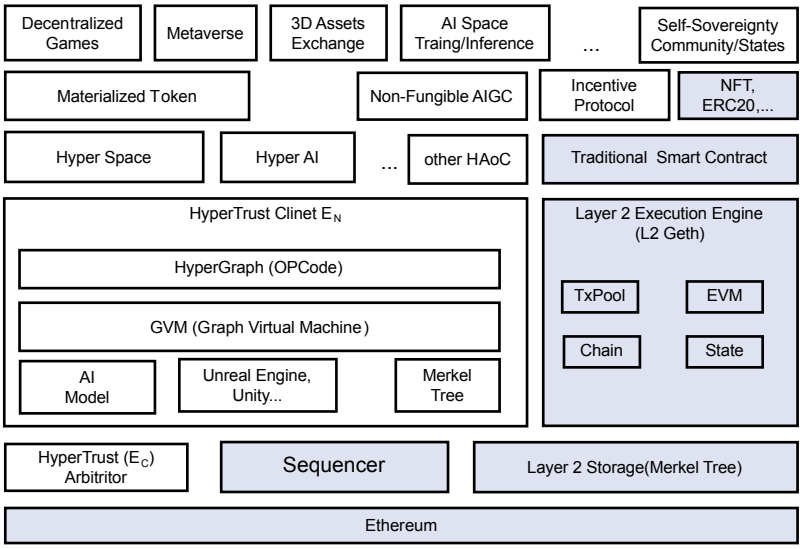


Figure : Sytem Architecture of Hyperdust

HyperTrust (E_c) is an smart contract deployed on the underlying blockchain, which can be deployed on Layer 2 chains or directly on Ethereum. The former option offers lower costs, while the latter provides better security.

3.2. Hyperdust Account and Autonomous Agent

Like Ethereum, Hyperdust's state consists of objects called "accounts" and state transitions that transfer value and information between these accounts. An account comprises four parts:

1. A random number used as a counter to ensure each transaction can only be processed once.
2. The current token balance of the account.
3. The account's Agent code, if it exists.
4. The account's storage (which is empty by default).

HYPT is the primary cryptographical fuel within Hyperdust, used to pay transaction fees.

In the Hyperdust system, Autonomous Agents are classified into two main categories. One category consists of smart contracts executed by the EVM, while the other category involves off-chain clients executing on-chain smart contract validations for agents, such as the Hyper Space Agent or Hyper AI Agent. These agents are extensions of smart contracts and have accounts similar to those of regular Hyperdust accounts. Whenever an Agent account receives a message, the Agent's internal code is activated, allowing it to read and write to its internal storage, send other messages, or create new Agents.

3.3. Hyper Space Agent

The HyperSpace Agent, like smart contracts, represents a verifiable and trusted 3D computing, AI gym space. Users interact with it using their private keys through the Hyper Space interface. A Hyper Space address is a mapping of an account address and balance, similar to a regular on-chain smart contract. Like smart contracts, a Hyper Space address is composed of an address and a hash of the corresponding storage space file (referred to as a level file in the gaming industry). For example, creating a space token on Hyperdust creates a smart space account, and the resulting address is a smart space address without a private key.

As previously described in Section 3, unlike smart contract agents, the Hyper Space Agent does not run on the EVM but operates locally on EN (Entity Node) through the HyperTrust protocol, utilizing the smart contract Ec for validation.

Hyperdust does not concern itself with all the code running locally; the focus is on the code defined by the computation graph (Section 4.4), which is abstracted into EOP sequences in HyperTrust. These computation results go through a submission and verification mechanism and are ultimately confirmed or rejected by the smart contract Ec. Tokens within the smart contract agent are transferred into the smart space agent through a bridging mechanism.

Similar to smart contracts, once a Hyper Space Agent is deployed, the corresponding HyperGraph and EN are saved on various nodes through a synchronization mechanism. The HyperGraph defines all spatial interactions (transactions) and generates intermediate OP sequences during runtime.

Any object used in space must be materialized as an NFT (Materialized NFT). Regular external NFTs need to undergo materialization before entering space, and new materialized NFTs can also be issued through the Hyper Space Agent. Materialized NFTs are interactive objects in space and possess attributes such as volume and mass. The operations of materialized NFTs in space (i.e., the aforementioned OPs) require GAS to execute.

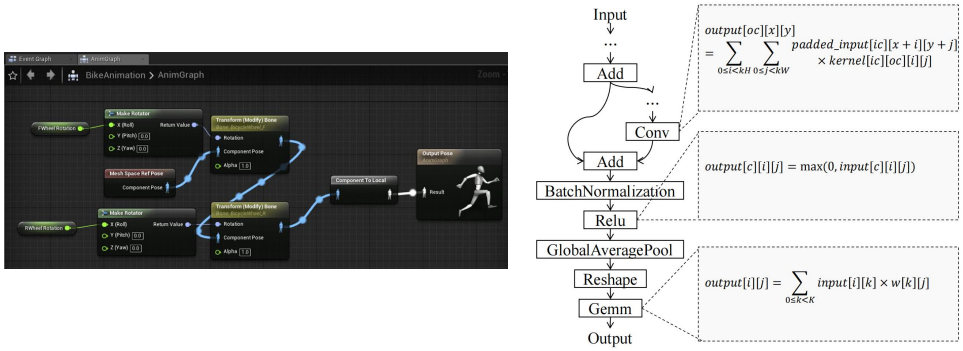
The HyperGraph specifies the rules for the operations of materialized NFTs, which are the transactions of the Hyper Space Agent and are saved as a Merkel Tree. This data is also stored on the (L2) chain to ensure its validity.

Transactions between external accounts involving materialized NFTs in Hyperdust are still handled by traditional smart contracts, and the execution of these traditional smart contracts also requires the payment of GAS fees.

3.4. Hyper AI Agent

While both AI and 3D pipeline calculations are based on the HyperTrust protocol, they operate on different clients to perform calculations. As a result, the mechanisms for overhead are different. Therefore, it is necessary to treat AI as a new autonomous on-chain agent, referred to as the Hyper AI Agent.

3.5. Graph Virtual Machine (GVM)



a. AnimaGraph in Unreal Engine Blueprints

b. The computation graph of ResNet50

Figure : Computational Graphs in 3D and AI computaion

A game engine's toolkit typically includes visual scripting, which is generally represented as a graph consisting of nodes representing computations and edges representing messages. Using these graphs, developers can achieve zero-code development through actions like drag-and-drop, as seen in tools like UE's Blueprint and Unity's Visual Scripts. AI calculations also employ similar computation graphs, such as in TensorFlow. In the case of deep neural networks, each layer of the network represents a computational node, and the network parameters form the edges of the computation graph. Below are two examples.

Hyperdust also provides developers with a similar graphical language called Hypergraph. In the graph, vertices represent OPs (Operations), and edges represent events. The hypergraph generates the corresponding BOP (Basic Operation) sequence at runtime. Before execution, the specific execution path is not known. Therefore, each client generates the execution of OPs at runtime. Results are periodically submitted, and if the results differ, the system identifies which OP caused the difference, down to the level of BOPs. Then, executing that specific OP's corresponding BOP on the blockchain can determine which party, the submitter or verifier, computed correctly.

Hypergraph is loaded and executed by the GVM (Graph Virtual Machine), and executing GVM requires paying GAS in HYPT.

3.6. Fees

There are different types of Hyper Agents, such as 3D and AI Hyper Agents. These Hyper Agents are powered by network nodes with GPU capabilities and storage and charge a certain amount of Hyper Agent GAS.

Hyper Agent Gas represents the workload and can correspond to the number of BOPs (Basic Operations) in Hyper Agent calculations. The cost of Hyper Agent GAS fluctuates based on network congestion levels.

When a user deploys a 3D Hyper Agent, they can transfer NFTs to or from the smart space and incur Agent GAS fees for smart space execution (hypergraph execution). The fees are determined by the workload and congestion levels. Minting and trading 3D NFTs occur within the 3D Pipeline and involve 3D GAS fees. Similarly, when a user triggers an AI Hyper Agent for computation, they will pay AI Hyper Agent GAS fees accordingly.

3.7. Computation and Turing-Completness

Both Unreal Engine Blueprints and Tensorflow computational graphs are turing-complete. At present only partial operation of Blueprint and computational graphs are supported in Evaluator EOP. Eventually , EOP will be turing-complete.

3.8. Blockchain and Mining

Utilizing the HyperTrust protocol, we have implemented trustworthy validation for 3D and AIGC (Artificial Intelligence Graphics Computation) calculations. With the addition of suitable incentive mechanisms, we can establish a continuous computing (cloud 3D/XR rendering and AI generation etc.) network, where computational resources are provided by contributors. The incentive mechanisms can include PoW (Proof of Work), PoS (Proof of Stake), or hybrid models. Clearly, since both 3D and AIGC involve computationally intensive tasks, we should initially consider a PoW-based incentive model.

Bitcoin, being one of the earliest and simplest PoW systems, uses computational power to guess inputs that satisfy certain mathematical conditions for hashing results, without any practical application. We propose a PoW model based on HyperTrust. In this model, the HyperTrust validation network checks whether network nodes (computational

resource providers) have correctly completed specified 3D, AI computations, or equivalent computationally intensive tasks and then rewards these nodes accordingly.

Computational resource providers download the E_N client and run it on their nodes. The node includes a matching algorithm.

4. Applications

The first category applications on Hyperdust is semi-financial applications. Such as self-enforcing bounties for solutions to intensive computational problems, AAA games, metaverse and AIGC are perfect examples. And some immersive social, virtual shows, performance that are not financial at all.

4.1. Higher image quality cloud rendering service

The key feature of computing power driven business is the marginal costs of computing power is never close to zero, which is the most important factor of traditional internet business(a.k.a zero marginal cost business), such as search engine, commercial etc. But computation intensive businesses have non-zero marginal cost business models.

4.2. Decentralized AI Application

Deep learning gets popularity in recent years. Lots of AI apps are building right now, but often the builders need to pay high cost of cloud AI computation. With Hyperdust protocol the computation powers are accessible for AI application without any intermediates. With Hyperdust the large-scale brain model such as Spaun 2.0 [12] will be possible with sustainable computing power economy and collaboration method. The Large language models, or LLMs, are a type of AI that can mimic human intelligence, requires vast computing power and investment. Along with the progress of LLMs, the concerns about so called AI abyss is also increasing. With Hyperdust AI could grow and owned by token holders that will decrease the centralized control from AI capital [13].

4.3. Materialized Token

3D NFT is not just NFT associated with 3D model files but also its material-like merits, such as width, length and depth and scales, as digital materialism. Digital materialism is actually a matter of digital economics. it addressed how much computing power is

needed to make it “alive” or “real” for us. Every physical world's assets need space to be existing. All 3D NFTs need computing power to be stored and rendered to be existing. Otherwise as we find out in 2D NFT’s fading market 3D NFTs may be hard to be a solid foundation of a new digital world.

Different from smart contracts, HyperSpace primarily manages and provides various read-write accesses to vector tokens denoted as "H." These vector tokens, unlike typical tokens described with a single value, have three scalars that can be represented as Tx, Ty, and Tz. Their exchange unit is Value = Tx * Ty * Tz, and the value of the token represents the tensor product of the intelligent space.

Similar to fungible token (ERC20), there is MFT (Materialized Fungible Token) in hyperspace. MFT token has the same geometries. For instance, the 1000x1000x1000 meters hyper space could release 1,000,000,000 materialized tokens that each has one cubic meter volume.

4.4. Virtual Environment for embodied AI

There has been an emerging paradigm shift from the era of “internet AI” to “embodied AI,” where AI algorithms and agents no longer learn from datasets of images, videos or text curated primarily from the internet. Instead, they learn through interactions with their environments from an egocentric perception similar to humans. Such as 3D Gaussian Splatting training and rendering could be hosted on Hyperdust that could create an virtual environment for all purposes.

4.5. Cloud Games/Digital Gemstones

Cloud gaming is an obvious application of Hyperdust. Moreover with the anti-cheating nature of HyperTrust new Proof-of-Gaming could be easily implemented such as digital gemstones of Brilliantcrypto.

4.6. Non-Fungible AIGC/AI Model as Assets

AIGC is often in form of text, image or videos as ordinary digital contents rather digital assets. AIGC should represents the invest and computing efforts behind the visual or auditory forms. With Hyperdust it is easily to implement AI assets such as non-fungible AI generative presented in MOSSAI, virtual islands generated by AI as Non-Fungible Generatives (NFG) [14].

5. Put it together： Decentralized Computation Intensive Application Platform

There are two remarkable milestones in the web3 industry, Bitcoin and Ethereum. The first one is the genesis of everything. Bitcoin is the first time we built up something in virtual digital form and last as real physical matter. Then Ethereum built a smart contract execution machine with such digital matter (blockchain). Almost ten years have passed, tremendous efforts and speculation have been drawn from the worldwide. Bitcoin becomes digital gold and the web3 finance applications prospers as expected in the white paper of Ethereum released in 2014. However, what is still missing is the good practice of semi-finance application identified as second category application in Ethereum whitepaper.

Hyperdust enhances Ethereum’s computation capability by creating new types of autonomous agent after smart contract. That makes decentralized computationally intensive application driven by tokens possible. Permission-less metaverse, self-owned virtual world, large scale artificial intelligence network with no so called abyss of AI capitalism.

6. Appendix A HyperTrust and bisection protocol[1]

You've correctly identified a significant challenge when it comes to validating high-density computations in a decentralized manner—the overhead can become prohibitively expensive. In scenarios like a typical metaverse or multiplayer online game, physical nodes are organized in a P2P or star-shaped network. The world state of the metaverse or game needs essential network state synchronization to ensure that all participants see a consistent scene and character status, including elements like weather, the positions of digital avatars, and the status of objects (translation, rotation, scaling), among others. These changes in state are synchronized using various mechanisms, such as by frame ticks or message-based penalties, resulting in a real-time data stream. We can think of these changes as transactions from state A to state B.

At regular time intervals, these transactions can be organized into a block based on a Merkle tree. If you include a hash pointing to the previous block in these blocks, you form a blockchain—essentially, this is the fundamental principle of a blockchain. However, all nodes are generating a portion of this data stream, including local user input that alters the world state and the synchronized collection of the world state from other nodes. If all these nodes join in an permission-less manner (meaning they are untrusted nodes), then you'd have to validate the data streams submitted by each party one by one using a specific algorithm.

This validation process, as you rightly pointed out, can result in a substantial overhead. For instance, in a simple world scene with 100 characters, even at a frequency of 15 transactions per second, you'd have 1500 transactions per second. This already exceeds the transaction processing capabilities (TPS) of many high-performance blockchains. Even with a 3-5 second delay, it could potentially consume the entire TPS of the blockchain and generate significant gas fees.

This challenge highlights the need for innovative solutions that can efficiently handle high-density computations and validate them in a decentralized manner without incurring excessive overhead. It's an area where further research and development are essential to enable complex real-time applications like metaverses and multiplayer online games to function smoothly on blockchain networks.

Genesis Block	block 1	block 2
Null	<------Hash of previous block	<-----Hash of previous block
CheckPoint Root	CheckPoint Root	CheckPoint Root
TX1...TXn	TXn+1...TXn+m	TXn+m+1...TXn+k

6.1. Related Technologies

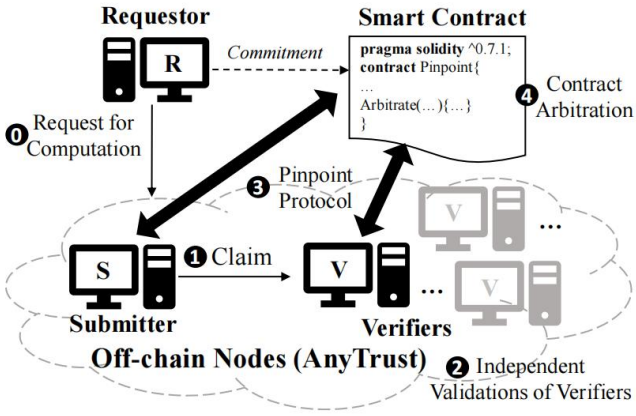


Figure 1: The essence of the previous approaches

There are various assumptions about the verifiers to guarantee security and liveness, such as AnyTrust , Financially Rational Players and the hybrid Byzantine assumption. For simplicity, we use the AnyTrust assumption from Arbitrum.

The AnyTrust assumption.
AnyTrust, rather than “majority trust”, assumes that, for every claim, there is at least one honest node. Namely, either the submitter is honest, or at least one verifier is honest and

will challenge within the predefined period. Suppose there are m verifiers, m−1 of whom collude to stay silent about a submitter's wrong claim. Under AnyTrust, the only honest verifier will still succeed in disproving the wrong claim in front of the smart contract, and the wrong claim is thus rejected. Like the previous work, we also assume data availability (i.e., D and should be accessible to all verifiers) and anti-censorship (i.e., every verifier can always interact with the contract). They are solved by orthogonal countermeasures.

The pinpoint protocol.

The pinpoint protocol also needs more explanations. The goal of the protocol is shown in Figure 2. The computation consists of a sequence of VM instructions (denoted as VMI_1, VMI_2, . . . , VM_n) and the initial VM state is S_0. However, the verifier and the submitter get different states $S \neq S'$. The goal of the protocol is to pinpoint VMI, such that $S_{k-1} = S'_{k-1}$ but $S_k \neq S_k'$, and to send (VMI, S_{k-1} , S_k) to contract for arbitration. The submitter and the verifier are forced to get the same with a challenge-response mechanism with a timeout penalty. The pinpoint protocol is very efficient: for time complexity, the verifier and the submitter only need (log) rounds of challenge response for both parties to commit to the number , and the contract only needs a constant time to arbitrate the disagreement about VMI; regarding the space complexity, the state is structured as a Merkle tree (MT), corresponding to a logarithmic-size message to the contract.

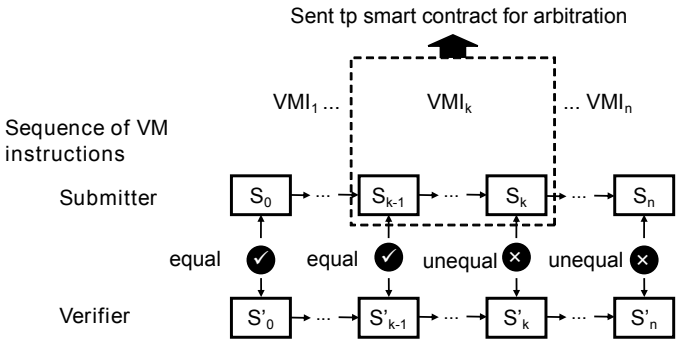


Figure : Pinpointing a disputed step in VM execution

The overhead of off-chain computation.

Besides the overhead of contract execution (i.e., on-chain part), the overhead of off-chain execution is also critical since the off-chain overhead determines the workload and throughput of off-chain nodes. For example, submitters or verifiers in TrueBit earn bounties proportional to the length of VM instructions, which is the cost to post the computation task for the requestor; Arbitrum VM leverages the Intel SHA extension to achieve a high off-chain throughput. Furthermore, in DeFi applications, the off-chain

latency directly determines the efficiency of the market because of the off-chain racing counterparties (e.g., liquidity providers and traders).

6.2. Hypertrust Assumption

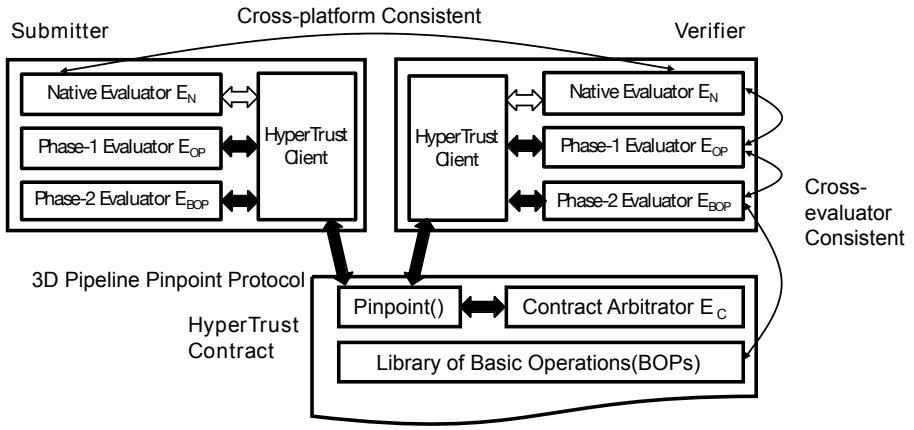


Figure : The overview of HyperTrust system

HyperTrust comprises smart contracts deployed on a blockchain and several standalone HyperTrust clients.The HyperTrust system consists of a smart contract deployed on blockchain and several independent HyperTrust clients. For a metaverse world/space computation task, the HyperTrust system focuses on the verification process after one HyperTrust client submits a result of the task (specifically, the hash value of the result). If any client finds the result incorrect, and then wins the challenge-response dispute following the HyperTrust pinpoint protocol, the contract will reject the submitter's result. Otherwise, the submitter's result is accepted.

Similar to the previous technologies, HyperTrust has a workflow to reduce the on-chain overhead. However, the key difference is that HyperTrust enables the native 3D and AI computation which reduces the off-chain overhead significantly leveraging the multiple (redundancy) computing nodes in the decentralized network. The design of the HyperTrust system is shown in Figure 3 and explained in this section.

6.3. Native execution

Similar to Agatha, HyperTrust is enabling native execution in scalable contracts for 3D or AI computation (see Table 1). There is a native evaluator in each client. In HyperTrust, both submitter and verifier execute 3D or AI computation with E_N, instead of a restricted single-threaded VM. E_N is a module on top of 3D engine (such as Unreal Engine, Unity or O3DE etc.) or LLM model. Using E_N significantly reduces both the submitter's execution latency and the verifier's validation latency.

Steps	Previous	Hypothetical	Agatha
Submitter's execution	VM	Native	Native
Verifiers' validation	VM	Native	Native
Pinpoint protocol	VM	VM	Semi-native
Contract arbitration	VMI	VMI	BOP

Table 1: Comparison of the previous work, a hypothetical method and Agatha

Similar to game engines, every object within the 3D AI gym scene is referred to as an "Actor." Each Actor can be individually configured for synchronization (and can have multiple components attached to it, each of which can be synchronized independently). The "Replicated" property of an Actor at a given moment is essentially its state information. In the 3D AI gym, when a node (let's call it Node En) ticks each frame, it determines which Actors should be synchronized, which attributes need to be synchronized, and then serializes them into binary form (think of it as an incremental snapshot of the current world state). This binary data is sent to the corresponding Node En, which can further process it through callback functions upon receipt.

6.4. Unified 3D-Pipeline-Based paradigm

We express a 3D computation as a matrix computation consisting of operations. A 3D pipeline is a queue with 3D related operations. For example, Figure 4 .... An operation is defined mathematically using operations, such as translation, rotation and scale. In the rest of the paper, we use the terminology “basic operation”, or BOP, to refer to an individual computation of actors. The terminology “operation” or OP, without a preceding “basic”, means the computation of 3D transformation pipeline.

Each HyperTrust client leverages a phase-1 evaluator and a phase-2 evaluator, used for our two-phase pinpoint protocol. In phase 1, the 3D transformation computation in dispute invokes to compute the results of the operations, and then the HyperTrust contract can locate an operation in dispute. In phase 2, the operation in dispute further invokes to generate the results of the basic operations, and thus the contract can locate a basic operation in dispute. After pinpointing, the contract performs the arbitration by calculating the basic operation on blockchain with its on-chain evaluator .

The simplicity comes from the world state stream. For this protocol, the arbitration contract only needs to understand 2D/3D leaner algebras (matrix), but not other complicated machinery; the client only needs to understand 3D Transformation Pipeline, but not sophisticated VM instructions and states.

The efficiency benefits from both native execution and the nature of 3D transformation computation. E_{OP} can invoke the high-performance E_N, and E_{BOP} can make arbitration simple. Imagine two alternative pinpoint protocols. The first one, without phase 1, uses E_{BOP} to generate internal intermediate results of the whole graph, and lets the contract locate a basic operation in dispute. This is too inefficient for a 3D transformation computation, which usually has billions of basic operations. The second one, without phase 2, demands the contract to directly locate an operation in dispute, and then arbitrate the operation by re-executing it on-chain. This is not practical for 3D transformation computations, whose operations are too complex to implement and execute in smart contracts.

Translation, Rotation, and Scaling: the Three Fundamental Transformations

Whenever we move objects around in 3D space, we do so using mathematical operations called transformations. We've already seen two kinds of transformation: translation, stored in an object's .position property, and rotation, stored in the .rotation property. Along with scaling, stored in the .scale property, these make up the three fundamental transformations that we'll use to move objects around in our scenes. We'll sometimes refer to transform, rotate, and scale using their initials, TRS.

$$R_x(\phi) = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\phi & -\sin\phi & 0 \\ 0 & \sin\phi & \cos\phi & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

$$R_z(\phi) = \begin{pmatrix} \cos\phi & -\sin\phi & 0 & 0 \\ \sin\phi & \cos\phi & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

$$T(t) = \begin{pmatrix} 1 & 0 & 0 & t_x \\ 0 & 1 & 0 & t_y \\ 0 & 0 & 1 & t_z \\ 0 & 0 & 0 & 1 \end{pmatrix} \quad R_y(\phi) = \begin{pmatrix} \cos\phi & 0 & \sin\phi & 0 \\ 0 & 1 & 0 & 0 \\ -\sin\phi & 0 & \cos\phi & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \quad S(s) = \begin{pmatrix} s_x & 0 & 0 & 0 \\ 0 & s_y & 0 & 0 \\ 0 & 0 & s_z & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

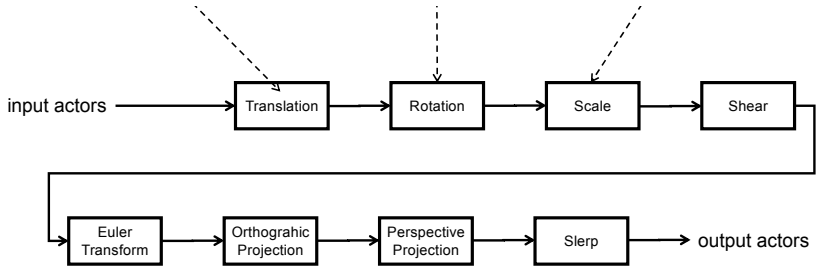
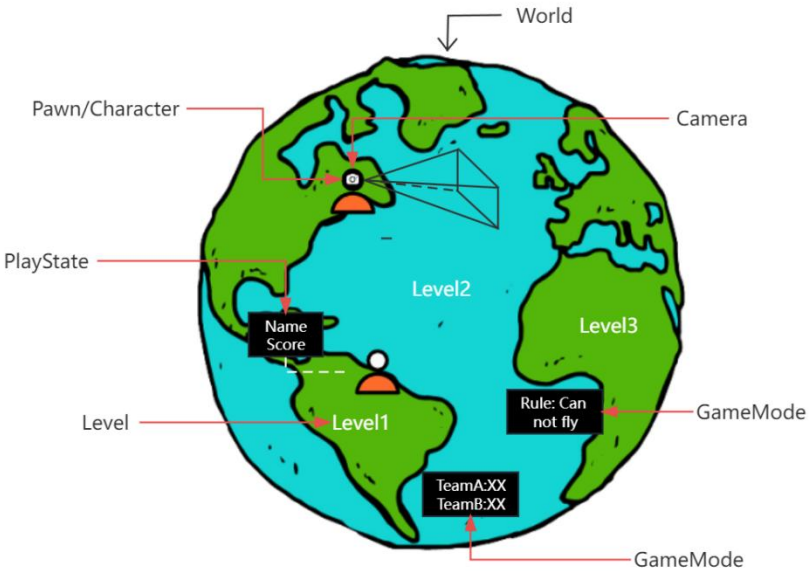


Figure :

"Transform" is an operation that is applied to points, vectors, or colors in a specific way. Through Transform operations, you can position, reshape, and animate objects, lights, and cameras. They allow you to bring all calculations into a common coordinate system and project objects in various ways. The image you mentioned shows geometric transformations in a 3D Pipeline. Other types of transformations can include view pipelines, rendering pipelines, and more. It's important to note that these pipelines are dynamically generated by Electronic Operations (EOP), where each node represents an operation, such as Translation, Rotation, and so on.

Notation	Name	Characteristics
$\mathbf{T}(\mathbf{t})$	translation matrix	Moves a point. Affine.
$\mathbf{R}_x(\rho)$	rotation matrix	Rotates ρ radians around the x-axis. Similar notation for the y- and z-axes. Orthogonal & affine.
$\mathbf{R}$	rotation matrix	Any rotation matrix. Orthogonal & affine.
$\mathbf{S}(\mathbf{s})$	scaling matrix	Scales along all x-, y-, and z-axes according to s. Affine.
$\mathbf{H}_{ij}(s)$	shear matrix	Shears component i by a factor s, with respect to component j. $i, j \in \{x, y, z\}$. Affine.
$\mathbf{E}(h, p, r)$	Euler transform	Orientation matrix given by the Euler angles head (yaw), pitch, roll. Orthogonal & affine.
$\mathbf{P}_o(s)$	orthographic projection	Parallel projects onto some plane or to a volume. Affine.
$\mathbf{P}_p(s)$	perspective projection	Projects with perspective onto a plane or to a volume.
$\mathtt{slerp}(\hat{\mathbf{q}}, \hat{\mathbf{r}}, t)$	slerp transform	Creates an interpolated quaternion with respect to the quaternions $\hat{\mathbf{q}}$ and $\hat{\mathbf{r}}$, and the parameter t.

In the image, we can see that both the input and output of the 3D transformation pipeline are "Actors." Here, we are using the terminology from Unreal Engine, which essentially refers to objects within the virtual scene.



Similar to a game engine, the design of the metaverse scene follows an object-oriented approach and involves common concepts (classes) as follows:

- **World**: Corresponds to a world.
- **dSpace**: Corresponds to a sub-level or sub-space; a world can have multiple dSpaces.
- **Controller/PlayerController**: The player controller, capable of accepting player input and setting the observation target, among other functions.
- **Pawn/Character**: A controllable object; Characters offer additional functionality compared to Pawns, particularly for humanoid characters, including actions like movement, crouching, jumping, etc.
- **CameraManager**: Manages all camera-related functions, such as camera position and camera shake effects.
- **GameMode**: Used to control the rules of a match or activity.
- **PlayerState**: Records data information for each player, such as player scores.
- **GameState**: Records information about the entire match, including the stage of the match or activity and information about participants from different teams.

6.5. 3D-Pipeline-based Pinpoint Protocol

As mentioned earlier, pinpoint protocol is the bridge between native evaluator EN and contract arbitrator EC. To align EN and EC, HyperTrust does not design the pinpoint protocol based on VM,
which is not aligned with the intrinsic nature of DNN computation. Instead, we design a transformation-based pinpoint protocol (TPP). TPP includes the following three components and defines how to use them for the pinpoint protocol: (1) Phase-1 evaluator EOP which aligns with EN and executes in the granularity of operation; (2) Phase-2 evaluator EBOP which aligns with both EC and EOP, and executes in the granularity of basic operation (BOP), along with the setup tools for EOP ; (3) an enhanced evaluator EC which can arbitrate the floating-point arithmetic in BOPs.
Next, we begin with explaining the components for EBOP , because EBOP plays a central role in TPP. It not only determines the BOPs that EC needs to support, but also is the basis of EOP.

6.6. Enabling 3D-Pipeline-based pinpoint with EBOP

Every actor/object has many properties for transformation. Such as position, scale, rotation and matrix etc.

Categories	Basic Operations
Assignment	= (the operand type can be f32, i32 or u8)

Floating-point Arithmetic	f32_add,f32_sub,f32_mul,f32_div, f32_min,f32_max,f32_sqrt,f32_round, f32_floor
Integer Arithmetic	i32_add, i32_sub, i32_mul
Type Casting	f32_to_u8, u8_to_f32, u8_to_i32, i32_to_f32, f32_to_i32

"BOP" includes the most fundamental transformations, such as translation, rotation, scaling, shearing, various transformations, rigid body transformations, normal transformations, and inverse operations.

1.1.1.1. Translation

The translation matrix T is used to move an object from one position to another. This matrix is used to translate an object based on the vector t = (tx, ty, tz). The matrix T is represented as follows:

$$T(t) = T(t_x, t_y, t_z) = \begin{pmatrix} 1 & 0 & 0 & t_x \\ 0 & 1 & 0 & t_y \\ 0 & 0 & 1 & t_z \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

As shown in Figure 4.1, this is an example of a translation matrix. You can observe that when you multiply a point P = (px, py, pz, 1) by T (t), it generates a new point P0 = (px + tx, py + ty, pz + tz, 1), that is a translation.

1.1.1.2. Rotation

Rotation transform is used to rotate a vector (position or direction) along an axis passing through the origin by a specified angle. Like the translation matrix, it's a rigid-body transform, meaning it preserves distances between transformed points and doesn't cause left-right swapping.

In the 3D domain, rotation matrices are denoted as Rx(), Ry(), Rz(), corresponding to rotations along the x, y, and z axes by an angle θ. The formulas are as follows:

$$R_x(\phi) = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\phi & -\sin\phi & 0 \\ 0 & \sin\phi & \cos\phi & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix},$$

$$R_y(\phi) = \begin{pmatrix} \cos\phi & 0 & \sin\phi & 0 \\ 0 & 1 & 0 & 0 \\ -\sin\phi & 0 & \cos\phi & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix},$$

$$R_z(\phi) = \begin{pmatrix} \cos\phi & -\sin\phi & 0 & 0 \\ \sin\phi & \cos\phi & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

1.1.1.3. Scale

The scaling matrix, S(s) = S(Sx, Sy, Sz), scales an object along the x, y, and z axes by the factors Sx, Sy, and Sz, respectively. In other words, the scaling matrix can enlarge or shrink an object. A larger scaling factor Si corresponds to a greater scaling ratio along that axis. If the scaling factor is 1, there is no scaling along that direction. The scaling matrix is represented as follows:

$$S(s) = \begin{pmatrix} s_x & 0 & 0 & 0 \\ 0 & s_y & 0 & 0 \\ 0 & 0 & s_z & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}.$$

Figure 4.4 illustrates the effect of the scaling matrix. When the scaling factors are the same for all three axes, it's referred to as uniform scaling; otherwise, it's called non-uniform scaling. Sometimes, it's also known as isotropic (uniform) and anisotropic (non-uniform). The inverse of the scaling matrix, S^(-1)(s) = S(1/Sx, 1/Sy, 1/Sz).

1.1.1.4. Shearing

Another type of transformation is called a shearing matrix. For example, these can be used in games to distort the entire scene for psychedelic effects or to alter the appearance of models. There are six basic shearing matrices: Hxy(s), Hxz(s), Hyx(s), Hyz(s), Hzx(s), and Hzy(s). The first subscript indicates the coordinate that the shearing matrix changes, and the second subscript indicates the coordinate of shearing. The formula for the shearing matrix Hxz(s) is as follows:

$$H_{xz}(s) = \begin{pmatrix} 1 & 0 & s & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}.$$

It's worth noting that the subscripts are used to determine the position of the parameters in the matrix. In this case, x represents the 0th row, and z represents the 2nd column, so s is located at that position.

1.1.1.5. Matrix Composing

From time t0 to t1, an Actor may undergo a series of matrix transformations based on user input, which involves matrix multiplication operations.

$$P' = P \times \begin{pmatrix} 1 & 0 & 0 & t_x \\ 0 & 1 & 0 & t_y \\ 0 & 0 & 1 & t_z \\ 0 & 0 & 0 & 1 \end{pmatrix} \times \begin{pmatrix} 1 & 0 & 0 & t_x \\ 0 & 1 & 0 & t_y \\ 0 & 0 & 1 & t_z \\ 0 & 0 & 0 & 1 \end{pmatrix} \times \begin{pmatrix} 1 & 0 & 0 & t_x \\ 0 & 1 & 0 & t_y \\ 0 & 0 & 1 & t_z \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

where, $P = (p_x, p_y, p_z, 1)$.

We implement HyperPrint to visualize 3D script recordings of BOP sequences (circuits) for Actor's 3D pipeline transformation calculations in the metaverse space. This includes transformation matrices and input vectors. Since metaverse 3D transformation calculations occur in real-time, these BOP sequences are also generated in real-time. The maximum number per second could be Actor count * frames per second * BOP type count * 4 * 4 (for 4x4 transformation matrices).

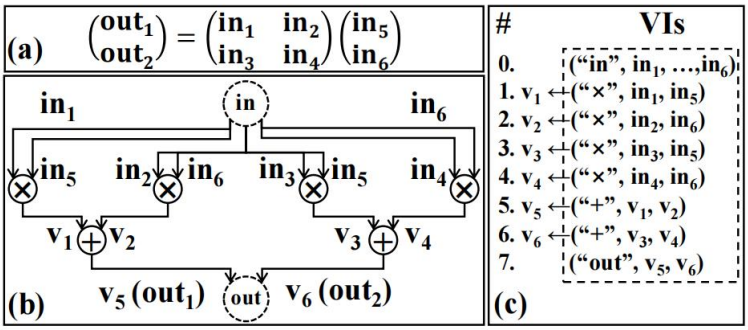


Figure 5: A circuit of matrix multiplication: (a) the formula, (b) the circuit, (c) serialized Vertices with Inputs (VIs).

With the setup work of circuit generation, serialization and the evaluator, when a verifier disagrees about the computation result of the submitter, the pinpoint protocol begins. In this subsection, we assume the whole 3D transformation computation is expressed as a circuit. Then, our pinpoint protocol can be based on a commitment scheme (Minimum disclosure proofs of knowledge) leveraging Merkle trees (MTs) and a smart contract.

Therefore, the pinpoint problem can be defined using VIS and VIC as follows. Prerequisite: the submitter and the verifiers agree on the same VIS-sequence and all input values. Problem: the submitter and a verifier get different VIC-sequences, so the smart contract needs to determine who is incorrect (note that it does not imply that the other is correct).

The protocol allows the submitter and the verifier to locate their leftmost divergent VI^c. The complexity is (log), where is the length of the VI sequence. The commitment scheme guarantees the non-repudiation of the disputing parties during the process. The protocol includes one preparation step and three interaction steps, as intuitively shown in Figure 6.

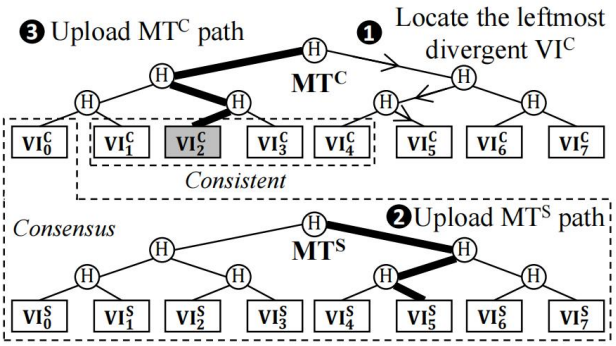


Figure 6: The MT-based commitment scheme to locate and arbitrate the divergent VI.

Up to this point, we have explained how to pinpoint a disputed BOP in a DNN computation, assuming the protocol can process the entire computation graph. However, this is difficult in reality. We will show in Section 7 that the number of basic operations can be as large as 30 billion, the memory size to include these BOPs is over 600 GB. If we assume the number of OPs in the first-layer is 1 and each OP has 2 BOPs, the space complexity would be $(1 \cdot 2)$ either for generating or for evaluating the graph. To reduce the significant overhead, we propose a two-phase pinpoint protocol, which introduces the Phase-1 evaluator.

6.7. Introducing EOP for two-phase pinpoint

To make pinpoint protocol practical, the pinpointing process needs to be done in two phases. Phase-1 identifies the disputed OP with another highly efficient evaluator . As shown in Figure 7(a), the leaves of the Merkle tree for Phase-1 are OPs, so the size of the tree (i.e., P1_MT) is much smaller. The procedure in Phase-1 is as what we described above, with two important details to note: (1) the VI represents an OP (e.g., Scale) not a BOP (e.g., “×”, 1, 5). Variables can be tensors with arbitrary shape and data type. The “BOP” field of VI in the previous section is replaced as one of operations (e.g., Rotation) that takes input variables with certain dimensions. In our protocol, this OP is represented by the Merkle root of MTof the operation’s symbolic circuit (i.e., P2_MT), rather than the string “MatMulInteger”, so it is a well-defined concrete computation. (2) Different from evaluating a circuit, it leverages to evaluate every operation with native performance and computes based on those operation results, which is a feature of current 3D pipeline APIs.

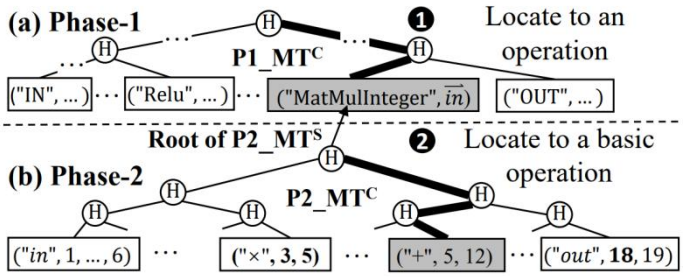


Figure 7: Two-phase pinpointing: (a) Phase-1 (b) Phase-2

Once the disputed operation is pinpointed in Phase-1, the submitter and the verifier start Phase-2, which is within the Merkle tree P2_MT in Figure 7(b). Generating P2_MTand P2_MTis on-demand, so the space complexity of the protocol is reduced to (1 + 2). In the end, a disputed BOP is submitted to the contract for arbitration.

6.8. Contract arbitrator Ec

We implement the contract arbitrator for all the BOPs in Table 3 via the integer-based emulation, which is fully compliant with the current Ethereum contract standard. In other words, is aligned with with the granularity of BOP. The implementation carefully complies with IEEE-754 and demonstrates a negligible on-chain overhead for arbitration (see Section 7).

6.9. Summary

From the above content, we can see that Hyperdust draws inspiration from Agatha by applying BOP to 3D pipeline calculations. The overall software pattern of both systems is the same*. In the subsequent implementation based on HyperTrust, it can be relatively easy to integrate the two, meaning integrating AI and 3D calculations onto the blockchain. This upgrades the capabilities of smart contracts, allowing them to support AI and 3D calculations. We refer to such smart contracts as compute-intensive on-chain autonomous agents or "fat contracts." This can pave the way for a decentralized application platform focused on compute-intensive tasks. In Chapter 4, we will provide a detailed description of this platform – Hyperdust.

* There is also a notable difference between the two systems: the overhead in 3D pipeline calculations can be incurred without additional costs, such as when the submitter and verifier in the network are nodes that provide services to users (3D/XR clients). However, the overhead in AI calculations both the submitter and verifier need to perform local AI inference calculations, which can increase computational costs. But in practice, since there are multiple nodes load and run different layers (blocks) of the same AI model (e.g. Meta Llama 3 70B) , those multiple

“chained” LLM instances could be used for this overhead.

References

[1] Zihan Zheng, Peichen Xie, Xian Zhang, etc. Agatha: Smart Contract for DNN Computation (arXiv:2105.04919)

[2] Optisim Architecture
https://10xpub.notion.site/Optimism-b66d7fb846d441b89bec4a7977060fca

[3] https://github.com/ethereum/EIPs/blob/master/EIPS/eip-1559.md

[4] https://github.com/nrichardsonphd/gvm

[5] Arbitrum: Scalable, private smart contracts Harry Kalodner, Steven Goldfeder, Xiaoqi Chen, S. Matthew Weinberg, and Edward W. Felten, Princeton University

[6] Graph compilers for AI training and inference Karthee Sivalingam, Nina Mujkanovic – CRAY EMEA Research Lab

[7] Ethereum: A Secure Decentralised Generalised Transaction Ledger Berlin Version 2bcdb2d – 2023-08-25 Dr. Gavin Wood Founder, Ethereum & Party

[8] Ethereum: A Next-Generation Smart Contract and Decentralized Application Platform. By Vitalik Buterin (2014).

[9] https://brilliantcrypto.net/

[10] https://repo-sam.inria.fr/fungraph/3d-gaussian-splatting/

[11] Bernhard Kerbl, Georgios Kopanas etc, 3D Gaussian Splatting for Real-Time Radiance Field Rendering ,SIGGRAPH 2023 (ACM Transactions on Graphics)

[12] Chris Eliasmith, How to Build a Brain, A Neural Architecture for Biological Cognition,Apr 2013 · Oxford University Press

[13] Nick Dyer-Witheford, Inhuman Power: Artificial Intelligence and the Future of Capitalism, Pluto Press, ISBN 0745338607

[14] https://aws.amazon.com/cn/solutions/case-studies/mossai-case-study/