CryptOS: A Federated Machine Learning Network Leveraging Blockchain Technology for Rewards

Nick Spanos
nick@cryptos.com
04/06/2023

Abstract

CryptOS (ctOS) is a decentralized, federated machine learning network designed to incentivize users to contribute computing power in exchange for tokens. CryptOS employs a novel consensus mechanism and economic model to allocate resources efficiently and reward contributors of GPU compute for machine learning applications equitable to their training throughput using a proprietary, patented method outlined in patent US9608829B2 (Spanos, Dixon, et al., 2017).

Enclosed herein, we present the architecture, consensus mechanism, tokenomics, and future technical roadmap of the CryptOS project in a technologically articulated manner that delivers a clear framework for creation, execution, and deployment of a software platform in the format of a lite Linux-derived software distribution that provides bloat-free hardware utilization and registration to a decentralized network.

Introduction

Machine learning (ML) and artificial intelligence (AI) have become critical components of modern technology and business applications. However, the computational power required for training large-scale ML models is significant, and access to these resources is often limited to organizations with substantial financial means. CryptOS aims to democratize access to computational resources by creating a federated machine learning network that rewards individuals and organizations for contributing compute power.

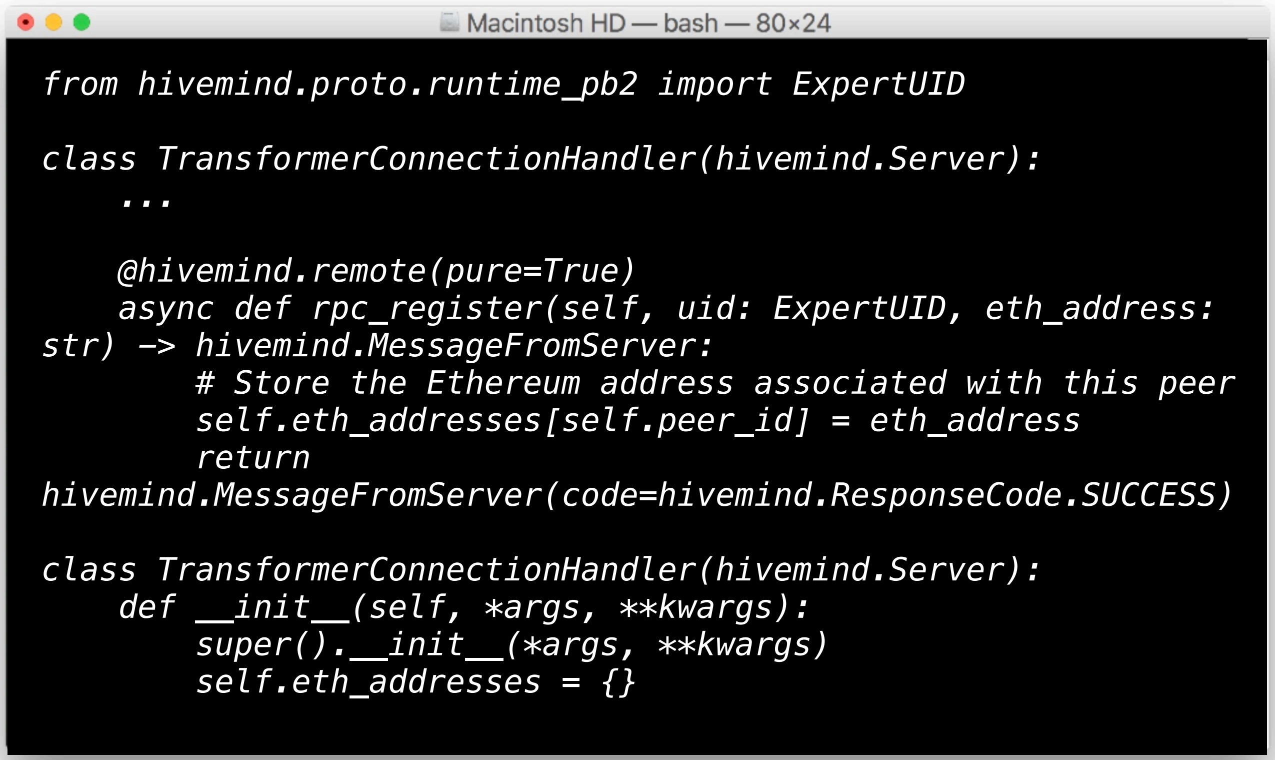
CryptOS will have a multi-faceted approach to delivering this proof-of-concept through building our initial rewards system on top of Petals.ML (Borzunov, A., Baranchuk, D., Dettmers, T., Ryabinin, M., Belkada, Y., Yandex, A., Samygin, P., & Raffel, C., March 2023) the core conceptual system will be deployed as an EVM compatible smart contract system that uses a modified fork of Petals.ML to substantiate the Proof of Compute (PoC) system necessary to provision rewards based on each contributing machine’s throughput which is quantified as an integer (RPS).

Architecture

1.1 Testnet Architecture

The testnet rewards system for CryptOS (ctOS) will rely on a smart contract oracle listener within an external EVM compatible smart contract. Our goal is to develop a reasonable proof of concept utilizing popular technology to create an implementation of our vision that can be tested and revised for efficiency. The key modification being made to the existing Petals.ML system as aforementioned and cited in the introduction exists within the *handler.py* file present in the code here and demonstrated in Figure 1: *bigscience-workshop/petals/src/petals/server/handler.py*

Lines 34-49 Appended with the asynchronous oracle hook

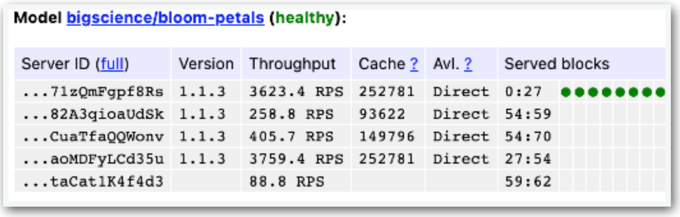


```
from hivemind.proto.runtime_pb2 import ExpertUID

class TransformerConnectionHandler(hivemind.Server):
    ...

    @hivemind.remote(pure=True)
    async def rpc_register(self, uid: ExpertUID, eth_address:
str) -> hivemind.MessageFromServer:
        # Store the Ethereum address associated with this peer
        self.eth_addresses[self.peer_id] = eth_address
        return
hivemind.MessageFromServer(code=hivemind.ResponseCode.SUCCESS)

class TransformerConnectionHandler(hivemind.Server):
    def __init__(self, *args, **kwargs):
        super().__init__(*args, **kwargs)
        self.eth_addresses = {}
```

Figure 1.

1.1.2 PseudoCode Function

The code above exists to allow users of CryptOS to associate their EVM compatible address with the Server ID of their Petals.ml deployment shown in Figure 2 and henceforth earn rewards based on the throughput of their compute contributions. (in the example Ethereum is used, but gas fees will necessitate the use of our proprietary chain in production builds)



Model bigscience/bloom-petals (healthy):

Server ID (full)	Version	Throughput	Cache ?	Avl. ?	Served blocks
...71zQmFgpf8Rs	1.1.3	3623.4 RPS	252781	Direct	0:27 ●●●●●●●●
...82A3qioaUdSk	1.1.3	258.8 RPS	93622	Direct	54:59
...CuaTfaQQWonv	1.1.3	405.7 RPS	149796	Direct	54:70
...aoMDFyLCd35u	1.1.3	3759.4 RPS	252781	Direct	27:54
...taCatlK4f4d3		88.8 RPS			59:62

Figure 2.

Throughput is the desired mechanism of measuring compute contributions because it allows us to reward users via an algorithm that is equitable to their contributions to the network.

1.2 Registration Endpoint Oracle Relay Smart Contract

The next portion of code will exist to aggregate the network contributors that have registered with the simple pairing of an EVM compatible address and Server ID (e.g. [ETH Address]:[ServerID]) as outlined in the previous step. This simple smart contract shown in Figure 3 will accomplish the rewards mechanism necessary to make this possible.

1.2.1 Chainlink & OpenZeppelin Libraries

The smart contract imports the ChainlinkClient and IERC20 OpenZeppelin libraries for interfacing with Chainlink oracles and issuing ERC20 $CTOS tokens as rewards.

1.2.2 Oracle Configuration

The contract defines several private variables: oracle, jobId, fee, rewardToken, and apiURL. These are used to configure the Chainlink oracle, the job ID for the RPS throughput metrics, the fee for making requests, the ERC20 token to be used for rewards, and the API URL for the Petals.ml service.

1.2.3 Reward Security

The contract defines a public mapping called lastRewardTimestamp. This mapping is used to keep track of the last time a user claimed a reward, so that they can only claim rewards once every 24 hours.

1.2.4 Initial Contract Parameters

The constructor function sets the initial contract parameters: the oracle address, the job ID, the fee, the reward token address, the API URL, or the Zap Protocol .ZAP address.

1.2.5 Claim Rewards Based on RPS Throughput

The claimReward function allows users to claim rewards based on their RPS throughput. When a user calls this function, the contract checks whether they have already claimed a reward within the last 24 hours. If they have not, the contract makes a Chainlink request to the Petals.ml API to retrieve the user's RPS throughput metrics. The result is then passed to the fulfill function.

1.2.6 Reward Fulfillment

The fulfill function is called by the Chainlink oracle once the RPS throughput metrics have been retrieved. It calculates the reward amount based on the RPS throughput metrics and transfers the tokens to the user. It also updates the lastRewardTimestamp mapping to record that the user has claimed a reward.

1.2.7 Reward Calculation Algorithm

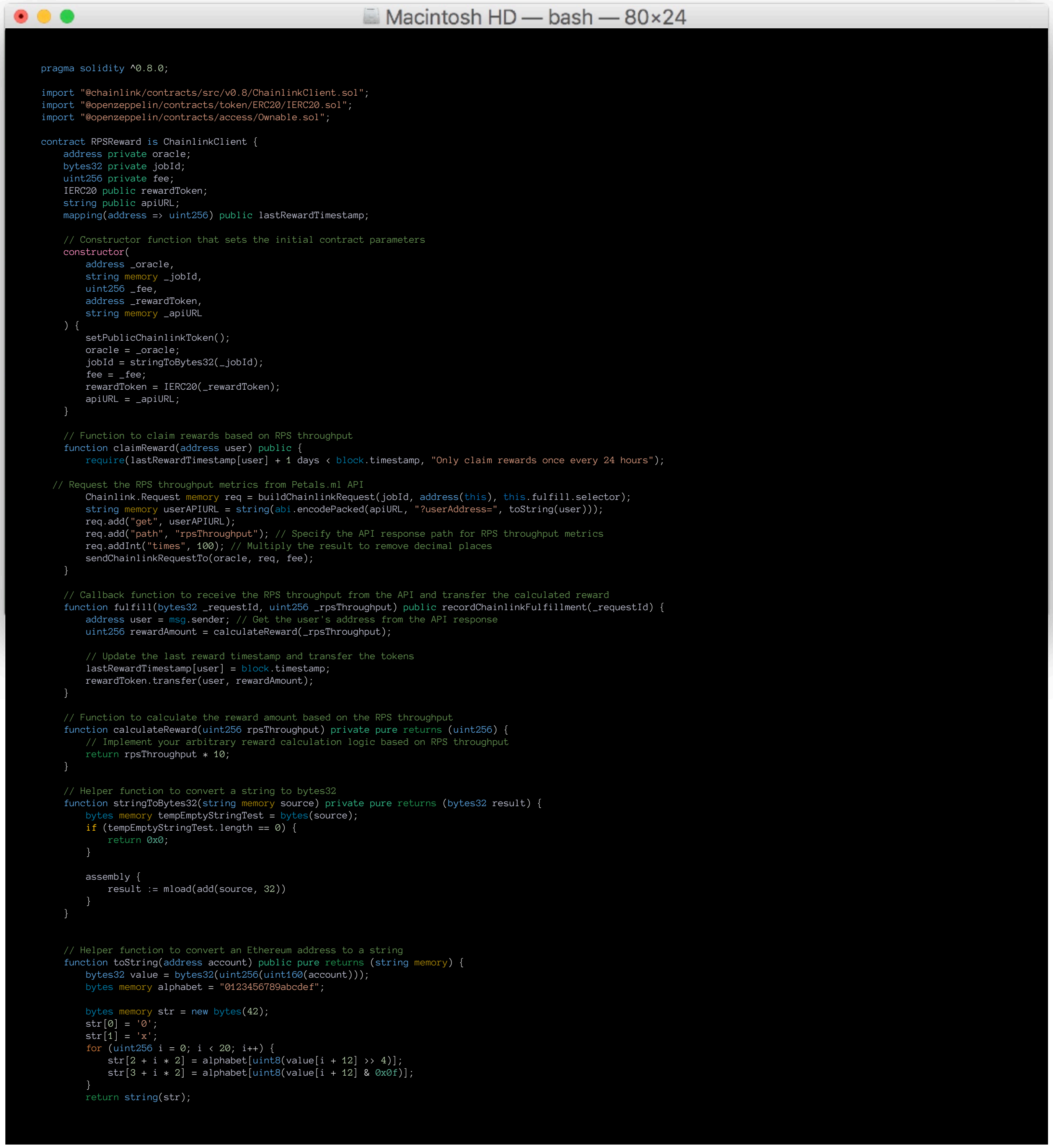
The calculateReward function is used to determine the amount of tokens to be rewarded based on the RPS throughput metrics. This function can be modified to implement arbitrary reward calculation logic based on the RPS throughput metrics.

1.2.8 Deployment on Testnet

The contract outlined in Figure 3 will subsequently be revised and deployed to testnet with the features outlined in this section operating within as a basic proof of concept that operates and assigns rewards on an EVM compatible blockchain to the participants of the Petals.ML public swarm.

This testing phase will occur on the Goerli testnet and be assumed to provide no financial incentive to participants outside of advancing efforts to build this solution and a collectible Ethereum NFT with no presumed resale value as a record of having participated in testnet for CryptOS (ctOS).

Our overarching goal is to utilize this phase to optimize efficiency and direct development efforts accordingly to build a system that has the best interests of machine learning put first and foremost over making a blockchain product that sacrifices compute effectiveness in favor of optimizations that are favored by typical products in that industry. Federated machine learning that brings the ability to operate 100B+ parameter language models in a democratized, decentralized fashion is the main goal of the testing phase.



```
pragma solidity ^0.8.0;

import "@chainlink/contracts/src/v0.8/ChainlinkClient.sol";
import "@openzeppelin/contracts/token/ERC20/IERC20.sol";
import "@openzeppelin/contracts/access/Ownable.sol";

contract RPSReward is ChainlinkClient {
    address private oracle;
    bytes32 private jobId;
    uint256 private fee;
    IERC20 public rewardToken;
    string public apiURL;
    mapping(address => uint256) public lastRewardTimestamp;

    // Constructor function that sets the initial contract parameters
    constructor(
        address _oracle,
        string memory _jobId,
        uint256 _fee,
        address _rewardToken,
        string memory _apiURL
    ) {
        setPublicChainlinkToken();
        oracle = _oracle;
        jobId = stringToBytes32(_jobId);
        fee = _fee;
        rewardToken = IERC20(_rewardToken);
        apiURL = _apiURL;
    }

    // Function to claim rewards based on RPS throughput
    function claimReward(address user) public {
        require(lastRewardTimestamp[user] + 1 days < block.timestamp, "Only claim rewards once every 24 hours");

  // Request the RPS throughput metrics from Petals.ml API
        Chainlink.Request memory req = buildChainlinkRequest(jobId, address(this), this.fulfill.selector);
        string memory userAPIURL = string(abi.encodePacked(apiURL, "?userAddress=", toString(user)));
        req.add("get", userAPIURL);
        req.add("path", "rpsThroughput"); // Specify the API response path for RPS throughput metrics
        req.addInt("times", 100); // Multiply the result to remove decimal places
        sendChainlinkRequestTo(oracle, req, fee);
    }

    // Callback function to receive the RPS throughput from the API and transfer the calculated reward
    function fulfill(bytes32 _requestId, uint256 _rpsThroughput) public recordChainlinkFulfillment(_requestId) {
        address user = msg.sender; // Get the user's address from the API response
        uint256 rewardAmount = calculateReward(_rpsThroughput);

        // Update the last reward timestamp and transfer the tokens
        lastRewardTimestamp[user] = block.timestamp;
        rewardToken.transfer(user, rewardAmount);
    }

    // Function to calculate the reward amount based on the RPS throughput
    function calculateReward(uint256 rpsThroughput) private pure returns (uint256) {
        // Implement your arbitrary reward calculation logic based on RPS throughput
        return rpsThroughput * 10;
    }

    // Helper function to convert a string to bytes32
    function stringToBytes32(string memory source) private pure returns (bytes32 result) {
        bytes memory tempEmptyStringTest = bytes(source);
        if (tempEmptyStringTest.length == 0) {
            return 0x0;
        }

        assembly {
            result := mload(add(source, 32))
        }
    }


    // Helper function to convert an Ethereum address to a string
    function toString(address account) public pure returns (string memory) {
        bytes32 value = bytes32(uint256(uint160(account)));
        bytes memory alphabet = "0123456789abcdef";

        bytes memory str = new bytes(42);
        str[0] = '0';
        str[1] = 'x';
        for (uint256 i = 0; i < 20; i++) {
            str[2 + i * 2] = alphabet[uint8(value[i + 12] >> 4)];
            str[3 + i * 2] = alphabet[uint8(value[i + 12] & 0x0f)];
        }
        return string(str);
```

Figure 3.

Future Developments and Technical Roadmap

As CryptOS (ctOS) progresses, our focus will be on enhancing the platform's capabilities and refining the system to better serve the machine learning community. The following technical roadmap outlines the key steps we plan to take to achieve our goals:

2.1 Mainnet Launch

After a successful testnet phase, we will launch the CryptOS mainnet, utilizing our proprietary blockchain to address the limitations of the EVM-compatible blockchain. This mainnet launch will enable the platform to operate with lower transaction fees and improve the overall efficiency of the system.

2.2 Integration of Additional Machine Learning Frameworks

Initially, the ctOS network will be built on top of the Petals.ML framework. However, to support a broader range of machine learning applications, we plan to integrate additional popular machine learning frameworks such as TensorFlow, PyTorch, and MXNet. This will facilitate the seamless integration of ctOS with various machine learning projects and foster a more inclusive ecosystem.

2.3 Development of a User-Friendly Interface

To encourage the adoption of ctOS among a wider audience, we plan to develop an intuitive and user-friendly interface that simplifies the process of contributing computing resources and managing rewards. This will make it easier for users with varying levels of technical expertise to participate in the ctOS network.

2.4 Expansion of the Ecosystem with Strategic Partnerships

In order to further enhance the capabilities of the ctOS network and increase its adoption, we will actively pursue strategic partnerships with leading technology companies, research institutions, and organizations in the AI and ML domain. These partnerships will help drive innovation and facilitate the integration of ctOS into various machine learning projects and use cases.

2.5 Establishment of a CryptOS Foundation

To govern the development and ongoing maintenance of the ctOS platform, we will establish the CryptOS Foundation. This non-profit organization will be responsible for overseeing the platform's direction, coordinating research efforts, and managing the allocation of resources. The foundation will also engage with the community to ensure that the project remains transparent and aligned with the needs of the machine learning community.

2.6 Research and Development of Advanced Consensus Mechanisms

As the ctOS network evolves, we will invest in research and development of advanced consensus mechanisms to further optimize the efficiency of resource allocation and reward distribution. This will ensure that the network remains adaptive and responsive to the ever-changing needs of the machine learning community.

2.7 Security and Privacy Enhancements

As the ctOS network grows, ensuring the security and privacy of user data and transactions will become increasingly important. We will implement robust security measures, such as end-to-end encryption, to protect user data and communications within the network. Additionally, we will actively collaborate with the cybersecurity community to conduct regular audits and vulnerability assessments, ensuring that our platform remains resilient against potential threats.

2.8 Scalability and Performance Optimization

To accommodate the increasing demand for computational resources in the machine learning community, we will continuously work on improving the scalability and performance of the ctOS network. This includes researching and implementing layer 2 scaling solutions, parallel processing techniques, and efficient resource allocation algorithms that optimize the throughput and responsiveness of the network.

2.9 Community-driven Development and Open Source Contributions

We believe that an active and engaged community is crucial for the success and longevity of the ctOS platform. As such, we will encourage community-driven development by making our codebase open source and fostering a collaborative environment where developers can contribute to the project. By involving the community in the development process, we aim to accelerate innovation and ensure that the platform addresses the real-world needs of machine learning practitioners.

2.10 Cross-chain Interoperability

To further enhance the utility of the ctOS token and increase the overall value of the ecosystem, we will explore cross-chain interoperability solutions that enable seamless integration with other blockchain networks. By enabling the ctOS token to be easily exchanged for other cryptocurrencies or used across various blockchain platforms, we aim to increase the liquidity and adoption of the token, benefiting both users and the broader ctOS ecosystem.

2.11 Education and Outreach Initiatives

Recognizing the importance of raising awareness and promoting understanding of the ctOS platform, we will implement various education and outreach initiatives. This includes organizing workshops, webinars, and hackathons to educate users on how to effectively utilize the ctOS network for their machine learning projects. Additionally, we will collaborate with educational institutions to integrate ctOS into their curricula and promote its adoption among the next generation of machine learning practitioners.

2.12 Environmental Sustainability

In light of growing concerns around the environmental impact of blockchain technology, we will actively work towards making the ctOS network more energy-efficient and environmentally sustainable. This includes researching and implementing eco-friendly consensus mechanisms, optimizing the energy consumption of the network, and supporting renewable energy initiatives.

By following this comprehensive technical roadmap, we aim to establish CryptOS (ctOS) as a leading platform for democratizing access to computational resources in the machine learning space, fostering innovation, and driving the advancement of AI and ML technologies across various industries and applications.

US Patent US9608829B2: A Deep Dive into the Core Technology

3.1 Overview of US9608829B2

US Patent US9608829B2, filed by Spanos, Spanos, et al. in 2017, lays the foundation for the CryptOS (ctOS) project by outlining a novel method for allocating resources and rewarding contributors in a decentralized, federated machine learning network. This patented technology forms the basis of ctOS's consensus mechanism and economic model, ensuring efficient resource allocation and equitable rewards distribution to participants based on their training throughput.

3.2 Key Components of US9608829B2

The patent comprises several key components that contribute to the uniqueness and effectiveness of the ctOS platform:

3.2.1 Decentralized Resource Allocation

The patented technology enables a decentralized approach to resource allocation, allowing the ctOS network to harness the power of distributed computing resources. This eliminates the need for a central authority to manage resource distribution and prevents potential bottlenecks or single points of failure.

3.2.2 Federated Machine Learning Network

By employing a federated machine learning network, the patent ensures that the ctOS platform can support the training and execution of large-scale machine learning models across a diverse range of devices and nodes. This design promotes collaboration and sharing of resources, thereby democratizing access to computational power for machine learning applications.

3.2.3 Novel Consensus Mechanism

The patent describes a unique consensus mechanism that enables the ctOS network to efficiently allocate resources and reward contributors based on their training throughput. This mechanism, also referred to as Proof of Compute (PoC), evaluates the performance of each node by quantifying its training throughput as an integer (RPS). This allows the network to dynamically adjust resource allocation and rewards distribution according to the real-time contributions of each participant.

3.2.4 Economic Model

The patented technology also encompasses an innovative economic model that leverages tokens to incentivize users to contribute their computing power to the ctOS network. Users can earn tokens based on their contributions, which can be traded or used to access additional resources within the network. This economic model encourages active participation and fosters the growth of the ctOS ecosystem.

3.3 Implications for the CryptOS Project

The patented technology outlined in US9608829B2 plays a crucial role in shaping the architecture, consensus mechanism, and tokenomics of the CryptOS project. By leveraging this innovative technology, the ctOS platform can effectively:

- Democratize access to computational resources for machine learning applications
- Facilitate efficient and dynamic resource allocation within the network
- Incentivize users to actively contribute their computing power
- Establish a decentralized, federated machine learning network that supports large-scale model training

As a result, the ctOS project can effectively address the challenges posed by limited access to computational resources in the machine learning domain, paving the way for a more inclusive and collaborative ecosystem that fosters innovation and accelerates the development of AI and ML technologies.

3.3.1 Layering The Petals.ML Federated Learning Model Over A Blockchain

If we refer to Figure 1 & 2 from US09608829 we can clearly see there is a mechanism to cryptographically chain together and reward participants while keeping a ‘genealogy’ of language models, fine-tunes and otherwise that take place within the Petals.ML ecosystem.

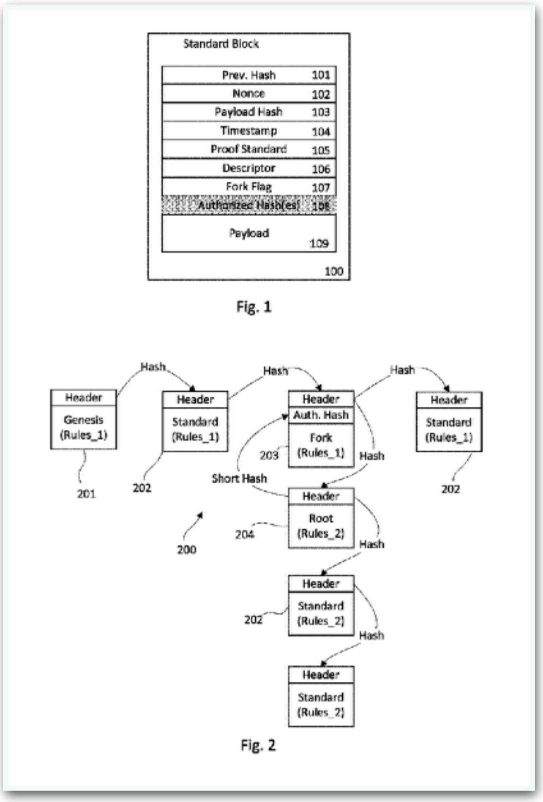


Figure 1 & 2 from US09608829

3.3.2 Blockchain Benefits for Petals.ML in CryptOS

The integration of blockchain technology with the Petals.ML federated learning model within the ctOS network offers several key advantages:

Trust and Transparency: By maintaining an immutable record of all transactions, contributions, and model developments, blockchain technology can instill trust and transparency within the Petals.ML ecosystem. This enables participants to verify the authenticity of models and contributions, preventing potential fraud or misuse of resources.

Secure and Decentralized: Blockchain's decentralized nature removes the need for a central authority, allowing the ctOS network to distribute resources and decision-making power across a network of participants. This ensures that no single party can manipulate or control the system, enhancing its security and resilience against potential threats.

Incentivization and Rewards: The integration of blockchain allows ctOS to effectively reward users for their contributions through a token-based system. This incentivizes active participation and fosters the growth of the ctOS ecosystem, encouraging users to contribute their computational resources for machine learning applications.

Scalability and Efficiency: By combining the Petals.ML federated learning model with blockchain, the ctOS platform can achieve improved scalability and efficiency. The decentralized nature of the blockchain enables the platform to distribute resources across a wide range of devices and nodes, optimizing the overall performance of the system.

The Combined Approach: Blockchain & Federated Machine Learning

4.1 Symbolized Combined Approach: PetalsML & ctOS

This symbol denotes a 2 part model being distributed across two compute modules and represented on the blockchain as a single model fork. (Such as GPT-J)

This symbol represents a 3 part model being divided across federated compute and represented on the blockchain as a single job. (Such as Bloom)

This symbol represents a 2 part model forked from the parent model being divided across machines. (Such as BloomZ - a fork of Bloom)

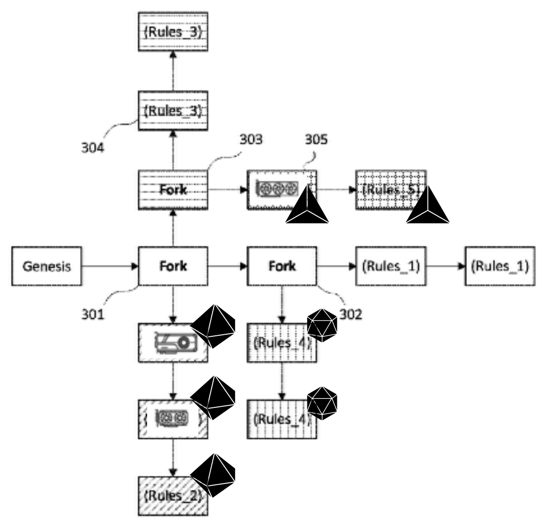


Fig. 3

Figure 3 from US09608829

4.2 Understanding Our Approach

These symbols serve as visual representations of the different ways in which models can be distributed and managed within the PetalsML & ctOS ecosystem. By utilizing these symbols, users can easily identify the specific type of model distribution and its corresponding configuration on the blockchain.

4.2.1 Inter-relational Models

The symbolized combined approach simplifies the understanding of various model distributions and enables a more intuitive way of representing the complex relationships between models, jobs, and blockchain representations. This not only enhances the overall user experience but also promotes transparency and collaboration within the PetalsML & ctOS ecosystem, ultimately driving innovation and the development of advanced AI and ML technologies.

WebAssembly for Federated Machine Learning in CryptOS

5.1 Introduction to WebAssembly

WebAssembly (Wasm) is a binary instruction format designed as a portable target for the compilation of high-level languages such as C, C++, and Rust. It enables the deployment of web applications with near-native performance by providing a low-level virtual machine that runs code at near-native speed. WebAssembly's compatibility across different platforms and browsers makes it a suitable candidate for building and deploying the ctOS federated machine learning model.

5.2 Advantages of Using WebAssembly for Federated Machine Learning

The use of WebAssembly for implementing the ctOS federated machine learning model offers several advantages, including:

Improved Performance: WebAssembly provides near-native execution speed, allowing machine learning workloads to be executed more efficiently compared to traditional JavaScript implementations.

Cross-platform Compatibility: WebAssembly runs in all major browsers and operating systems, enabling users to access and participate in the ctOS network from various devices without the need for specialized software.

Language Flexibility: Developers can leverage various programming languages, such as C, C++, and Rust, to build and optimize the machine learning models for WebAssembly.

5.3 WebAssembly-based Machine Learning Frameworks and Libraries

Several machine learning frameworks and libraries can be used in conjunction with WebAssembly to facilitate the implementation of the ctOS federated machine learning model. Some popular options include TensorFlow WebAssembly, ONNX Runtime WebAssembly, and Apache TVM.

5.4 Implementing Federated Machine Learning with WebAssembly

To implement the ctOS federated machine learning model using WebAssembly, the following steps can be taken:

Model Compilation: Compile the machine learning models into WebAssembly format using appropriate compilers or frameworks.

Model Distribution: Distribute the WebAssembly-compiled machine learning models across the ctOS network, enabling users to execute the models on their devices.

Local Execution: Users execute the WebAssembly-compiled machine learning models on their devices, leveraging the near-native performance of WebAssembly to efficiently process the data.

Model Aggregation: Collect and aggregate the locally updated models from users, and update the global model accordingly.

Secure Communication: Ensure secure communication between users and the ctOS network to maintain privacy and data integrity.

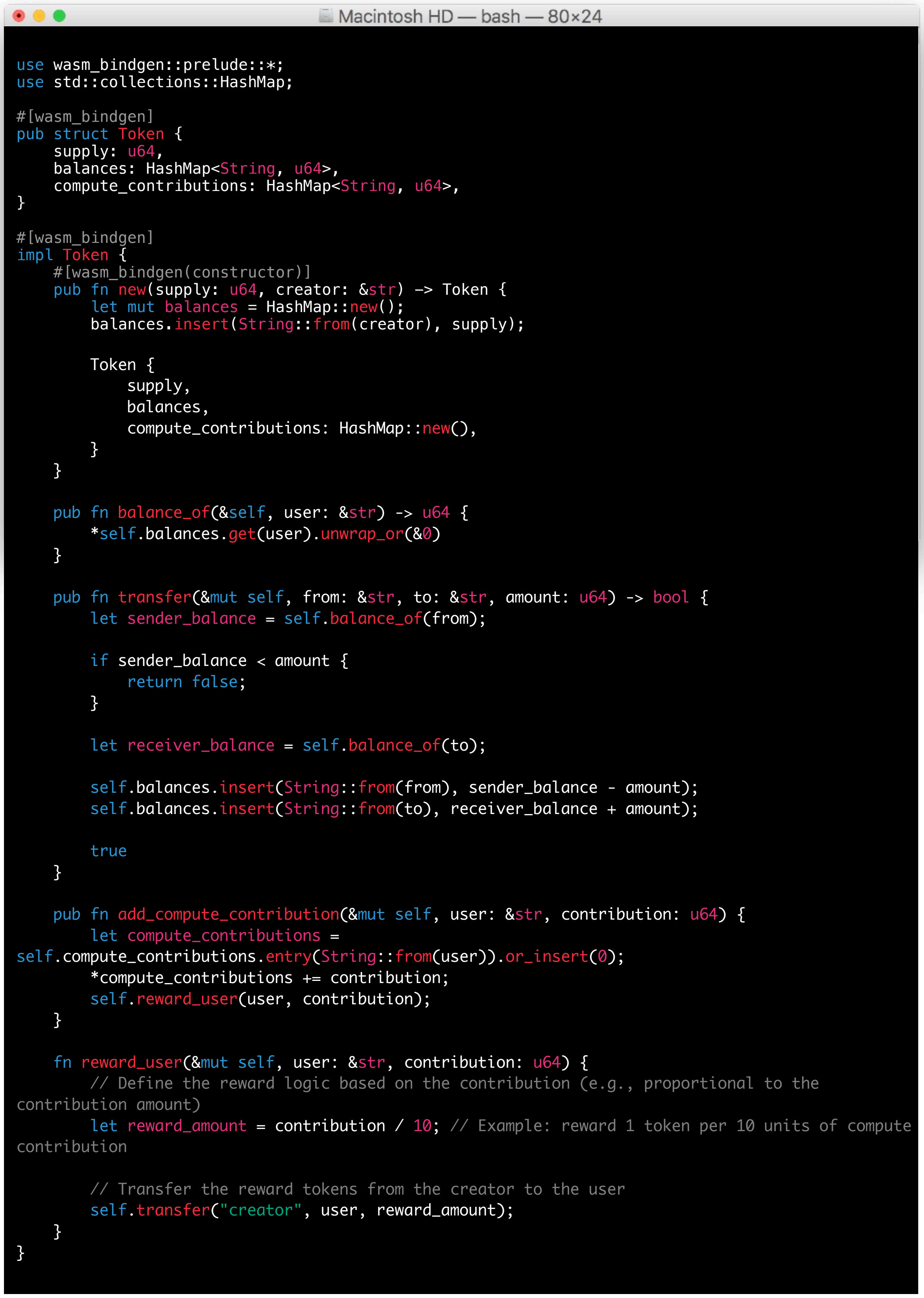
5.4.1 Implementation Of Tokenized Federated Machine Learning In Rust

The simple implementation of this technology using WebAssembly and Rust involves creating a token structure, managing user balances, and rewarding tokens to users upon their donation of compute resources. The provided code in Figure 4. examples showcase the key components to accomplish this:

1. The Token struct stores the total supply, user balances, and compute contributions made by the users.
2. The add_compute_contribution method allows users to report their compute contributions to the system. Upon receiving a user's contribution, this method calls the reward_user function to calculate and distribute the appropriate token rewards.
3. The reward_user function is a private method within the Token struct that defines the reward logic based on the user's compute contribution. In this simple example, users receive 1 token per 10 units of compute contribution. The function transfers the reward tokens from the creator's balance to the user's balance.

By integrating this Rust implementation with WebAssembly, it is possible to create a tokenized federated machine learning system that operates efficiently within web applications. Users can call the add_compute_contribution method to register their compute contributions and receive tokens as rewards.

Please note that this implementation serves as a basic example to demonstrate the concept. In a production environment, more sophisticated reward algorithms, secure user authentication, and additional features for managing the federated machine learning network should be considered.



```
use wasm_bindgen::prelude::*;
use std::collections::HashMap;

#[wasm_bindgen]
pub struct Token {
    supply: u64,
    balances: HashMap<String, u64>,
    compute_contributions: HashMap<String, u64>,
}

#[wasm_bindgen]
impl Token {
    #[wasm_bindgen(constructor)]
    pub fn new(supply: u64, creator: &str) -> Token {
        let mut balances = HashMap::new();
        balances.insert(String::from(creator), supply);

        Token {
            supply,
            balances,
            compute_contributions: HashMap::new(),
        }
    }

    pub fn balance_of(&self, user: &str) -> u64 {
        *self.balances.get(user).unwrap_or(&0)
    }

    pub fn transfer(&mut self, from: &str, to: &str, amount: u64) -> bool {
        let sender_balance = self.balance_of(from);

        if sender_balance < amount {
            return false;
        }

        let receiver_balance = self.balance_of(to);

        self.balances.insert(String::from(from), sender_balance - amount);
        self.balances.insert(String::from(to), receiver_balance + amount);

        true
    }

    pub fn add_compute_contribution(&mut self, user: &str, contribution: u64) {
        let compute_contributions =
self.compute_contributions.entry(String::from(user)).or_insert(0);
        *compute_contributions += contribution;
        self.reward_user(user, contribution);
    }

    fn reward_user(&mut self, user: &str, contribution: u64) {
        // Define the reward logic based on the contribution (e.g., proportional to the
contribution amount)
        let reward_amount = contribution / 10; // Example: reward 1 token per 10 units of compute
contribution

        // Transfer the reward tokens from the creator to the user
        self.transfer("creator", user, reward_amount);
    }
}
```

Figure 4.

5.5 Challenges and Future Directions with Zap.org Oracle Integration

Integrating Zap.org oracle into the ctOS federated machine learning model using WebAssembly offers numerous advantages, but also presents several challenges and future directions that warrant consideration:

Security: Given that WebAssembly operates within a browser environment and Zap.org oracle endpoints manage data exchange, it is essential to address potential security concerns, particularly those related to data privacy and the integrity of computations. Implementing rigorous security measures, best practices, and secure oracle-based endpoints will help ensure the confidentiality, integrity, and availability of the ctOS platform and its data.

Interoperability: To fully harness the potential of WebAssembly for ctOS, the compatibility and interoperability between various machine learning frameworks, libraries, devices, and Zap.org oracle endpoints must be assessed and enhanced continuously. Developing open standards and fostering collaborations within the machine learning and oracle communities will promote seamless integration of diverse tools and resources in the ctOS ecosystem.

Scalability: As the ctOS platform expands and supports increasingly complex machine learning models, it is crucial to evaluate and optimize the scalability of WebAssembly-based solutions and Zap.org oracle endpoints. This will ensure that the platform continues to deliver high performance and efficient resource allocation even as the network and user base grow.

Performance Optimization: While WebAssembly provides near-native performance, there is room for further optimization. Investigating and implementing advanced optimization techniques will help ensure that ctOS, in conjunction with Zap.org oracle endpoints, delivers the best possible performance for federated machine learning.

In conclusion, the integration of WebAssembly and Zap.org oracle offers a promising approach to implementing the ctOS federated machine learning model, providing near-native performance, cross-platform compatibility, and language flexibility. By leveraging the capabilities of WebAssembly and secure oracle endpoints, ctOS aims to advance the development and deployment of decentralized machine learning models, democratizing access to advanced AI technologies for a wider range of users and organizations.

Implementation of a Tiered Validator Node Sales Model

To further enhance the decentralized nature of CryptOS and ensure a robust and secure network, we propose the introduction of a tiered sales model for validator nodes. This model is designed to encourage early participation, reward commitment, and ensure sustained growth and security of the network.

6.1 Objective of the Tiered Sales Model

The primary objectives of introducing a tiered sales model for validator nodes in CryptOS are:

Early Network Support: Encourage early adopters by offering initial nodes at a lower cost, thereby attracting a foundational group of validators who are invested in the network's success.

Risk-Reward Balance: Gradually increase the cost of validator nodes to balance the risk and reward as the network matures and proves its stability and potential.

Sustainable Economic Model: Create a self-sustaining economic model that attracts diverse stakeholders and ensures a continuous investment into the network's development and expansion.

6.2 Details of the Tiered Validator Node Sales

The validator node sales are structured into three distinct tiers, reflecting the developmental phases of the CryptOS network:

First Tier – Launch Phase:

Nodes Available: 10
Cost per Node: $100,000
Focus: This phase targets blockchain enthusiasts and visionary investors who understand the potential of CryptOS and are willing to support the project from its early stages.
Second Tier – Growth Phase:

Nodes Available: 10
Cost per Node: $150,000
Focus: Aimed at more cautious investors who need visible progress and a somewhat established network but are still willing to engage at a relatively early stage.
Third Tier – Expansion Phase:

Nodes Available: 10
Cost per Node: $200,000

Focus: Targeted towards institutional investors and large stakeholders who look for a well-established and lower-risk environment.

6.3 Implementation Strategy

The implementation of this sales model will involve several key strategies:

Marketing and Communication: Transparent and targeted marketing campaigns to explain the benefits and responsibilities of owning a validator node in CryptOS.

Technological Integration: Seamless integration of new validator nodes into the network with support for hardware and software requirements.

Compliance and Security: Ensuring that all sales and operations comply with global regulatory standards and implementing state-of-the-art security measures to protect both the investors and the network.

6.4 Expected Impact

By adopting this tiered sales model, CryptOS expects to achieve a balanced and diversified validator network, which is crucial for maintaining the decentralized integrity and security of the platform. This approach also allows for scaling the network capacity as it grows, supporting the broader goal of democratizing access to machine learning resources.

Conclusion

The CryptOS (ctOS) project presents a groundbreaking approach to democratizing access to computational resources for machine learning and artificial intelligence applications. By combining the Petals.ML federated learning model with blockchain technology, ctOS creates a secure, transparent, and efficient ecosystem that promotes collaboration and innovation among its users.

This whitepaper has outlined the architecture, consensus mechanism, tokenomics, and future technical roadmap of the ctOS project, detailing the integration of the Petals.ML framework with blockchain technology. Through the implementation of a novel consensus mechanism and economic model, ctOS effectively allocates resources and rewards contributors based on their training throughput.

By layering the Petals.ML federated learning model over a blockchain and utilizing a symbolized combined approach, ctOS streamlines the representation and understanding of various model distributions and their corresponding configurations on the blockchain. This fosters trust, transparency, and collaboration within the ecosystem, ultimately driving the development of advanced AI and ML technologies.

As the ctOS project continues to evolve and grow, it holds significant potential to revolutionize the machine learning landscape by making advanced computational resources accessible to a broader range of individuals and organizations. Through the power of decentralized and federated machine learning, CryptOS aims to empower users to harness the capabilities of large-scale ML models in a democratized and collaborative environment.

Citations:

Borzunov, A., Baranchuk, D., Dettmers, T., Ryabinin, M., Belkada, Y., Yandex, A., Samygin, P., & Raffel, C. (March 2023). *PETALS: Collaborative Inference and Fine-tuning of Large Models*. Retrieved April 7, 2023, from https://arxiv.org/pdf/2209.01188.pdf

Spanos, Spanos, et al. System and method for creating a multi-branched blockchain with configurable protocol rules. USOO9608829B2, United States Patent and Trademark Office, 28 Mar. 2017. patents.google.com/patent/US9608829B2/en.