提出AI market的概念，但是没有做出原型。将设备划分为三类。数据产生IoT设备，数据处理网关，模型训练云。文章出发点和contract-based FL一样，但是比较偏概念，作者展示了一些可用实现该平台的一些web3应用

2022 IEEE International Conference on Cloud Computing Technology and Science (CloudCom) | 978-1-6654-6367-6/22/$31.00 ©2022 IEEE | DOI: 10.1109/CloudCom55334.2022.00041

Decentralizing Machine Learning Operations using Web3 for IoT Platforms

John Wickström
Aalto University
Helsinki, Finland
john.wickstrom@aalto.fi

Magnus Westerlund
Arcada University of Applied Sciences
Helsinki, Finland
magnus.westerlund@arcada.fi

Emmanuel Raj
Relex Solutions
Helsinki, Finland
emmanuelraj7@protonmail.com

***Abstract*—Combining machine learning with IoT offers new and exciting possibilities for service innovation. A core challenge for such systems is model training and pipeline monitoring/management while not revealing sensitive data or introducing new attack points. Innovative solutions that reveal network structure and topology are often in conflict with security/privacy design principles. In this paper, we consider the system communication between sensors, edge, and the cloud that does not reveal the origin source or destination of the data. A core contribution is how to apply Machine Learning Operations (MLOps) to decentralized architectures that are not under the direct control of a service provider. The aim is to decouple the Machine Learning (ML) solution, data platform, and sensors while avoiding service level degradation. We utilize an unlinkable end-to-end encrypted asynchronous communication protocol called Whisper that is based on the Ethereum blockchain to achieve sender and receiver anonymity. The Whisper protocol provides a level of darkness that maintains anonymity in communication.**

***Keywords*-Web3, MLOps, IoT, edge, cloud, privacy**

I. Introduction

Introducing IoT-infused AI services (or Artificial Intelligence of Things, AIoT) creates new privacy and security challenges [1], [2]. Architectures that stream data from the edge to a centralized cloud computing infrastructure often demand a trust relationship towards both the controller (collector of data) and the processor (the AI service producer). The CIA triad (Confidentiality, Integrity, and Availability) has long guided secure systems development. However, the platform business model that aims to combine as much data as possible, demands a greater emphasis on privacy. The use of IoT devices in homes or businesses are often to be considered high-risk in terms of both payload data (primary sensor data) and metadata (secondary data, including telemetry). Further, the inferences made using this data must also be considered under the same security and privacy framework.

中心化ML需要一个可信任的中间服务提供商

Let us consider some example types of AIoT services, and list them in the perceived order of increased infrastructure complexity. First, these include direct B2C services such as personal assistants that persistently listen in on conversations in their environment while processing somewhere on the cloud and then delivering the intelligence directly back to the customer. Secondly, B2C services that include human oversight, such as security companies that use video surveillance to monitor remote locations and provide a human operator access to the gathered data and alleged intelligence. A third category is B2B AIoT services, whereby a company outsources the intelligence creation to a processor that may offer the controller data cleaning and modeling solutions. The fourth category, still in development, is data and modeling markets that may, for example, utilize public smart city data that is refined or combined with data from other sources and used to train models that may also be sold through the market.

The development of blockchain technology has shown that mediators (legal entities) are not required to achieve finality in transactions between two parties without an established trust relationship. While much of the public attention has been given to the cryptocurrency phenomena, the promise of distributed cloud infrastructures is technically much more intriguing [3]. As discussed below, the emergence of a new distributed software stack, often referred to as Web3, would possibly help to establish an improved level of trust toward the system. Given that the software is distributed means that systems must be built by default to enforce both security and privacy. A lesser-known fact is that the use of Web3 technology makes it considerably easier to achieve these aims. The paper contributes by defining a Web3 architecture for IoT communication and a first attempt at defining an MLOps process to assist in training and monitoring ML models.

总结的很好，但是blockchain上的数据必须完全透明

The paper is structured as follows. Section 2 reviews the core concepts for privacy and security while ending with breaking down the MLOps process into three parts. Section 3 reviews the relevant parts of the Web3 software stack. Section 4 presents the conducted experiment. Section 5 defines the MLOps process for continuous integration and delivery pipelines that can be used for model training and monitoring.

II. Literature Review

Data mesh networks that facilitate the gathering of data and offer just-in-time availability to the processing entity are often insufficient for guaranteeing both the security and privacy of data [4]. The guarantee for privacy is often considered to be anonymity. However, in data mesh networks, anonymity is hard to guarantee, particularly regarding internal threats from operators and administrators or external actors gaining unauthorized access to the network. Anonymity has been defined as protection against traceability, linkability, and identifiability. While this definition often overlaps with the CIA triad in terms

978-1-6654-6367-6/22/$31.00 ©2022 IEEE
DOI 10.1109/CloudCom55334.2022.00041

Authorized licensed use limited to: National University of Singapore. Downloaded on November 09,2023 at 11:57:34 UTC from IEEE Xplore. Restrictions apply.

of non-functional requirements, it also introduces a more strict view of architectural design. Further, it may even become impractical or illegal to guarantee anonymity to end-users, and as a result, most solutions settle for a degree of pseudonymity.

Related discussions within security by design are the End-to-End Encryption (E2EE) scheme and the post-quantum guarantee, meaning that the data collected is safe from attacks for the foreseeable future [5]. Further, the NIST proposal for Zero Trust Architectures (ZTA) as a requirement for systems development means that authentication and authorization within the network are always performed per transaction, and access is recorded [6]. Hence, systems that adhere to the Zero Trust principles can potentially achieve an improved level of security. When combined, these three ideals offer basic guidelines for secure architectures.

The third aspect we must consider is the processing itself, the process that supports the use of reliable and resilient machine learning. We can define three distinct parts that must be considered, firstly, data acquisition and feature engineering, secondly, model training, and thirdly, inference. In prior work, we have proposed Machine Learning Operations as a general operations method for automating the machine learning workflow [7]. MLOps defines continuous integration and delivery steps for data and models in the machine learning workflow. However, as a general operations method, MLOps does not directly define or address security and privacy requirements. Instead, we can consider proposals for particular modeling approaches, such as ML on homomorphically encrypted data [8] and federated learning [9], which aim to keep input data private in the cloud environment and on the edge, respectively. However, in communication within the network, we must also strive for privacy preserving techniques that do not leak information about the network itself.

III. Web3 Development Stack

The full scope of the Ethereum project [10] is commonly misunderstood. Their ambitious goal is to offer a decentralized alternative to the current software stack. This includes a logical backend, data storage, and web hosting, as well as asynchronous communication. Together, these are often referred to as the *Web3 stack*.

Web3 components are language and platform-agnostic and strive to mimic their conventional counterparts in function to a great degree. However, the important differences can be identified on a foundational level due to a set of opposing priorities. Web3 protocols prioritize privacy and security over metrics such as throughput or latency, whilst the inverse approach is popular for industry solutions. These properties are mutually exclusive such that to enact the former necessarily requires the reduction of the latter and vice versa. For example, to increase privacy, an HTTP request may be routed through a proxy server to hide the request's origin. However, this often requires a routing detour which increases the request's cumulative latency in comparison to the straight-line distance.

A unique strength of the Ethereum Web3 stack is its underlying uniformity. Each component utilizes the same identity-based communication protocol, namely *DEVp2p* [11] . If you consider the blockchain protocol trustworthy, then by the transitive property, this trust can also be extended to the remaining protocols. Another shared feature among the protocols is the *Public Key Infrastructure* (PKI). Public and private keys are derived from unique identifiers, using the *Keccak-256* one-way hash function. These keys can in turn be used for digital signatures as well as for asymmetric data encryption through standard practices. Having a robust PKI built into the platform allows for effortless and frequent cycling of key-pairs, which drastically limits exposure even in worst-case scenarios where a key-pair is deciphered.

A. Smart Contract

The most prominent Ethereum protocol is called *eth*. It entails the blockchain API and intends to resolve the problem of decentralized computation through a *Smart Contract* (SC). These can be thought of as conventional object oriented classes with methods and variables that exist in the memory of a distributed virtual machine called the *Ethereum Virtual Machine* (EVM). The EVM is operated by a multitude of nodes, called miners, in accordance with the blockchain's consensus algorithm. For the Web3 stack, SCs fulfill the role of a highly secure logical backend.

Since the blockchain's content is persistent, SCs inherit the capability of long-term data storage. However, utilizing them for this purpose with large quantities of data is expensive to the point of economic infeasibility. Every injection or mutation of *on-chain* data is considered a transaction with a cost that is equivalent to the amount of work required to apply said action. Thus, the more data you want to inject, the more expensive the transaction. Another glaring problem concerns both privacy and security. Data permanence is a proverbial double-edged sword because the storage of any form of sensitive data is an insurmountable anti-pattern, even if the data in question is encrypted. Any and all data that exists *on-chain* is public domain throughout perpetuity, thus enabling a malicious entity to mount large-scale offline cracking attacks against memory locations that are known to contain something valuable [12].

B. Storage

Ethereum's response to the storage problem is the lesser-known *bzz* protocol, which is colloquially known as *Swarm* [13]. Swarm's purpose is to function as a large-scale decentralized data storage platform that is interlinked with the EVM. This gives SCs seamless access to data at a fraction of the cost through pointers and references. In addition, administrative functions such as managing the ownership of web domains are perfectly suited for a SC.

While there are many competitors in this market, such as *IPFS* [14] or *Filecoin* [15], there is little to no consensus on how to resolve the incentive structure of hosting content. If a host only gets rewarded based on fulfilled requests, over time, only the upper percentile of popular content would end up being hosted in favor of less popular content. This would reduce the utility of the service to near redundancy.

Authorized licensed use limited to: National University of Singapore. Downloaded on November 09,2023 at 11:57:34 UTC from IEEE Xplore. Restrictions apply.

Due to Swarm's close proximity to the already established EVM, this solution may have the greatest chance of eventual success. However, due to Swarm's very experimental and underdeveloped state, it is still an unproven protocol.

In addition to these delineations, it is imperative to understand that decentralized storage platforms are not a replacement for the cloud's near-infinite resources. In fact, due to how constrained standalone SCs are for real-world use cases, the community came up with the concept of *Oracles* [16]. Oracles are autonomous services that make external information available *on-chain* through a SC. For example, a bank can make its data records queriable through a globally recognized SC Oracle at a fraction of the cost compared to uploading all of their data to the blockchain. This clearly distorts the absolute transparency of a blockchain, but the benefits may very well be what enables eventual widespread adoption. Great efforts have been made in an attempt to solve the *Oracle Problem* [16], but a unified solution is yet to be found.

C. Communication

The third relevant Ethereum protocol provides end-to-end encrypted asynchronous communication and is named *shh*, but is more commonly known as *Whisper* [17]. While Whisper is no longer developed, it offers a path to asynchronous messaging protocols that use the Ethereum blockchain network for routing messages without linking sender and receiver. Compared with other asynchronous messaging protocols such as MQTT, that do not offer similar privacy-preserving mechanisms, in its current form, Whisper has scalability challenges and should not be considered for production environments.

Unparalleled privacy on the Whisper network is obtained through a concept referred to as *Darkness*. When a peer publishes a new message to the network, they propagate said message to each one of their peers, who in turn propagate it forward to their peers. This pattern continues with a probabilistic peer ordering until everyone on the network has intercepted the message. This forwarding technique makes deciphering a message's origin or intended destination impractical. In a sense, every network peer functions as a pseudo proxy for the message's sender and recipient. With that said, compared to MQTT or Kafka, the latency of Whisper is both many orders of magnitude higher as well as being unpredictable. The unpredictability stems entirely from the randomized peer propagation process of Darkness. Naturally, a higher latency is directly correlated with the reduction of the hypothetical upper boundary of the networks throughput.

One prominent quandary for asynchronous messaging protocols concerns *Denial Of Service* (DOS) attacks. DOS protection for Whisper is handled by attaching a Proof of Work (POW) requirement to messages. Before a message is published to the network, the sender must first solve a cryptographic puzzle with an equal or greater complexity than the network's minimum threshold. This prevents malicious entities from flooding the network without extraordinary hardware. In scenarios where this threshold is breached, the network can merely increase the threshold slightly to make it exponentially more difficult for an attacker while barely affecting legitimate peers. Individual peers also have the ability to set their personal minimum POW threshold for both incoming and outgoing messages.

A trivial method of parsing incoming messages is to use a first-in-first-out rule, but in practice, messages tend to have a different level of urgency. One way of expressing this urgency in a meaningful way is to set an abnormally high POW threshold for a message. This is significant because you are putting in extra work in advance, in contrast to merely placing a request. An alternative way to express urgency is to set an abnormally low *Time to Live* (TTL) value for the message. TTL specifies the number of seconds after which network peers considers the message to be invalid and drops it. Similar to POW, the network sets a global TTL threshold, but a peer can modify this value on a message-to-message basis. A short TTL value signals that a message needs to be dealt with quickly before the network marks it as invalid. To measure a message's cumulative urgency level, it is common practice to use a function of both POW and TTL with a weighting that favours one.

IV. Experiment

The premise of our experiment is to investigate how to utilize the Web3 stack in conjunction with MLOps processes that facilitate cloud-based machine learning workflows. A commonly identified issue with IoT networks is a lack of security, but often an even bigger problem is a lack of privacy that may also lead to security issues. When an IoT network communicates directly with the cloud, it often reveals information about the devices in and topology of the network that may cause both privacy and security issues. Direct access is a standard procedure for API-based architectures, which means that even if the IoT device uses privacy-preserving methods and is a validated network resource, the metadata leaked during communication is still sensitive. In particular, situations involving data transfers over jurisdictional borders or between two or more parties are seen as problematic.

To achieve our aim of privacy by design, we build a bridge between Web3 nodes and the cloud that in all communication retains a high degree of privacy, see Figure 1. We create a node network consisting of three types of autonomous workers with distinctly different roles, while maintaining a traditional edge-cloud architecture [7]. *Type-1* (T1) workers resemble low-powered IoT devices that primarily produce data and defer decision-making to a more capable device. *Type-2* (T2) workers are closely related to conventional edge devices with substantially better hardware, such as GPUs, which enables them to perform modern machine learning inference. The primary function of T2 workers is to digest intercepted data from T1 workers and subsequently respond with an appropriate recommendation. Finally, *Type-3* (T3) workers effectively function as an anonymizing gateway into the cloud through which MLOps pipelines can be reached by T2 workers.

Authorized licensed use limited to: National University of Singapore. Downloaded on November 09,2023 at 11:57:34 UTC from IEEE Xplore. Restrictions apply.

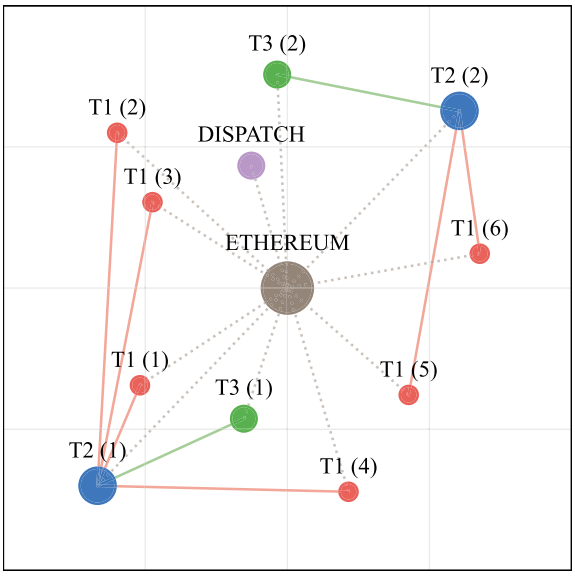

Fig. 1. A hypothetical software defined network topology

A. Service Contract

While devices could hypothetically function without a backend system, distributed or otherwise, having support structures that are decoupled from the device itself enables numerous life-cycle and quality assurance features. For example, if a device was abruptly shut down in an unsafe manner, a backend system can allow for the device to pick up from where it left off prior to the problem occurring. With our proposed solution, all workers are connected to a central SC that is used for various tasks. The benefit of this is threefold.

Firstly (1), it allows participating workers to synchronize by reading the latest shared Whisper configuration from the same source during start-up. The service owner retains the capability to update this configuration through a protected function. Upon such a mutation, all workers are automatically notified through an emitted blockchain event and can dynamically adjust to it. In contrast to having a static Whisper configuration, this approach allows the service to frequently cycle through identities and encryption keys with minimal disruption to the worker nodes.

Secondly (2), participating workers are required to opt-in to the service by registering their Whisper identity as well as defining the address of the device owner. Two workers are not permitted to work together until both are registered and marked as active. Registration also works as a deterrent against undesired behavior due to the increased barrier of entry. This barrier can be substantially raised by attaching an arbitrary fee to the transaction, which simultaneously works as a revenue source for the service.

Thirdly (3), as registered workers are associated with an owner, we can enact a form of indirect management through protected contract functions. For example, if a device was experiencing some form of malfunction, the owner could disable the device through the service contract. This action automatically severs all formed connections for both parties through an emitted blockchain event. In a scenario where malfunctioning device A would continue sending data to device B after the connection has been severed, device B would silently disregard the incoming data for being invalid. We refer to this style of indirect communication pattern as *"Push then Pull"*.

B. Worker Communication

We argue that indirect communication via SC is best suited for *User-to-Machine* (U2M) use cases due to there being effectively no meta-data involved when best practices are followed. As this type of communication is reserved for infrequent actions, such as toggling a device's active status, the associated transaction cost is negligible. However, for ephemeral and frequent *Machine-to-Machine* (M2M) communication, Whisper scales to more realistic proportions as messages only need to be propagated rather than mined.

Messages on the Whisper network have unique identifiers which are exclusively known by the sender and the intended recipient. When two workers communicate using our proposed solution, they do so through a continuous exchange of message identifiers. For example, worker A sends a request to worker B, which generates a message ID of 123. When worker B responds to the request, the payload includes the ID 123, which links it to the proceeding message. This recursive pattern of responding to responses allows a device to have many simultaneous asynchronous conversations that play out in synchronous time. Additionally, it allows both parties to protect themselves by setting predefined expectations for a response, such as the author, turnaround time or payload schema, if a response is expected at all. Consequently, attack vectors that are related to freshness or replayability can be detected and handled in realtime.

C. Dispatch Oracle

The prolific growth of IoT has sparked a drastic evolution of industry standards through innovation and optimization. One such optimization is the IoT-Edge paradigm, where a data consumer is strategically positioned in close proximity to numerous data producers [7]. This nullifies the need to send large quantities of raw data over long distances since a nearby data consumer can now extract any value from the data before sending a minimal subset for long-term storage. While being efficient, this paradigm is also predictable and potentially dangerous because a compromised edge device functions as a gateway into an entire network of other devices. When two entities communicate using Whisper, the notion of proximity does not exist because messages are propagated in a randomized order. In other words, the aforementioned proximity-based optimization is impossible to enact. In fact, the way in which two devices discover each other without publicly announcing some form of attribute or capability is highly ambiguous at best and impossible at worst. Implementing a service discovery system through a SC is less

Authorized licensed use limited to: National University of Singapore. Downloaded on November 09,2023 at 11:57:34 UTC from IEEE Xplore. Restrictions apply.

创建一个中心的dispatch节点？类似contract platform

than ideal due to data permanence and advances in data mining. Similarly, a dedicated Whisper channel where workers regularly announce their services would ultimately be affected by the same vulnerability as SCs.

Our solution for this problem is an autonomous *Dispatch Oracle* (DO) that works over Whisper. When a worker wants to list their services for others or to be connected with someone who is offering a specific service, they do so by sending the DO an encrypted message. For cooperation requests, the DO will broker an anonymous deal on behalf of both parties before an exchange of identities takes place. Once two workers form a connection, they start communicating directly with each other rather than through the DO. While this solution is undeniably centralized, we argue that the potential dangers of an autonomous service are lesser than the aforementioned alternatives. We do not foresee problems concerning scalability as the DO's event loop is highly replicable with tools such as Kubernetes, and the fact that workers only interact with the DO during the handshake phase of conversations. For use cases where no dynamic service discovery is necessary, this step can be entirely disregarded.

V. MLOps workflow

MLOps is an emerging method of fusing machine learning with software development processes such as DevOps and data engineering. It enables automated and efficient building, deployment, and maintenance of complex ML pipelines for various devices and use cases. The MLOps process synchronizes the orchestration of subsystems which increases the potential time-to-market of services as well as real time monitoring of running pipelines.

Figure 2 presents an architecture that facilitates the communication of nodes, as defined in Section IV, and how the MLOps pipeline is maneuvered and leveraged through Whisper. In the following subsections we first describe the MLOps processes in the cloud controller and then discuss how the pipeline connects to the nodes.

A. Type 3 - Cloud Controller

In the Cloud controller layer, multiple services are running on the cloud to perform various synchronous and asynchronous jobs that facilitate the MLOps process. ML models are used to enable fast and reasonably accurate predictions for newly generated data, which is often the case in an AIoT setting. We use the cloud to train models due to the abundance of resources, and we use an ML pipeline to automate each step in the process. The steps for handling specialized ML models per device include training, testing, selecting the best model candidates for deployment, monitoring performance in production, and maintaining prediction quality. In a scalable setting, these steps must become automated tasks that cannot become sufficiently resilient if manually addressed. As part of our previous work [7], we propose to handle them with a cloud orchestration layer composed of five functionalities:

1) Machine Learning Pipelines: : ML pipelines used to build and deploy models are typically proposed by cloud providers as the basis for cloud-based MLOps. Such services provide scalable and programmatically defined computing resources and data storage to enable the training of machine learning models. This service typically also includes an ML model repository and container storage for faster deployments. We consider the following five steps to train and validate our ML pipeline on a given dataset.

- **Data Ingestion**: The ingestion script procures data (based on parameters) and versions the data, which will be used for ML model training. As a result of this step, any experiment (*i.e.,* model training or retraining) can be audited and traced back.
- **Model Training and Retraining**: A script that performs all the traditional steps in ML, such as data pre-processing, feature engineering, and feature scaling before training or retraining any model. Usually, ML models have a set of hyperparameters to tune for fitting the model to the training set. This step can be done manually, but efficient and automatic solutions, such as GridSearch or RandomSearch [18] exist.
- **Model Evaluation**: The inference of the trained model is evaluated according to desired metrics on a separate set of data points named test set. The output of this step is a report on the model performance.
- **Model Packaging**: Once the trained model has been tested, it is serialised and packaged into a container for standardized deployments.
- **Model Registering**: The serialised model candidate is registered and stored in the model registry, from where it is ready for quick deployment into an edge device.

2) Storage: : Storage services are commonly available in a cloud vendor's catalogue for storing structured and unstructured data. Cloud storage solutions are usually auto-scaled depending on the demand, which makes them convenient and efficient to use. In our experiments, we store data for a time period of the last (one) month and auto-erase the historical data to ensure proper data-agency and ownership. Versioning and governance are handled by enabling these features on the storage and in the ML pipeline to track all data movements at each stage of the ML pipeline and deployments.

3) Fleet Analytics: : Fleet analytics is a tool for data analysis and monitoring that includes both the predictions provided by the deployed models and the data collected from edge and IoT devices. Cloud vendors typically offer a few premade tools for evaluating the functionality and health of IoT and edge devices. Additionally, a personalized dashboard can be made to track these revelations in realtime. [19].

4) CI/CD: : Continuous delivery to the edge layer is made possible through a pipeline for Continuous Integration and Continuous Deployment. Deploying, retraining, and maintaining models while retaining end-to-end traceability are the objectives of the CI/CD process (e.g., maintaining logs, tracking versions of models deployed as well as datasets and source code used for model training). The CI/CD pipeline also

Authorized licensed use limited to: National University of Singapore. Downloaded on November 09,2023 at 11:57:34 UTC from IEEE Xplore. Restrictions apply.

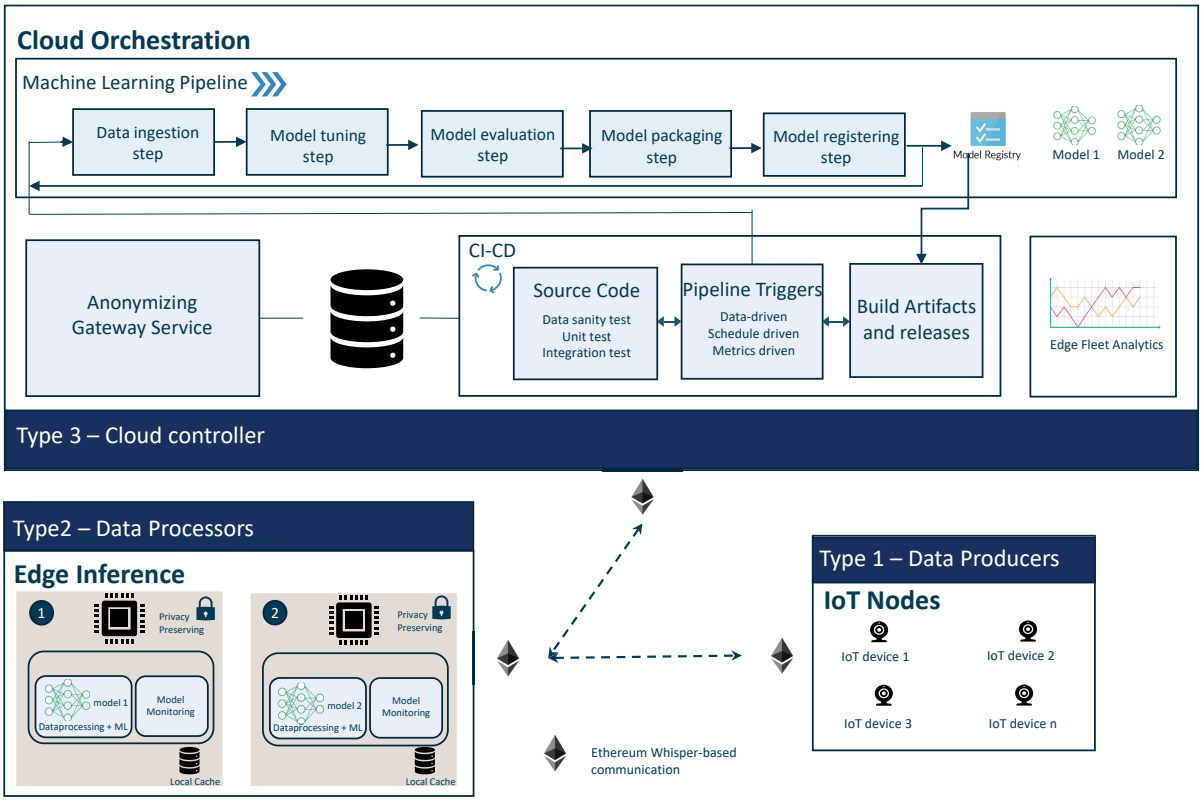

Fig. 2. Proposed Framework Architecture for Edge MLOps for AIoT Applications.

enables triggers to produce artifacts, release for deployment to edge devices, and perform essential tasks in parallel.

5) Anonymizing Gateway Service: : The final cloud controller component is the gateway that implements the Whisper protocol in the cloud and connects the pipeline to the data storage bucket, continuous delivery and monitoring to the type 2 processor. The communication is bidirectional and an analogy for traditional architectures is the publicly accessible endpoint API.

B. Type 2 - Data Processors

The data processors focus on orchestrating operations for the IoT and the edge devices. It enables realtime machine learning inference and works in sync with the Cloud controllers for deploying the ML models to the edge and transferring data from devices for training and testing purposes. More details on each capability of the data processors are given below:

1) Continuous Integration and Delivery from Cloud to Edge: In our experiments for continuous delivery, a high-speed and low-latency network for stable and robust communication and operations is used. For stability and standardization reasons, all operations are run inside *Docker containers*. Continuous deployment pipeline facilitates model deployments in containers to the edge, data transfer, and monitoring (ML models and edge devices).

2) Privacy-preserving communication: Data received from IoT sensors may need anonymization. The component handles communication with the cloud controller over the Whisper protocol, and further maintains the privacy of the IoT devices by cleaning data. While we do not employ homomorphic encryption, this can potentially be incorporated in future work, depending on the use case. Feature engineering is also included as a means to pseudonymize data and prepare data for model use.

3) Hyper-personalization at the edge: Edge and IoT devices in a network can be used in different environments, consisting of different hardware, and be located in separate geographical locations. In many cases, the devices need to perform tasks customized to their respective environments. In such cases, edge nodes can enable customization for each device using a custom ML model. This way, ML models deployed at the edge can optimize and retrain themselves autonomously, while constantly learning to serve better. We train ML models on the cloud as described in sub-section V-A, using data from respective nodes to train customized ML models.

Devices often come with different hardware architectures (e.g., arm64X, x86, etc.) or sensor setups, so personalizing a model for deployment demands customization for specialized data and the variety in the edge hardware. By using

Authorized licensed use limited to: National University of Singapore. Downloaded on November 09,2023 at 11:57:34 UTC from IEEE Xplore. Restrictions apply.

appropriate docker containers tailored to each architecture, we improve hardware stability and standardization among nodes in the network. Further, extensive versioning (data, model, and environment) allows for the explicit targeting of a model per sensor.

4) Automated Machine Learning at the edge: Machine Learning inference and monitoring are automated as part of Continuous Deployment operations. A periodic trigger from the CI/CD process in the Cloud Orchestration layer is implemented to invoke the monitoring feature in the edge devices, to evaluate model drift and to replace or retrain the existing ML model with an alternative model [20]. With this approach, the whole process of ML inference at the edge is automated in realtime.

C. Type 1 - Data Producers

The type 1 nodes are sensors and actuators that collect specific data and perform simple interactions with their environment. In the last decade, we have seen explosive growth in connected devices, which is pioneering the world into the era of Internet-of-Things. The data producers are typically IoT devices that generate data and broadcast it back across the network. However, in future experiments, we aim to evaluate actuators that also interact with their environment.

D. Monitoring ML Operations

In deploying a model or an ML system, it must be continuously monitored [21]. Hence, we systematically monitor our deployed models and other related aspects of the MLOps pipeline in line with the framework proposed in Figure 3. The framework focuses on monitoring realtime data, analyzing in batches and realtime, and based on the analysis govern the system to achieve desirable goals.

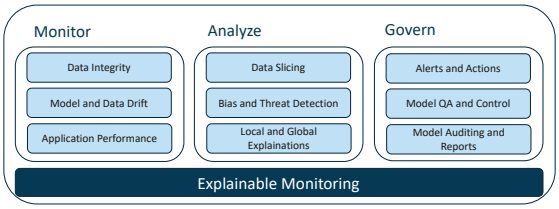

Fig. 3. Explainable monitoring framework for MLOps pipelines, adapted from [22]

Here are the major areas of monitoring in the system that we performed during experimentation. Further work must be done to complete all the framework considerations.

1) Data Integrity and Management: Having correct and accurate data is an essential part of a system based on IoT. In monitoring automated systems, we must also ensure the audit trail of "Who? What? When? and Why?". Likewise, changes that occur through the CI/CD must be adapted into the monitoring of the system. For example, if an update suddenly leads to data samples being dropped, the monitoring must alert of unexpected incidents and trigger appropriate fallback measures. Additionally, as part of the data integrity monitoring, the type 1 sensor may also have to be recalibrated. The recalibration may be automatically triggered at even or uneven intervals in more sophisticated setups.

Finally, as previously stated, the data received by the cloud controller is stored for a time period of one month and historical data is auto-erased to preserve data privacy. The type 2 device uses a local cache for storing recent data for up to a week that can also be used for monitoring purposes and ensuring the system's effective working.

2) Data Drift: The data stored in the type 3 cloud bucket is used to monitor data drift (the degree of change in the nature of data over a period of time). The data drift is measured by using Wasserstein distance, a separate drift model which looks at the baseline and compares inference data [23]. This comparison results in a simple statistical percentage or degree of change in data. If the data has drifted more than the set threshold (e.g., 70 percent in our case), we triggered a retraining of the model that would use the ML pipeline described in Section V-A1 and make a new retrained model available in the model registry, which is later deployed to edge nodes using CI/CD pipelines described in Section V-A4. If the drift is drastic or anomalous, the quality of input data sent to the system needs to be verified.

3) Model Drift: Given that the world is dynamically changing the environment into which an ML model is deployed to perform a task or make predictions is continually evolving. This is where the properties of dependent variables change. In our case, we evaluate the Root Mean Square Error (RMSE) for model predictions for a period of one week (using scheduled batch jobs in the type 2 nodes) to gauge model performance in the real world. When the RMSE is below a set threshold, a new deployment is triggered for a retrained model from the model store and is deployed to the respective edge device.

4) Explainable Monitoring: In an initial version of the explainable monitoring framework, we analyzed the performance of data producers, processors and the cloud controller, and further governed the system based on these indicators to ensure the robust working of the ML system. Monitoring system performance is critical to ensure data integrity, efficient model performance, and robustness of the applications running in the MLOps pipeline. We monitored data and model performance as described in the above sections, whereas application performance is monitored using telemetry data. Telemetry data depict the device performance of a system over a period of time. Telemetry data that measure the device condition can be acquired from sensors such as an accelerometer, gyroscope, humidity, magnetometer, pressure, and temperature. Using this environmental data the system's performance, health, and longevity can be monitored.

VI. Conclusions

This paper details the use of a distributed privacy-preserving application protocol for AIoT services. The comparable difference to more traditional communication protocols is that the revealed metadata about the sender and receiver is minimized. Due to the use of the Whisper protocol, we also find that the

Authorized licensed use limited to: National University of Singapore. Downloaded on November 09,2023 at 11:57:34 UTC from IEEE Xplore. Restrictions apply.

public endpoints achieve improved security. The limitations of distributed protocols in comparison to direct and conceptually centralized are that the throughput is more limited and that latency is unpredictable due to the probabilistic peer forwarding technique used in communication. Suitable use cases include, for example, environmental or personal health monitoring, while it is not intended for sending video imagery. In cases that require a large amount of data to be transferred, a distributed storage service can be utilized, and event communication can still be handled over the Whisper protocol.

The paper also discusses the use of MLOps processes for handling model training, inference, and monitoring. The success of AIoT services is dependent on reliability in used modeling solutions. MLOps defines and automates the CI/CD workflow and the needed ML pipelines. In our experiment, we utilize the same privacy preserving communication protocol for deployment. This means that while models are trained in the cloud (type 3), the pipeline is decoupled from the processor node (type 2) and IoT node (type 1). Decoupling these components allows for interoperability in service design and easy exchange of service providers without having to replace the complete system. Decentralizing the control of the system improves choice but may also improve security as the system designers have to be more transparent.

The Web3 stack offers a promise of improved privacy and security, as evidenced by our experiment. A system based on Web3 technology must follow a privacy and security by design architectural implementation by default. A noteworthy consideration is that using the Web3 stack makes the secure handling of transactions between different parties much easier. Web3 also offers the promise of simple monetization logic through the use of smart contracts. In future research, we will consider how AI markets for AIoT can become autonomous while not revealing sensitive data.

REFERENCES

[1] J. Zhang and D. Tao, "Empowering things with intelligence: a survey of the progress, challenges, and opportunities in artificial intelligence of things," *IEEE Internet of Things Journal*, vol. 8, no. 10, pp. 7789–7817, 2020.

[2] J. Ni, K. Zhang, X. Lin, and X. Shen, "Securing fog computing for internet of things applications: Challenges and solutions," *IEEE Communications Surveys & Tutorials*, vol. 20, no. 1, pp. 601–628, 2017.

[3] M. Westerlund and N. Kratzke, "Towards distributed clouds: a review about the evolution of centralized cloud computing, distributed ledger technologies, and a foresight on unifying opportunities and security implications," in *2018 International Conference on High Performance Computing & Simulation (HPCS)*. IEEE, 2018, pp. 655–663.

[4] N. J. Podlesny, A. V. Kayem, and C. Meinel, "Cok: A survey of privacy challenges in relation to data meshes," in *International Conference on Database and Expert Systems Applications*. Springer, 2022, pp. 85–102.

[5] ENISA, "Good practices for security of iot - secure software development lifecycle," European Union Agency for Cybersecurity, Tech. Rep., 2019.

[6] S. Rose, O. Borchert, S. Mitchell, and S. Connelly, "Zero trust architecture," NIST SP 800-207, National Institute of Standards and Technology, Tech. Rep., 2020. [Online]. Available: https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-207.pdf

[7] E. Raj, D. Buffoni, M. Westerlund, and K. Ahola, "Edge MLOps: An automation framework for aiot applications," in *2021 IEEE International Conference on Cloud Engineering (IC2E)*. IEEE, 2021, pp. 191–200.

[8] A. Acar, H. Aksu, A. S. Uluagac, and M. Conti, "A survey on homomorphic encryption schemes: Theory and implementation," *ACM Computing Surveys (Csur)*, vol. 51, no. 4, pp. 1–35, 2018.

[9] B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y Arcas, "Communication-efficient learning of deep networks from decentralized data," in *Artificial intelligence and statistics*. PMLR, 2017, pp. 1273–1282.

[10] V. Buterin, "Ethereum white paper: A next generation smart contract & decentralized application platform," 2013. [Online]. Available: https://github.com/ethereum/wiki/wiki/White-Paper

[11] Ethereum, "The rlpx transport protocol," 2022. [Online]. Available: https://github.com/ethereum/devp2p

[12] N. Atzei, M. Bartoletti, and T. Cimoli, "A survey of attacks on ethereum smart contracts sok," in *Proceedings of the 6th International Conference on Principles of Security and Trust - Volume 10204*. Berlin, Heidelberg: Springer-Verlag, 2017, p. 164–186. [Online]. Available: https://doi.org/10.1007/978-3-662-54455-6

[13] S. team, "Storage and communication infrastructure for a self-sovereign digital society, v.1.0," 2021. [Online]. Available: https://www.ethswarm.org/swarm-whitepaper.pdf

[14] J. Benet, "Ipfs-content addressed, versioned, p2p file system," *arXiv preprint arXiv:1407.3561*, 2014.

[15] Y. Psaras and D. Dias, "The interplanetary file system and the filecoin network," in *2020 50th Annual IEEE-IFIP International Conference on Dependable Systems and Networks-Supplemental Volume (DSN-S)*. IEEE, 2020, pp. 80–80.

[16] G. Caldarelli, C. Rossignoli, and A. Zardini, "Overcoming the blockchain oracle problem in the traceability of non-fungible products," *Sustainability*, vol. 12, no. 6, 2020. [Online]. Available: http://dx.doi.org/10.3390/su12062391

[17]

[18] J. Bergstra and Y. Bengio, "Random search for hyper-parameter optimization," *J. Mach. Learn. Res.*, vol. 13, no. null, p. 281–305, Feb. 2012.

[19] E. Raj, M. Westerlund, and L. Espinosa-Leal, "Reliable fleet analytics for edge iot solutions," *Cloud Computing 2020: The Eleventh International Conference on Cloud Computing, GRIDs, and Virtualization*, p. 55, 2020.

[20] R. Akkiraju, V. Sinha, A. Xu, J. Mahmud, P. Gundecha, Z. Liu, X. Liu, and J. Schumacher, "Characterizing machine learning process: A maturity framework," *arXiv preprint arXiv:1811.04871*, 2018.

[21] S. Shankar, R. Garcia, J. M. Hellerstein, and A. G. Parameswaran, "Operationalizing machine learning: An interview study," *arXiv preprint arXiv:2209.09125*, 2022.

[22] E. Raj, *Engineering MLOps*. Packt Publishing, 2021.

[23] S. Kolouri, K. Nadjahi, U. Simsekli, R. Badeau, and G. Rohde, "Generalized sliced wasserstein distances," *Advances in neural information processing systems*, vol. 32, 2019.

Authorized licensed use limited to: National University of Singapore. Downloaded on November 09,2023 at 11:57:34 UTC from IEEE Xplore. Restrictions apply.