Promoting Distributed Trust in Machine Learning and Computational Simulation

Nelson Kibichii Bore,[1] Ravi Kiran Raman,[2,3] Isaac M. Markus,[1] Sekou L. Remy,[1] Oliver Bent,[1,4]
Michael Hind,[2] Eleftheria K. Pissadaki,[2] Biplav Srivastava,[2] Roman Vaculin,[2]
Kush R. Varshney[2] and Komminist Weldemariam[1]
IBM Research, [1]Nairobi, Kenya, [2]Yorktown Heights, NY, USA
[3]University of Illinois at Urbana-Champaign, Urbana, IL, USA
[4] University of Oxford, United Kingdom

***Abstract*—Policy decisions are increasingly dependent on the outcomes of simulations and/or machine learning models. The ability to share and interact with these outcomes is relevant across multiple fields and is especially critical in the disease modeling community where models are often only accessible and workable to the researchers that generate them. This work presents a blockchain-enabled system that establishes a decentralized trust between parties involved in a modeling process. Utilizing the OpenMalaria framework, we demonstrate the ability to store, share and maintain auditable logs and records of each step in the simulation process, showing how to validate results generated by computational workers. We also show how the system monitors worker outputs to rank and identify faulty workers via comparison to nearest neighbors or historical reward spaces as a means of ensuring model quality.**

***Index Terms*—blockchain, distributed trust, multi-party model sharing**

I. Introduction

Technological innovation has facilitated large-scale computation where data collection, management, analysis, and interpretation can be achieved as a multi-party process. The complexity of having multiple organizations or participants involved in the collection, utilization and modeling of data, including policies derived from such, results in an increased risk of under-resourced, negligent, or bad faith parties compromising the entire data and modeling pipeline. One domain where this scenario is highly relevant is disease modeling, where simulations are often performed in a distributed fashion, with both inputs and outputs generated or consumed by remote parties in under-resourced settings.

Let us consider a scenario where a disease modeling scientist (DMS) is interested in running experiments to simulate malaria epidemiology and control using a mechanistic simulation model [1], [2]. Currently to study the infection dynamics for a geographic region, a DMS will define parameters including demographics, entomology, health system and intervention policies. The DMS will then likely focus on evaluating different interventions such as distributing insecticide treated nets and commissioning indoor residual spraying in varying proportions. The model outputs can be mapped to appropriate evaluation metrics which allow comparison between different possible intervention policies, for example in terms of disability-adjusted life years (total productive life years lost as a result of infection). In addition, policies are also evaluated based on the cost of implementation, e.g., health system cost and intervention costs. These factors are a great concern in malaria modeling and decision making, where misinterpretation or compromise of an intervention in the model could have far-reaching public health consequences. Given the computational expense of modeling and computations, it is desirable for various decision makers (e.g., government health ministries and non-governmental organizations) to rely on efficacy evaluations computed by others, without re-running models themselves as this may be prohibitively expensive.

This scenario points to the need for both algorithmic accountability and decision provenance systems such that simulation or machine learning models can, at a minimum, be audited and tracked, and preferably have results validated before actions or policies are derived [3]–[5]. In defining the requirements of such a system, we consider two distinct types of functionality.

The first functionality is the ability to log each interaction and the associated metadata from every step in the data process pipeline, from data collection to model creation, utilization and results sharing. This includes identifying participants or organizations that interact with data and models, the methodologies and steps used in generating, handling or manipulating and sharing the data and/or models, the resources used in the process, and any other read, write or execute activities. The second functionality is a validation scheme which can enable users or systems to accept, reject or require further actions. This functionality is intended to provide multiple layers of enhanced trust regarding data and model provenance.

These two capabilities address concerns which arise from pervasiveness of ad-hoc methodologies, weak or unreliable access controls to data, and from readily introduced bias or other ways that data or models can be tainted. More broadly framed as a challenge in reproducibility, access, provenance and reliability, these are issues which impact data, models and derived policies. These are not only concerns for the

978-1-7281-1328-9/19/$31.00 ©2019 IEEE

disease modeling community, but they also arise in several other computational and scientific fields.

The mechanism embedded in these two capabilities is trust, and the need to provide added layers of trust is crucial for technical, practical and ethical or legal reasons [6], [7]. At present, however, the ability to implement the required overhead to address these concerns is neither mature nor reliable enough to gain wide adoption among the respective scientific communities.

There are efforts to address these maturity concerns from a system perspective. Reference [8] presents FairML, an end-to-end toolbox for evaluating predictive model fairness by examining their inputs and treating the algorithm as a black box, quantifying some of the cost associated with promoting fairness in a system. Other end-to-end machine learning pipelines have been proposed where there are multiple entities interacting with data, models and policies [9]. However, these systems assume that (computational) workers which are executing simulations, or learning tasks, can be relied upon to deliver trusted results, where the fault would mainly lie at the algorithm or data preparation level. As it is possible to envision a system where remuneration is paid commensurate with computation, it becomes essential to extend validation and verification schemes to the output produced by workers to maintain system integrity. For these reasons, a trusted system should address at least the following concerns:

Q1: How can we facilitate the sharing of data and models securely among different modeling communities where computation has been performed remotely?

Q2: How can we detect and eliminate invalid results and workers in the system to establish a digital trust policy among participating entities?

Q3: How can we quantify the computational cost to achieve the digital trust when checks are occurring at data, algorithm, and worker level?

As blockchain provides a mechanism to assist with these concerns, in this work we propose a blockchain-enabled distributed system which establishes and promotes decentralized trust between parties involved in disease modeling. There are two categories of blockchain: permissioned and permissionless blockchains [10]. Permissionless blockchain anyone can participate without a specific identity, on the other hand, permissioned blockchain are implemented among a set of participants that need some form of trusted identification because they don't fully trust each other with the goal of securing transactions [10]. We implemented a prototype version of the system based on permissioned blockchain Hyperledger Fabric (HLF) because of its strengths which include resiliency, flexibility, modularity, scalability, and confidentiality [10].

The system is designed and implemented to facilitate (i) the sharing of simulation and/or machine learning assets securely and (ii) validation of results or outputs of simulation or machine learning computations. The initial setup of the blockchain network mimics a disease modeling community consisting of several organizations with whom we have extensively engaged during the course of developing the prototype system and specifically based on our active involvement in the eradication malaria in developing countries [5]. In this blockchain network, each organization can operate its own blockchain node (peer) with its own copy of the shared, distributed and immutable ledger (database) which records all the audit logs as will be explained later. Importantly, potentially every organization running the blockchain peer can independently validate the process and submitted results as well as participate in identifying invalid or problematic results.

The rest of the paper is structured as follows. Section II motivates this work by presenting background materials. Section III discusses the various components and their implementation. Section IV describes the experimental setup and Section V presents results of our experiments. Finally, in Section VI, we conclude the work and also provide possible future directions.

II. Background and Motivation

Data and decision provenance are key considerations in the growing literature of algorithmic fairness, accountability, and transparency (FAT). Beyond the methodologies developed by researchers to ensure the former, there are several considerations around the infrastructure required to promote accountable and transparent models. End-to-end systems have been discussed by [9] and [11], noting required components and possible architectures for capturing and handling events in the modeling and decision making process.

Reference [12] explored a different dimension of model provenance and usage under the recent General Data Protection Regulation (GDPR) legislation, examining how not just data but also model provenance and access needs to be monitored due to the possibility of model inversion and inference attacks. For entities providing machine learning models as a service (e.g., DLaaS [13], MLaaS [14]), the liability exposure implies certain infrastructure requirements such as defined storage limitation (keeping data for only as long as needed), erasure rights, and information rights. Furthermore, it can be argued that under GDPR legislation, remote computation needs to be monitored and tracked in case third parties are contracted for analyzing and using data to build models.

Public policy FAT considerations also have an established literature in the field of decision-making or decision-support systems [15], [16]. Since the 1970s these types of systems have been introduced into multiple industries from manufacturing to health care to provide improved visibility and understanding into business process rules. Another more recent movement has been around the development of Open Algorithms under the OPAL project from MIT. This work outlines multiple challenges with providing trusted models, from secure member on-boarding, to data sharing processes, to worker remuneration [17]–[19]. Additionally, [20] surveyed blockchains as data processing platforms.

Further motivation for this work comes from the critical outcomes associated with the disease modeling community [3], [4], [21] where sub-optimal outcomes have direct consequences to human lives. Unfortunately, depending on the

specific disease modeled, geographical considerations create complicate scenarios for model sharing and development. Due to a lack of computing resources and a fragmented field, researchers are not able to access or further develop models not generated by themselves. Furthermore, the validation and verification (V&V) process can be impaired, since acquisition of real world data is difficult, costly or unreliable. Lastly, the actual complexity involved in evaluating the reward and policy space for managing disease transmission and/or prevention, requires high performance computing, which augments the needs for precise V&V schemes to prevent unnecessary expenditure of computing resources. Although there are various V&V methods at the data and model level [3], [22], [23]; this work explores the utilization of worker outputs for doing individual assessment of worker outputs.

Blockchain technology [10], [24] provides a decentralized platform with the objective of improving transparency, data integrity and security of transactions among involved participants [10], [25]–[27]. Blockchain platforms are based on distributed, shared, immutable ledgers with each participant of the blockchain network possibly having its own copy of the data. The algorithms and protocols of blockchain maintain data replication and consistency of each ledger copy, provide support for recording blockchain data and transactions, assure the integrity of transactions by providing mechanisms for achieving consensus among the participants, guarantee transactions finality and assure that transactions are executed by identifiable entities (non-repudiation). Additionally, generic blockchain platforms, such as Hyperledger Fabric [10], also support execution of business logic along with the transactions in the form of chaincode (or so called smart contracts) which typically have the form of an executable code in languages such as JavaScript, GoLang, etc. Given the above properties, blockchain technologies have been particularly suitable for problems in decentralized settings where there is a need for increased visibility, need of compliance or lack of trust.

In the case of our particular problem, all entries of, and changes to, records on blockchain ledger represent the entities associated with computational or machine learning assets and all relevant operations applied to these assets. Because of the built-in immutability and consensus mechanisms of blockchain, any alteration of an asset (e.g., trained model or simulation result) can be verified against a particular instance of the record on blockchain. Any participant of the blockchain network can audit the records and their validity at any time. Our work additionally contributes distributed validation and verification mechanisms specifically suitable for computational assets and processes, and it leverages a novel compression schema [28] built into the pipeline to address some scalability concerns of blockchains [29].

For the practical scenario of distributed trust from epidemiological simulation results, we present the use of the domain specific simulator, OpenMalaria. OpenMalaria and other epidemiological simulators are part of the class of stochastic mechanistic models, common across the modelling of many uncertain physical systems. They are characterised by a high computational cost associated with the nature of modelling expansive discrete event transitions for large atomic populations. This paper will not go into further detail with regards the chosen malaria model, as the approach presented here is extensible to any domain or non domain specific simulator and further detail may be found in the referenced work [1], [2].

III. The System

In this section, we provide a brief overview of the proposed system shown in Figure 1 and its prototype implementation.

A. Overview

The system leverages blockchain to allow users to track, validate and verify machine learning tasks and results securely. These include simulation outputs, policies, parameters, and other meta-data associated with the simulation process. Each of these can be treated as an asset in a blockchain network, defining a relationship schema as a template for process or provenance engine to map the transition of assets across a blockchain network. The blockchain network ensures that users can share data and associated models in a secure manner among distributed stakeholders in the disease modeling community. In addition, the system has validation engine service that the system uses to validate and verify experiment results, promoting digital trust of these results among participating entities. The core components of the system include: External Systems of Engagement, Process Engine, Message Processor, Learning Layer, Notification Manager and Validation Engine.

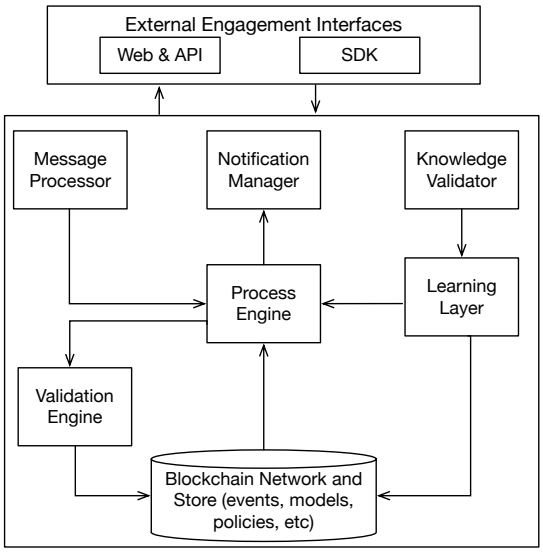

Fig. 1. High-level view of the system components

The *external engagement interfaces* exposes a set of cloud-enabled APIs. The APIs serve as input/output end-points to enable users to post their experiments, receive real time updates of policies, simulation progress and view validated simulation results. Some modeling or simulation requirements (e.g., country or organization-specific privacy acts) may restrict the sharing of sensitive data and models on cloud-enabled

infrastructure. In such cases, the system can be packaged as a software development kit (SDK) and deployed to private on-premise infrastructure. The SDK comes bundled with underlying services as shown in Figure 1, such that engineers can configure appropriately for on-premise experiment setup. Some of the configurations that they can perform include configuration of the application with their on premise blockchain network and still leverage the validation engine.

The *process engine* is responsible for executing model policies based on a specific executor (e.g., OpenMalaria or EMOD [4] [30]) that the users specify in their simulation setup. The system allows a user to upload their executor model, which is then available for other users or organizations who are onboarded onto the blockchain network. The shared executor model is securely stored and managed on the blockchain network to guard from malicious intentions or users through inbuilt cryptography [31]. The process engine receives experiment requests from a message queue (e.g., Kafka or RabbitMQ) and executes the experiment based on the executor model logic. The experimental results are thereafter securely stored on the blockchain network after they have been validated by the validation engine. Also note that the process engine is a decentralized micro-service configured to each node in our blockchain network. This service facilitates end-to-end experience of model runs as discussed in this paper.

The *message processor* is responsible for receiving tasks from external engagement interfaces. This component filters simulation payloads for duplicates to avoid resource wastage and also determines the degree of similarity from previously posted model policies. In the event that there exist simulation duplicates, this component will inform the process engine to retrieve the previous simulation results for further adjustments if needed. For new simulations this component sends the information to the process engine on how to execute simulations and which existing policy results to learn and validate from during model execution. This information is stored securely on the blockchain for validity and authenticity of the data.

The *blockchain layer* is responsible for ensuring decentralized trust among the parties involved in model creation, execution, simulation, storage and validation. This includes policies, simulation results, interaction events, validation data and executors being stored in a secure way to ensure that they are valid, secure and authentic. The component stores most of its data off-chain in an external encrypted IBM Cloudant storage [32], it then stores cryptographic hash and the encryption key on the ledger(on-chain) for data verification purposes. This data storage approach is useful in ensuring that the chain size does not grow exponentially due to the large volumes of data from simulations and computations. The component is also used in managing users and creating a long-lasting trackable reference that others will be able to consult when their model policies match. The use of peers in a Hyperledger Fabric (HLF) [10] blockchain network also ensures that the policies and simulation results are validated by human input to make them effective and useful.

We leverage specialized smart contracts/chaincodes in HLF (i.e., system chaincodes) that manage system level business logic and parameters as used during consensus, endorsement, etc. These chaincodes are trusted distributed applications which leverage tamper-proof properties of the blockchain and an underlying agreement between blockchain nodes which is referred to as an endorsement or endorsement policy [10]. In general, blockchain transactions must be endorsed before being committed to the blockchain while transactions which are not endorsed are disregarded. A typical endorsement policy allows chaincode to specify endorsers for a transaction in the form of a set of peer nodes that are necessary for endorsement. When a client sends the transaction to the peers specified in the endorsement policy, it validity is executed and checked. After validation, the transactions enter an ordering phase in which a consensus protocol is used to produce an ordered sequence of endorsed transactions grouped into blocks.

The *learning layer* is responsible for providing a service where different models can leverage results from other models that are deemed to be similar based on their inputs and the target model. This depends on the asset relationship schema (stored on the blockchain ledger) to externalize any events and metadata for other model simulation activities, in an attempt to use trust data and observations while reducing unnecessary computing cost. The usefulness of the existing simulations output is determined by users that give their report via the knowledge validator on to the learning layer.

The *notification manager component* coordinates the communication between users who have posted their model policies for execution and informing them once completed. This component is also useful when the learning layer requires user validation on new model policies for learning.

The *validation engine* is responsible for ensuring consistency and validity of experiment simulation results to provide digital trust. Experimental simulations and computations for machine learning tasks produce a large volume of data that are complex to validate manually. Adopting the notion of valid computing from [28], a reported computation is declared valid if it is close to the recomputed estimates, otherwise it is flagged as invalid (detailed example in Section IV & V). Drawing insight from the distributed endorsements used by cryptocurrencies, the validation engine, upon receiving computation results from the process engine, selects a set of peers, called endorsers, who recompute the report and endorse the computation if the deviation from recomputation is within an acceptable tolerance. Considering the challenge of system scalability owing to the communication and computation overheads, the validation engine employs the compression schema of [28] for approximate reporting and validation. Given secure access to the computational models and parameters, the endorsers can perform parallel validation through recomputation, while accounting for any randomness in the computational pipeline. The validated states, upon achieving consensus on the endorsements are then added to the blockchain ledger to maintain an audit of validated computations. This record of validated computations allows for simple verification by other agents who just need to ensure consistency of the ledger

(hash chain) to verify the results. Appropriate endorsement consensus policies and validation tolerance can be encoded into the validation engine depending on the extent of trust and accuracy desired in the computation.

B. Implementation

We implemented the system as a suite of microservices running on docker containers (see Figure 2) to enable running on different computing environments. Each computational worker is also packaged as a docker container containing OM executable, application and scripts to engage with message scheduler using RabbitMQ to consume input files, and post the output files as needed to task clerk. During deployment, the workers are capable of executing different models compatible with the OM framework, which can be passed along with the input files required to run a model. The computation results are first stored in the data store for further validation by the validation engine and thereafter persisted to the ledger. In addition to the results, the container also keeps track of the task progress and logs the events to the ledger. Together, these tools are referred to as a worker, and multiple workers can be deployed on a single machine or even distributed across the Internet.

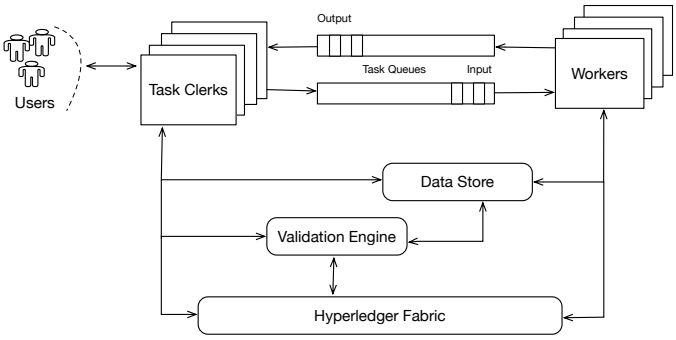

Fig. 2. An example of a deployment instance

The task clerk exposes an interface where model creators can submit their models to be evaluated by the community by running simulations. Model evaluators define a specific instantiation of a model that they would like to evaluate in the experiment. The definition will contain the right input specification that aligns with the model of their choice. The resulting file is then stored in the data store with the associated events persisted to the ledger. Thereafter, the task is posted to the task queues ready for computations.

The task request file, as well as the results for the processed task are stored in a scalable distributed non-relational storage —for our implementation, we used IBM's Cloudant [32] The task clerk, validation engine and workers are connected to the data store for ease of accounting and providing real-time progress of the tasks. To tie task clerks and workers together, we used a messaging fabric. The current implementation harness Advanced Message Queuing Protocol (AMQP) [33] as implemented in the RabbitMQ message queue. This ensures that the framework can keep track of the tasks completed by every worker. The task clerk posts jobs to a message queue which are subsequently picked up by the idle workers. When the worker completes processing the task, the results are then posted to a different channel on the same queue. This instantiation permits workers to be deployed in a wide range of environments, with little requirements on coordination.

Moreover, the implementatiojn of the blockchain layer is based on the HLF [10]. We developed a set of chaincodes to track, store and/or manage data, models and associated metadata as well as the interactions with the assets via process engine. In this work, the blockchain network includes a first, a second, and a third blockchain node. The first blockchain node is configured to receive executor models from one or more users associated with a modeling and create a blockchain transaction including the models metadata. The blockchain transaction is configured to store the models, simulation results and metadata to a shared ledger of the blockchain network. The second blockchain node is configured to request for model simulation results validation via validation engine based on the executor models. The third blockchain node is configured to compute the model simulation validation based on the executor models, outputs, metadata and stored parameters and provide the validation result to the second blockchain node. The model output validation is based on the validation engine component description in Section III-B. Please note that the current configuration of the blockchain network (i.e., the number of peers/nodes) is only for the purpose of this experiment.

IV. Experiments

To evaluate the proposed system, consider a scenario where a malaria data scientist (MDS) wishes to run simulations using a specific model, OpenMalaria (OM), to study malaria epidemiology, understand the disease spread and identify optimal disease interventions strategies. The MDS will therefore post simulations in an environment space where there could be a number of anonymous workers. For our experiment, we deployed 144 docker containers that were bundled with the OM framework as described in the previous section.

An example of a parameterized scenario model in the OpenMalaria framework is shown in Listing 1. Users can alter the *Interventions* section and run their simulations to determine the efficacy upon implementing the intervention. The simulations posted vary on the execution time which largely depend on the intervention chosen as they may require several iterations to evaluate [4]. Upon execution of the simulation, a *reward* in terms of the cost-adjusted disease affected life years saved is returned from the OM framework. After running the simulations for different intervention policies, the MDS will have to validate and verify these results to look for accuracy and precision which often takes time. Validation and verification can be achieved through different ways e.g. comparing results from other models and use of domain experts [4]. Both approaches are not easily achievable because of the extra resources required for either case (amongst other reasons as discussed in Section II).

{

```
  <Model (Parametrized with datafrom field studies)
      >,
  <Demography (demographics)>,
  <Entomology (specific details including
      seasonality)>,
  <Health System (case management, treatment seeking
      , coverage)>,
  <Interventions (e.g., vaccines, Insecticide-
      Treated bed nets (ITNs), indoor spraying (IRS)
       or changing health system) >,
  <...>
}
```

Listing 1. An example of an OpenMalaria scenario specification

To address the challenges of verification and validation, we use our validation engine and in the process try to detect anomalous agents, as evaluated by our experiment here. In our experimental setup, we model anomalous outputs as noisy rewards, where input dependent noise is added to the simulated reward. Here we use additive white Gaussian noise (AWGN), $N \overset{i.i.d.}{\sim} \mathcal{N}(c\sigma(ITN - IRS), \sigma)$ where c is a scaling constant and σ is the standard deviation of the computations. This represents workers who are biased toward promoting the use of ITNs over IRS, and the extent of this bias is characterized by c. The noise model is inspired by the central limit theorem and proves to be a simple benchmark for the noisy/adversarial agents. For this work we evaluate the use of validation engine by introducing noise to 10% of the workers.

The objective of finding anomalous workers, is achieved though the verification and validation of the experiment results, with workers associated with these anomalous results consquently decommisioned from the system. This is crucial in situations where users need to validate and verify experiment results on a large scale, along with providing confidence in comparison of their results with other validated and verified experiments. Thus providing a platform promoting that the results and workers, across all experiment results may be trusted based on sufficient computations.

V. Results

The experiment was designed with 144 workers running OpenMalaria executable models distributed across different clusters on IBM Cloud. Each worker performs 8 OpenMalaria simulations generating a total of 1152 valid OpenMalaria results. Of these sources, a randomly sampled subset of 10% of the sources are anomalous and thus generate noisy rewards as described in the previous section. That is, when an anomalous source is sampled, the reward that is returned is the true reward plus a noise $N \sim \mathcal{N}(c\sigma(ITN - IRS), \sigma)$.

In the experiment a client samples 500 policies, $(ITN, IRS) \in [0, 1]^2$ and queries the system for the corresponding reward. The system employs a worker to evaluate the reward corresponding to the policy and the output is validated using the multi-agent blockchain system. Over the course of the experiment, for each source, $j \in [1152] := \{1, \ldots, 1152\}$, we keep track of the validation statistic V_j and the deviation record Δ_j as follows. Let us assume that the output generated by a source i is being validated by an endorser set $\mathcal{E}_i$. Then, for each $j \in \mathcal{E}_i \cup \{i\}$, if the output is valid, update $V_j \leftarrow V_j - 1$ and if it is invalid, $V_j \leftarrow V_j + 1$. Also, for each $j \in \mathcal{E}_i \cup \{i\}$, record the deviation, $\Delta_j \leftarrow \Delta_j \cup \{\delta\}$, where

$$\delta = \left\| \tilde{Y}_i - \frac{1}{m} \sum_{j \in \mathcal{E}_i} \hat{Y}_j \right\|, \tag{1}$$

is the endorsement deviation. That is, it is the Euclidean distance between the state by the client i $\tilde{Y}_i$, and the average state recomputed by the validating agents, $\hat{Y}_j$, for $j \in \mathcal{E}_i$. The higher the deviation, the less reliable the reported state, and therein the more likely of an erroneous report.

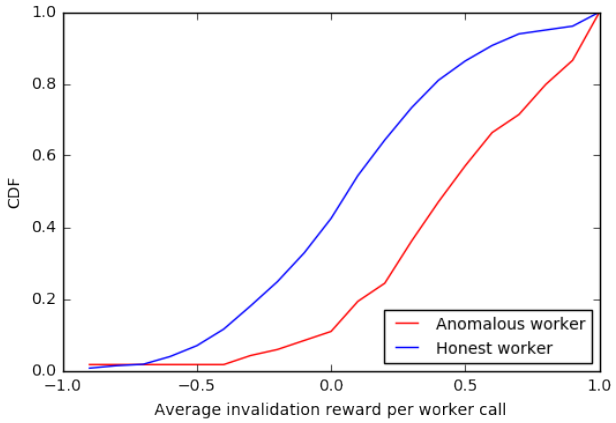

Fig. 3. Average invalidation reward per worker call

We run multiple batches of this experiment and collect the average validation statistics and deviation records for the sources across these batches. We then estimate the probability mass function of the average validation statistic P_{V_j} and that of the deviation P_{D_j} for each source j. We use these distributions to detect anomalous sources.

Note that all workers involved in a validation cycle record the same validation statistic and deviation. That is, if an anomalous worker is part of the validation cycle, resulting in the invalidation of an output, then all sources involved in that validation incur the corresponding deviation and invalidation. However, as there are only 10% anomalous sources, there is a characteristic difference in the deviation and validation distributions of honest and anomalous workers. This is observed in the form of a separation of the corresponding CDFs, as shown in Figures 3 and 4.

First, from Figure 3, we see that the CDF of anomalous workers is biased in comparison to that of the honest workers. This indicates that anomalous sources incur more invalidations than honest workers as expected as the probability of invalidation in the presence of anomalous sources is higher. We can also observe that the presence of anomalous sources affects the validation statistic of honest workers adversely as well, as they undergo more invalidations than they would otherwise.

Next, we consider the CDFs corresponding to the deviations incurred by the sources in Figure 4. We can again see a clear difference in the distributions of anomalous and honest

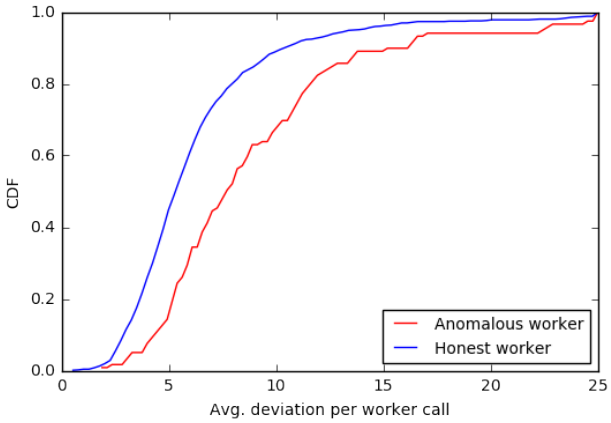

Fig. 4. Average deviation per worker call

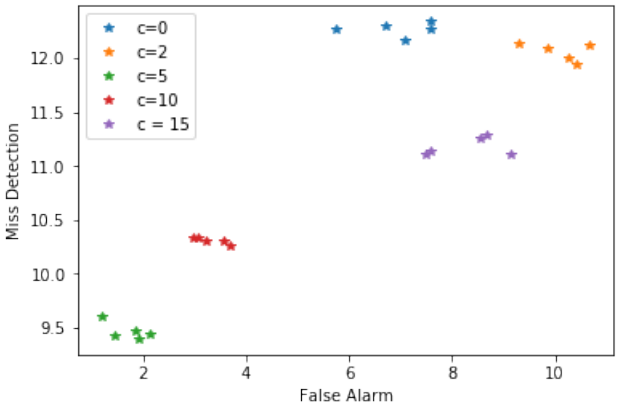

Fig. 5. Performance of anomaly detection using k-NN classifier.

TABLE I
KOLMOGOROV-SMIRNOV TEST STATISTICS BETWEEN HONEST AND ANOMALOUS WORKERS.

	Unbiased, $c = 0$	Biased, $c = 10$
KS stat for P_V	0.11	0.344
p-value for P_V	0.13	$1.9 * 10^{-11}$
KS stat for P_D	0.108	0.407
p-value for P_D	0.165	$3.4 * 10^{-16}$

sources. Also, the anomalous sources incur larger deviations more often than the honest sources as expected.

We invoke the Kolmogorov-Smirnov (KS) test to check for distributional similarity for two main types of anomalies: biased sources ($c > 0$) and unbiased, input-independent anomalies ($c = 0$). The KS test computes the maximum separation in the CDFs of any two data sources. This statistic is used to compute the p-value corresponding to the binary hypothesis test with null hypothesis that the two sources are identical and are statistically different sources.

The results of KS test between honest and anomalous workers for both validation and deviation data are shown in Table I. We can observe that in the biased case, both the validation and deviation profiles are starkly different for honest and anomalous workers, as highlighted by the extremely low p-value. On the other hand, for the unbiased anomalies case, we noted that the p-values are comparatively much higher indicating that whilst different, the statistical confidence in separating the two sources is much lower. This is understandable as detection of anomalies with bias is much easier than unbiased anomalies who just behave like noisy sources.

Next, we consider the task of detection of anomalies using $\{(P_{V_j}, P_{D_j}) : j \in [1152]\}$ as the feature representations. Treating the problem of anomaly detection as clustering into two clusters, we used the k-NN classifier to determine honest and anomalous workers. In particular, we used the total variational distances between the probability distributions as the notion of distance in the k-NN classifier. Other notions of divergence between probability measures can easily be used without loss of generality.

Let us now study the performance of the classifier for different types of anomalies. The false alarm and miss detection probabilities in percentages is represented in Figure 5. The plot considers the anomaly detection performance for unbiased ($c = 0$) and varying degrees of bias ($c > 0$).

A variety of important characteristics are to be noted:

1) First for $c = 0$, the anomalous sources behave like unbiased noisy compute nodes and thus are harder to detect, as indicated by its high miss detection and false alarm probabilities.
2) For small amounts of bias, such as $c = 2$, the false alarm probability worsens with minimal to no improvement in miss detection probability. This is owing to the fact that the biased anomalies create few more invalidations among honest sources, making the classifier misclassify them as anomalies. However, the bias being small also implies that the classifier does not get much more information about the anomalies than in the unbiased case—hence the comparable miss detection probabilities.
3) Next, as the bias increases to $c = 5$, we observe that the anomaly detection gets significantly easier as the bias becomes more recognizable through the invalidation profiles.
4) However, when the bias becomes significantly large, for instance $c \in \{10, 15\}$, the anomalous sources are so heavily biased that they end up creating far more frequent invalidations, which make the classifier categorize honest sources as anomalies. Thus, the false alarm probability increases. The miss detection probability also grows as the anomalies are now well-hidden amongst more honest sources with similar invalidation profiles.

Separate from the task of detecting anomalous workers, it is also important to benchmark the overheads in terms of computation and communication as incurred by the validation engine to address Q3 in the introduction section. Figure 6 considers the average number of recomputes (or approximate computes) per simulation as a function of the validation tolerance, Δ_{val}, for varying sizes of endorser sets, m. With de-

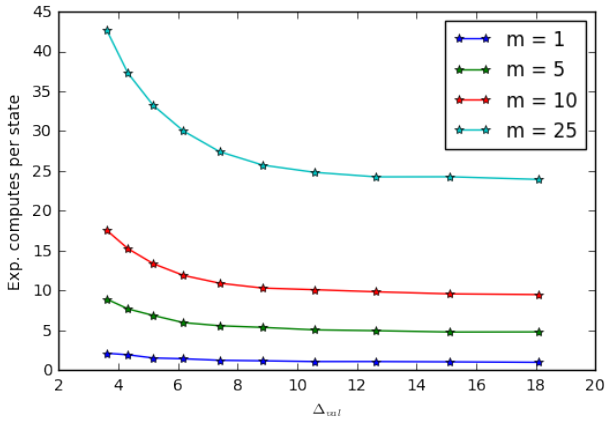

Fig. 6. Computational cost

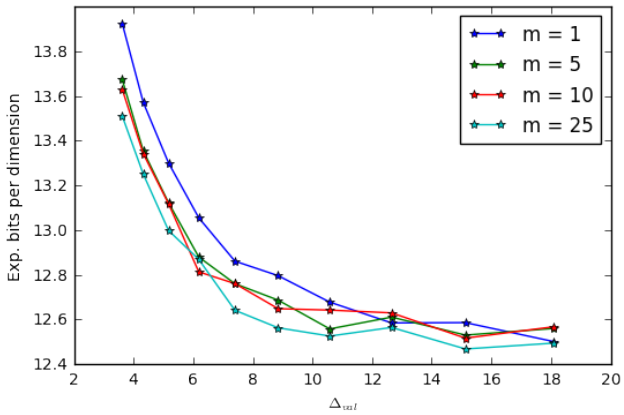

Fig. 7. Communication cost

creasing validation tolerance, Δ_{val}, the computation overhead increases sharply.

When m is small, validation translates to a requirement of collision in recomputation. Since the OM computational framework incurs significant variance, the collision probability is also lower therein increasing the chance of invalidation. Thus, the scaling of computational cost for a given Δ_{val} is not linear with m, especially for small values of Δ_{val}.

The compression schema enforced by the validation engine ensures a significant reduction in the average communication cost per simulation, per endorser, per dimension, as shown in Figure 7. To note the reduction in cost, observe that the average number of bits per simulation, per endorser, per dimension without compression costs 32 bits. The communication cost increases sharply with decreasing validation tolerance and decreases with the number of endorsers. This is because larger endorser sets result in fewer invalidations when comparing against more robust estimates of the mean. The reduced number of invalidations also imply reduced number of refinements and recomputations, i.e., lesser communication with each endorser.

The results discussed above were generated by the validation engine of the system whose main goal is to make a decision of either committing data to the ledger or flag the results as invalid. For invalid results, the engine notifies the users with possible reasons for invalidations with suggestion for the simulations to be recomputed by other workers. Users can then proceed with request for another simulation with possible updates if it was input dependent anomaly.

VI. Conclusion and Future Work

In this paper, we have considered the problem of establishing distributed trust in the computations performed over a distributed multi-agent system. Using a novel combination of blockchain logging and consensus with distributed computing, we presented a validation and verification scheme for large scale computational results. Beyond employing existing validation properties from the blockchain endorsement policy —e.g., identity keys and metadata of the chaincode, we added the validation mechanism to extend the endorsement policy.

We have studied the cost overheads associated with the proposed system and highlighted its capacity to detect faulty or anomalous workers using the validation engine. In addition, the systems capability proofs useful when validating and verifying computational experiment results in large scale which is tedious and complex without the solution. From our initial in-house evaluation, the creation of such trusted systems for distributed computation among 'un-trusting' peers can allow improved collaboration along with efficient data, model, and result sharing that is critical to improved policy design mechanisms. Additionally, such systems can result in creating unified platforms for sharing data and facilitating scientific reproducibility and rigor in machine learning, computational science, and related fields.

Our team's emphasis is on stochastic simulators in the form of an epidemiological model, however the validation and verification techniques presented in this paper are broadly applicable. Moving forward, we plan apply the design to other scientific and machine learning models and disease modeling environments where trust is key e.g., Epidemiological MODeling software (EMOD). In addition, the learning layer component can also be extended to provide meaningful insights to machine learning model creators that can be useful for partial/approximate computations. For example, the running time and resources of disease modeling simulations can be reduced if the computations can be run partially. Lastly, generalizing the system provides a robust framework to investigate fairness, accountability and transparency in broader domains making use of machine learning models.

References

[1] T. Smith, N. Maire, A. Ross, M. Penny, N. Chitnis, A. Schapira, A. Studer, B. Genton, C. Lengeler, F. Tediosi, D. de Savigny, and M. Tanner, "Towards a comprehensive simulation model of malaria epidemiology and control," *Parasitology*, vol. 135, 2018. [Online]. Available: https://doi.org/10.1017/S0031182008000371

[2] P. Eckhoff, "P . falciparum Infection Durations and Infectiousness Are Shaped by Antigenic Variation and Innate and Adaptive Host Immunity in a Mathematical Model," vol. 7, no. 9, 2012.

[3] S. N. Arifin and G. R. Madey, "Verification, validation, and replication methods for agent-based modeling and simulation: Lessons learned the hard way!" in *Concepts and Methodologies for Modeling and Simulation*. Springer, 2015, pp. 217–242.

[4] C. Ferris, B. Raybaud, and G. Madey, "Openmalaria and emod: a case study on model alignment," in *Proceedings of the Conference on Summer Computer Simulation*. Society for Computer Simulation International, 2015, pp. 1–9.

[5] O. Bent, S. L. Remy, S. Roberts, and A. Walcott-Bryant, "Novel exploration techniques (nets) for malaria policy interventions," in *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence, (AAAI-18), the 30th innovative Applications of Artificial Intelligence (IAAI-18), and the 8th AAAI Symposium on Educational Advances in Artificial Intelligence (EAAI-18), New Orleans, Louisiana, USA, February 2-7, 2018*, 2018, pp. 7735–7740. [Online]. Available: https://www.aaai.org/ocs/index.php/AAAI/AAAI18/paper/view/16148

[6] K. R. Varshney and H. Alemzadeh, "On the safety of machine learning: Cyber-physical systems, decision sciences, and data products," *Big data*, vol. 5, no. 3, pp. 246–255, 2017.

[7] M. Hind, S. Mehta, A. Mojsilović, R. Nair, K. N. Ramamurthy, A. Olteanu, and K. R. Varshney, "Increasing trust in AI services through supplier's declarations of conformity," arXiv:1808.07261 [cs.CY], Aug. 2018.

[8] J. A. Adebayo, "Fairml: Toolbox for diagnosing bias in predictive modeling," Ph.D. dissertation, Massachusetts Institute of Technology, 2016.

[9] S. Shaikh, H. Vishwakarma, S. Mehta, K. R. Varshney, K. N. Ramamurthy, and D. Wei, "An end-to-end machine learning pipeline that ensures fairness policies," *CoRR*, vol. abs/1710.06876, 2017. [Online]. Available: http://arxiv.org/abs/1710.06876

[10] E. Androulaki, A. Barger, V. Bortnikov, C. Cachin, K. Christidis, A. De Caro, D. Enyeart, C. Ferris, G. Laventman, Y. Manevich, S. Muralidharan, C. Murthy, B. Nguyen, M. Sethi, G. Singh, K. Smith, A. Sorniotti, C. Stathakopoulou, M. Vukolić, S. W. Cocco, and J. Yellick, "Hyperledger fabric: A distributed operating system for permissioned blockchains," in *Proceedings of the Thirteenth EuroSys Conference*, ser. EuroSys '18. New York, NY, USA: ACM, 2018, pp. 30:1–30:15. [Online]. Available: http://doi.acm.org/10.1145/3190508.3190538

[11] A. Bower, S. N. Kitchen, L. Niss, M. J. Strauss, A. Vargas, and S. Venkatasubramanian, "Fair pipelines," *CoRR*, vol. abs/1707.00391, 2017. [Online]. Available: http://arxiv.org/abs/1707.00391

[12] M. Veale, R. Binns, and L. Edwards, "Algorithms that Remember: Model Inversion Attacks and Data Protection Law," *ArXiv e-prints*, Jul. 2018.

[13] B. Bhattacharjee, S. Boag, C. Doshi, P. Dube, B. Herta, V. Ishakian, K. R. Jayaram, R. Khalaf, A. Krishna, Y. B. Li, V. Muthusamy, R. Puri, Y. Ren, F. Rosenberg, S. R. Seelam, Y. Wang, J. M. Zhang, and L. Zhang, "IBM deep learning service," *IBM Journal of Research and Development*, vol. 61, no. 4, p. 10, 2017. [Online]. Available: http://ieeexplore.ieee.org/document/8030274/

[14] M. Ribeiro, K. Grolinger, and M. A. M. Capretz, "Mlaas: Machine learning as a service," in *14th IEEE International Conference on Machine Learning and Applications, ICMLA 2015, Miami, FL, USA, December 9-11, 2015*, T. Li, L. A. Kurgan, V. Palade, R. Goebel, A. Holzinger, K. Verspoor, and M. A. Wani, Eds. IEEE, 2015, pp. 896–902. [Online]. Available: https://doi.org/10.1109/ICMLA.2015.152

[15] C. Zhengmeng and J. Haoxiang, "A brief review on decision support systems and it's applications," in *IT in Medicine and Education (ITME), 2011 International Symposium on*, vol. 2. IEEE, 2011, pp. 401–405.

[16] B. Lepri, N. Oliver, E. Letouzé, A. Pentland, and P. Vinck, "Fair, transparent, and accountable algorithmic decision-making processes," *Philosophy & Technology*, pp. 1–17, 2017.

[17] T. Hardjono and S. Pentland, "Open algorithms for identity federation," *CoRR*, vol. abs/1705.10880, 2017. [Online]. Available: http://arxiv.org/abs/1705.10880

[18] T. Hardjono, D. L. Shrier, and A. Pentland, *TRUST:: DATA: a new framework for identity and data sharing*. Visionary Future, 2016.

[19] A. Basak, T. Hardjono, and A. Stopczynski, "Fortifid, inc."

[20] T. T. A. Dinh, R. Liu, M. Zhang, G. Chen, and B. C. Ooi, "Untangling blockchain: A data processing view of blockchain systems," *IEEE Transactions on Knowledge and Data Engineering*, vol. 30, no. 7, pp. 1366–1385, Jul. 2018.

[21] J. Tsai, "Transform blockchain into distributed parallel computing architecture for precision medicine," in *2018 IEEE 38th International Conference on Distributed Computing Systems (ICDCS)*, July 2018, pp. 1290–1299.

[22] N. Arifin, G. J. Davis, and Y. Zhou, "Verification & validation by docking: a case study of agent-based models of anopheles gambiae," in *Proceedings of the 2010 Summer Computer Simulation Conference*. Society for Computer Simulation International, 2010, pp. 236–243.

[23] N. Arifin, R. C. Kennedy, K. E. Lane, A. Fuentes, H. Hollocher, and G. R. Madey, "P-sam: a post-simulation analysis module for agent-based models," in *Proceedings of the 2010 Summer Computer Simulation Conference*. Society for Computer Simulation International, 2010, pp. 350–357.

[24] S. Nakamoto, "Bitcoin: A peer-to-peer electronic cash system," http://bitcoin.org/bitcoin.pdf, 2008.

[25] M. Swan, *Blockchain: Blueprint for a New Economy*, 1st ed. O'Reilly Media, Inc., 2015.

[26] D. Tapscott and A. Tapscott, *Blockchain Revolution: How the Technology Behind Bitcoin Is Changing Money, Business, and the World*. Brilliance Audio, 2016.

[27] A. Hari and T. V. Lakshman, "The internet blockchain: A distributed, tamper-resistant transaction framework for the internet," in *HotNets '16*. New York, NY, USA: ACM, 2016, pp. 204–210.

[28] R. K. Raman, R. Vaculin, M. Hind, S. L. Remy, E. Pissadaki, N. K. Bore, R. Daneshvar, B. Srivastava, and K. R. Varshney, "Trusted multi-party computation and verifiable simulations: A scalable blockchain approach," arXiv:1809.08438 [cs.DC], Sep. 2018.

[29] S. Meiklejohn, "Top ten obstacles along distributed ledgers path to adoption," *IEEE Security Privacy*, vol. 16, no. 4, pp. 13–19, Jul. 2018.

[30] A. Bershteyn, J. Gerardin, D. Bridenbecker, C. W. Lorton, J. Bloedow, R. S. Baker, G. Chabot-Couture, Y. Chen, T. Fischle, K. Frey, J. S. Gauld, H. Hu, A. S. Izzo, D. J. Klein, K. A. McCarthy, J. C. Miller, A. L. Ouedraogo, S. Titova, B. G. Wagner, P. A. Welkhoff, E. A. Wenger, C. N. Wiswell, and I. f. D. Modeling, "Implementation and applications of emod, an individual-based multi-disease modeling platform," *Pathog Dis*, vol. 76, no. 5, p. fty059, Jul 2018, 29986020[pmid]. [Online]. Available: https://www.ncbi.nlm.nih.gov/pubmed/29986020

[31] H. Halpin and M. Piekarska, "Introduction to security and privacy on the blockchain," in *2017 IEEE European Symposium on Security and Privacy Workshops, EuroS&P Workshops 2017, Paris, France, April 26-28, 2017*, 2017, pp. 1–3. [Online]. Available: https://doi.org/10.1109/EuroSPW.2017.43

[32] I. Cloudant, "Ibm cloudant," IBM Cloudant, Tech. Rep., 2018.

[33] S. Vinoski, "Advanced message queuing protocol," *IEEE Internet Computing*, vol. 10, no. 6, pp. 87–89, Nov. 2006. [Online]. Available: http://dx.doi.org/10.1109/MIC.2006.116