Received 28 September 2023, accepted 18 October 2023, date of publication 20 October 2023, date of current version 26 October 2023.

*Digital Object Identifier 10.1109/ACCESS.2023.3326418*

RESEARCH ARTICLE

# An Evaluation of Storage Alternatives for Service Interfaces Supporting a Decentralized AI Marketplace

**ANDREI TARA[1], HJALMAR K. TURESSON[2], NICOLAE NATEA[3], AND HENRY M. KIM[2], (Member, IEEE)**
[1]Lucian Blaga University of Sibiu, 550024 Sibiu, Romania
[2]York University, Toronto, ON M3J 1P3, Canada
[3]Openfabric Network SRL, 555324 Sibiu, Romania

Corresponding author: Andrei Tara (research@andreitara.com)

**ABSTRACT** Given the exploding interest in generative AI and the concern that a few companies like Microsoft will monopolize access to such models, we address this centralization risk in the context of a DApp that matches buyers and sellers of various AI services. A key question for a decentralized marketplace is where and how to store the metadata that specifies the services' properties in human and machine-readable formats. Having one or a few actors controlling access to that data constitutes undesirable centralization. We explore data storage alternatives to ensure decentralization, equitable match-making, and efficiency. Classifying decentralized storage alternatives as simple peer-to-peer replication, replication governed by a permissionless consensus, and replication governed by a private consensus, we select an exemplar for each category: IPFS, Tendermint Cosmos and Hyperledger Fabric. We conduct experiments on performance and find that read and write speeds are fastest for IPFS, about two times slower for Tendermint and slowest for Hyperledger. Writing using IPFS and Tendermint takes significantly longer than reading, and finally, specifically with IPFS, write speeds strongly depend on configuration. Given these results and the properties of the storage technologies, we conclude that simple peer-to-peer storage is the best option for the proposed AI marketplace.

**INDEX TERMS** Blockchain, decentralized AI decentralized storage, distributed ontology, semantic models.

## I. INTRODUCTION

Artificial Intelligence (AI) has been one of the key research themes in Computer Science since its early days. Researchers made most of the major theoretical breakthroughs over the last six-seven decennia and developed the mathematics and algorithms [1], [2], [3], architectures [4], [5], and data systems that power AI today [6], [7], [8], [9]. However, the data sets and hardware were lagging, leaving the potential of these systems unrealized for an extended "AI Winter" [10].

In the last decade, with the upgrade of computing power, the associated lowering of cost and power consumption, and the massive amounts of data available through the internet, AI has witnessed explosive growth and increased attention from the scientific community, data scientists, and software engineers and industry at large. This remarkable progress has promised to reshape our society in ways never imagined before by delivering a technological revolution that could positively impact all aspects of our lives [11]. Practical applications of AI, and especially machine learning, have permeated most aspects of our society in applications like autonomous driving [12], [13], [14], intelligent assistants [15], industrial robots [16], infrastructure systems control [17], [18], and supply chain management.

Despite the recent advancements, the promise of AI risks being stifled due to the massive resources needed for state-of-the-art AI concentrating in the hands of a few organizations [19], [20], [21]. Over the past few years, the dominating trend in deep learning has been the expansion of models in terms of parameter count and volume of training data [22], [23].

The associate editor coordinating the review of this manuscript and approving it for publication was Dominik Strzalka.

According to [19], the number of calculations required for cutting-edge performance in AI tasks like natural language understanding, game playing, and common-sense reasoning rose an estimated 300,000 times between 2014 and 2020. With increasing resource requirements, control of AI is being concentrated in fewer and fewer hands.

Furthermore, the fragmentation and complexity of information and data standards and inter-organizational differences make effective collaboration difficult [24]. Many organizations construct moats around their technology by utilizing heavy, monolithic, and rigid data formats [25], [26]. Thus, there is an increasing demand for cross-organizational integration and data exchange protocols that communicate in regulated yet decentralized contexts [27].

Only by rethinking the AI ecosystem – where models are currently hard to combine, closely tied to particular use cases, and difficult for software engineers to apply in real-world applications – can further improvements be made. The existing monolithic data models need to be replaced with a comprehensive strategy capable of connecting multiple simple, specialized and reusable AI modules to achieve the desired results while having a smaller environmental impact and fewer computing and integration requirements.

In smart manufacturing and the Internet of Things, machine-to-machine communication is made possible by a semantic standard – an ontology – providing classification and unambiguous descriptions of products and services [28], [29], [30]. A similar standard for AI could enable a rich ecosystem of cross-organizational, interacting AI agents [31].

This paper explores a novel communication paradigm utilizing a multilayered ontology model designed [31], [32] to stimulate the communication between AI modules in a decentralized multi-organizational context. An ontology is a formal conceptualization of the knowledge representation that provides definitions for concepts and relationships, encapsulating the knowledge of a domain [33]. Ontologies aim to make domain knowledge explicit and remove ambiguities, enabling machines to reason and facilitate knowledge sharing between distributed entities, thus allowing interoperability [34], [35]. The proposed architecture encapsulates data by dividing a monolithic ontology model into separate semantic layers, enabling more flexibility and effective data manipulation [31], [32].

The proposed ontology model facilitates modular machine-to-machine communication, enabling effortless data exchange and discovery between AI modules, but also provides core mechanics for simplifying the human-to-machine interaction and configuration. Recent instances of this kind of machine-to-machine chaining of tools and models (i.e. modules) are Auto-GPT, an AI system that uses a variety of tools together with GPT-4 in an autonomous loop to complete a task [36], HuggingGPT that, similarly to Auto-GPT, relies on GPT-4 to parse a question and generate a plan, then delegates sub-tasks to different specialized models, and finally combines their outputs into a response to the user [37], or ChatLLM a network of language models proposed by [38].

In the presented architecture, ontology concepts provide semantic definitions for the AI modules' input and output communication boundaries. Ontology definition rules enable the modules to detect communication requirements and create dynamic processing pipelines that allow them to process data in an emergent manner that is, to our knowledge, currently not supported in existing implementations. This approach provides a novel perspective on building heterogeneous AI agents by enabling high-level manipulation where the individual modules cooperate as interconnected black boxes to achieve the overall task.

Critical for this decentralized ecosystem of AI services is that the ontology concepts are persisted and replicated on a decentralized storage medium and not stored by a single entity. This system has to allow the effortless composition of AI modules into heterogeneous agents and thus maintain the binaries specifying the composition of the agents and the blueprints of inter-module messages. Thus, we benchmark and evaluate various decentralized storage technologies to uncover their strengths and trade-offs by exploring diverse topologies, parameters and configuration profiles. The experiments explore how the mediums behave on average and at the extreme boundaries of definition.

While decentralized technologies and blockchain protocols offer great diversity, the underlying mechanisms can be classified into three major categories, depending on the nature of replication and the complexity of the consensus mechanism. We have simple peer-to-peer replication, replication governed by a public consensus mechanism, and replication governed by a private consensus mechanism [39], [40].

Based on this general classification, the current paper selected exponents for each category:

- Interplanetary Filesystem (IPFS) for Peer-To-Peer (P2P) Replication [41];
- Tendermint Cosmos for Public Blockchain Consensus [42];
- Hyperledger Fabric for Private Blockchain Consensus [43].

These technologies are evaluated to assess their suitability for storing and replicating data in a decentralized manner. We evaluate and fine-tune their available parameters to detect configurations and profiles that achieve superior results over the baseline configurations for persisting ontology concepts. Furthermore, this work aims to detect bottlenecks in communication and resource utilization, should they appear in the benchmarking process. This is a critical part of a wider effort, namely enabling AI modules to communicate in a decentralized environment by using instances of ontology concepts, thus facilitating the widespread usage of AI.

## II. RELATED WORK

Storing ontology concepts or related resource description framework (RDF) data for knowledge graphs on distributed systems (distributed hash tables (DHT) and blockchains) have attracted significant interest from the research community [44], [45], [46], [47], [48], [49], [50],

[51], [52], [53], [54], [55], [56], [57], [58], [59], but most of the existing work has focused on theoretical aspects without providing quantitative experimental results [44], [45], [47], [49], [51], [52], [53], [54], [55]. To our knowledge, only five papers [46], [56], [57], [58], [59] report on any experimental benchmarking results for storage of these data types. One additional paper compares compression algorithms for RDF triplets [48], but this is not relevant here.

Ruta et al. [46] proposed a blockchain-based architecture for storing and discovering IoT resources where meta-descriptions were stored on a blockchain in RDF triples. The system was implemented on Hyperledger Iroha using 10, 50 and 150 nodes. They reported processing times for discovery and selection, but the results do not reveal the reading time of the RDF, nor are any comparisons with other storage alternatives provided.

Cano-Benito et al. [56] present experimental results from a case study exploring the viability of storing a knowledge graph on Ethereum-based blockchains. They analyze the cost of storing RDF compared to JSON, both in terms of gas and time, and whether a virtualization approach, where JSON is transformed on the fly to RDF, is a better alternative to directly storing RDF. They find that while storing RDF has a clear advantage in that the data can be directly queried, reading and writing JSON are significantly faster. Thus, the proposed virtualization approach captures both advantages. However, they do not compare multiple storage alternatives, only locally run Ethereum blockchains. Beyond the limited scope, the particular choice of blockchain makes this study less relevant as a storage alternative due to Ethereum's unpredictable and sometimes high fees and congestion [60], [61], [62].

Wang et al. [57] investigate knowledge graph storage on IPFS with the file index hashes stored and made accessible on Hyperledger Fabric. They present experimental results of write times using Hyperledge Fabric with four peer nodes running on a single machine. While this is a relevant study, IPFS hashes are small, and they consequently only explore data sizes between one and 64 KB. The reported write speeds of the hashes are between less than one and 10 ms, with a steep increase of around 16 KB. The upper limit of the tested range, 64 KB, corresponds roughly to concepts with 15 properties [58]. This is below the practical range of concept sizes for the type of ontology investigated in the current study [28] making these findings partially complementary to the current study. However, several other questions have not been addressed such as the speeds for reading and writing the actual knowledge base to IPFS or how read and write speeds depend on network size (i.e. number of nodes).

Djeddai and Khemaissia [59] explore a knowledge graph storage solution where the knowledge graph itself is stored on an off-chain (not decentralized storage) database (MongoDB), and JSON descriptions of entities and relations are stored on a blockchain, Hyperledger Fabric. Hyperledger Fabric was configured with two organizations, each with a single node and CouchDB as a storage engine. A critical shortcoming for current purposes is that they don't report the write time for individual entries but only for entire data sets (22.5 to 64 MB and 24 KB to 3 MB for entity and relation entries, respectively), making a comparison difficult. However, the reported write durations range from 5.32 s (24 KB entities) to 740.196 s (64 MB relations). Another limitation is that they do not explore how the configuration of Hyperledger Fabric affects read and write speeds.

Finally, [58] compares the read-write performance across three different systems for decentralized storage of ontology concepts in JSON format. Specifically, IPFS, Hyperledger Fabric and Tendermint Cosmos are compared while varying the number of peers (or nodes) and concept size. They conclude that IPFS provides the best performance, both for reading and writing speeds, especially as the number of peers increases. However, a shortcoming of the study is that each decentralized storage technology comes with a plethora of possible configurations, all influencing the read and write times differently. This makes drawing firm conclusions about the optimal technology and parameter set distributed storage and data propagation difficult. Thus, a systematic comparison exploring a comprehensive set of parameters is needed.

This paper aims to compare storage performance on three distributed systems comprehensively. It furthers the research presented in [58] by exploring a broader set of variables relevant to decentralized storage, like the datastore engine, block size, transaction pool and memory pool. Furthermore, the current body of work is expanded by studying the influence of block sizes on storage latency when persisting ontology concepts on blockchain implementations.

## III. BACKGROUND

In order to achieve AI communication, three preconditions must be met. First, the message structure for the agents must be declared so that modules can interface with each other. Second, there should be an intuitive method for allowing software engineers to link modules via their inputs and outputs. Third, the environment in which they operate must be defined; that is, there should be a medium where agents can be deployed along with their definition of inputs and outputs. The three aspects of the matter are depicted in Figures 1, 2, and 3 respectively, and their complexities will be described in the following sections.

### A. ONTOLOGY DATA-MODEL ARCHITECTURE

Fig. 1 depicts the modular architecture of the messages that can be passed between AI modules. The design criteria for the ontology are described in [32]. Data passed between the modules follow a flexible structure that enables the modules to declare their inputs and outputs. The blueprint for the messages is declared in ontology concepts that conform to the layers presented in Fig. 1, and the data is an actual instance of an ontology concept. In ontology-based systems, one crucial challenge represents naming conventions. The current architecture addresses this issue by introducing a

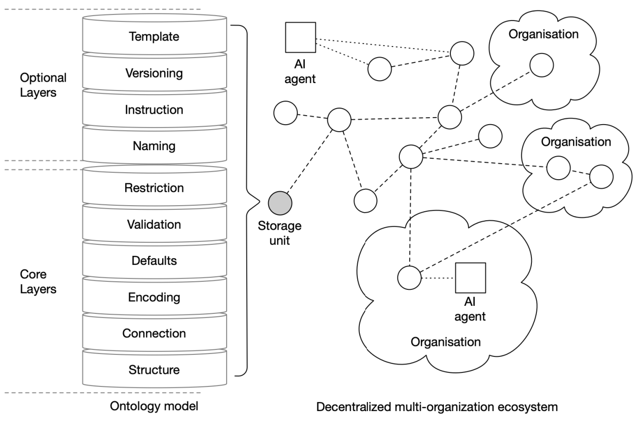


**FIGURE 1.** **Ontology Data-Model Architecture.**

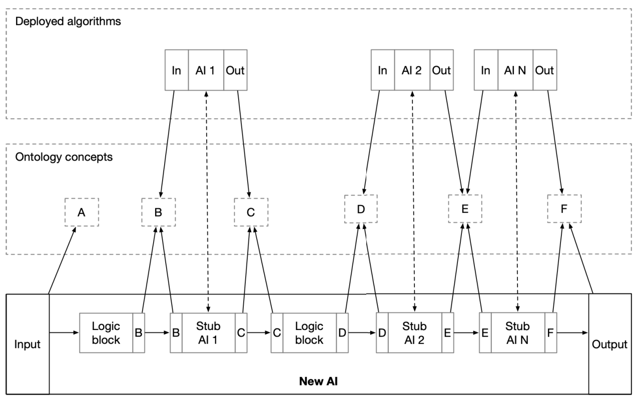


**FIGURE 2.** **Agent Composition.**

dedicated naming layer. This naming layer tags schema, properties, concepts, and relations in a standardized, human-readable format. Organized in contextual profiles, these ensure compatibility across multiple languages and locale formats. An algorithm for describing the naming layer is available in [31]. The model's versatility lies in the layered architecture, which enables multiple modules to operate on different levels of the data without having to fetch it in its entirety. The purpose of each layer is as follows:

- The Structural Layer holds the definition of a message's fields, including their name and data type. The data type can be a primitive (e.g., string, number) or a complex type (e.g., another concept)
- The Connection Layer holds the addresses of the complex concepts declared in the previous layer. These addresses are used for fetching the definitions of the complex data types (concepts) from the decentralized storage.
- The Encoding Layer describes how the concept's fields are encoded. This layer specifies how an AI module should parse or encode the data upon send or receive.
- The Defaults Layer contains implicit values. This layer specifies sensible default values for the concept's fields.
- The Validation Layer stores the rules that define whether or not a given value can be stored in a given field of a concept.
- The Restriction Layer contains cross-field validation rules for a given concept.
- The Naming Layer provides user-friendly labels for the concept's fields.
- The Restriction Layer provides information on the concept's purpose and usage.
- The Versioning Layer holds all the requests suggested for evolving the concept's structure.
- The Template Layer provides partial views over the concept's fields. This allows the concept to be used for multiple purposes when only subsets of a concept's fields are required.

## B. AGENT COMPOSITION

Fig. 2 depicts the process of connecting AI modules from a logical and structural perspective. At a logical level, modules are connected in code through sequences of stubs, thus creating a composite AI agent. These stubs (modules) have known inputs and outputs (ontology concepts) so that a software engineer can instantiate concepts and interconnect modules (bottom layer of the figure). At a structural level, the ontology concepts are stored in a decentralized medium (middle layer of Fig. 2) from where they can be imported either in the logical definition of AIs (bottom layer) or in the actual, running instances of the modules, which are deployed in execution environments (upper layer).

## C. EXECUTION ENVIRONMENT ARCHITECTURE

Execution environments for the AI agents are but a part of the infrastructure necessary for successful deployment. Fig. 3 paints a comprehensive picture of the components that must be communicated to ensure the execution of composite AI agents.

- The Clients upload data and AI definitions onto the platform. They are also responsible for extracting and reporting results to the end user. Furthermore, an end-user can issue the execution of agents using the Client.
- The Coordinator Nodes form the backbone of the decentralized network and serve multiple purposes. Primarily, they translate the Client's commands into actionable instructions for the underlying components. This means that they have to inform Execution Environment Managers about their tasks, and they must listen for results and update the application state by generating and validating transactions for the Blockchain State Machines.
- The Blockchain State Machines hold the application state. Information about algorithms, data, and ontology concepts is written to the blockchain ledger. Furthermore, the transactions should attest to events such as execution starts, data and algorithm uploads, and successful or unsuccessful executions.

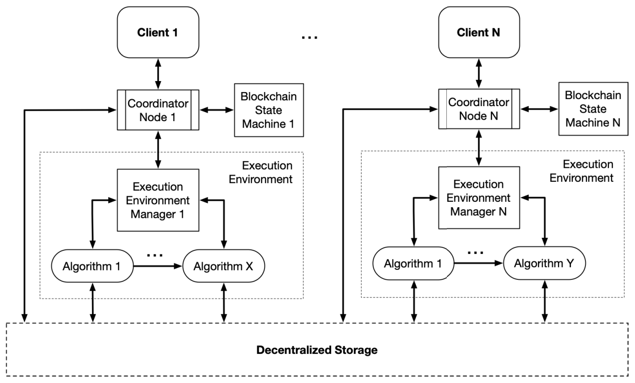


**FIGURE 3.** Execution Environment Architecture.

- The Execution Environment Managers act as supervisors for the execution of agents. As such, they must inspect the responsiveness of the agent execution environments and report the execution results.
- The execution environments act as containers that house the AI agents. Managers instantiate them, and they must be given the AI executable.

### D. DECENTRALIZED STORAGE TECHNOLOGIES

The practical part of this study tackles evaluating the performance of implementations of the Decentralized Storage in Fig. 3. There is a wide range of decentralization technologies, but this paper focuses on a suite that is representative of the major directions in which these technologies are headed. In this sense, the chosen technologies are IPFS, Tendermint Cosmos, and Hyperledger Fabric networks.

IPFS is a decentralized storage technology specifically designed to store files on nodes in the network and generate their addresses by hashing over their content [41]. This behavior also ensures immutability, as a file's address would change if the content changes. One of the advantages of this network is that it does not require reaching an agreement across all the nodes in the network. IPFS's throughput depends on the data size requested, where greater size results in greater throughput up to around 16 MB. Specifically, for 16 KB request size, the throughput is approximately 50 KB/s; for 16 MB, it's close to one MB/s [63], [64].

Tendermint Cosmos is a blockchain technology which ensures that every node in the network writes the transactions in the same order and the network agrees on the transaction order by using a PBFT-based consensus algorithm [42]. Proof-of-stake protects against Sybil attacks by requiring validating nodes to hold a stake to participate in the consensus. Tendermint provides Byzantine Fault Tolerance (BFT) guarantees [65], [66], that is, it can reach consensus in the face of up to N/3 node failures, including adversarial (Byzantine) nodes. The throughput for a 128-node network is around 400 transactions per second [66]. Reference [66] investigated Tendermint's scalability by measuring how throughput and transaction latency depends on the number of nodes. Testing 16, 32, 64 and 128 nodes, they found gracefully decreasing performance with latencies increasing from 2.72 to 3.45 s and the throughput dropping from 535 transactions per second (tps) to 438 tps. A performance decrease with an increasing number of nodes is expected due to the message passing required by the PBFT-based consensus algorithm [67].

Hyperledger Fabric is a permissioned blockchain technology where the nodes present cryptographic certificates to confirm identities, and these identities allow them to take part in the network [43]. The nodes have specialized roles in the network, such as transaction orderers, transaction validators and certificate generators. This approach is often taken in enterprise environments. Hyperledger Fabric has Crash Fault Tolerance (CFT) guarantees [68], meaning it can withstand up to N/2 node failures (crashes). However, it does not guarantee fault tolerance in the face of adversarial (Byzantine) nodes, which makes sense given that it's a permissioned blockchain. Hyperledger Fabric has sub-second transaction latency under throughput up to 200 tps [69]. However, the blockchain doesn't scale well as the number of nodes increases above 12 with the result that the latency and throughput drop sharply [70].

## IV. EXPERIMENTAL METHODOLOGY

A series of experiments were devised to study the efficiency of the various decentralized storage back-ends for persisting ontology concepts. They were designed to persist and retrieve concepts from the underlying decentralized environment. Four orthogonal parameters were varied to assess the environment's performance: concept sizes, infrastructure topology configurations and storage engines for each decentralization technology.

### A. CONCEPT SIZES

Each concept comprises ten files - one for each layer - plus a metafile that identifies the concept and its author (see Fig. 1). These files are the ones that persist on the decentralized storage. Out of the ten layers, we varied the number of properties in the structural layer from ten to 10,000 in exponential steps, and, since each property at the structural layer is given a new name, also the naming layer. The upper limit is of the same magnitude as the 30,000 properties listed in ECLASS [28]. The final concept sizes in bytes were approximately 41.4 KB, 54.5 KB, 186.1 KB, 1.5 MB and 14.6 MB.

### B. INFRASTRUCTURE TOPOLOGY CONFIGURATIONS

#### 1) P2P ENVIRONMENT

The IPFS topology was tested in two modes. The first topology only includes a network of IPFS daemons (regular peers), which transmit information among themselves only when requested. The second topology includes a Cluster network alongside the Daemon network, and each peer in the Daemon network has one correspondent in the Clustering network. Having a clustering network brings the benefit of disseminating the files received by one member of the Daemon network to all the peers in the same network by emitting signals in the Clustering network and thus improving replication and fault-tolerance. We tested three different storage engines for IPFS: FlatFs, BadgerDs and Lowpower, see Table 1.

#### 2) PUBLIC BLOCKCHAIN ENVIRONMENT

The Tendermint network comprises only one type of node, which is instantiated the appropriate number of times according to the network size. However, the nodes operated in two modes: Commit and Sync. Further details about them will be given in the Results section. For Tendermint, we tested five different storage engines: ClevelDb/GoLevelDb, BadgerDb, BoldDb, RocksDb and MemDb, see Table 2.

#### 3) PERMISSIONED BLOCKCHAIN ENVIRONMENT

Hyperledger Fabric, being a permissioned blockchain, requires multiple types of nodes to form an operational network. Every configuration contains one orderer node that receives signed transactions from the peers, packages them into blocks, and appends them to the blockchain. The components varied in this topology are the peer nodes, responsible for validating, executing and signing the transactions. They were grouped into two organizations, and each organization contains one Certificate Authority node that is responsible for generating certificates that attest to the identity and trustworthiness of the nodes in the organization. The Hyperledger framework was tested with two different configurations: LevelDb and CouchDb, see Table 3.

### C. HARDWARE CONFIGURATION

The suite of experiments was implemented using the Java Microbenchmark Harness Tool. The experiments adhere to the guidelines presented in [71], and each experiment consisted of five warm-up iterations of 10 seconds each, followed by ten 30-second measurement iterations. The experiments were run on a machine leveraging an Intel 7700HQ CPU, 24 GB of RAM and one TB of SSD.

## V. EXPERIMENTAL RESULTS

Here we report the results of the experiments conducted to evaluate the replication environments with regard to optimal parameter configurations and feasibility in the context of the proposed model. The experiments measured how the latencies for persisting and retrieving concepts from the underlying environments depended on four types of parameters mentioned in the previous section.

We report the results for 660 individual experiment configurations averaged over multiple runs. The results were obtained by averaging the values obtained in 64 hours of continuous simulation, each experiment composed of five warm-ups and ten execution iterations where each iteration was limited to ten and 30 seconds, respectively.

**TABLE 1. IPFS Simulation Parameters.**

| Parameter | Values |
|---|---|
| Network setup | Default I Cluster |
| Topology (peers count) | 2 I 4 I 8 |
| Concept size (properties count) | $10^1$ I $10^2$ I $10^3$ I $10^4$ I $10^5$ |
| Operation | Read I Write |
| Datastore engine | BadgerDS, FlatFs, Lowpower Profile |
| Warmup iterations | 5 (each running for 10s) |
| Execution iterations | 10 (each running for 30s) |

### A. PEER-TO-PEER NETWORK RESULTS

In this section, we report the impact of various configurable parameters listed in Table 1 on the performance of the proposed model by utilizing IPFS as the underlying environment. The results presented in this section were averaged over multiple runs. In total, 2700 iterations were conducted, demanding 19.5 hours of continuous simulations.

Fig. 4 and Fig. 5 show the experimental results for reading and writing operations providing various observations and insights into the different elements and their operational impact.

*Observation 1:* The measurements depicted in Fig. 4a, Fig. 4c, Fig. 4e, Fig. 5a, Fig. 5c, and Fig. 5e suggest that the network size has a limited influence on the read time. This effect can be explained by the fact that the network configuration is stable; all nodes are active all the time, and no node is joining or leaving the network. By eliminating hazardous circumstances and downtime, if the information cannot be discovered on the target node, it can be efficiently resolved by one of the neighbouring peers.

*Observation 2:* With IPFS *Default* network topology and the same datastore engine context, the network size has a minor influence on the writing operations latencies, as shown in Fig. 4b, Fig. 4d, and Fig. 4f. While the network arrangement is stable, the data replication mechanism utilized by peer-to-peer networks relies on an uncomplicated propagation protocol where the information propagates progressively and only when a peer requests it. Therefore is no need for multiple rounds of synchronization and messages exchanged between nodes, as in the case of consent-driven networks.

*Observation 3:* The measurements depicted in Fig. 4d, Fig. 4e, and Fig. 4f indicate that *BadgerDS* performs better than *FlatFS* and *Lowpower* when writing with smaller concepts. The difference is mainly explained by *BadgerDS* use of LSM tree structures, which provide an optimized indexing mechanism leading to better operation throughput [72]. However, as expected, the gap diminishes as the concept size increases since network communication becomes the dominant operation.

*Observation 4:* Fig. 5 depicts the results for the IPFS *Cluster* network topology; this delivers data orchestration across a swarm of IPFS instances by automatically allocating, replicating, and tracking stored information across all peers on the network. When comparing the results from Fig. 4 with Fig. 5, we notice a significant difference in both read

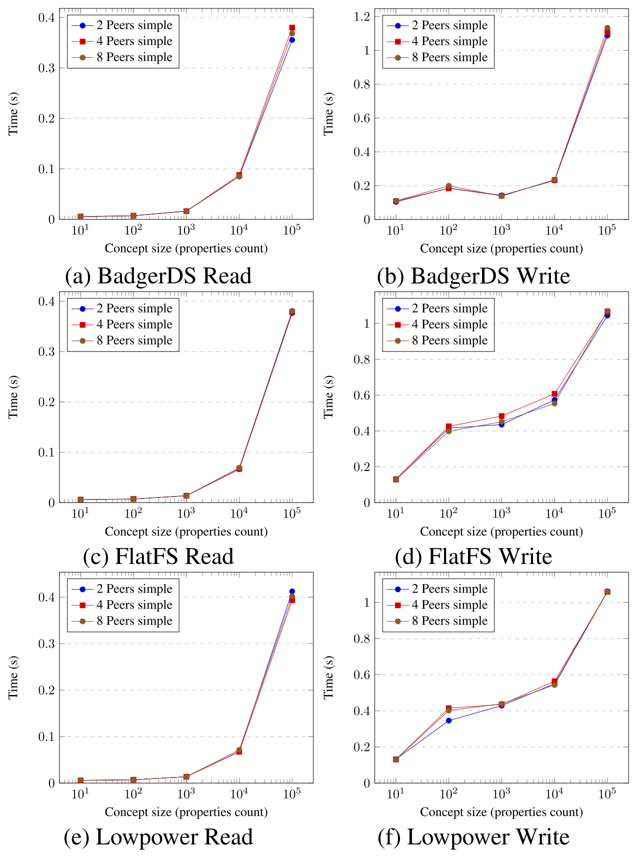


**FIGURE 4.** IPFS Default Setup Results.

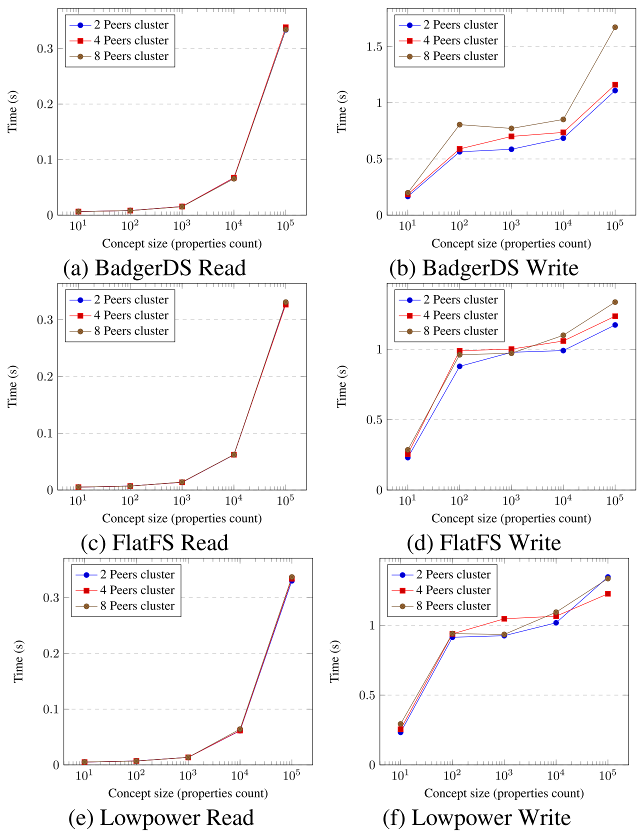


**FIGURE 5.** IPFS Cluster Setup Results.

and write operations. Generally, read operations are slightly faster, as the data is already replicated on all peers and, therefore, available locally. On the other hand, compared to the *Default* topology, the *Cluster* write operations are considerably slower, even though we are still writing to a single peer because each node in the cluster is informed that it should replicate the newly published data.

*Observation 5:* As expected, at the level of individual configurations, the *Cluster* network has slower write latencies, see Figs. 5b, 5d, and 5f. The latency increase is a consequence of the number of nodes on the network. A final observation is that the *Cluster* network setup will require significantly more space since all the information is replicated on all nodes.

*Observation 6:* The measurements shown in Fig. 5b and Fig. 5d indicate that when writing small concepts, *BadgerDS* performs better than *FlatFS* and *Lowpower*, but the gap closes as the size of the concepts increases, and *FlatFS* even surpasses *BadgerDS* for concepts of a large enough size.

*Observation 7:* The *Lowpower* profile is a mode modifying IPFS node configurations to reduce resource utilization by lowering communications and discovery capabilities. Fig. 4e, Fig. 4f, Fig. 5e and Fig. 5f show the result of running the IPFS nodes with the *Lowpower* profile for read and write configurations. The *Lowpower* profile results are comparable to the *FlatFs* configuration, particularly for the small-size concepts. However, as anticipated, for the big-size concepts, the performance decreases since the network and memory throttling of *Lowpower* are starting to materialize.

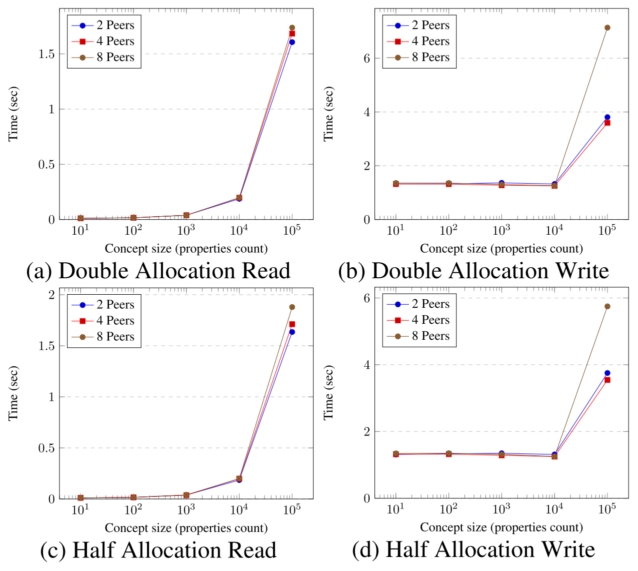
**TABLE 2.** Tendermint Simulation Parameters.

| Parameter | Values |
|---|---|
| Topology (peers count) | 2 \| 4 \| 8 |
| Concept size (properties count) | $10^1$ \| $10^2$ \| $10^3$ \| $10^4$ \| $10^5$ |
| Operation | Read \| Write |
| Memory pool | Half \| Double |
| Transaction pool | Half \| Double |
| Datastore engine | GolevelDB, BadgerDB, BoltDB, CleveldDB, RocksDB, MemDB |
| Warmup iterations | 5 (each running for 10s) |
| Execution iterations | 10 (each running for 30s) |

## B. PUBLIC CONSENSUS BLOCKCHAIN RESULTS

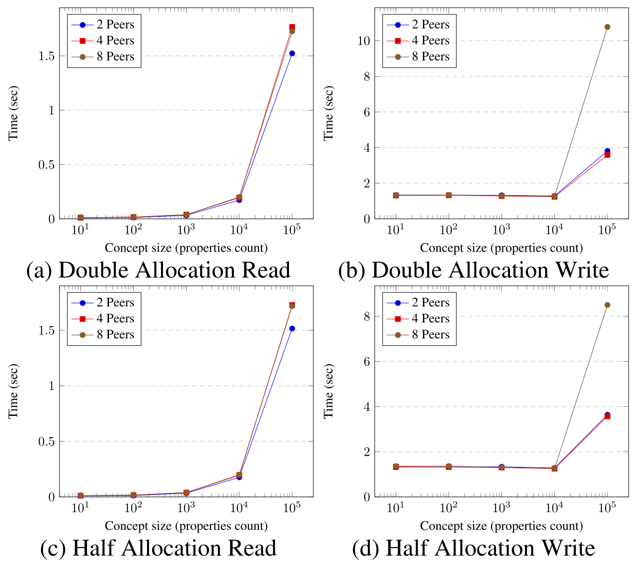
In this section, we report on the impact of various configurable parameters listed in Table 2 on the performance of the proposed model by utilizing Tendermint Cosmos as the underlying environment; the results presented in this section were averaged over multiple runs. In total, 1800 iterations were conducted, demanding 12 hours of continuous simulations for the transaction and memory pool variation experiments and 2700 iterations demanding 18 hours of continuous simulations for the datastore engine variation experiments.

Fig. 6 and Fig. 7 illustrate the results for writing and reading concepts by adjusting transaction and memory pool size. Fig. 8 portrays the results for writing and reading concepts by adjusting various datastore engines. These results



(a) Double Allocation Read (b) Double Allocation Write
(c) Half Allocation Read (d) Half Allocation Write

**FIGURE 6.** **Tendermint Transaction Pool Variation Results.**



(a) Double Allocation Read (b) Double Allocation Write
(c) Half Allocation Read (d) Half Allocation Write

**FIGURE 7.** **Tendermint Memory Pool Variation Results.**

provide observations and insights into different elements and their impact described below.

*Observation 8:* As can be observed in Figs. 6a, 6c, 7a, and 7c, the time required to perform the reading of concepts is only slightly impacted by the size of the transaction and memory pools. As expected, the read operation does not require accumulating transactions into the transaction pool or allocating extra memory.

*Observation 9:* Figs. 6b, 6d, 7b, and 7d depict the results for writing concepts. The time required to write small concepts is consistent between different topologies, suggesting that the layered design of the current proposed ontology model successfully exploits the blockchain fragmentation structure. Notably, for big concepts and extensive network configuration, the latency increases first when the memory or transaction pool doubles. At first glance, it may look counter-intuitive, but this can be explained by the fact that big concepts are not mapping very well on the blockchain structure, and bigger pools require more management, especially in more extensive networks; thus, they require multiple rounds of synchronization and messages exchanged between nodes.

*Observation 10:* When analyzing Fig. 8a, Fig. 8c, Fig. 8e, Fig. 8g, Fig. 8i and Fig. 8k, one can see that the network sizes have a weak influence on the read latency for all datastore engines. Especially for small-size concepts, the points on the graph are almost overlapping, emphasizing how the current ontology model successfully exploits the blockchain fragmentation structure. However, the simulation results show a nearly exponential increase for the big-size concepts. This behavior is expected since big concepts are split before being persisted into blockchain structure, and the read operation implies reassembling them together. This pattern is consistent and independent of the storage engine, depicting the stability and versatility of the proposed ontology model.

*Observation 11:* Fig. 8b, Fig. 8d, Fig. 8f, Fig. 8h, Fig. 8j and Fig. 8l show the results of persisting data to the Tendermint blockchain. The combinations of the number of peers and the database back-end show no noticeable differences for the small concept sizes. All the values are constantly above one second, representing the consensus time; this is explained by the fact that all concepts are persisted in one consensus round. The differences appear only for the big-size concepts, where the concepts must be fragmented across multiple transactions and possible blocks, requiring multiple rounds of consensus. Like the read case, the pattern is consistent and independent of the storage engine.

### C. PRIVATE CONSENSUS BLOCKCHAIN RESULTS

In this section, we report the impact of the various configurable parameters listed in Table 3 on the performance of the proposed model by utilizing Hyperledger Fabric as the underlying environment. The results presented in this section were averaged over multiple runs. In total, 1800 iterations were conducted for a total of 12 hours of continuous simulations for the transaction and block size variation experiments. Furthermore, the datastore engine variation experiments required 900 iterations and six hours of continuous simulations.

Fig. 9 and Fig. 10 present the results for writing and reading concepts by adjusting the sizes of the transaction pool and the blocks while varying the rest of the parameters. Fig. 11 shows the results for writing and reading concepts by adjusting the size and topology for various datastore engines.

*Observation 12:* As can be observed in Fig. 9a, Fig. 9c, Fig. 10a, and Fig. 10c, the time required to read concepts is impacted by the size of the transaction pool for big-size concept sizes when reading data from the eight-peer network. As expected, the read operation does not require accumulat-

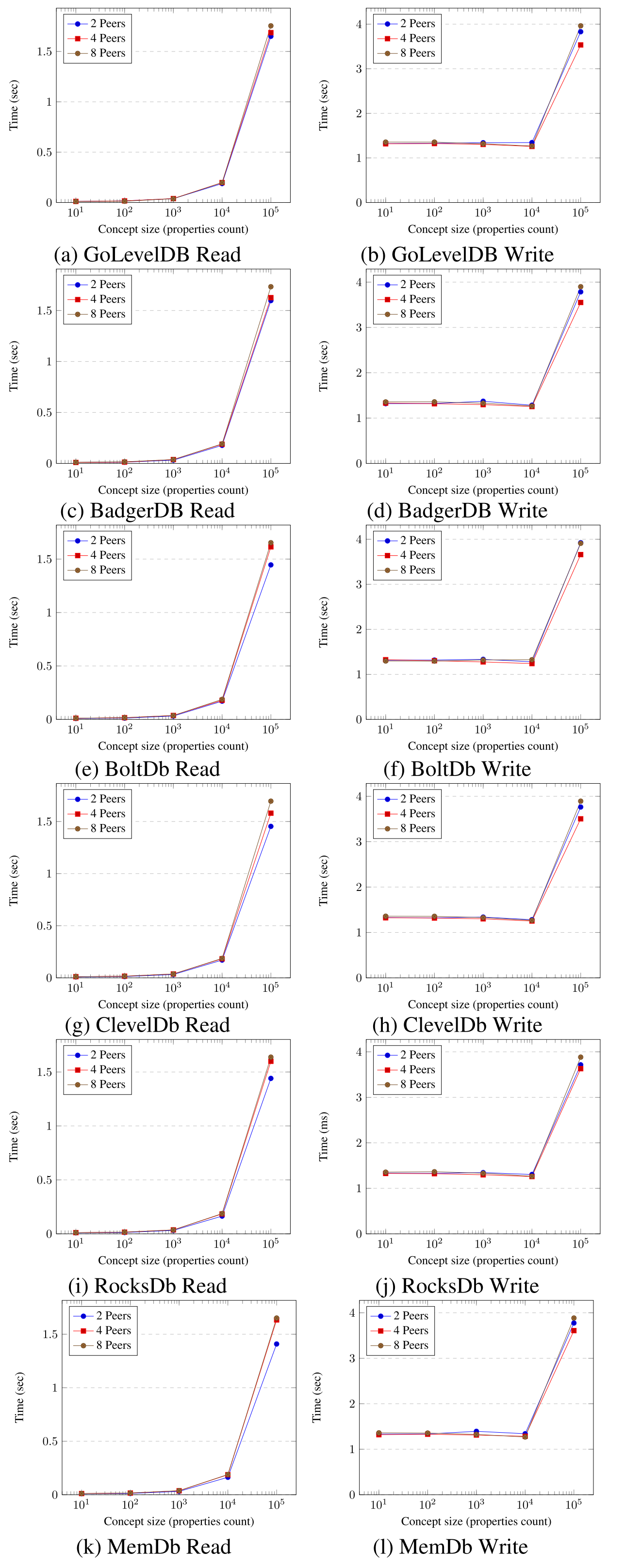


**FIGURE 8.** Tendermint Datastore Variation Results.

ing transactions into the transaction pool or allocating extra memory.

**TABLE 3.** Hyperledger Fabric Simulation Parameters.

| Parameter | Values |
|---|---|
| Topology (peers count) | 2 \| 4 \| 8 |
| Concept size (properties count) | $10^1$ \| $10^2$ \| $10^3$ \| $10^4$ \| $10^5$ |
| Operation | Read \| Write |
| Block size | Half \| Double |
| Transaction pool | Half \| Double |
| Datastore engine | GolevelDB \| CouchDB |
| Warmup iterations | 5 (each running for 10s) |
| Execution iterations | 10 (each running for 30s) |

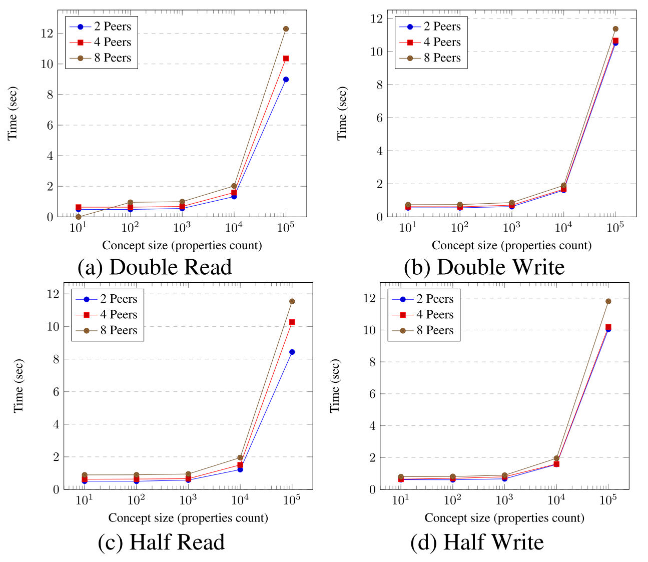


**FIGURE 9.** Hyperledger Transaction Pool Variation Results.

*Observation 13:* As can be observed in Fig. 9b, Fig. 9d, Fig. 10b, and Fig. 10d, the time required to write concepts shows a similar pattern for all configurations in the case of small-size concepts. For big-size concept configurations, the effect of varying the transaction pool size is not visible because the transaction pool limit is never reached. However, when doubling the block size, we experience an increase in latency for big-size concepts on the eight-peer network. This effect is explained by the fact that the Hyperledger consensus mechanism is not well-optimized for big-size blocks, especially for extensive networks.

*Observation 14:* The measurements depicted in Fig. 11 suggest that the network size has a noticeable impact on both read and write operations. The throughput decreases as the network size increases.

*Observation 15:* The results in Fig. 11 show that the type of database has a noticeable impact on both read and write operations. *GoLevelDB*, has a faster response time, compared to *CouchDB* for both read and write operations. The performance difference could be because *GolevelDb* is integrated locally, while *CouchDB* is available through a different process. However, as the number of peers increases, the benefit disappears, and *CouchDB* seems better suited when writing on a bigger network that manages concepts with many properties.

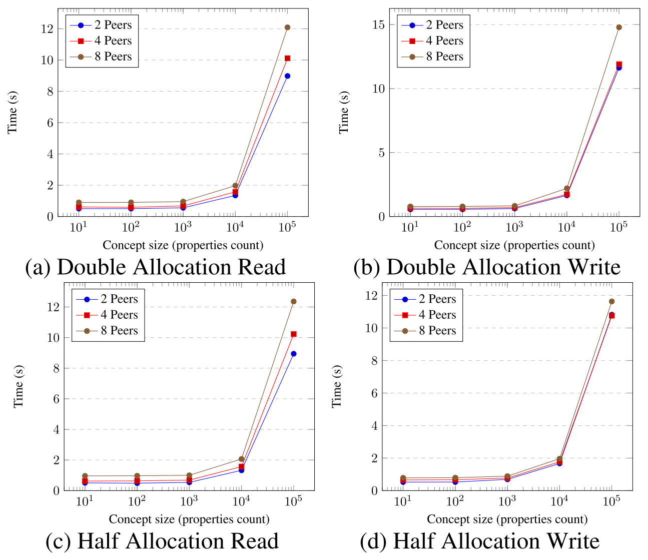


FIGURE 10. Hyperledger Block Size Variation Results.

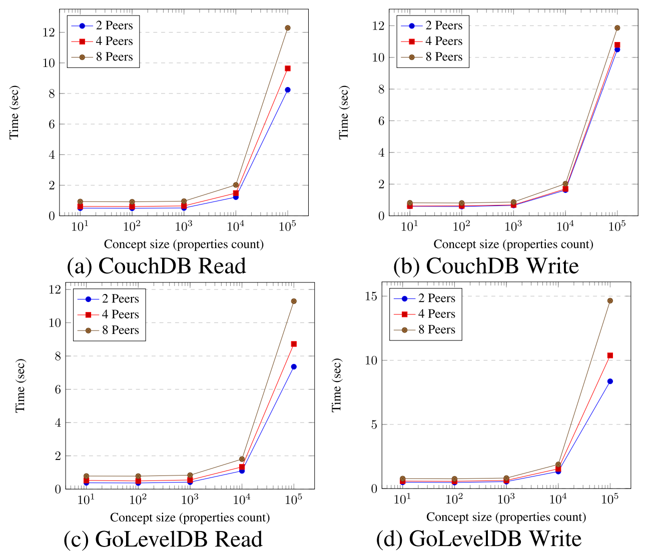


FIGURE 11. Hyperledger Datastore Variation Results.

## D. AVERAGED RESULTS

In this section, we present the results obtained by averaging each configuration's time by eliminating the extreme use cases for concepts with ten and $10^5$ properties. The aim is to give the reader a better overview of the high-level behavior or the proposed model related to each architecture and configuration parameter.

*Observation 16:* Fig. 12 shows the results for the IPFS configuration. On average, persisting the proposed model information is twice as expensive as reading it. However, compared to Fig. 14 and Fig. 13, the IPFS setup provides the best throughput for both reading and writing.

*Observation 17:* In the case of Tendermint Cosmos, as depicted in Fig. 13, the proposed model behaves predictably and stably for both reads and write use cases in all configurations. As expected, once persisted on the network, retrieving the information is, on average, three times faster than persisting.

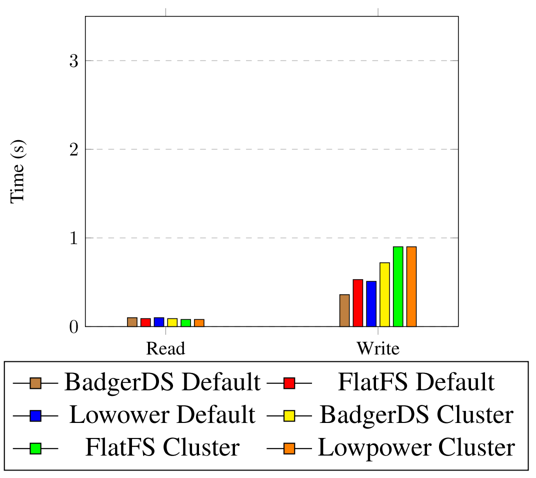


FIGURE 12. IPFS Averaged Results.

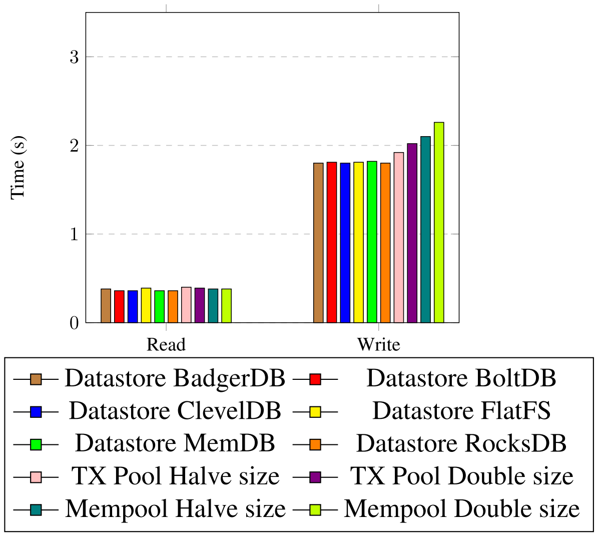


FIGURE 13. Tendermint Averaged Results.

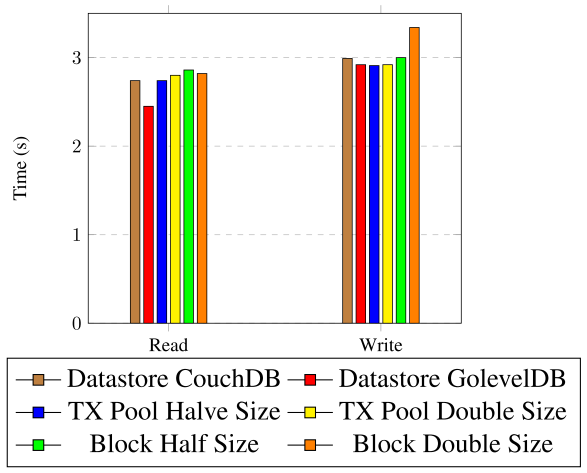


FIGURE 14. Hyperledger Averaged Results.

*Observation 18:* Fig. 14 summarises the results for the Hyperledger Fabric configuration parameters. The proposed model is stable and behaves predictably for all the measured configurations, with a slight improvement for the Couch DB storage configuration. Also, the measured latencies for reading and writing concepts are in the same range, suggesting that a component responsible for transporting the information in Hyperledger heavily impacts the overall system performance (since there is no difference between reading and writing times).

## VI. CONCLUSION

Elaborating practical solutions where AI modules can interoperate and connect dynamically requires an efficient mechanism for resolving the concept model describing each module's outer boundaries. From an AI module's perspective, retrieving concept models is the immediate priority, so selecting a configuration that reduces the read latencies is mandatory. Low latency in resolving the concepts will ensure a prompt response when configuring and deploying multiple agents for the composition case relevant to a complex setup.

For the edge use cases requiring the usage of large concepts containing more than $10^5$ properties, IPFS provides sub-second latency, which is sufficiently fast for the use cases where AI Agents need to perform intensive and time-critical tasks. In contrast, Hyperledger Fabric requires more than ten seconds for both read and write operations, making it unsuitable for similar use cases.

The proposed ontology model provides a stable and consistent behavior on all exported profiles, topologies and configurations for the average size concepts. Selecting the suitable replication medium is a decision based on the multi-organizational structure, policies and constraints on managing the data:

- IPFS has high read and write speeds, but lacks a consensus algorithm [41], thus it is the most suitable option when few actors are involved in defining and maintaining the ontology model or when decisions regarding the model are simple.
- Cosmos Tendermint - a public blockchain [42] - is the best option when decisions regarding the ontology model are more complex, and consensus has to be reached among multiple actors while maintaining unrestricted access to the content.
- Hyperledger Fabric, being a permissioned blockchain [43] is most suitable when a small set of private organizations has to take, possibly, complex decisions about the ontology model and where access to it must be restricted.

To conclude, this paper contributes by presenting a flexible AI-to-AI communication paradigm based on ontology concepts stored using decentralized technologies. Three different storage technologies were evaluated on read and write speeds, and a design space exploration was performed for each. The study explored topology parameters, infrastructure variables, concept sizes and database engines. Furthermore, in the case of blockchain technologies (Tendermint Cosmos and Hyperledger Fabric), additional parameters were varied to study the impact of transaction pool and block sizes. While these results are of great interest in the context of an AI marketplace, distributed data storage has attracted much broader interest [44], [45], [46], [47], [48], [49], [50], [51], [52], [53], [54], [55], [56], [57], [58], [59]. Reading and writing data to decentralized storage is relevant to, for example, the semantic web [44], supply chains [46], and knowledge graphs [57]. To our knowledge, this is the most comprehensive evaluation of multiple storage technologies with results relevant beyond AI-to-AI communication.

Experimental results, obtained under controlled conditions, can exhibit bias and discrepancies when applied to real-world scenarios, where uncontrolled variables come into play. Real-world systems frequently encounter downtime, delays, and errors in underlying protocols, which are generally not accounted for in experimental settings. These unforeseen factors can impose additional penalties, leading to a difference in actual performance compared to experimental data.

The dynamic interconnection and deployment of AI modules will be evaluated quantitatively and qualitatively in future research.

## REFERENCES

[1] S. Linnainmaa, "Taylor expansion of the accumulated rounding error," *BIT*, vol. 16, no. 2, pp. 146–160, Jun. 1976, doi: 10.1007/bf01931367.

[2] P. J. Werbos, "Applications of advances in nonlinear sensitivity analysis," in *System Modeling and Optimization*. New York, NY, USA: Springer, 1982, pp. 762–770.

[3] S. Amari, "A theory of adaptive pattern classifiers," *IEEE Trans. Electron. Comput.*, vol. EC-16, no. 3, pp. 299–307, Jun. 1967.

[4] M. Minsky and S. Papert, *Perceptrons; an Introduction to Computational Geometry*. Cambridge, MA, USA: MIT Press, 1969. [Online]. Available: https://books.google.ca/books?id=Ow1OAQAAIAAJ

[5] D. E. Rumelhart and J. L. McClelland, *Parallel Distributed Processing: Explorations in the Microstructure of Cognition: Foundations*, vol. 1. Cambridge, MA, USA: MIT Press, 1986.

[6] J. Corchado and B. Lees, "An overview of intelligent frameworks," in *Colloquium On Intelligent Systems*. London, U.K.: IEE, 1998.

[7] Y. Bengio, Y. LeCun, and G. E. Hinton, "Deep learning for AI," *Commun. ACM*, vol. 64, pp. 58–65, Jun. 2021.

[8] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. Cambridge, MA, USA: MIT Press, 2016.

[9] A. Zhang, Z. Lipton, M. Li, and A. Smola, "Dive into deep learning-release 0.16. 0," Tech. Rep., 2021.

[10] S. Umbrello, "Ai winter," in *Encyclopedia of Artificial Intelligence: The Past, Present, and Future of AI*, M. Klein and P. Frana, Eds. Santa Barbara, CAS, USA: ABC-CLIO, 2021, pp. 7–8.

[11] B. C. Smith, *The Promise of Artificial Intelligence: Reckoning and Judgment*. Cambridge, MA, USA: MIT Press, 2019.

[12] A. Faisal, T. Yigitcanlar, M. Kamruzzaman, and G. Currie, "Understanding autonomous vehicles," *J. Transp. Land Use*, vol. 12, no. 1, pp. 45–72, Jan. 2019.

[13] B. B. Elallid, N. Benamar, A. S. Hafid, T. Rachidi, and N. Mrani, "A comprehensive survey on the application of deep and reinforcement learning approaches in autonomous driving," *J. King Saud Univ. Comput. Inf. Sci.*, vol. 34, no. 9, pp. 7366–7390, Oct. 2022.

[14] J. P. Zmud and I. N. Sener, "Towards an understanding of the travel behavior impact of autonomous vehicles," *Transp. Res. Proc.*, vol. 25, pp. 2500–2519, Jan. 2017.

[15] F. Nasirian, M. Ahmadian, and O.-K. D. Lee, "Ai-based voice assistant systems: Evaluating from the interaction and trust perspectives," Tech. Rep., 2017.
[16] R. Benotsmane, L. Dudás, and G. Kovács, "Survey on artificial intelligence algorithms used in industrial robotics," *Multidiszciplináris Tudományok*, vol. 10, no. 4, pp. 194–205, 2020.
[17] S. F. Smith, "Smart infrastructure for future urban mobility," *AI Mag.*, vol. 41, no. 1, pp. 5–18, Mar. 2020.
[18] W. Zhang, N. Aung, S. Dhelim, and Y. Ai, "DIFTOS: A distributed infrastructure-free traffic optimization system based on vehicular ad hoc networks for urban environments," *Sensors*, vol. 18, no. 8, p. 2567, Aug. 2018.
[19] R. Schwartz, J. Dodge, N. Smith, and O. Etzioni, "Green AI," *Commun. ACM*, vol. 63, no. 12, pp. 54–63, Dec. 2020.
[20] E. M. Bender, T. Gebru, A. McMillan-Major, and S. Shmitchell, "On the dangers of stochastic parrots: Can language models be too big," in *Proc. ACM Conf. Fairness, Accountability, Transparency*, Mar. 2021, pp. 610–623.
[21] T. Babina, A. Fedyk, A. X. He, and J. Hodson, "Artificial intelligence, firm growth, and industry concentration," in *Firm Growth and Market Concentration*, vol. 22, 2020.
[22] J. Kaplan, S. McCandlish, T. Henighan, T. B. Brown, B. Chess, R. Child, S. Gray, A. Radford, J. Wu, and D. Amodei, "Scaling laws for neural language models," 2020, *arXiv:2001.08361*.
[23] S. Biderman, H. Schoelkopf, Q. Anthony, H. Bradley, K. O'Brien, E. Hallahan, M. Aflah Khan, S. Purohit, U. Sai Prashanth, E. Raff, A. Skowron, L. Sutawika, and O. van der Wal, "Pythia: A suite for analyzing large language models across training and scaling," 2023, *arXiv:2304.01373*.
[24] L. Saukko, K. Aaltonen, and H. Haapasalo, "Inter-organizational collaboration challenges and preconditions in industrial engineering projects," *Int. J. Manag. Projects Bus.*, vol. 13, no. 5, pp. 999–1023, May 2020.
[25] D. Chen and G. Doumeingts, "European initiatives to develop interoperability of enterprise applications-basic concepts, framework and roadmap," *Annu. Rev. Control*, vol. 27, no. 2, pp. 153–162, 2003.
[26] G. Zacharewicz, S. Diallo, Y. Ducq, C. Agostinho, R. Jardim-Goncalves, H. Bazoun, Z. Wang, and G. Doumeingts, "Model-based approaches for interoperability of next generation enterprise information systems: State of the art and future challenges," *Inf. Syst. e-Bus. Manage.*, vol. 15, no. 2, pp. 229–256, May 2017.
[27] K. Fischer, "Advanced technologies for interoperability of heterogeneous enterprise networks and their applications," in *European Coordination Action for Agent-Based Computing*, vol. 34, 2005.
[28] *ECLASS Technical-Specification*. Accessed: Jun. 7, 2023. [Online]. Available: https://eclass.eu/support/technical-specification/data-model/iso-13584-32-ontoml
[29] V. R. S. Kumar, A. Khamis, S. Fiorini, J. L. Carbonera, A. O. Alarcos, M. Habib, P. Goncalves, H. Li, and J. I. Olszewska, "Ontologies for Industry 4.0," *Knowl. Eng. Rev.*, vol. 34, p. e17, Nov. 2019.
[30] N. Shilov, A. Smirnov, and F. Ansari, "Ontologies in smart manufacturing: Approaches and research framework," in *Proc. 26th Conf. Open Innov. Assoc. (FRUCT)*, Apr. 2020, pp. 408–414.
[31] A. Tara, A. Butean, C. Zamfirescu, and R. Learney, "An ontology model for interoperability and multi-organization data exchange," in *Proc. Comput. Sci. Line Conf.* Cham, Switzerland: Springer, 2020, pp. 284–296.
[32] A. Tara, N. Taban, C. Vasiu, and C. Zamfirescu, "A decentralized ontology versioning model designed for inter-operability and multi-organizational data exchange," in *Proc. Comput. Sci. Line Conf.* Cham, Switzerland: Springer, 2021, pp. 617–628.
[33] G. Wang, T. N. Wong, and X. H. Wang, "A negotiation protocol to support agent argumentation and ontology interoperability in MAS-based virtual enterprises," in *Proc. 7th Int. Conf. Inf. Technol., New Generat.*, Apr. 2010, pp. 448–453.
[34] C. Persson and E. O. Wallin, "Engineering and business implications of ontologies—A proposal for a minimum viable ontology," in *Proc. 13th IEEE Conf. Autom. Sci. Eng. (CASE)*, Aug. 2017, pp. 864–869.
[35] J. I. Olszewska and I. Allison, "Odyssey: Software development life cycle ontology," Tech. Rep., 2018.
[36] *Auto-GPT Repository*. Accessed: Jun. 7, 2023. [Online]. Available: https://github.com/Significant-Gravitas/Auto-GPT
[37] Y. Shen, K. Song, X. Tan, D. Li, W. Lu, and Y. Zhuang, "HuggingGPT: Solving AI tasks with ChatGPT and its friends in hugging face," 2023, *arXiv:2303.17580*.
[38] R. Hao, L. Hu, W. Qi, Q. Wu, Y. Zhang, and L. Nie, "ChatLLM network: More brains, more intelligence," 2023, *arXiv:2304.12998*.
[39] B. Lashkari and P. Musilek, "A comprehensive review of blockchain consensus mechanisms," *IEEE Access*, vol. 9, pp. 43620–43652, 2021.
[40] J. Nijsse and A. Litchfield, "A taxonomy of blockchain consensus methods," *Cryptography*, vol. 4, no. 4, p. 32, Nov. 2020.
[41] J. Benet, "IPFS–content addressed, versioned, P2P file system," 2014, *arXiv:1407.3561*.
[42] J. Kwon and E. Buchman, "Cosmos whitepaper," *A Netw. Distrib. Ledgers*, vol. 27, pp. 1–32, May 2019.
[43] E. Androulaki, A. Barger, V. Bortnikov, C. Cachin, K. Christidis, A. De Caro, D. Enyeart, C. Ferris, G. Laventman, and Y. Manevich, "HyperLedger fabric: A distributed operating system for permissioned blockchains," in *Proc. 13th EuroSys Conf.*, 2018, pp. 1–15.
[44] M. English, S. Auer, and J. Domingue, "Block chain technologies & the semantic web: A framework for symbiotic development," in *Proc. Comput. Sci. Conf. Univ. Bonn Students*, J. Lehmann, H. Thakkar, L. Halilaj, and R. Asmat, Eds., 2016, pp. 47–61.
[45] J. D. Kruijff and H. Weigand, "Understanding the blockchain using enterprise ontology," in *Proc. Int. Conf. Adv. Inf. Syst. Eng.* Cham, Switzerland: Springer, 2017, pp. 29–43.
[46] M. Ruta, F. Scioscia, S. Ieva, G. Capurso, and E. Di Sciascio, "Supply chain object discovery with semantic-enhanced blockchain," in *Proc. 15th ACM Conf. Embedded Netw. Sensor Syst.*, Nov. 2017, pp. 1–2.
[47] J. J. Sikorski, J. Haughton, and M. Kraft, "Blockchain technology in the chemical industry: Machine-to-machine electricity market," *Appl. Energy*, vol. 195, pp. 234–246, Jun. 2017.
[48] L. D. Ibáñez, H. Fryer, and E. P. B. Simperl, "Attaching semantic metadata to cryptocurrency transactions," in *Proc. DeSemWeb@ISWC*, 2017.
[49] M. R. Hoffman, L.-D. Ibáñez, H. Fryer, and E. Simperl, "Smart papers: Dynamic publications on the blockchain," in *Proc. Eur. Semantic Web Conf.* Cham, Switzerland: Springer, 2018, pp. 304–318.
[50] H. Zhu, Y. Wang, X. Hei, W. Ji, and L. Zhang, "A blockchain-based decentralized cloud resource scheduling architecture," in *Proc. Int. Conf. Netw. Netw. Appl. (NaNA)*, Oct. 2018, pp. 324–329.
[51] H. M. Kim, M. Laskowski, and N. Nan, "A first step in the co-evolution of blockchain and ontologies: Towards engineering an ontology of governance at the blockchain protocol level," 2018, *arXiv:1801.02027*.
[52] M.-Á. Sicilia and A. Visvizi, "Blockchain and OECD data repositories: Opportunities and policymaking implications," *Library Hi Tech*, vol. 37, no. 1, pp. 30–42, 2018.
[53] D. Graux, G. Sejdiu, H. Jabeen, J. Lehmann, D. Sui, D. Muhs, and J. Pfeffer, "Profiting from kitties on Ethereum: Leveraging blockchain RDF data with SANSA," in *Proc. SEMANTiCS Conf.*, 2018, pp. 1–4.
[54] A. Le-Tuan, D. Hingu, M. Hauswirth, and D. Le-Phuoc, "Incorporating blockchain into RDF store at the lightweight edge devices," in *Proc. Int. Conf. Semantic Syst.* Cham, Switzerland: Springer, 2019, pp. 369–375.
[55] J. Cano-Benito, A. Cimmino, and R. García-Castro, "Towards blockchain and semantic web," in *Proc. Int. Conf. Bus. Inf. Syst.* Cham, Switzerland: Springer, 2019, pp. 220–231.
[56] J. Cano-Benito, A. Cimmino, and R. García-Castro, "Benchmarking the efficiency of RDF-based access for blockchain environments," in *Proc. SEKE*, 2020, pp. 554–559.
[57] Y. Wang, X. Yin, H. Zhu, and X. Hei, "A blockchain based distributed storage system for knowledge graph security," in *Artificial Intelligence and Security*, X. Sun, J. Wang, and E. Bertino, Eds. Cham, Switzerland: Springer, 2020, pp. 318–327.
[58] A. Tara, N. Taban, and H. Turesson, "Performance analysis of an ontology model enabling interoperability of artificial intelligence agents," in *Proc. Comput. Sci. Line Conf.* Cham, Switzerland: Springer, 2022, pp. 395–406.
[59] A. Djeddai and R. Khemaissia, "Keeping the privacy and the security of the knowledge graph completion using blockchain technology," in *Proc. 4th Int. Conf. Pattern Anal. Intell. Syst. (PAIS)*, Oct. 2022, pp. 1–6.
[60] Y. Faqir-Rhazoui, M.-J. Ariza-Garzón, J. Arroyo, and S. Hassan, "Effect of the gas price surges on user activity in the DAOs of the Ethereum blockchain," in *Proc. Extended Abstr. CHI Conf. Human Factors Comput. Syst.* New York, NY, USA: Association for Computing Machinery, May 2021, pp. 1–7, doi: 10.1145/3411763.3451755.
[61] S. Leonardos, B. Monnot, D. Reijsbergen, E. Skoulakis, and G. Piliouras, "Dynamical analysis of the EIP-1559 Ethereum fee market," in *Proc. 3rd ACM Conf. Adv. Financial Technol.* New York, NY, USA: Association for Computing Machinery, Sep. 2021, pp. 114–126, doi: 10.1145/3479722.3480993.

[62] J. E. de Azevedo Sousa, V. Oliveira, J. Valadares, G. Dias Gonçalves, S. M. Villela, H. S. Bernardino, and A. Borges Vieira, "An analysis of the fees and pending time correlation in Ethereum," *Int. J. Netw. Manage.*, vol. 31, no. 3, May 2021, Art. no. e2113.

[63] D. Trautwein, A. Raman, G. Tyson, I. Castro, W. Scott, M. Schubotz, B. Gipp, and Y. Psaras, "Design and evaluation of IPFS: A storage layer for the decentralized web," in *Proc. ACM SIGCOMM Conf.* New York, NY, USA: Association for Computing Machinery, Aug. 2022, pp. 739–752, doi: 10.1145/3544216.3544232.

[64] J. Shen, Y. Li, Y. Zhou, and X. Wang, "Understanding I/O performance of IPFS storage: A Client's perspective," in *Proc. IEEE/ACM 27th Int. Symp. Quality Service (IWQoS)*, Jun. 2019, pp. 1–10.

[65] E. Buchman, R. Guerraoui, J. Komatovic, Z. Milosevic, D.-A. Seredinschi, and J. Widder, "Revisiting tendermint: Design tradeoffs, accountability, and practical use," in *Proc. 52nd Annu. IEEE/IFIP Int. Conf. Dependable Syst. Netw. Supplemental Volume (DSN-S)*, Jun. 2022, pp. 11–14.

[66] D. Cason, E. Fynn, N. Milosevic, Z. Milosevic, E. Buchman, and F. Pedone, "The design, architecture and performance of the tendermint blockchain network," in *Proc. 40th Int. Symp. Reliable Distrib. Syst. (SRDS)*, Sep. 2021, pp. 23–33.

[67] M. Castro and B. Liskov, "Practical Byzantine fault tolerance and proactive recovery," *ACM Trans. Comput. Syst.*, vol. 20, no. 4, pp. 398–461, Nov. 2002, doi: 10.1145/571637.571640.

[68] S. Shalaby, A. A. Abdellatif, A. Al-Ali, A. Mohamed, A. Erbad, and M. Guizani, "Performance evaluation of hyperledger fabric," in *Proc. IEEE Int. Conf. Informat., IoT, Enabling Technol. (ICIoT)*, Feb. 2020, pp. 608–613.

[69] M. Kuzlu, M. Pipattanasomporn, L. Gurses, and S. Rahman, "Performance analysis of a hyperledger fabric blockchain framework: Throughput, latency and scalability," in *Proc. IEEE Int. Conf. Blockchain (Blockchain)*, Jul. 2019, pp. 536–540.

[70] M. Q. Nguyen, D. Loghin, and T. T. A. Dinh, "Understanding the scalability of hyperledger fabric," Tech. Rep., 2021.

[71] D. Costa, C.-P. Bezemer, P. Leitner, and A. Andrzejak, "What's wrong with my benchmark results? Studying bad practices in JMH benchmarks," *IEEE Trans. Softw. Eng.*, vol. 47, no. 7, pp. 1452–1467, Jul. 2021.

[72] L. Lu, T. S. Pillai, H. Gopalakrishnan, A. C. Arpaci-Dusseau, and R. H. Arpaci-Dusseau, "WiscKey: Separating keys from values in SSD-conscious storage," *ACM Trans. Storage*, vol. 13, no. 1, pp. 1–28, Feb. 2017.

**ANDREI TARA** was born in Sibiu, Romania, in 1987. He received the B.S. and M.S. degrees in computer science from the Faculty of Engineering, Lucian Blaga University of Sibiu, Sibiu, in 2010 and 2012, respectively, where he is currently pursuing the Ph.D. degree in computer science, with a focus on decentralized systems with practical use-case in creating interoperable artificial intelligence semantic models on top of blockchain.

From 2012 to 2018, he was involved in researching and developing data master repository systems based on ontologies and semantic models for companies, such as Siemens, ALV, NXP Semiconductors, Zeiss, and Volkswagen.

**HJALMAR K. TURESSON** was born in Helsingborg, Sweden. He received the B.Sc. degree in biology from Lund University, Lund, Sweden, the M.Sc. degree in behavioral and neural sciences from the University of Tübingen, Germany, and the Ph.D. degree in neuroscience from Princeton University, Princeton, NJ, USA. He is currently an Adjunct Professor with the Schulich School of Business, York University, Toronto, ON, Canada.

**NICOLAE NATEA** was born in Sibiu, Romania, in 1987. He received the B.S. and M.S. degrees in computer science from the Faculty of Engineering, Lucian Blaga University of Sibiu, Sibiu, in 2010 and 2012, respectively. Since 2010, he has been an Automotive, Telecom and IoT System Engineer. He is currently a part of the team with Openfabric Network SRL, involving on a new platform that aims to simplify the usage of artificial intelligence.

**HENRY M. KIM** (Member, IEEE) received the Ph.D. degree in industrial engineering from the University of Toronto, Toronto, ON, Canada. He is currently an Associate Professor with the Schulich School of Business, York University, Toronto, and the Director of the blockchain.lab, York University. He has written extensively on blockchain, authored more than 30 blockchain-related articles in venues, such as IEEE Transactions on Engineering Management and the IEEE Computer Society. He served as a senior research advisor for several blockchain startups and co-organized the two IEEE International Conferences on Blockchain and Cryptocurrencies (ICBC).