Agora: Bringing Together Datasets, Algorithms, Models and More in a Unified Ecosystem [Vision]

Jonas Traub Zoi Kaoudi Jorge-Arnulfo Quiané-Ruiz Volker Markl

Technische Universität Berlin DFKI GmbH

ABSTRACT

Data science and artificial intelligence are driven by a plethora of diverse data-related assets, including datasets, data streams, algorithms, processing software, compute resources, and domain knowledge. As providing all these assets requires a huge investment, data science and artificial intelligence technologies are currently dominated by a small number of providers who can afford these investments. This leads to lock-in effects and hinders features that require a flexible exchange of assets among users. In this paper, we introduce Agora, our vision towards a unified ecosystem that brings together data, algorithms, models, and computational resources and provides them to a broad audience. Agora (i) treats assets as first-class citizens and leverages a fine-grained exchange of assets, (ii) allows for combining assets to novel applications, and (iii) flexibly executes such applications on available resources. As a result, it enables easy creation and composition of data science pipelines as well as their scalable execution. In contrast to existing data management systems, Agora operates in a heavily decentralized and dynamic environment: Data, algorithms, and even compute resources are dynamically created, modified, and removed by different stakeholders. Agora presents novel research directions for the data management community as a whole: It requires to combine our traditional expertise in scalable data processing and management with infrastructure provisioning as well as economic and application aspects of data, algorithms, and infrastructure.

1. INTRODUCTION

As data and data science technologies have become production factors, it is clear that they must be accessible by *everyone* [1]. Academia and industry have made progress towards the goal of providing access to data (e.g., [11, 20]), AI algorithms (e.g., [2, 4, 6, 19]), or computational resources [13]. However, the users still require significant expertise to combine all these *data-related assets* (assets, for short) from different marketplaces and cloud providers. For instance, a social scientist, who has no expertise in data science techniques and does not own any data, can hardly validate her assumptions about a social phenomenon, even if the required data and technology are theoretically available.

We envision *Agora*, an ecosystem that enables and eases the creation and composition of data science pipelines as well as their scalable execution. Agora provides *unified* access to all types of assets (e.g., data, algorithms, and compute resources) and treats them as *first-class citizens*: The social scientist in our example would not only find all relevant assets, but also executable compositions of them. This combination of abstraction and accessibility makes Agora attractive even for non-expert users. Agora brings together asset providers and consumers. It allows providers to offer any type of assets to a broader audience. For consumers, Agora provides access not only to data sources but to the entire data value chain.

We envision this ecosystem playing a dual role: (i) It is composed of a set of marketplaces where providers and consumers can exchange assets, and (ii) it provides the means to users to run their tasks (composition of assets) in Agora instead of using their own computing infrastructure. The key aspect of Agora is the fine-grained exchange of *any* asset. Each type of assets corresponds to a specialization of a provider, leading to different user roles. Agora hides the complexity of each role. For example, (i) a researcher can subscribe to an event stream without knowing details about the infrastructure that captures those events; (ii) a company can acquire a classification pipeline without understanding the details of all involved algorithms; (iii) researchers and companies can book a stream processing cluster with uptime guarantees without having knowledge on cluster operations; and (iv) system operators can focus on cluster monitoring and maintenance without knowing details about the running tasks.

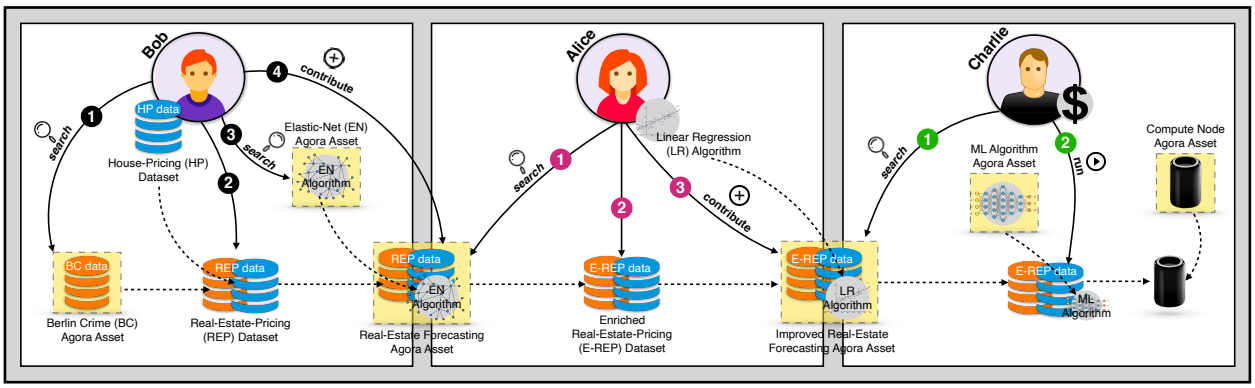

Figure 1: Motivating examples: Bob, Alice, and Charlie use Agora to discover assets, improve them, and contribute them back to the ecosystem. Agora also provides the infrastructure to run asset-based pipelines.

Related work. Most advanced data science pipelines require huge amounts of data, cutting edge data science innovations, and powerful computational infrastructure. Agora aims to connect providers and users of these key assets in an open ecosystem. In contrast, recent works tackle only parts of the solution provided by Agora: For example, OpenAI [21] primarily builds custom solutions and shares them via free software for training, benchmarking, and experimenting. Ocean Protocol [20] has similar goals with Agora but it focuses only on developing a decentralized protocol and network for data sharing. Datum [15] focuses on the privatization and secure storage of data sharing and proposes a network based on blockchain technology. Enigma [13] offers a protocol for computations on encrypted data by enabling computational resources to be shared securely in a decentralized manner. Nebula [18] forms a cloud of edge computers to perform distributed data-intensive computing. Data lake solutions, such as [16, 8], focus only on how to index and find datasets using metadata. In the space of machine learning, ML Bazaar [25] proposes a unified ML API to ease the development and sharing of ML algorithms. Agora goes beyond a simple abstraction to a holistic solution. There are also initiatives in providing marketplaces for sharing data [10, 12] and data science/AI solutions [4, 14, 19, 6, 2] as well as for storing data in a flexible manner [17]. However, their approach is single-facet, which makes it hard for users to combine them. The industry has also brought storage, computational, and cloud resources at the reach of the masses, such as Amazon EC2 and Microsoft Azure. Nevertheless, such cloud-based solutions result in lock-in effects: Users must stick to one provider for the entire pipeline of their solutions. We envision an open ecosystem where one can freely combine resources from different marketplaces. All above efforts go in the right direction for building a data ecosystem, however, it is still hard to combine them for devising end-to-end new solutions. Our work envisions a single data ecosystem where data, data science technologies, and storage/compute resources can seamlessly be combined to extract data insights.

Requirements. To see Agora become a reality, the following requirements should be met: (i) asset sharing and discovering – users should be able to easily provide or consume assets; (ii) asset privacy and security – users must be able to set privacy and security constraints to their assets; (iii) asset interoperability – users should be able to easily combine different (types of) assets; (iv) asset equivalence – users should be able to achieve their desired goals without being concerned about the specifics of the underlying algorithms; and (v) hardware independence – users should be able to run their assets on heterogeneous hardware seamlessly. We see Agora as an umbrella system, uniting all pieces of data management research. We believe that the database community should strive to realize this vision.

2. MOTIVATION

Imagine Bob, a freelance data scientist, who wants to create a machine learning (ML) model for real-estate price forecasting in Berlin (see Figure 1). His dataset is missing the criminality rate of each area, which he knows also affects the prices. He, thus, goes to Agora to find data about the crime rates in Berlin ❶. He finds the data, augments his initial dataset with this feature ❷, and builds an ML model using the elastic-net algorithm ❸. He then decides to provide his composed asset in Agora ❹. Bob's asset consists of the 'real-estate-pricing' dataset for Berlin and the elastic-net algorithm to estimate a potential price of apartments.

Alice, another data scientist, finds Bob's asset in

Agora ❶ and decides to improve it ❷. She enriches the original 'real-estate-pricing' dataset with several feature engineering techniques, adds the 'linear-regression' algorithm for prediction, and contributes it back to Agora to gain some revenue ❸.

Charlie, a consumer who is looking for a real-estate pricing predictor, queries Agora for available assets on price forecasting that yield the average error rate below 5,000 euros ❶. As he does not have the infrastructure to run assets, he decides to use Agora to execute his discovered assets (e.g., train the found ML pipelines) ❷. Although he wants to complete the training as fast as possible, his budget is limited. To overcome his budget limitation, Agora recommends to replace the linear regression algorithm by a logically equivalent neural network implementation having lower license cost. Next, Agora decides to run the resulting asset on an execution node registered within Agora. The latter can be a machine of a cloud provider registered within Agora or an idle cluster of an individual user. This enables users to gain some revenue by offering personal compute resources in competitive prices.

Allowing asset exchange in Agora leads to the following main benefits:

(1) Secondary use of existing assets. Users can reuse any (composed) asset (e.g., data and algorithms) offered in Agora. In most cases, companies own a plethora of highly valuable assets. However, as these assets are fragmented across companies, their economical potential remains unused as secondary asset usage is extremely rare. Similarly individuals can offer any asset, including compute resources, with the goal of gaining revenue. A fine-grained asset sharing would allow for combining existing resources to derive new insights and services.

(2) Leveraging specializations. Agora creates an ecosystem of highly specialized providers who provide assets of a very high quality. Such an ecosystem is comparable with the automotive industry where many companies specialize in certain parts (e.g., brakes, tires, or lights), which get combined to one high-quality car. This enables small and medium-sized companies to offer assets that they would not be able to bring in the market otherwise.

(3) Hiding complexity. Agora hides the complexity and intricacies of assets from the consumers. It is aware of logical equivalence of assets, i.e., assets that yield the same results (e.g., a nested loops join is equivalent with a hash join for equi-joins). Implementations of logically equivalent assets can have very different properties: They may use different programming languages (e.g., C++ and Java), be tailored to different systems (e.g., Flink and Spark),

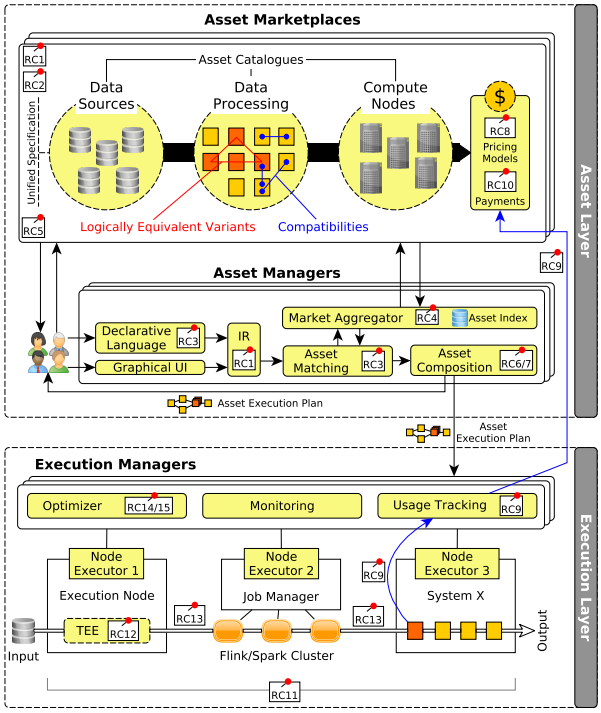

Figure 2: An overview of the architecture of Agora with 15 selected *Research Challenges (RCs)*.

be optimized for specific hardware (e.g., CPU and GPU), and run in a parallelized, distributed, or sequential setting. In addition, each provider can define different pricing for her implementation. To optimize execution, Agora chooses the best combination out of the available implementations based on the requirements of the incoming task.

3. AGORA ARCHITECTURE

Agora builds around assets which we define as *any data-related unit of production that allows users to exploit the value of data.* We identify six major categories of assets: *data sources*, *algorithms*, *pipelines*, *storage and compute resources*, *systems*, and *applications*, discussed in more detai in [26].

Agora consists of two layers (Figure 2): the *Asset Layer* and the *Execution Layer*. In this way we decouple the execution infrastructure from the platform for asset composition and discovery as they come with different requirements. Still, a major strength of Agora is its seamless connection between these two layers. It goes beyond stand-alone marketplaces, stand-alone execution engines, and cloud services with the goal of facilitating the use of DS tools for a broader group of users.

The **asset layer** constitutes an "intelligent" ecosystem of multiple *asset marketplaces* and en-

ables not only offering and finding assets but also composing them in a smart way via *asset managers*. Recall our motivation example described in Section 2. Bob, who is searching for a dataset, has the choice of going directly to his favorite marketplace or to an asset manager to find his desired dataset. In the former case, he either browses the marketplace or uses keywords to search within it. In the latter case, he specifies his request in a declarative manner and the asset manager responds by potentially accessing multiple marketplaces.

The **execution layer** optimizes and runs asset execution plans via *execution managers* and *node executors*. For instance, Charlie, in our running example, finds his pipeline via an asset manager and decides to execute it in Agora. For this reason, the asset manager translates the pipeline into an execution plan together with its equivalent assets, which are logically equivalent variants satisfying the same request. Logically equivalent variants can be different physical implementations of the same logical operator, alternative compute nodes with similar properties, or alternative data sources, such as weather data from different providers. Next, the asset manager passes the execution plan to an execution manager, which optimizes the plan and finds the best possible asset options that respect Charlie's budget. The execution manager accesses processing nodes through a *node executor*, which is a standardized component to interface arbitrary execution environments with execution managers. For example, NodeExecutor 1 in Figure 2 provides access to a Trusted Execution Environment (TEE), such as an Intel SGX Enclave [9], and NodeExecutor 2 provides access to a Flink or Spark cluster.

In Table 1, we show an overview of research challenges (RCs) present in the two layers and also illustrated in Figure 2. We offer a detailed discussion of all RCs in [26]. In the next sections, we present the two layers of Agora referring to all our RCs.

Table 1: Overview of Research Challenges in Agora. A detailed discussion of all challenges can be found at: https://arxiv.org/pdf/1909.03026.pdf [26]

Asset Layer	**RC1: Unified specification.** The design of a unified specification (a standard) for assets. Such a specification is the precondition for sharing, searching, and discovering assets in the ecosystem.
	RC2: Automated specification generation. Asset specifications should automatically be extracted to prevent overhead from providers. Such an automated extraction opens up a new research direction.
	RC3: Matchmaking. Agora should effectively and efficiently identify required assets based on consumer requests. It is a challenge to formalize such requests and match them with assets.
	RC4: Market aggregator. Agora is composed of independent asset marketplaces. It is important for users to be aware of the different marketplaces and their assets through a market aggregator.
	RC5: Constraint specification. Enable the declarative specification of constraints which define the rules for asset sharing and execution.
	RC6: Constraint satisfaction. Efficiently process queries in a manner compliant with respect to asset constraints. Our initial implementation allows expressing constraints on shipping data across geographical borders using extended-SQL statements.
	RC7: Capturing asset provenance. In order to audit compliance with respect to data usage and its sharing policies, we need a technique for capturing provenance in an asset-centric marketplace.
Execution Layer	**RC8: Pricing models.** Our ecosystem should allow providers to define prices of their assets based on different pricing models. Agora should also propose prices based on market monitoring.
	RC9: Asset usage tracking. To ensure fair asset payments, the execution manager needs to track the usage of the assets. Tracking fine-granular operations in a set of assets (e.g., in a pipeline), which may run in parallel, is a challenging task.
	RC10: Payments. Ensuring a safe way to charge and pay the use of assets is crucial for the ecosystem health. A payment process should be distributed such that components can receive payments and split them among their sub-components.
	RC11: Establishing trust through certificates. We need a way to establish trust through certificates in an ecosystem with an extremely large number of actors.
	RC12: Trusted execution environments. A Trusted Execution Environment (TEE) provides a solution for secure computation, which does not require to trust the owner of a compute node. We explore TEE-based solutions in the context of a large data ecosystem.
	RC13: Secure data transfer. We need to enable secure data exchange in Agora. Therefore, we investigate techniques which enable data trading, ensure encryption, and guarantee data integrity.
	RC14: Heterogeneous asset deployment. We want to determine the ideal deployment environment, i.e., the processing system for deploying each asset. For example, if the asset is a stream processing algorithm, Agora might decide to run it on Apache Flink
	RC15: Heterogeneous asset execution. We want to automatically assign assets to compute resources. E.g., matching processor-specific algorithm implementations with the respective processors.

4. ASSET LAYER

The asset layer consists of many connected *asset marketplaces* and *asset managers*. The former allow for sharing assets, while the latter allow for discovering assets across multiple marketplaces.

Each **asset marketplace** contains catalogues that keep track of the available assets and their properties. To make this possible, Agora unifies assets under a common specification which enables easy asset discovery and composition across all the marketplaces in the ecosystem. Providers should conform with this unified specification when they offer new assets to the marketplaces. This can be

a barrier for new asset providers. Therefore, it is crucial that Agora provides the means for automatically generating asset specifications from more intuitive user inputs, such as query and programming languages or graphical interfaces. Defining such a specification and determining ways for its automated extraction is challenging due to the the large heterogeneity of assets (*RC1, RC2*). Our initial efforts [22] towards a unified specification is a declarative intermediate representation of data science assets which is automatically extracted from Python code using static code analysis.

Providers can also define a pricing model (e.g., subscriptions or pay-per-use) for their assets usage (*RC8*). Ideally, Agora proposes a pricing model and a price based on monitoring the current trend of the market. When a provider chooses a pay-per-use pricing model, Agora ensures to track the asset's usage and report usage counters back to marketplace (*RC9*). Marketplaces then perform the invoicing and initiate (micro-)payments between users (*RC10*).

Asset managers are the entry point for users who want to declaratively: find assets across different marketplaces; combine multiple assets into execution plans; and run asset execution plans. An asset manager provides a graphical user interface and/or a declarative language for finding and composing assets (*RC3*). A user request is then converted to an intermediate representation (IR), which allows for matching asset specifications with user requests (*RC1*). The asset manager matches user requests to assets that are compatible with each other and satisfy the requests (*RC3*). For this, it needs to aggregate the assets of all marketplaces and build an asset index (*RC4*). Next, the asset manager composes all the relevant assets (with their equivalent assets) such that they fulfill the request. As a result, the asset manager outputs an asset execution plan, which allows the execution layer to further optimize, deploy, and run the plan.

Asset provenance and usage constraints are crucial in this context: We address these points in RC5, RC6, and RC7. Providers can specify usage constraints to their assets, such as location requirements (e.g., private data may not be moved out of a country) or vendor requirements (e.g., my algorithm may not be used by a competitor). Identifying the best way to describe constraints over assets is an interesting research challenge because of the asset heterogeneity and different constraint granularity (*RC5*). This opens up a completely new dimension of "compliant query (asset) processing" that aims at finding efficient ways to process requests in a compliant manner respecting the assets constraints (RC6). In our efforts towards realizing Agora we provide support for compliant geo-distributed query processing. Our initial implementation allows expressing constraints on shipping data across geographical borders using our extended-SQL statements. A query optimizer aims at finding distributed execution plans that are compliant with respect to shipping of intermediate data between compute sites. This concept can be generalized to model other types of constraints as well. For example, a constraint could require an ISO certification for compute nodes, which would prevent assets from being moved out of ISO-certified data centers.

In RC7, we address the challenge of capturing asset provenance in an open asset ecosystem. This is important to ensure compliance with regulations such as *the right to be forgotten* in GDPR and CCPA. We envision a solution based on a provenance graph which allows for tracing back the source(s) of each asset. Combined assets will thereby inherit the graphs of their sub-assets, forming a new, joined graph. By navigating in the provenance graph, one will be able to trace the use of each asset and to withdraw assets if needed. To ensure that withdrawn assets cannot be accessed any more, we plan to explore techniques, which allow for expiring digital assets using encryption keys [5].

5. EXECUTION LAYER

Agora's execution layer consists of *execution managers*, which receive execution plans, and *node executors*, which run consumers' assets.

An **execution manager** is a core component of the execution layer. It is responsible for optimizing an asset execution plan, deploying it on compute nodes, and monitoring its execution. As the plan may contain different variants of operations, the execution manager can schedule an operation of an execution plan on different execution environments (node executors). Achieving this multi-environment execution of a plan is very challenging as the search space of all possibilities to execute a plan becomes very large (*RC14* and *RC15*). The selection of existing variants and the selection of node executors goes hand-in-hand with possible algorithm adaptations, which increases the performance on a particular target system. We already made the first step towards this direction with Rheem [3] and have shown that using multiple data processing platforms significantly decreases the execution time of a single processing task. However, considering highly diverse assets is still an open research problem.

In addition to determining which processing sys-

tem to execute an asset, Agora also determines how to allocate the asset to compute resources. Agora must adapt algorithms to the specific processor they run on to exploit the full potential of heterogeneous computing resources. For this, we must automatically generate such processor-specific algorithm implementations. Our previous work [7, 23, 24] demonstrates that this is indeed feasible: Data processing systems can learn processor-specific implementations during installation or at runtime.

A **node executor** is Agora's interface component to connect arbitrary execution environments with execution managers. For example, in Figure 2 the asset execution plan is deployed to three node executors with different characteristics: NodeExecutor 1 provides access to a trusted execution environment (TEE), which provides additional security because the owner of the node has no access to the executed source code nor the processed data (*RC12*); NodeExecutor 2 provides access to a Flink or Spark Cluster; and NodeExecutor 3 provides direct access to hardware resources on a dedicated server. When dealing with multiple node executors, Agora provides a secure way to transfer data among nodes to validate data integrity and to pay for data that is traded as an asset. This is hard to achieve especially when data is large or data streams have high bandwidth (*RC13*). Note that we provide secure data transfer for transmitting data in processing pipelines. However, in general, Agora tries to avoid data transfer if possible. For example, consider a classification task over a data stream. Agora can compute a model on one provider's infrastructure, then only transfer the model to another provider, and use the model to classify items from a high bandwidth stream at this other provider. The size of the transferred model would be small, but the underlying data processed at each provider can be large.

Node executors and execution managers are responsible for tracking the usage of assets, which is crucial to ensure fair payments. This is a challenging task because it also assumes tracking fine-granular operations in a composition of assets (*RC9*). Agora adopts certificates to ensure transparency and trust between consumers and providers. For example, one can certify the physical location of a node, security standards, compliance with asset usage tracking, or energy efficiency. The main challenge remains in the standardization of certificates and assets requirements (*RC11*).

6. CONCLUSION

We presented Agora, our vision towards a unified asset ecosystem where assets are fine-grained data-related units of production, such as data and algorithms. Agora provides the technical infrastructure for offering, discovering, and combining assets to form novel data-driven applications. One can share assets through marketplaces, use and combine them through asset managers, and execute them through execution managers. We pointed out multiple open research challenges that need to be addressed to make such an asset ecosystem a reality. This paper is a call for action as we believe that the database community is well positioned to lead the efforts towards such a unified asset ecosystem.

Acknowledgments: This work has been supported by the German Ministry for Education and Research as BIFOLD (01IS18025A, 01IS18037A)

7. REFERENCES

[1] D. Abadi et al. The Seattle Report on Database Research. https://db.cs.washington.edu/events/other/2018/Seattle_DBResearch_Report-Full.pdf.
[2] Acumos. https://www.acumos.org.
[3] D. Agrawal, S. Chawla, B. Contreras-Rojas, et al. Rheem: enabling cross-platform data processing – may the big data be with you! In *PVLDB*, 2018.
[4] AWS Marketplace. https://aws.amazon.com/ marketplace.
[5] S. Blumenau and M. Barnes. Systems and methods for expiring digital assets using encryption key, May 18 2006. US Patent App. 11/282,463.
[6] Bonseyes. https://bonseyes.com/.
[7] S. Breß, B. Köcher, H. Funke, et al. Generating custom code for efficient query execution on heterogeneous processors. *VLDB Journal*, 27(6):797–822, 2018.
[8] R. Castro Fernandez, Z. Abedjan, F. Koko, G. Yuan, S. Madden, and M. Stonebraker. Aurum: A data discovery system. In *ICDE*, pages 1001–1012, 2018.
[9] V. Costan and S. Devadas. Intel SGX explained. *IACR Cryptology ePrint Archive*, (086):1–118, 2016.
[10] Data Space. https://www.datapace.io.
[11] Datahub. https://datahub.io.
[12] Datawex. https://www.dawex.com.
[13] Enigma. https://enigma.co.
[14] G Suite Marketplace. https://gsuite.google.com/marketplace.
[15] R. Haenni. Datum network - the decentralized data marketplace - white paper v15. 2017. https://datum.org/assets/Datum-WhitePaper.pdf.
[16] A. Halevy, F. Korn, N. F. Noy, C. Olston, N. Polyzotis, S. Roy, and S. E. Whang. Goods: Organizing google's datasets. In *SIGMOD*, page 795–806, 2016.
[17] A. Jindal, J. Quiané-Ruiz, and J. Dittrich. WWHow! Freeing Data Storage from Cages. In *CIDR*, 2013.
[18] A. Jonathan et al. Nebula: Distributed edge cloud for data intensive computing. *IEEE TPDS*, 2017.
[19] Microsoft Azure. https://azure.microsoft.com.
[20] Ocean Protocol. https://oceanprotocol.com.
[21] OpenAI. https://openai.com.
[22] S. Redyuk. Automated documentation of end-to-end experiments in data science. In *ICDE*, 2019.
[23] V. Rosenfeld, S. Breß, S. Zeuch, T. Rabl, and V. Markl. Performance analysis and automatic tuning of hash aggregation on gpus. In *DaMoN*, 2019.
[24] V. Rosenfeld, M. Heimel, C. Viebig, and V. Markl. The operator variant selection problem on heterogeneous hardware. In *ADMS*, 2015.
[25] M. J. Smith et al. The machine learning bazaar: Harnessing the ml ecosystem for effective system development. In *SIGMOD*, 2020.
[26] J. Traub, Z. Kaoudi, K. Beedkar, et al. Agora: A unified asset ecosystem going beyond marketplaces and cloud services. https://arxiv.org/abs/1909.03026, 2020.