Unified Data Analytics: State-of-the-art and Open Problems

Zoi Kaoudi
Technische Universität Berlin
Berlin, Germany
zoi.kaoudi@tu-berlin.de

Jorge-Arnulfo Quiané-Ruiz
Technische Universität Berlin
Berlin, Germany
jorge.quiane@tu-berlin.de

ABSTRACT

There is an urgent need for unifying data analytics as more and more application tasks become more complex: Nowadays, it is normal to see tasks performing data preparation, analytical processing, and machine learning operations in a single pipeline. Despite this need, achieving this is still a dreadful process where developers have to get familiar with many data processing platforms and write ad hoc scripts for integrating them. This tutorial is motivated by this need from both academia and industry. We will discuss the importance of unifying data processing as well as the current efforts to achieve it. In particular, we will introduce a classification of the different cases where an application needs or benefits from data analytics unification and discuss the challenges in each case. Along with this classification, we will also present current efforts known up to date that aim at unifying data processing, such as Apache Beam and Apache Wayang, and emphasize their differences. We will conclude with open problems and their challenges.

PVLDB Reference Format:
Zoi Kaoudi and Jorge-Arnulfo Quiané-Ruiz. Unified Data Analytics: State-of-the-art and Open Problems. PVLDB, 15(12): 3778 - 3781, 2022.
doi:10.14778/3554821.3554898

1 INTRODUCTION

Over the last decade, we have embarked on an endless race to develop specialized data processing platforms (*platforms* for short) with the goal of enabling users to extract value out of their big data assets [43]. Just under the umbrella of NoSQL, there are reportedly over 200 different platforms[1]. Although each of these platforms excels in different aspects in the design space, users typically end up running their data analytics on suboptimal platforms. This is because choosing the right platform among the myriad of big data platforms is simply a daunting task.

Furthermore, data analytics are moving beyond the limits of a single platform, which makes the task of choosing and integrating the right platforms much more difficult. There are plentiful applications requiring several platforms to perform data analytics. For example, (i) IBM reported that North York hospital needs to process 50 diverse datasets, which are on a dozen different internal platforms [26], (ii) oil & gas companies need to process large amounts of data they produce everyday [25], e.g., a single oil company can produce more than 1.5TB of diverse (structured and unstructured) data per day [13], and (iii) airlines need to analyze large datasets, which are produced by different departments and reside on multiple data sources, to produce global reports for decision makers [1]. These are few examples of applications that use or can exploit the use of a diversity of platforms for effectiveness or efficiency.

Thus, there is a clear need for *unifying data analytics*. Such a need can stem from a very simple task, such as k-means clustering, to a very complex data analytical pipeline, e.g., one that includes data cleaning, preparation, feature extraction, and model training. Unfortunately, achieving data analytics unification is quite challenging, because applications are typically tied to one single platform. The common practice is to develop several specialized analytic applications on top of different platforms and write ad-hoc programs (or scripts) to glue them all together. This is not only a tedious and costly task, but it also requires knowing the intricacies of the different platforms to achieve high efficiency and scalability. The research community has recently recognized the need for a systematic solution that enables data analytics unification [8, 19, 20, 27, 37, 42]. The holy grail is to allow users to express the logics of their application while an intermediate system decides on which platforms to execute each incoming query with the goal of minimizing its cost (e.g., runtime or monetary cost). There have been many research efforts towards this goal, from wrappers and polystores to cross-platform systems [10, 16, 20–22, 28, 36, 41]. Specifically, the recently introduced task of federated learning [2, 5, 14], falls into the category of unifying data analytics.

Tutorial's Goal, Length & Outline. The goal of this tutorial is threefold: (i) First, to introduce the different use cases where unifying data analytics is necessary, (ii) present and classify the state-of-the-art in unifying data analytics, ranging from federated databases and polystores to federated learning, and (iii) make a call for arms by presenting the challenges and open problems in this domain. This will be a 1.5 hours tutorial, which we structure as follows:

(1) Introduction (10min): We will start with the motivation behind using unified analytics nowadays and introduce the necessary background.
(2) Use cases (20 min): We will introduce a classification for the different cases where an application needs unifying data analytics. We will show that, surprisingly, most of us have required more than once support for this unification even without noticing it.
(3) Unified data analytics (45 min): We will then discuss the challenges for providing data processing unification. We will present current approaches and algorithms that enable data processing unification. Especially, we will discuss their benefits and limitations.
(4) Challenges and Opportunities (15 min): At the end of the tutorial, we will highlight the research challenges and open problems where the database community needs to focus on.

Proceedings of the VLDB Endowment, Vol. 15, No. 12 ISSN 2150-8097.
doi:10.14778/3554821.3554898

[1]http://db-engines.com

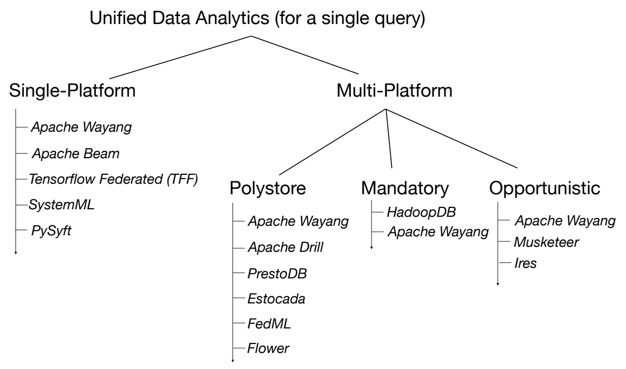

Figure 1: Taxonomy of modern data analytics and current unified analytics systems.

Related tutorials. A small part on the use cases has been presented in ICDE 2018 [31]. However, the ICDE tutorial put emphasis only on the use cases and challenges of cross-platform data processing. In this tutorial, we will provide a complete overview of unifying analytics, discuss the relationship with federated learning, and present the state-of-the-art efforts that did not exist back in 2018. For example, we will introduce efforts on federated learning which is part of unified analytics. A tutorial on federated learning has been recently presented in NeurIPS 2020 [29]. However, that tutorial focused mostly on cross-device federated learning, and in particular federated optimization and differential privacy. In contrast, we will show the relationship of unifying analytics and federated learning from a data management or system perspective. We believe looking at federated learning (and unified analytics in general) from a system perspective will be of high interest to the database community.

Targeted audience and required background. The tutorial targets researchers, developers, and system architects who are keen to know: (i) how they can benefit by combining multiple systems into a single unification layer; (ii) which solutions offer such a unification layer for today's applications; and (iii) how to speed-up their multi-system applications in an easy and systematic manner. The tutorial requires the audience some familiarity with basic database, big data management, and ML concepts.

2 USE CASES

In the first part of the tutorial, we will go through the different use cases where an application requires or benefits from unifying data analytics via the taxonomy shown in Figure 1. In this figure, we illustrate the different approaches for answering a single query. We will showcase these use cases with real data processing examples. In the space of unified data analytics systems, we first identify two cases: An entire query is run in one platform but can be ported to different platforms or a single query can be split into parts which are executed on different platforms. Specifically, we identify four different cases: *platform-independence* (single platform), *opportunistic cross-platform*, *mandatory cross-platform*, and *polystore* (multiple platforms). In addition, we will contrast with parallel, distributed, and federated database systems, such as Garlic[18, 39] and Interbase [17], in order to highlight its uniqueness.

- *Platform-independence (single platform)*: applications may require to switch platforms for two main reasons. First, as new and more efficient platforms become available, developers need to re-implement existing applications on top of faster platforms. For example, Capacitor [38] is the Google Colossus counterpart of Parquet in HDFS. Similarly, SparkSQL is the Spark counterpart of Hive in Hadoop. Second, for different applications, a different platform may be the most efficient one. For instance, running a specific query on a big data platform for very large datasets is often more beneficial than running it on a single-node platform, such as a DBMS. In contrast, for smaller datasets, running the same query on a single-node DBMS might be much faster due to little overhead costs. In fact, several applications in companies and organizations leverage two different processing platforms, such as the machine learning system of IBM [15].
- *Opportunistic cross-platform*: applications might benefit from using multiple platforms throughout a single query. For instance, efficiently supporting hybrid workloads (OLTP and OLAP) might require storing a subset of attributes from a relation as a small table (row layout) and the rest attributes individually (column layout) [11]. Another example of this case is that users can run a gradient descent algorithm, such as SGD, on top of Spark relatively fast. However, mixing the SGD Spark execution with a standalone Java program significantly increases performance [32]. Currently, developers must spot such opportunities for improving performance and write ad-hoc programs or scripts to move data and integrate different platforms.
- *Mandatory cross-platform*: applications need to go beyond the functionalities offered by the platform on which the data is stored. This is because there is no platform that can fit all the data analytics spectrum (following the *one-size-does-not-all* dictum). This leads to complex ad-hoc ETL processes of moving data out of the place it resides and importing it into the system that can perform the processing. Imagine for example that a dataset is stored on a relational database and a user needs to perform a clustering query. Doing so inside the relational database might simply be disastrous in terms of performance. Thus, the user needs to move the data out of the relational database. For example, she might move the data to HDFS in order to use Spark, which is known to be efficient for iterative queries. A similar situation occurs in complex data analytics applications with disparate subqueries. As an example, an application might extract a graph from a text corpus to perform subsequent graph analytics on. This might require using both a text and a graph analytics system. The required integration of platforms is tedious, repetitive, and particularly error-prone.
- *Polystore*: applications might require to use multiple platforms because the input data is stored on multiple data sources (data lakes) and hence a query must be divided accordingly. Datasets in data lakes reside natively on their format and hence on different storage platforms, such as DBMSs, document stores, key-value stores, and pure file systems. Polyglot persistence [40] is another example of using multiple data stores in the same application. Oil & gas [13, 25], health care [26], airline [1] industries, and business intelligence [41] are just few examples of such scenarios.

3 UNIFIED DATA ANALYTICS

In the main part of the tutorial, we will discuss some of the main challenges to be addressed to achieve efficient unified data analytics. After the challenges, we will discuss the different current efforts that have been done to support unified data analytics along different dimensions.

• **Query model and architecture**. One of the core problems to achieve a unified data processing layer is definitely the query model and architecture. The goal is to provide a general query processing model that is able to connect with underlying platforms of different data models. We will discuss different types of data models followed by Apache Beam [3] and Apache Wayang (formely known as Rheem) [4, 10]. In addition, we distinguish between two main types of current systems: (i) providing an internal processing engine (e.g., Apache Drill [6], prestoDB [7], ESTOCADA [12]) or (ii) utilizing only the underlying platforms (e.g., Apache Beam [3], Apache Wayang [10]) for query processing.

• **Automatic platform selection**. One of the challenges towards automatically achieving unified data processing is deciding on which platform(s) an incoming task should be executed. This can be achieved through query optimization with the goal of improving performance without the user's intervention. Although it seems like a traditional query optimization problem, the search space grows not only with the number of physical operators but also with the number of available platforms. In addition, a crucial piece in a cost-based optimizer is an accurate cost model. However, one typically has little control over the platforms in a cross-platform setting which makes cost modeling a big challenge. We will discuss a number of systems that provide different techniques to cross-platform optimization, such as Ires [20], Musketeer [22], Myria [44], and Wayang [30, 34]. These optimizers follow different methodologies, namely rule-based, cost-based and ML-based.

• **Cross-platform data movement and transformation**. Moving data across different platforms is a crucial aspect in unifying data processing. There is an inherent trade-off between choosing the most efficient execution platform and minimizing/optimizing inter-platform data movement. A large overhead of inter-platform data movement would defeat the purpose of using multiple platforms for data processing. Current efforts follow two orthogonal directions. While Wayang aims at minimizing data movement costs [35], Weld [37] and PipeGen [23] propose different techniques on how to optimize data movement. In addition, different platforms may support different data formats and types. For example, moving data from a batch processing platform to a graph processing system requires transforming the data to the right format so that the graph system can ingest the data. The challenge is how to perform such transformations in an automatic and efficient way.

• **Extensibility**. Unifying data processing systems should be designed to be extended to new platforms and operators. This is crucial as existing data processing systems get updated with new functionalities and new data processing systems become available. To this extend, Apache Wayang allows developers to plug new platforms and operators by creating new execution operators, their mappings to Wayang operators, and the appropriate communication channels. For example, Wayang comes with a custom made inequality join operator [33], Then, the optimizer will automatically take these mappings and channels into account without requiring any code changes from the developer [34].

• **Federated learning**. The increased complexity of modern data analytics pipelines also stems from the diverse data sources and storage engines that the data reside. This had led to the recently introduced federated learning: Instead of bringing the datasets in one centralized place to perform the model training, local models are trained on the storage engines that the date reside and a server node gathers and aggregates the local models. Most federated learning systems, such as TensorFlow Federated [2], Syft/Grid [45], FATE [5] are based on such a client-server architecture. All of them support either a single ML backend engine or a couple of ML engines (Syft). In contrast, Flower [14] allows for arbitrary communication patterns. It separates the code for workers and aggregators and supports any ML backend as long as a user implements the required interfaces. FedML [24] allows users to define the low-level communication behavior for each individual participant to realize arbitrary communication topologies.

4 PRESENTERS

Zoi Kaoudi is a Senior Researcher in the DIMA team of the Technical University of Berlin. She has previously worked as a Scientist in the Qatar Computing Research Institute (QCRI) of the Hamad Bin Khalifa University in Qatar, in IMIS-Athena Research Center as a research associate and Inria as a postdoctoral researcher. She received her PhD from the National and Kapodistrian University of Athens in 2011. Her research interests lie in the intersection of machine learning systems, data management and knowledge graphs. She is currently Associate Editor of SIGMOD 2022 and has been the proceedings chair of EDBT 2019, co-chair of the TKDE poster track co-located with ICDE 2018, and co-organizer of the MLDAS 2019 held in Qatar. She has previously presented tutorials at ICDE 2013, SIGMOD 2014, and ICDE 2018. She has co-authored articles in both database and ML communities and served as member of Program Committee for several international database conferences. She has recently received the best demonstration award at ICDE 2022 for her work on "Training data generation for ML-based query optimization". Personal webpage: https://www.user.tu-berlin.de/zkaoudi/

Jorge-Arnulfo Quiane-Ruiz is the head of the Big Data Systems research group at the Berlin Institute for the Foundations of Learning and Data (BIFOLD) and a Principal Researcher at DIMA (TU Berlin). Earlier in his career, he was a Senior Scientist at the Qatar Computing Research Institute (QCRI) and a Postdoctoral Researcher at Saarland University. He obtained his PhD in computer science from INRIA (Nantes University). His current research is in the broad area of big data: mainly in federated data analytics, scalable data infrastructures, and distributed query processing. He has previously presented tutorials at VLDB 2012 and ICDE 2018 and has received an Excellent Presentation Award at VLDB 2014. He has published numerous research papers on data management and novel system architectures, and served as PC member for several international database conferences. He has recently been honoured with the ACM SIGMOD Research Highlights Award and the Best Paper Award at ICDE 2021 for his work on "Efficient Control Flow in Dataflow Systems" as well as the best demonstration award at ICDE 2022. He holds five patents in core database areas and on machine learning. He is a senior PC member of EDBT'23 and has been associate editor of PVLDB 2021 and DASFAA'19. Personal webpage: https://www.user.tu-berlin.de/quiane/

Note: The presenters have several works on unifying data analytics [8, 9, 27, 28, 30, 31, 34, 35] and are the core designers as well as PPMC members of Apache Wayang (Incubating) [2].

ACKNOWLEDGMENTS

This project has been supported by the German Ministry for Education and Research as BIFOLD — "Berlin Insti- tute for the Foundations of Learning and Data" (01IS18025A and 01IS18037A).

REFERENCES

[1] 2019. Fortune magazine. http://fortune.com/2014/06/19/big-data-airline-industry/.

[2] 2019. TensorFlow Federated. https://www.tensorflow.org/federated.

[3] 2021. Apache Beam. https://beam.apache.org.

[4] 2021. Apache Wayang (incubating). https://wayang.apache.org/.

[5] 2021. FATE (Federated AI Technology Enabler). https://github.com/FederatedAI/FATE.

[6] 2022. Apache Drill. https://drill.apache.org.

[7] 2022. PrestoDB Project. https://prestodb.io.

[8] Divy Agrawal et al. 2016. Road to Freedom in Big Data Analytics. In *EDBT*. 479–484.

[9] Divy Agrawal, Lamine Ba, Laure Berti-Equille, Sanjay Chawla, Ahmed Elmagarmid, Hossam Hammady, Yasser Idris, Zoi Kaoudi, Zuhair Khayyat, Sebastian Kruse, Mourad Ouzzani, Paolo Papotti, Jorge-Arnulfo Quiané-Ruiz, Nan Tang, and MohammedJ. Zaki. 2016. Rheem: Enabling Multi-Platform Task Execution. In *SIGMOD*. 2069–2072.

[10] Divy Agrawal, Sanjay Chawla, Bertty Contreras-Rojas, Ahmed K. Elmagarmid, Yasser Idris, Zoi Kaoudi, Sebastian Kruse, Ji Lucas, Essam Mansour, Mourad Ouzzani, Paolo Papotti, Jorge-Arnulfo Quiané-Ruiz, Nan Tang, Saravanan Thirumuruganathan, and Anis Troudi. 2018. RHEEM: Enabling Cross-Platform Data Processing - May The Big Data Be With You! -. *Proc. VLDB Endow.* 11, 11 (2018), 1414–1427.

[11] Mohammed Al-Kateb, Paul Sinclair, Grace Au, and Carrie Ballinger. 2016. Hybrid Row-Column Partitioning in Teradata. *PVLDB* 9, 13 (2016), 1353–1364.

[12] Rana Alotaibi, Damian Bursztyn, Alin Deutsch, Ioana Manolescu, and Stamatis Zampetakis. 2019. Towards Scalable Hybrid Stores: Constraint-Based Rewriting to the Rescue. In *SIGMOD*. 1660–1677.

[13] Abdelkader Baaziz and Luc Quoniam. 2014. How to use Big Data technologies to optimize operations in Upstream Petroleum Industry. In *21^{st} World Petroleum Congress*.

[14] Daniel J. Beutel, Taner Topal, Akhil Mathur, Xinchi Qiu, Titouan Parcollet, and Nicholas D. Lane. 2020. Flower: A Friendly Federated Learning Research Framework. *CoRR* abs/2007.14390 (2020). arXiv:2007.14390 https://arxiv.org/abs/2007.14390

[15] Matthias Boehm, Michael Dusenberry, Deron Eriksson, Alexandre V. Evfimievski, Faraz Makari Manshadi, Niketan Pansare, Berthold Reinwald, Frederick Reiss, Prithviraj Sen, Arvind Surve, and Shirish Tatikonda. 2016. SystemML: Declarative Machine Learning on Spark. *PVLDB* 9, 13 (2016), 1425–1436.

[16] Francesca Bugiotti, Damian Bursztyn, Alin Deutsch, Ioana Ileana, and Ioana Manolescu. 2015. Invisible Glue: Scalable Self-Tuning Multi-Stores. In *CIDR*.

[17] Omran A. Bukhres et al. 1993. InterBase: An Execution Environment for Heterogeneous Software Systems. *IEEE Computer* 26, 8 (1993), 57–69.

[18] Michael J. Carey et al. 1995. Towards Heterogeneous Multimedia Information Systems: The Garlic Approach. In *RIDE-DOM*. 124–131.

[19] Jens Dittrich and Alekh Jindal. 2011. Towards a One-Size-Fits-All Database Architecture. In *CIDR*.

[20] Katerina Doka, Nikolaos Papailiou, Victor Giannakouris, Dimitrios Tsoumakos, and Nectarios Koziris. 2016. Mix 'n' match multi-engine analytics. In *IEEE BigData*. 194–203.

[21] Jennie Duggan et al. 2015. The BigDAWG polystore system. *ACM SIGMOD Record* 44, 2 (2015), 11–16.

[22] Ionel Gog et al. 2015. Musketeer: all for one, one for all in data processing systems. In *EuroSys*. 2:1–2:16.

[23] Brandon Haynes, Alvin Cheung, and Magdalena Balazinska. 2016. PipeGen: Data Pipe Generator for Hybrid Analytics. In *SoCC*. 470–483.

[24] Chaoyang He, Songze Li, Jinhyun So, Mi Zhang, Hongyi Wang, Xiaoyang Wang, Praneeth Vepakomma, Abhishek Singh, Hang Qiu, Li Shen, Peilin Zhao, Yan Kang, Yang Liu, Ramesh Raskar, Qiang Yang, Murali Annavaram, and Salman Avestimehr. 2020. FedML: A Research Library and Benchmark for Federated Machine Learning. *CoRR* abs/2007.13518 (2020).

[25] Adam Hems, Adil Soofi, and Ernie Perez. 2014. How innovative oil and gas companies are using big data to outmaneuver the competition. Microsoft White Paper, http://goo.gl/2Bn0xq.

[26] IBM. 2017. Data-driven healthcare organizations use big data analytics for big gains. White paper, http://goo.gl/AFIHpk.

[27] Alekh Jindal, Jorge-Arnulfo Quiane-Ruiz, and Jens Dittrich. 2013. WWHow! Freeing Data Storage from Cages. In *CIDR*.

[28] Alekh Jindal, Jorge-Arnulfo Quiane-Ruiz, and Samuel Madden. 2013. Cartilage: Adding Flexibility to the Hadoop Skeleton. In *SIGMOD*. 1057–1060.

[29] Peter Kairouz, Brendan McMahan, and Virginia Smith. 2020. Federated Learning and Analytics: Industry Meets Academia. In *NeurIPS (tutorial)*.

[30] Zoi Kaoudi, Jorge-Arnulfo Quiané-Ruiz, Bertty Contreras-Rojas, Rodrigo Pardo-Meza, Anis Troudi, and Sanjay Chawla. 2020. ML-based Cross-Platform Query Optimization. In *ICDE*. 1489–1500.

[31] Zoi Kaoudi and Jorge-Arnulfo Quiané-Ruiz. 2018. Cross-Platform Data Processing: Use Cases and Challenges. In *ICDE (tutorial)*.

[32] Zoi Kaoudi, Jorge-Arnulfo Quiane-Ruiz, Saravanan Thurumuruganathan, Sanjay Chawla, and Divy Agrawal. 2017. A Cost-based Optimizer for Gradient Descent Optimization. In *SIGMOD*.

[33] Zuhair Khayyat, William Lucia, Meghna Singh, Mourad Ouzzani, Paolo Papotti, Jorge-Arnulfo Quiané-Ruiz, Nan Tang, and Panos Kalnis. 2015. Lightning Fast and Space Efficient Inequality Joins. *PVLDB* 8, 13 (2015), 2074–2085.

[34] Sebastian Kruse, Zoi Kaoudi, Bertty Contreras-Rojas, Sanjay Chawla, Felix Naumann, and Jorge-Arnulfo Quiané-Ruiz. 2020. RHEEMix in the data jungle: a cost-based optimizer for cross-platform systems. *VLDB J.* 29, 6 (2020), 1287–1310. https://doi.org/10.1007/s00778-020-00612-x

[35] Sebastian Kruse, Zoi Kaoudi, Jorge-Arnulfo Quiané-Ruiz, Sanjay Chawla, Felix Naumann, and Bertty Contreras-Rojas. 2019. Optimizing Cross-platform Data Movement. In *ICDE*.

[36] Harold Lim, Yuzhang Han, and Shivnath Babu. 2013. How to Fit when No One Size Fits. In *CIDR*.

[37] Shoumik Palkar, James J. Thomas, Anil Shanbhag, Malte Schwarzkopt, Saman P. Amarasinghe, and Matei Zaharia. 2017. A Common Runtime for High Performance Data Analysis. In *CIDR*.

[38] Mosha Pasumansky. April 26, 2016. Inside Capacitor, BigQuery's Next-Generation Columnar Storage Format. Google Cloud Platform.

[39] Mary Tork Roth and Peter M. Schwarz. 1997. Don't Scrap It, Wrap It! A Wrapper Architecture for Legacy Data Sources. In *VLDB*. 266–275.

[40] Pramod J. Sadalage and Martin Fowler. 2012. *NoSQL distilled: A brief guide to the emerging world of polyglot persistence.* Addison-Wesley Professional.

[41] Alkis Simitsis, Kevin Wilkinson, Malu Castellanos, and Umeshwar Dayal. 2012. Optimizing Analytic Data Flows for Multiple Execution Engines. In *SIGMOD*. 829–840.

[42] Michael Stonebraker. July 13, 2015. The Case for Polystores. ACM SIGMOD Blog.

[43] Michael Stonebraker and Ugur Çetintemel. 2005. "One Size Fits All": An Idea Whose Time Has Come and Gone (Abstract). In *ICDE*.

[44] Jingjing Wang, Tobin Baker, Magdalena Balazinska, Daniel Halperin, Brandon Haynes, Bill Howe, Dylan Hutchison, Shrainik Jain, Ryan Maas, Parmita Mehta, Dominik Moritz, Brandon Myers, Jennifer Ortiz, Dan Suciu, Andrew Whitaker, and Shengliang Xu. 2017. The Myria Big Data Management and Analytics System and Cloud Services. In *CIDR*.

[45] A. Ziller, A. Trask, A. Lopardo, et al. 2021. PySyft: A Library for Easy Federated Learning. In *Federated Learning Systems: Towards Next-Generation AI*. 111–139.

[2] https://github.com/apache/incubator-wayang