ECDP: A Big Data Platform for the Smart Monitoring of Local Energy Communities

(Application Paper)

Luca Gagliardelli[1], Luca Zecchini[1], Domenico Beneventano[1], Giovanni Simonini[1], Sonia Bergamaschi[1], Mirko Orsini[2], Luca Magnotta[2], Emma Mescoli[2], Andrea Livaldi[2], Nicola Gessa[3], Piero De Sabbata[3], Gianluca D'Agosta[3], Fabrizio Paolucci[3] and Fabio Moretti[3]

[1] *University of Modena and Reggio Emilia, Modena, Italy,* `{name.surname}@unimore.it`

[2] *DataRiver S.r.l., Modena, Italy,* `{name.surname}@datariver.it`

[3] *Italian National Agency for New Technologies, Energy and Sustainable Economic Development (ENEA), Bologna, Italy,* `{name.surname}@enea.it`

Abstract

In this paper we present the Energy Community Data Platform (ECDP), a middleware platform designed to support the collection and the analysis of big data about the energy consumption inside local energy communities, with the aim of encouraging a more conscious use of energy by the users. The big data platform, commissioned by ENEA, acquires data of different nature (e.g., describing the measurement of the energy consumption and production, weather conditions, etc.) in a heterogeneous format from multiple sources. We describe the architecture of ECDP, designed to support a *Data Integration Workflow* and a *Data Lake Workflow*, conceived for different uses of the data, motivating our technological choices. Then, we illustrate several dataflows reflecting real-world use cases, which highlight the advantages offered by the designed architecture for different types of users. The main strengths of the presented big data platform are flexibility and scalability (guaranteed by its modular architecture), which allow its applicability to any type of local energy community.

Keywords

Big Data Integration, Energy Communities, Big Data Platform

1. Introduction

The Energy Community Data Platform (ECDP), commissioned by ENEA (the Italian National Agency for New Technologies, Energy and Sustainable Economic Development), is a middleware platform we designed to collect and analyze big data about the energy consumption inside Local Energy Communities (LEC), with the aim of encouraging a conscious use of energy by the users, at home and in the workplace.

ECDP is designed for the acquisition of data from different dataflows in a heterogeneous format, the proper management of the workflows to retrieve and store the great amount of data acquired from different sources and utilities (of public or private nature), and functionalities of data integration, transformation, and cleaning, required to make the data ready to run queries on it and for its use in data analysis and visualization operations.

The design of the big data platform was driven by several real-world use cases, which allowed to detect two main requirements to be satisfied: *(i)* it must allow to preprocess the data in a batch and periodical way, then to store it, making it available for data analysis without having to repeat these operations every time; *(ii)* it must integrate heterogeneous data acquired from multiple sources with different characteristics. To comply with the first requirement, ECDP exploits an efficient data lake storage layer, namely Delta Lake [1]. Storing the data in Delta Lake allows to mitigate the excessive execution time needed to import large files or to import data in bulk from database management systems. For the second requirement, the architecture of the big data platform is centered on MOMIS (Mediator EnvirOnment for Multiple Information Sources), an open-source data integration system [2, 3, 4] which adopts a semantic approach to the integration of different data sources, also making the acquisition of new sources easier. ECDP was developed following the wrapper/mediator architecture of MOMIS, which allows to aggregate information from heterogeneous data sources (both structured and semi-structured) and make it homogeneous, in a semi-automatic way. Thus, it makes possible to obtain a single unified source, without any redundancy or conflict in data.

In Section 2, we describe the architecture of the big data platform, motivating the structural choices and the adopted technologies. The main strengths of the pro-

Published in the Workshop Proceedings of the EDBT/ICDT 2022 Joint Conference (March 29–April 1, 2022, Edinburgh, UK)

0000-0001-5977-1078 (L. Gagliardelli); 0000-0002-4856-0838 (L. Zecchini); 0000-0001-6616-1753 (D. Beneventano); 0000-0002-3466-509X (G. Simonini); 0000-0001-8087-6587 (S. Bergamaschi); 0000-0002-5087-9530 (M. Orsini)

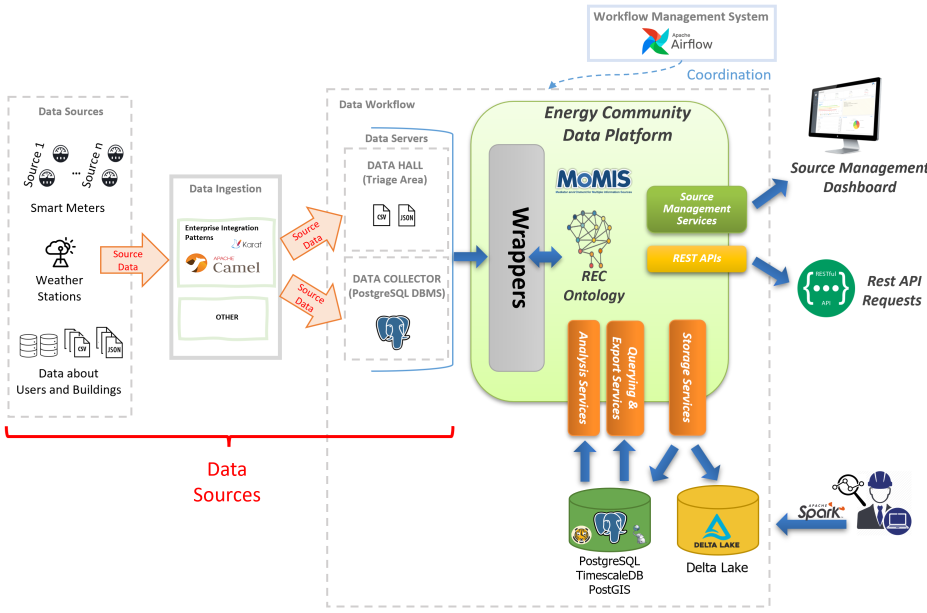

Figure 1: Architecture of the Energy Community Data Platform (ECDP).

posed solution are flexibility and scalability. ECDP was designed in a modular manner as a flexible and open system, to ease future extensions for the implementation of new functionalities, different ways for connecting to and retrieving data from devices or services, and the integration with the other components of the software architecture or software systems present inside the LEC. These features allow an extremely wide range of applications for ECDP, making it suitable for any type of local energy community.

The flexibility of ECDP is highlighted in Section 3, where we illustrate its adaptability to several real-world use cases that drove the design process, addressing different application areas. These use cases are related to projects carried out by ENEA[1], namely *SelfUser*[2], aiming at maximizing the collective self-consumption based on renewable energy in a condominium (Section 3.2), and *PELL*[3], to optimize the energy consumption of public lighting (Section 3.3). Moreover, also the European project *GECO*[4] about green energy communities was considered as a reference for the design of the architecture.

Contributions by the groups

The presented big data platform was designed and developed within the project “Smart Monitoring of a Local Energy Community” (2020-2021), supervised by ENEA. The design of the platform was carried out by the Database Research Group (DBGroup[5]) of the University of Modena and Reggio Emilia, leveraging its experience in big data integration and analysis [5, 6, 7, 8, 9], jointly with DataRiver[6], which subsequently took care of its implementation. Both phases complied with the specifications and the technical documentation provided by ENEA and saw a continuous interaction with its project team.

2. The Energy Community Data Platform (ECDP)

This section illustrates the architecture of the Energy Community Data Platform (ECDP), represented in Figure 1, whose main characteristics and requirements were presented in Section 1.

2.1. Data Sources

The three modules identified as "Data Sources" in Figure 1 reflect the existing scenario and the solutions adopted by ENEA, and constitute the basis to build the big data platform on.

Data sources are multiple and heterogeneous, and will certainly increase in number and change over time. The data about the functioning of the LEC can be static or dynamic. Static data is collected once and stored in tables

[1]https://www.enea.it
[2]http://www.selfuser.it
[3]https://www.pell.enea.it
[4]https://www.gecocommunity.it

[5]https://dbgroup.unimore.it
[6]https://www.datariver.it/en

of a relational DBMS; it can be related to people (e.g., personal information), buildings (e.g., geopositioning or exterior insulation), or apartments/offices (e.g., air conditioning). Dynamic data, which is collected by sensors (continuously or at different intervals) or received from outside the LEC (e.g., from energy providers), is mainly related to the energy consumption, production, or accumulation. In the considered use cases, for example, ECDP has to deal with data about the energy consumption/production in condominiums and single apartments, measured with a granularity of one second and sent weekly in CSV format as ZIP archives (i.e., *condominium data*) or measured with a granularity of 15 minutes and sent continuously as CSV files (i.e., *smart meter data*), but also *meteorological data* collected by weather stations with a granularity of one minute and stored as JSON files, and many other types of data, such as information about the presence of the personnel in the offices (in TXT format).

Furthermore, the considered projects also present some specific additional data. SelfUser adds to the described scenario structured data stored in a PostgreSQL DBMS, providing information about the energy consumption, the energy production by photovoltaic panels, and weather conditions. This structured data presents a granularity of 15 minutes and is obtained by preprocessing some of the raw data described above. PELL platform deals with static data (e.g., the position of the lighting devices), stored using a MySQL DBMS, and dynamic data (e.g., information about the energy consumption, measured by smart meters), stored in HDFS [10] according to the UrbanDataset data structure[7].

The ingestion of the described input data can be managed in different ways. The main solution adopted by ENEA is based on the open-source system integration framework Apache Camel[8], running in a deeply customized OSGi Apache Karaf[9] container, named SignalMix. In this environment, both specific custom components and Camel allow to use domain specific languages (namely, a Java DSL and a Spring XML DSL) to define routes, containing flow and logic of integration between different systems, protocols, and formats, supporting most of the enterprise integration patterns. However, also different solutions are adopted depending on the source (e.g., some JSON/CSV files are sent directly by the users via e-mail).

ENEA exploits two web servers to store the ingested data. The central element is the Data Hall server, which supports the so-called "Triage Area", which is the reference for the storage of unstructured data and contains the files (whose format can be CSV, JSON, etc.) as they are acquired from the sources by data ingestion systems, without applying further preprocessing operations on them. From the Data Hall server it is possible to access the second server, called Data Collector, via remote port forwarding. This server supports a PostgreSQL[10] DBMS, which represents the reference for structured data instead (e.g., the ones related to SelfUser). Moreover, it is possible to exploit additional database management systems for testing which are available on other local network servers.

[7] https://smartcityplatform.enea.it/UDWebLibrary
[8] https://camel.apache.org
[9] https://karaf.apache.org
[10] https://www.postgresql.org

2.2. Data Workflow

The module on the right in Figure 1 represents ECDP, the big data platform designed to extract value from this great amount of ingested data. To ensure maximum flexibility, allowing different types of users to operate on data in different ways, it is possible to define two specific workflows (with the related functional modules): *(i)* a *Data Integration Workflow*, designed to perform data integration, which provides the main access point to the unified data for every application; *(ii)* a *Data Lake Workflow*, which allows to store the whole historical data in a raw (i.e., not integrated) form, available to be handled by expert users if needed. Both systems are coordinated through a *Data Workflow Management System*, used to define the settings for the operations needed to manage the data workflow (e.g., new input data activating triggers/ETL, failure alerts, error-handling). These three modules are detailed in the following subsections.

2.2.1. Data Integration Workflow: MOMIS

The chosen data integration tool is MOMIS, and it was a natural choice, since this open-source system currently managed by DataRiver was designed by the DBGroup and played a central role in its research activities for several years [11, 12].

MOMIS is based on a wrapper/mediator architecture: the wrapper makes available a data source to be integrated, then the mediator performs data fusion [13] to generate in a semi-automatic way a mediated schema, called Global Virtual View (GVV) or Global Schema (GS), of the schemas of the local sources. The user can query this schema (through the query manager or through third-party applications) to obtain a complete and unified view of the data contained in the local sources. MOMIS is a virtual data integration system, i.e., the data is retrieved from the sources at query time. This allows to avoid data replication, so that each query returns updated data. However, it is possible to materialize some global classes (*materialized views*); this can be useful to optimize the retrieval of frequently used data or complex queries.

In particular, ECDP is based on MOMIS I4.0, the specific industrial IoT extension of MOMIS for Industry 4.0 [4], a web and mobile application designed to effectively collect and manage big data generated by machinery and sensor networks in industrial processes, exploiting artificial intelligence and machine learning techniques. This platform provides tools for the continuous monitoring and advanced services for the real-time analysis of production and quality performance, which allows to learn from experience for predictive maintenance policies, production process optimization, and energy consumption reduction. The technology stack of MOMIS I4.0 is illustrated in Figure 2.

Data ingestion is performed by software modules called *wrappers*, which allow to connect to data sources, extracting their schemas and features. The interfaces created through the analysis of these schemas allow to represent the heterogeneous sources in a common language, making them homogeneous through data integration services. Several wrappers were created for ECDP to obtain the representation of data from different dataflows: *(i)* a version of the CSV wrapper for detecting the ZIP archives about *condominium data* on the Data Hall server and extracting their content; *(ii)* a version of the CSV wrapper for detecting the files about *smart meter data* on the Data Hall server; *(iii)* a version of the JSON wrapper for detecting the files about *meteorological data* on the Data Hall server; *(iv)* the PostgreSQL wrapper for connecting to the DBMS on the Data Collector server to ingest *SelfUser data*; *(v)* the Delta Lake wrapper to ingest *PELL data* from the data lake. Moreover, an MQTT, an OPC UA, and a WoT wrapper, not used in the final version, were implemented and initially considered to directly ingest data from smart meters.

The software module for data integration exploits the MOMIS mediator to obtain the semantic representation of data sources, allowing to create an ontology of the domain, called Renewable Energy Community (REC) Ontology. The semantic integration is used to perform the mapping between data sources and the mediated schema. Novel services to support data integration were developed for ECDP, allowing to intervene on dataflows with transformation operations such as applying mathematical operators, computing average/minimum/maximum values, performing time conversion to different formats (e.g., UNIX format) and time-based missing value imputation, adding or removing tuples according to data timestamp.

ECDP exploits a hybrid storage, based on two solutions for different uses: *(i) Delta Lake* storage layer, used to store big data for offline data historicization and analysis (if the data timestamp has passed the retention time, the data is historicized in Delta Lake and removed from PostgreSQL); *(ii) PostgreSQL* relational DBMS optimized through *PostGIS* and *TimescaleDB* extensions for querying temporal series, used to answer most frequent queries on recent data, and containing synthesis of the data historicized in Delta Lake.

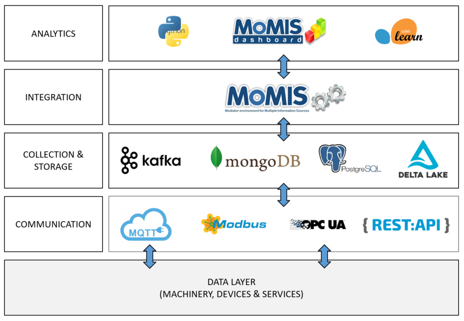

Figure 2: MOMIS Industrial IoT Technology Stack.

Data querying and export services are used to allow authorized users and applications to access a certain portion of data or aggregate view. The access is managed through a role-based access control to guarantee to each type of user the right to access the needed information and at the same time the security of the information for which the authorization was not granted. ECDP relies on two distinct APIs: *(i) ExportUD*, to query the integrated table of readings and export the values in UrbanDataset-XML, UrbanDataset-JSON, or raw CSV format; *(ii) SQL_API*, to run the user queries based on the queries defined in the application and the related access authorizations.

ECDP also supports a customized script scheduling service, which can be used to run scripts on MOMIS server. This service simply runs the main class of the script (in Bash or MATLAB language) and is extremely flexible, since it can use the GUI to collect different scripts and to define the configuration for their scheduling.

Finally, the source management and storage services allow to monitor the data collected and processed by the other software modules. In particular, the source management dashboard was realized using MOMIS data analytics module (i.e., MOMIS Dashboard). It supports a unified view of the business data integrated with external data sources, searching and monitoring aggregate data from distributed and heterogeneous data sources, visualizing indicators on charts and dynamic tables, managing security and visibility of data based on roles and user groups.

2.2.2. Data Lake Workflow: Delta Lake

The estimated size of data and the available resources allow to store raw data (i.e., data as it is acquired, without applying any preprocessing operation on it) in a data lake which lays on HDFS. The main goal of this workflow is to support data analysis performed by advanced users who need to access raw data to retrieve additional information that cannot be obtained from integrated data. For example, raw sensor data collected with a granularity of one second is stored in the data lake, while in Data Integra-

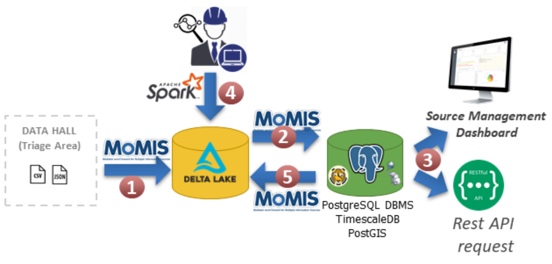

Figure 3: Data Hall Dataflow.

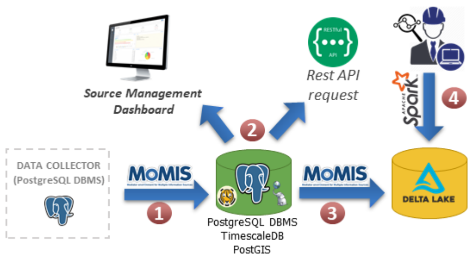

Figure 4: Data Collector Dataflow.

tion Workflow it is aggregated according to a granularity of 15 minutes. Thus, if an advanced user needs data at the finest granularity, it is necessary to exploit Data Lake Workflow. The chosen technology for the data lake is Delta Lake [1], an open-source project which allows to manage a great amount of data using existing storage tools such as HDFS. Delta Lake is integrated with the whole Apache Spark[11] ecosystem, allowing the native use of all its libraries (e.g., MLib for machine learning) and to execute efficient elaborations on data both in batch and in streaming mode. The possibility of exploiting this powerful engine to operate on raw data was one of the core reasons for the choice of Delta Lake, also considering the significant role played by Spark in the DBGroup research activity [14, 15, 16]. Delta Lake is a mature device, used by multiple cloud service providers such as Databricks and Microsoft Azure, which supports ACID transactions, guaranteeing the highest level of isolation (this allows to execute at the same time read and write operations without issues about data integrity). Delta Lake exploits Apache Parquet[12], an open-source format which operates data compression, requiring much less space for data storage than the one required by the original format (usually CSV). Moreover, it supports data versioning, tracking the operations executed on data through a system of logs and allowing to execute rollback, it allows to modify data schema, and it handles metadata as it were data.

2.2.3. Data Workflow Management System: Apache Airflow

The goal of the Data Workflow Management System is the orchestration of the different functional modules. In particular, when new data is made available from the sources, it has to provide the mechanisms to automatically identify it and to raise the appropriate workflows. Furthermore, it has to support the handling of the presence of unexpected errors and it must allow to compose workflows by combining different software modules. The chosen tool has to be compatible with these modules, programmable, and simple to use. Considering these requirements, we chose Apache Airflow[13], which allows to: *(i)* manage task workflows through simple Python scripts defining the dependencies among tasks, whose execution order can be represented with a directed acyclic graph (each module is a black box, so it can be executed by any tool and implemented with any programming language); *(ii)* monitor the workflow (even through a GUI) and manage the possible errors generated by tasks, using scripts to define the trigger rules.

[11]https://spark.apache.org
[12]https://parquet.apache.org

3. Use Cases and Scenarios

In this section we report three dataflows related to real-world use cases that show how ECDP can be used in different application scenarios. In particular, we want to illustrate: *(i)* how unstructured data is ingested (Section 3.1); *(ii)* how structured data is managed (Section 3.2); *(iii)* how it is possible to use Spark to directly import data in the platform (Section 3.3).

The chosen dataflows let us highlight the advantages and the flexibility offered by the coexistence of the two distinct workflows, using a data lake to store the entire raw data (i.e., *Data Lake Workflow*) in combination with a structured database to store the integrated data (i.e., *Data Integration Workflow*). In fact, this solution allows the standard users to access an integrated and clean view of the data (e.g., they can see in a dashboard the hourly energy production by the photovoltaic panels related to the weather conditions), while advanced users can use the raw data to perform advanced tasks (e.g., a data scientist can use the data about the energy production by the photovoltaic panels collected at the finest granularity combined with the solar radiation collected by the sensors to train a machine learning model to predict the energy production).

3.1. Data Hall Dataflow

Data Hall is the web server managed by ENEA on which is located the so-called "Triage Area" (see Figure 1), that hosts unstructured raw data (e.g., JSON and CSV files).

[13]https://airflow.apache.org

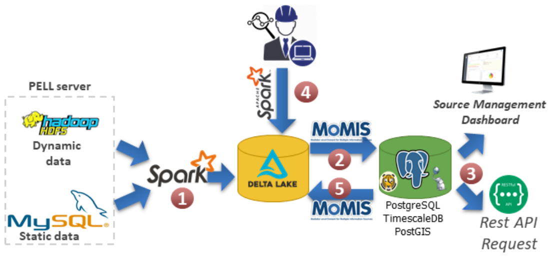

Figure 5: Delta Lake Dataflow.

The web server mainly contains data of the SelfUser project, which aims to test the Clean Energy Package directive by creating an innovative plant, in a pilot form, to support the energy transition through the promotion of energy produced by renewable sources (in this case, photovoltaics). The data, such as the energy consumption, the energy production by photovoltaic panels, and weather conditions, is collected from several sensors placed in smart buildings. The data can be considered as a time series with a different time granularity: the energy consumption/production is measured every second, while the weather information every minute.

The dataflow is described in Figure 3. Firstly, MOMIS (Figure 3-1) acquires the new data from the Data Hall server by using its wrappers and stores it into Delta Lake, where it can be used for further analysis by directly querying the data lake with the connectors provided by Spark (Figure 3-4). Then, raw data is processed by MOMIS to clean and integrate it. The resulting integrated data is materialized in a PostgreSQL database with PostGIS and TimescaleDB extensions: this data can be considered as a time series with also geographic coordinates, reflecting the position of the sensors (Figure 3-2). The integrated data is made available for external services (e.g., MOMIS Dashboard, REST APIs) that can be used to analyze the most recent data (Figure 3-3). Then, after a predefined retention time, the integrated data is moved from PostgreSQL to Delta Lake (Figure 3-5) to avoid losing in performance, creating an historical view of the data.

3.2. Data Collector Dataflow

Data Collector is the web server managed by ENEA that hosts the PostgreSQL database containing the structured data of the SelfUser project. The database stores a cleaned and aggregated version of the raw data about the energy consumption, the energy production, and weather conditions; all measurements are aggregated at intervals of 15 minutes.

The dataflow to import this data in ECDP is described in Figure 4. MOMIS connects to the Data Collector server by using the apposite wrapper, acquires the data never read before, and stores it in a PostgreSQL database optimized to manage time series (Figure 4-1). Since this data was already preprocessed and aggregated, no further transformations are needed and can be made available to external data analysis services, such as MOMIS Dashboard and REST APIs (Figure 4-2). Again, the integrated data is periodically moved from PostgreSQL to Delta Lake (Figure 4-3), where advanced users can operate on it using Spark (Figure 4-4).

3.3. Delta Lake Dataflow

Public Energy Living Lab (PELL) is a platform developed by ENEA to collect data about the energy consumption of public lighting, which represents a key task for urban renewal. The platform uses MySQL to store static information about the lighting devices (e.g., the position and other technical details) and HDFS to store the consumption of each device (dynamic data).

The dataflow is described in Figure 5. The static and dynamic data collected from the PELL server is combined by using a Spark script that produces a DataFrame which is stored into Delta Lake (Figure 5-1). From Delta Lake, the data can be directly queried by advanced users through the connectors provided by Spark (Figure 5-4) or it can be processed by MOMIS (Figure 5-2). MOMIS reads the data from Delta Lake, performs data integration operations, and stores the data in a PostgreSQL database equipped with extensions to manage time series; in fact, the data about the energy consumption can be seen as a time series, since every record is associated with a specific time. The users can access the integrated data by querying the database through a dashboard with predefined views, or by using REST APIs (Figure 5-3). Periodically, after a predefined retention time, to avoid losing in performance the integrated data is moved from PostgreSQL to Delta Lake (Figure 5-5).

4. Conclusions

We presented the Energy Community Data Platform (ECDP), a big data platform designed for the smart monitoring of local energy communities, to encourage a more conscious use of energy by the users. For this purpose, ECDP can be useful both for the administrators of the energy communities, allowing a better monitoring of the community performance and a classification of the energy consumption profiles, and for the users, which can set energy (self-)consumption targets and receive feedback about their adherence to these plans, adapting their behavior accordingly in a conscious way. The modular architecture of ECDP, conceived to support different uses of data by different types of users, has the goal of maximizing flexibility and scalability. As illustrated through real-world use cases, these features represent the main strengths of the presented big data platform and make ECDP suitable for being applied to any type of local energy community.

The main lesson that we learned from this project, which makes our experience relevant and reusable for related tasks, is that maintaining the workflows separated through a modular architecture allows to combine the strengths of each adopted tool, guaranteeing the availability of data at different abstraction levels, making the system more flexible to better support future changes, and meeting the needs of the different types of users. The data lake (Delta Lake) can store in an efficient way a huge amount of raw data, allowing to perform further analysis operations on it. The data integration system (MOMIS) can clean and integrate the raw data and store it into the relational DBMS (PostgreSQL with TimescaleDB extension). The relational DBMS can be used as a cache to store materialized views of the most recent integrated data (old ones are moved to the data lake) fastening the access by the data analytics tools (MOMIS Dashboard). A single workflow cannot guarantee this flexibility: directly integrating and storing the whole data in a relational DBMS would not allow to exploit raw data to perform new types of analysis in the future, while relying only on Delta Lake raises significant efficiency issues, since its Spark-based interaction requires to load the whole dataframe in memory every time. Abstracting from the specific case, this lesson can be useful for every project dealing with the management and the analysis of time series, proposing an approach to face this challenging task for big data platform design and implementation.

Acknowledgements

The project was funded by the Italian Ministry of Economic Development as a part of the 2019-2021 National Electricity System Research Plan.

References

[1] M. Armbrust, T. Das, S. Paranjpye, R. Xin, S. Zhu, A. Ghodsi, B. Yavuz, M. Murthy, J. Torres, L. Sun, P. A. Boncz, M. Mokhtar, H. V. Hovell, A. Ionescu, A. Luszczak, M. Switakowski, T. Ueshin, X. Li, M. Szafranski, P. Senster, M. Zaharia, Delta Lake: High-Performance ACID Table Storage over Cloud Object Stores, Proc. VLDB Endow. 13 (2020) 3411–3424.

[2] S. Bergamaschi, S. Castano, M. Vincini, Semantic Integration of Semistructured and Structured Data Sources, SIGMOD Rec. 28 (1999) 54–59.

[3] S. Bergamaschi, P. Bouquet, D. Giacomuzzi, F. Guerra, L. Po, M. Vincini, An Incremental Method for the Lexical Annotation of Domain Ontologies, Int. J. Semantic Web Inf. Syst. 3 (2007) 57–80.

[4] L. Magnotta, L. Gagliardelli, G. Simonini, M. Orsini, S. Bergamaschi, MOMIS Dashboard: A Powerful Data Analytics Tool for Industry 4.0, in: TE 2018, volume 7 of *Advances in Transdisciplinary Engineering*, IOS Press, 2018, pp. 1074–1081.

[5] G. Simonini, S. Bergamaschi, H. V. Jagadish, BLAST: a Loosely Schema-aware Meta-blocking Approach for Entity Resolution, Proc. VLDB Endow. 9 (2016) 1173–1184.

[6] G. Simonini, G. Papadakis, T. Palpanas, S. Bergamaschi, Schema-Agnostic Progressive Entity Resolution, in: ICDE 2018, IEEE Computer Society, 2018, pp. 53–64.

[7] G. Simonini, L. Zecchini, S. Bergamaschi, F. Naumann, Entity Resolution On-Demand, Proc. VLDB Endow. 15 (2022).

[8] G. Papadakis, L. Tsekouras, E. Thanos, N. Pittaras, G. Simonini, D. Skoutas, P. Isaris, G. Giannakopoulos, T. Palpanas, M. Koubarakis, JedAI^3: beyond batch, blocking-based Entity Resolution, in: EDBT 2020, OpenProceedings.org, 2020, pp. 603–606.

[9] F. Guerra, G. Simonini, M. Vincini, Supporting Image Search with Tag Clouds: A Preliminary Approach, Adv. Multim. 2015 (2015) 439020:1–439020:10.

[10] D. Borthakur, HDFS Architecture Guide, Hadoop Apache Project (2008).

[11] S. Bergamaschi, D. Beneventano, F. Guerra, M. Orsini, Data Integration, in: Handbook of Conceptual Modeling, Springer, 2011, pp. 441–476.

[12] S. Bergamaschi, D. Beneventano, F. Mandreoli, R. Martoglia, F. Guerra, M. Orsini, L. Po, M. Vincini, G. Simonini, S. Zhu, L. Gagliardelli, L. Magnotta, From Data Integration to Big Data Integration, in: A Comprehensive Guide Through the Italian Database Research, volume 31 of *Studies in Big Data*, Springer International Publishing, 2018, pp. 43–59.

[13] J. Bleiholder, F. Naumann, Data fusion, ACM Comput. Surv. 41 (2008) 1:1–1:41.

[14] G. Simonini, L. Gagliardelli, S. Bergamaschi, H. V. Jagadish, Scaling entity resolution: A loosely schema-aware approach, Inf. Syst. 83 (2019) 145–165.

[15] L. Gagliardelli, G. Simonini, D. Beneventano, S. Bergamaschi, SparkER: Scaling Entity Resolution in Spark, in: EDBT 2019, OpenProceedings.org, 2019, pp. 602–605.

[16] L. Gagliardelli, S. Zhu, G. Simonini, S. Bergamaschi, BigDedup: A Big Data Integration Toolkit for Duplicate Detection in Industrial Scenarios, in: TE 2018, volume 7 of *Advances in Transdisciplinary Engineering*, IOS Press, 2018, pp. 1015–1023.