IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received 5 March 2024, accepted 8 April 2024, date of publication 10 April 2024, date of current version 18 April 2024.

Digital Object Identifier 10.1109/ACCESS.2024.3387315

APPLIED RESEARCH

IW-NET BDA: A Big Data Infrastructure for Predictive and Geotemporal Analytics of Inland Waterways

NIKOLAOS CHALVANTZIS[1], (Graduate Student Member, IEEE), ARISTOTELIS VONTZALIDIS[1], EVDOKIA KASSELA[1], ARIS SPYROU[1], NIKOLAOS NIKITAS[1], NIKODIMOS PROVATAS[1], IOANNIS KONSTANTINOU[2], AND NECTARIOS KOZIRIS[1], (Member, IEEE)

[1]School of Electrical and Computer Engineering, National Technical University of Athens, 115 27 Athens, Greece
[2]Department of Informatics and Telecommunications, University of Thessaly, 351 00 Lamia, Greece

Corresponding author: Nikolaos Chalvantzis (nchalv@cslab.ece.ntua.gr)

This work was supported by the European Union's Horizon 2020 Research and Innovation Program (IW-NET) under Grant 861377.

ABSTRACT The recent shift towards digitalization in traditional sectors like logistics and transportation has unlocked new avenues for gaining valuable insights and streamlining operations. This transformation is facilitated by the abundance and specificity of data now available, including fleet IoT data, transactional documents, and event notifications. These businesses leave a substantial digital footprint, ripe for analysis when combined with external data sources. However, harnessing this information requires robust computing infrastructure and adaptable software capable of handling vast amounts of data. In this paper, we introduce IW-NET BDA, a big-data analytics framework built on open-source technologies to address the storage and processing demands of massive datasets from various origins. Developed within the framework of the EU-funded research and innovation project IW-NET (Innovation driven Collaborative European Inland Waterways Transport Network), our system caters to the logistics domain but offers a versatile IT service backbone due to its agnostic design, focusing on infrastructure-as-a-service provision. Furthermore, it allows for the development and deployment of applications that encapsulate business logic, thus tailored to specific business needs. In the subsequent sections, we delve into the design principles, architectural components, and deployment possibilities of IW-NET BDA. Additionally, we present two illustrative use cases: firstly, the automated detection of areas of interest and vessel activity tracking for insightful geo-temporal data analytics along the River Weser corridor; secondly, the utilization of recurrent neural networks to forecast water levels in the Danube River corridor. These examples highlight the adaptability and efficacy of IW-NET BDA in tackling diverse challenges across different contexts, underscoring its versatility and utility.

INDEX TERMS Big data, cloud computing, DBSCAN, machine learning, recurrent neural networks.

I. INTRODUCTION

With over 40.000 km of navigable waterways and 250 inland ports, Inland Waterway Transport (IWT) within the EU is responsible for the movement of circa 550 million tonnes of goods on a yearly basis, according to the European Inland Waterway Transport Platform.[1] Given the European Commision's official commitment of Europe becoming the first climate-neutral continent by adopting policies that aim at the critical reduction of GHG emissions and the development of an environmentally sustainable economy,[2] over the recent years IWT has emerged as a promising alternative to a more traditional road-bound logistics eco-system. To help further develop IWT – by means of improving its efficiency,

The associate editor coordinating the review of this manuscript and approving it for publication was Md. Moinul Hossain.

[1]https://www.inlandwaterwaytransport.eu/

[2]https://commission.europa.eu/strategy-and-policy/priorities-2019-2024/european-green-deal_en

competitiveness, and reliability – the **IW-NET**[3] EU-funded research and innovation project started in May 2020 and was active until October 2023.

The project addressed operations and focused on use cases inspired by real-life scenarios in different regions across the continent - in particular rivers Danube, Spree/Oder and Weser, and the regions of Brussels and Ghent. In the context of the project, we tried to suggest solutions to problems such as infrastructure bottlenecks, insufficient IT integration along the chain and slow adoption of technologies, such as new vessel types, alternative fuels, automation, IoT advanced data analytics and machine learning, aiming to deliver a multi-modal optimisation process across the EU Transport System. The project's main objectives have been organised in three distinctive directions:

- **Digitalization**, focusing on the optimization of the planning of barge operations in dense urban areas and navigability in unknown water conditions by employing a data-driven approach which involved high-volume data collection from various sources, statistical and predictive analytics and machine learning tools.
- **Sustainable Infrastructure and Intelligent Traffic Management**, focusing on the management of the infrastructure to reduce uncertainty in voyage planning (eg. lock management, berth planning with mandatory shore power supply and other services).
- **Innovative vessels**, focusing on new barge designs that fit corridor conditions and target markets.

Concerning digitalization, our efforts focused on the introduction and adoption of state-of-the-art technologies in a business that has traditionally lagged in that regard. Having extensively documented the requirements and specificities of the logistics business in general and the Inland Waterway operations more specifically, a general architecture was designed where discreet components based on bleeding-edge technologies would interact in order to automate, facilitate and provide visibility to a large part of the supply chain, thus offering a new perspective to many the stakeholders involved in these operations.

A very significant aspect of our approach to the full software stack design has been the reproducibility of our solution. We have made the choice to rely on publicly available, open-source software components to utilise as building blocks for the largest part of the presented infrastructure. Of course, in a market as large as the global logistics one, the problems we have encountered are already being addressed but in such a highly competitive environment, the norm is that software is developed in isolation. Organizations and companies active in the business are investing hefty amounts of resources in research and innovation [1]. What differentiates our initiative in comparison to existing solutions is our open-source and open-data approach which derives from the EU Commission's commitment to fund research that will benefit the global community.

The suggested architecture, displayed in Figure 1, comprises various pluggable and loosely coupled components that interact through secure communication channels. As visualized in the diagram, our data sources can be services implemented by logistics agents, IoT devices or any other service that is designed to collect data. Data is pushed from these services into the system, via a Publish/Subscribe system based on Apache Kafka [2]. The secure communication is safeguarded by a deployed instance of an Identity and Access Management [3] component, based on the state-of-the art open-source solution Keycloak [4]. All interactions marked with a black key in the architecture diagram indicate a secure access communication pattern. The architecture is designed to be modular - all interconnected connected only need to implement an interface which allows them to exchange (receive and dispatch) data with the Publish/Subscribe distributed framework. To satisfy state-of-the-art business requirements, our services were designed to produce and consume data compliant with the GS1 [5] and UN/CEFACT [6] standards for the representation of logistics actions and operations.

In this paper, the focus of our report is IW-NET BDA, an open-source[4] Big Data Storage and Processing System (presented in the diagram as ''Big Data Analytics'', top right). Its role in the suggested architecture is to store data, as well as perform general-purpose data analysis using technologies that support those operations on Big Data scale. In Section II we initially present the architecture and deployment plan of the most significant software components comprising the computing infrastructure where data was stored and processed. In Sections III and IV we present two distinct use cases which we believe provide a representative view of the complexity of the issues that need to be resolved in real-world applications. The solutions to the problems presented here have been utilized to collect and analyze data, producing insights that have been used by the infrastructure owners to monitor the IWT networks, as well as in the implementation of a complex syncrho-modal collaboration algorithm. More specifically, Section III concentrates on geo-temporal data analytics, tracking the live movements of vessels. In Section IV, we focus on the examination of methods for the forecasting of water levels utilising state-of-the-art tools and techniques based on machine learning while Section V concludes our work.

II. INFRASTRUCTURE AND SERVICE DEPLOYMENT

In the heart of every system that aims at enhancing the operational decision-making of modern-day businesses lies the ability to store and process large amounts of data [7], [8], [9], [10], [11]. The software solution that IW-NET proposes is no different. In the current section, we present the IW-NET Big Data Analytics Subsystem (BDA). Specifically designed to support applications that can provide on-the-fly insights on live, streaming data, the BDA is a reliable and highly

[3]https://www.inlandwaterwaytransport.eu/iw-net-project/

[4]https://github.com/iwnet/digitalization-infrastructure

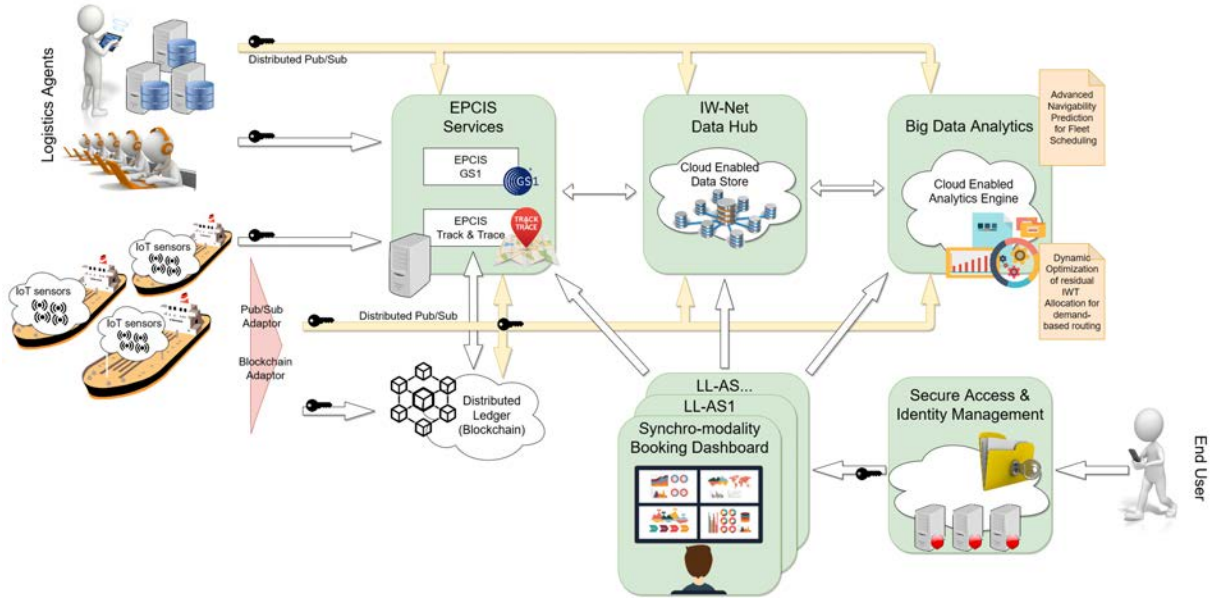


FIGURE 1. High-level architecture of the IW-NET digitalization ecosystem.

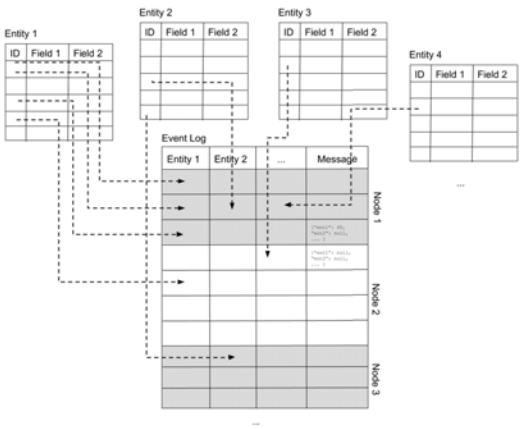


FIGURE 2. The star schema employed by the BDA storage engines.

performant infrastructural layer. Implemented using state-of-the-art technologies adopted by industry and academia alike, such as Apache Spark [12], Hadoop's distributed file system HDFS [13], HBase [14] and Postgres [15] amongst others, it consists of a set of interconnected modules that interact with each other closely, offering big data storage and parallel processing capabilities. It has been designed to be as autonomous and fault-tolerant as possible, requiring minimal intervention by its administrators to perform its intended tasks, self-monitor, recover from failures etc.

From a high level, the BDA's functionality is implemented by its 3 functional components: the Data Storage Engines, the Processing Engines and the KPI Database and Service. However, on a lower-level detailed analysis, it is built upon five discreet modules: the Datastore module, the Analytics and Machine Learning (ML) module and the Controller and Connector modules, the two latter of which implement the interfacing and scheduling tasks it requires. Additionally, a KPI Service has been developed and implemented as a supportive module which offers persistent storage for the output of the Analytics and ML workloads.

The BDA Subsystem follows the "*lambda*" architecture concept [16], to provide both batch and real-time processing with high performance and fault-tolerance. The Data Storage Engines is the functional component responsible for storing all data and meta-data. Incoming data is stored across tables of both NoSQL and RDBMS systems implementing a star schema to optimize query responsiveness. Meta-data are exclusively stored in dedicated RDBMS tables.

The Execution Engines is the component where Analytics or ML tasks are performed to calculate KPIs, predict variables and produce metrics that are required per use case. By design, the BDA tries to abstract business logic, expecting the latter to be implemented in pluggable applications we call "*recipes*".

The KPI Service stores the outcomes of these data processing workloads. Services developed in the context of IW-NET architecture external to the BDA can access the KPI Service and query its data to create reports or visualize the results.

Finally, the Controller and Connector modules act as the gateways that facilitate communication either via REST or the dedicated Publish/Subscribe (Pub/Sub) module, which is in charge of the message exchange in the general architecture and connects other IW-NET services to the BDA, while simultaneously coordinating the rest of the functional components. Except for the Publish/Subscribe mechanism, the BDA is also designed to interact with an additional external Identity and Access Management component, implemented by Keycloak [4] in our prototype. This software allowed

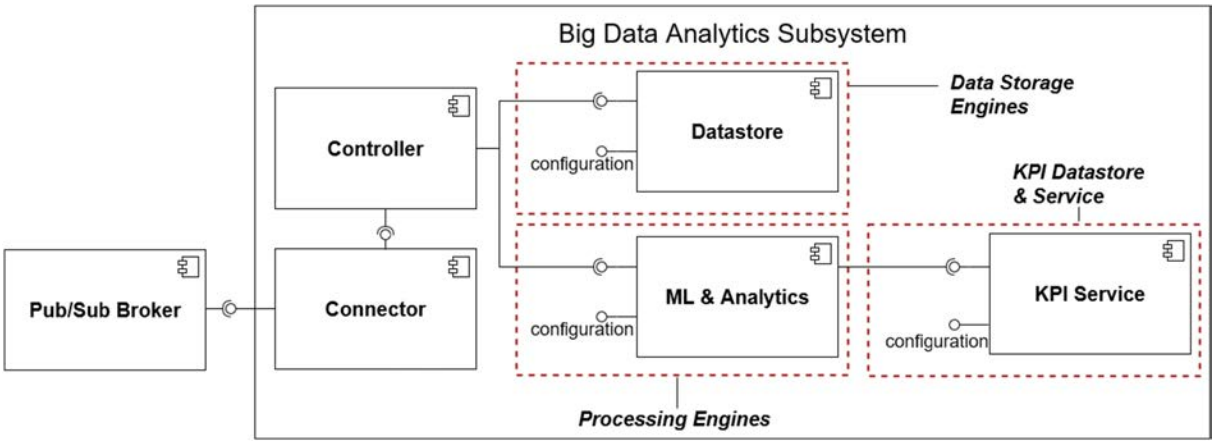


FIGURE 3. BDA subsystem low-level architecture.

us to monitor resource access patterns and define privileges between roles and actors in our use-cases. Data exchanges and interactions implement the TLS [17] protocol to ensure the security of all communications.

The low-level architecture of the BDA subsystem is shown in Figure 3 and includes the aforementioned functional components and the five software modules: the Datastore module, the Analytics and ML module, the KPI Service module and the Controller and Connector modules. In the next paragraphs, we will follow the high-level overview of the BDA and analyze its distinct modules one by one. We shall also dedicate a short paragraph to the connectivity interfaces with other IW-NET architectural components and close this section by providing information on the deployment scheme.

A. DATA STORAGE ENGINES

The Data Storage Engines are implemented within the Datastore module. The Storage Engines are utilized for two purposes: (a) to store data that arrive through the connectivity endpoints and (b) to provide the data for BDA execution engines to execute Analytics or ML workloads with. To support BDA operations and functionality, the Data Storage Engines also maintain meta-data catalogues of all resources created and utilized at the application and infrastructure level. Application level meta-data include information such as valid message types, recipes etc. Infrastructure level meta-data includes execution engine and environment information, specific recipe executable paths etc.

Data Storage: The Star Schema: Data in the Storage Engines are stored using a star schema, which consists of a fact table, which will be referred to as "*Event Log*", and of smaller dimension tables, also referred to as "*Entity Tables*". The Entity Tables contain what is essentially the static data describing the infrastructure of a supply chain, often included in message exchanges. They are dictated by the use case we are addressing and typically need to be populated in the initialization phase of the BDA subsystem. This information rarely changes and needs to be queryable.

On the other hand, the Event Log can be considered as an append-only log that captures every business message (i.e., event) that is flowing into the BDA. The exact schema of the Event Log table will need to be defined in a different step of the initialization process previously mentioned, according to the participating Entities. The basic concept is that every row represents a message and includes information about the entities it references (there might be multiple entities involved - or none, depending on the message type and the event it describes), and the main message payload. Finally, the Event Log is populated in a streaming manner by new event messages after the BDA has been integrated with the appropriate message exchange mechanism. The star schema of the BDA Data Storage Engines is shown in Figure 2. It contains numerous small, immutable Entity Tables and a single constantly growing Event Log table. Each message is appended as a new row in the Event Log and is linked to one or more of the existing Entities.

As a motivating example, consider a simple scenario where our use-case describes a fleet of barges sailing between a network of ports and the messages consist of arrivals and departures. In this case, there would exist two Entity Tables: one describing the meta-data related to barges and a second one, containing meta-data related to ports. The Entity Table dedicated to ports could additionally include data such as port location, capacity etc. Each message stored in the Event Log would either describe an event of type "*arrival*" or "*departure*" and would be linked to both a barge and a port, from both Entity tables.

Meta-Data Storage: As different message types, related to different events, can trigger different processing workloads, it is necessary to also create a persistent storage resource where the different message types, their attributes and workload-related meta-data are stored in dedicated, interconnected meta-data tables. We also store information

about the workloads, the configurations of the execution environments and other meta-data information per use case. These tables are updated/populated with new data upon new message type and workload creation. An example can be seen in Figure 4. The functionality of the entities introduced here is explained in the following subsection, on the Processing Engines.

Implementation Details: The Datastore module offers a RESTful interface to facilitate communication with the storage engines while being agnostic to the specific underlying systems utilized to implement them. The API has the capability to work with several supported storage systems by using dedicated integrated connectors depending on the BDA subsystem's configuration provided during the initialization. By default, different storage systems are used to store the Event Log table and the Entities tables to optimize performance: The Event Log, as an append-only structure that constantly grows is better suited to a distributed database or a distributed file system using the appropriate connectors, whereas the Entities tables contain limited, immutable information which may be accessed very often so a replicated RDBMS solution or an in-memory database are a more suitable choice [18]. Each provided system connector internally implements the required operations on the data.

Taking into consideration all of the above, as well as the non-functional requirements that must be met by the storage service, a containerized setup of the distributed NoSQL Apache HBase [14] database is employed to store the Event Log table. HBase offers out-of-the-box scalability and high-availability features. It supports storage in a semi-structured and sparse format thus allowing us to adopt any data schema that a use case requires in an efficient way. This feature is very useful as our design is use case agnostic and the flexibility it offers permits the infrastructure to be usable in various scenarios. In HBase, data rows are divided into multiple shards (i.e. partitions) and distributed to the cluster nodes. Therefore, consecutive rows of the Event Log are stored in different nodes which allows for quick data ingestion. Moreover, HBase allows user access to be controlled on a database, table or even file system folder level. We deploy an instance of Apache Zookeeper [19] to implement the high-availability feature.

For the Entities tables, on the other hand, a replicated, containerized deployment of the PostgreSQL [15] RDBMS is utilized. Multiple copies of the same database are created, using leader-follower streaming data replication [20]. Each copy is hosted by a dedicated machine in a cluster setup. One of the servers is considered as the leader, handling the write requests. Meanwhile, the follower machines only allow read access to the data that they contain. We use pgpool [21], a proxy layer that is used to achieve load-balancing and scalability for the read throughput when accessing the Entities tables. Pgpool ensures that, in the case where the primary leader server fails, a follower can replace it, thus providing high-availability. PostgreSQL - a typical RDBMS - offers adaptable database schema and user roles and permissions.

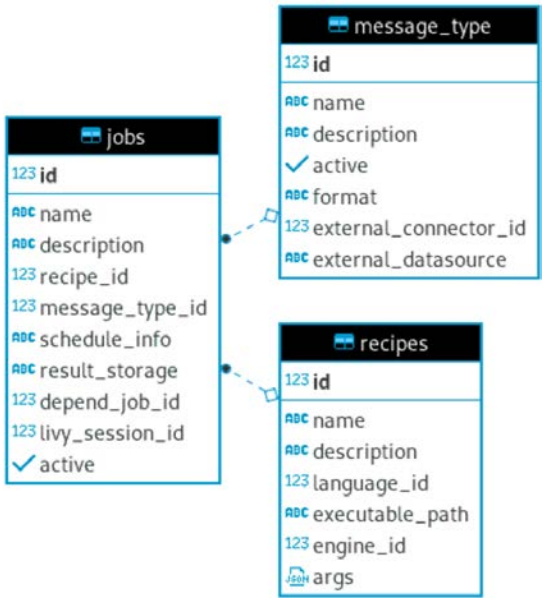


FIGURE 4. Schema of the application level meta-data.

Regarding the meta-data tables, since they contain limited volumes of information, they share the physical resources of the PostgreSQL deployment where the Entities tables are stored, implementing a separate, completely disjoint database schema. The module's REST API is implemented as a replicated Java-based service that runs on a Jetty HTTP Web server [22]. Multiple server instances are deployed in containers which are accessible using a proxy layer like HAProxy [23] bundled with Keepalived [24] to achieve high-availability and load balancing. We also use TLS encryption [25] and user authentication and authorization for enhancing the Web server security. More information on the deployment is offered in the relevant sub-section.

B. BIG DATA PROCESSING ENGINES

The Big Data Processing Engines of the BDA system are implemented within the Analytics and ML module which is responsible for performing any type of computations. Its main purpose is to calculate KPIs or perform predictive tasks by executing the corresponding applications, using the proper data. As already mentioned, when performing a computation task, the BDA essentially executes a *recipe* that implements a specific analytics or ML workload. The input for these workloads is stored in the Data Storage Engines. The recipes are provided as executable files that consume data consistent with a specified data schema, so the module is generally agnostic to the input data schema and location. A recipe requires the definition of the implemented algorithm's programming language, the corresponding binary executable location, the input data schema specification, and

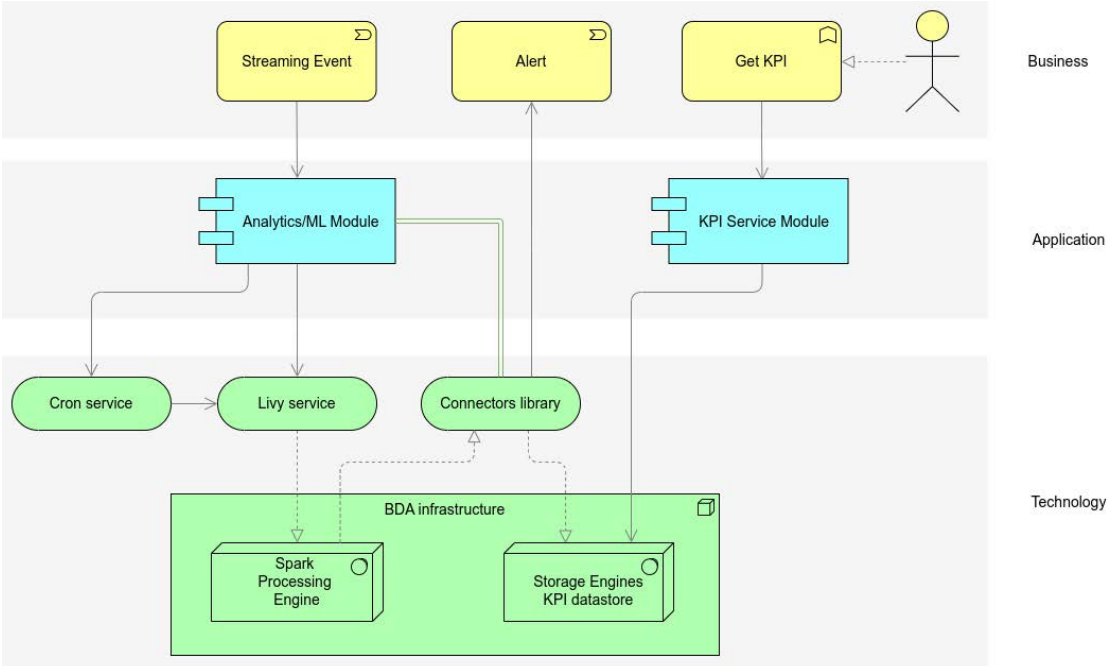


FIGURE 5. The event processing workflow from high to low-level components.

other possible user-defined arguments for the configuration of its execution environment.

Based on this design, we formulate two types of processing workloads, which we call "*jobs*": a) message-triggered jobs, used to make real-time predictions/calculations, and b) periodic jobs that will be used to (re)train an ML model or calculate aggregating statistics, cummulative KPIs, or group of KPIs over specified time windows (in Figure 10 we can see some examples of such queries). We define a job by specifying the execution pattern for a recipe: either defining the message type that triggers its execution or a time interval for its periodic execution. Multiple jobs can be scheduled with the same recipe.

Message-triggered jobs will be executed upon a message arrival of the specified message type, consuming that particular message, while periodic jobs will be executed when their specified time interval expires. Moreover, when a job is defined, the output can be forwarded to either of two data sinks. It will be either be stored in the appropriate Data Storage Engine, according to a well-predefined schema (making use of the Datastore or KPI service modules), or published as a message to a predefined Pub/Sub topic. Multiple jobs can be created with different execution parameters and output locations using the same recipe. These complex relations between messages, and workloads are on display in Figure 4, where the schema of the meta-data indicates the one-to-many connection between the jobs and recipes tables.

Implementation Details: The Analytics and ML module offers a Java-based interface used to define recipes and schedule the execution of a job and can connect to different execution engines using the appropriate integrated connector, to run the required executables. Such computations are performed under the job configuration provided to this module. A software library is also implemented inside the module that contains the appropriate connectors for the Storage Engines and the Pub-Sub which are provided by the module to the Execution Engines to handle the recipes' input and output data.

In general, a recipe that includes a ML algorithm will calculate (infer) some predicted variables using open-source ML libraries that will build and query respective models. The calculated variables are related to the current state and will then be published as new events to the Pub-Sub for all subscribed sub-systems to consume. Additionally, a model instance can be (re)trained provided the appropriate historical input data. The updated model can then be used to make real-time predictions consuming the incoming messages. The Analytics and ML module will indicate, before launching an ML Recipe, whether a model will be created/updated or which trained model will be used for inference.

The main execution environment of the BDA is built upon on Apache Spark [12]. It is implemented through the deployment of a containerized Apache Spark cluster that offers both offline and online processing (using interactive sessions) and out-of-the-box scalability and high-availability features which are supported by Zookeeper. Spark supports multiple programming interfaces (eg. python, Java and scala) and it also provides a standard SQL interface where analytics tasks can be programmed. We also support execution environments such as plain python, or Java as evidenced

in Section IV where the codebase developed is written in python.

In order to exploit Spark's interactive sessions for fast streaming calculations, the BDA uses Apache Livy [26], an Apache incubator project that provides a REST interface to facilitate programmatic and fault-tolerant submission of both interactive and batch Spark jobs. By configuring the Analytics and ML module to use the Livy service, we are able to easily launch a different live session for every streaming (message-triggered) job. Moreover, the jobs that are part of a complex DAG-shaped workflow are launched on the same live session, one after the other. Dependencies between jobs are stored as meta-data in the Data Storage Engines. Finally, we use Hadoop YARN [27] as a cluster manager for Apache Spark. The processing workflow starting from high to low-level components, is presented in Figure 5. The Streaming Event that appears in Figure 5 is received by the BDA and triggers the execution of a job using the Livy service. Batch jobs can also be periodically launched by the Cron [28] service. A recipe running in Spark can access the Data Storage Engines to use them as data sources. Processing results can be stored in the KPI datastore or directly broadcast through the Pub/Sub.

C. KPI DATASTORE AND SERVICE

The KPI datastore and service are utilized to store the results of the analytics workloads executed by the Processing Engines. Analytics workloads are often used to calculate KPIs - which explains the name of this module. However, despite what the name might suggest, the KPI datastore is designed to be a full-fledged storage service which can support every kind of data - not merely KPIs - both in terms of format and size. The KPI datastore and service was initially designed as a supportive component to the BDA, in the sense that was not considered as a part of its core functionality. However, the need to offer a persistent storage service for the outcomes of the calculations performed made us review that approach. In its current implementation, the KPI datastore is designed to use a classic RDBM system as its storage backbone, sharing the same infrastructure with the Entities Tables.

Every time a data-processing job is executed, we expect a result to be generated. The outcome of a processing job can either be a scalar or integer number, a matrix, or even a more complex data structure of higher level. Regardless of the data structure it might consist of, the output of all processing workloads defined in the BDA is persisted in a dedicated datastore for future reference and quick retrieval. More specifically, the output of processing workloads is wrapped inside a JSON object and pushed into the appropriate database table using a library specifically implemented for that purpose. The current implementation only allows for the storage of serializable data types, but we plan to extend this functionality to support any type of data - including binary. Each job instance is given a uniquely identifying index upon creation. It is important to point out that the same recipe can be used to implement multiple jobs. An application that calculates the averages of some metrics of interest could, for example, be utilized for a job that calculates daily averages (executed every day) and another that calculates weekly averages (executed once per week). Workload results are stored in a dedicated table per job. Result tables are automatically created using the recipe and job indices and contain information such as the data of the message that triggered the job execution (in case of streaming processing), the result, and the timestamp that corresponds to the moment the result is pushed into the database.

D. CONNECTIVITY INTERFACES

The Connectivity Interfaces of the BDA are implemented by the Controller and Connector modules respectively. The Controller module co-ordinates and orchestrates the operation of all other BDA subsystem's modules, handles incoming messages and acts as a job scheduler. It has four important functionalities:

(a) It manages the Pub/Sub subscriptions that need to be created by the Connector module in order for the BDA to receive messages.
(b) It is also responsible for forwarding these messages to the Datastore when they are received (data ingestion).
(c) It acts as a scheduler that can trigger recipes execution based on the type of incoming messages (streaming execution) or in a periodic basis (batch execution).
(d) It secures the REST APIs of all modules.

This module interacts with every other internal BDA module. Additionally, other external components can interact with it through its REST API, secured by the Identity and Access Management component. The Connector module acts as the data 'gateway' for the BDA sub-system as it receives messages from the Pub/Sub component and forwards them to the Controller to handle them, i.e., forward them to the Datastore for storage and launch any message-triggered jobs.

In particular, the Controller orchestrates the Pub/Sub subscriptions that need to be created according to the defined message types and forwards them to the Connector module. The Connector module then connects to a Pub/Sub server, subscribes to these topics and once it receives a message, forwards it back to the Controller which interacts with the Datastore module to store the event-related messages. The Controller therefore has no direct interaction with the Pub/Sub and only directs the traffic of the data that can flow into the BDA.

The Controller can also be configured initially (and later on during the BDA's life cycle) to execute specific Analytics or Machine Learning Recipes either in a batch or in a streaming manner. To perform a specific KPI or ML calculation, the corresponding Analytics or ML job is explicitly launched by the controller. In the case of periodical batch processing, it automatically launches the job every time the selected period expires. In the case of stream processing, the execution of a specific job is triggered upon the arrival of a specific type

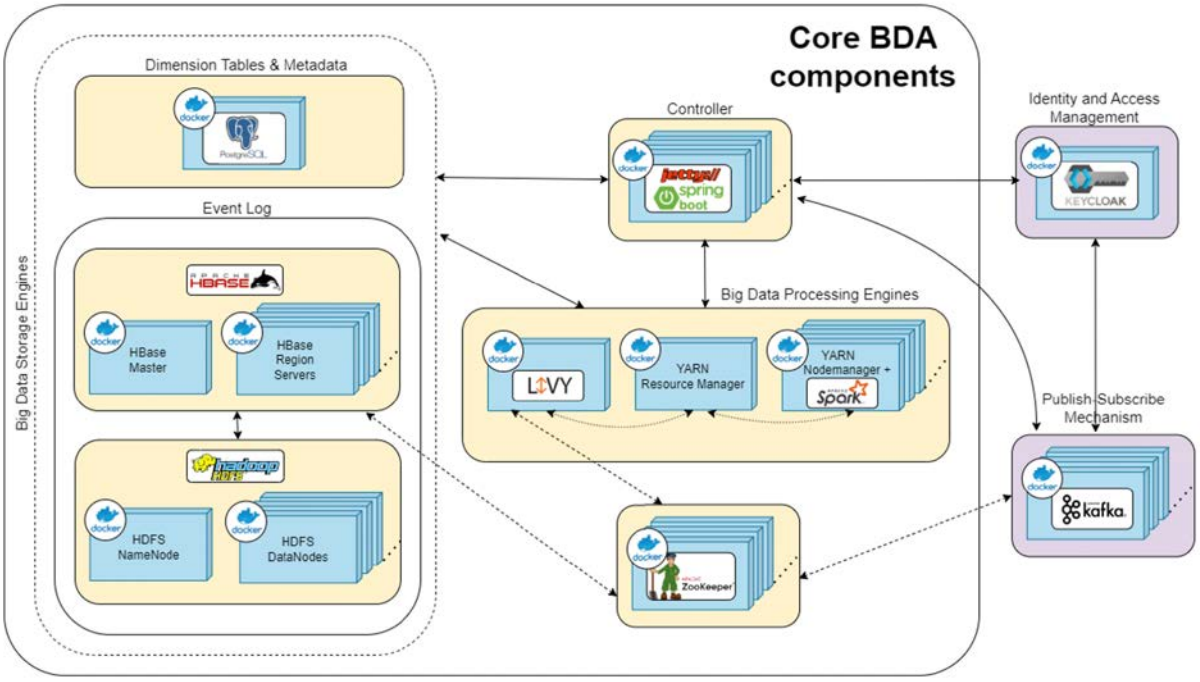


FIGURE 6. Containerization of the BDA.

of message. The configuration that will be used in each case is retrieved from the meta-data tables described in the Data Storage Engines sub-section.

The Connector is configured by the Controller to connect with the Pub/Sub and specifically, subscribe to topics corresponding to the documented message types. Upon receiving a message through the Pub/Sub, it then parses the incoming message and ingests the information stored in it after having validated its format against an XML schema that is declared during the message type definition process. Finally, it forwards the message to the Controller module.

Implementation Details: The Controller module uses all other modules of the BDA to facilitate the main functionalities of the sub-system (data storage and processing). It is implemented as a replicated Java-based service that runs in a Jetty HTTP Web Server [22], deployed in multiple container instances. These instances are accessible using a proxy layer to achieve both load balancing and scalability as well as high-availability. We use authentication and authorization mechanisms for every action that is submitted to the Controller through the REST API, as well as TLS encryption for the input and output data.

E. DEPLOYMENT

In Figure 6 we present a deployment diagram for the BDA and accompanying components such as the Pub/Sub and Identity and Access Management (IAM) service. Blue boxes represent containers hosting a service, while stacked blue boxes represent multiple containers (based on the same container image) that are part of a distributed service/system which can be scaled horizontally by adding more instances of the same worker. This design was implemented with the purpose of providing both scalability and fault-tolerance for all of the included software components of the BDA.

The Storage Engines utilize the Apache HBase [14] database to store the Event Log. PostgreSQL is also used to implement the Entity Tables, KPI data store, as well as store meta-data. An underlying Hadoop Distributed File System (HDFS) [13] installation supports HBase and the Apache Spark cluster, which implements Data Processing. These are distributed services that can horizontally scale. The same applies to the component implementing the Publish/Subscribe message exchange mechanism, based on Apache Kafka. We employ Zookeeper [29] to help with the management and configuration integrity of our distributed services. Finally, the Controller module, which implements the connectivity and interactions between the BDA sub-modules and external services such as the IAM and Pub/Sub can be launched in multiple instances to achieve higher throughput, if necessary. Deployment scripts for the infrastructure have been made open-source and are available online.[5]

III. ANALYTICS USING GEO-TEMPORAL DATA

After having presented the infrastructure, we move on to discuss some characteristic use cases which we tackled in the context of the project. This section will cover analytics workloads that process geo-temporal data provided by the

[5]https://github.com/iwnet/digitalization-infrastructure

automatic identification system (AIS) [30], a short-range coastal tracking system currently used on ships developed to provide identification and positioning information to both vessels and shore stations. We have identified two distinct use cases where analytical workloads can be utilised in order to extract interesting information from raw data, which will be presented in detail in the next paragraphs.

A. DATA DESCRIPTION

Before we discuss the specifics of each use case, we consider it's important to highlight the properties and characteristics of the data-set based on which this analysis has been designed and implemented. The Automatic Identification System (AIS) [30] is a maritime communication system that allows ships to exchange information with other vessels and shore-based facilities. AIS works by transmitting and receiving data over VHF radio frequencies and is used to provide vessel identification, position, speed, course, and other information that can be useful for navigation, collision avoidance, and maritime safety.

Our data set consists of an anonymized collection of AIS messages dispatched by vessels active in the inland corridor of River Weser. The data-set covers two calendar years - namely 2018 and 2019. The anonymization process has ensured that identification numbers (MMSI) for all vessels have been altered in a way that cannot be reversed - but maintains consistency by allocating the same altered identification number for every data point referring to the same vessel.

B. DISCOVERY OF AREAS OF INTEREST

1) PROBLEM DESCRIPTION

The first use case is the discovery of areas of interest in Inland Waterway networks using clustering techniques. In general, networks similar to those we can observe in inland waterways can be modelled as graphs, where edges can be mapped to navigable paths between stops, and nodes can be mapped to stops. In the inland waterway context, such stops can be terminals, bridges, locks, waiting areas, or even congested parts of the network where traffic needs to be moderated. Regardless of their function, these stops, or *Areas of Interest* (AoI), are of critical importance for the operational aspect of the network. They often function as bottlenecks, as vessels typically have to spend significant time there while tasks such as (un)loading, waiting for water levels to be adjusted, etc. are performed [31]. The task at hand is to detect Areas of Interest in a network without prior knowledge of the infrastructure.

This process can prove to be useful in cases where networks or parts of a network are unmapped. It can help provide a list of geo-fenced areas as candidates for business experts to tag, in order to map the network. It can, alternatively, be utilised to detect parts of the network where traffic is slowed down, therefore causing a degradation of the service for the entire network. This analytical processing workload was applied using the dataset described above, to locate AoIs

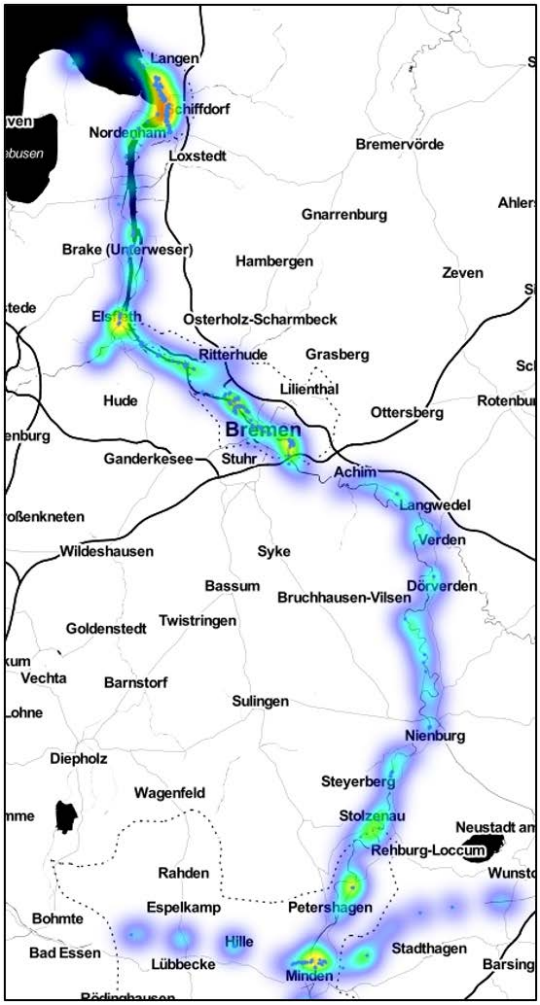


FIGURE 7. Heatmap of AIS messages from non-moving vessels.

in the corridor of River Weser. Our findings were shared with the authorities who are responsible for the maintenance and operation of these infrastructures and were also used to discover patterns of vessel traffic and calculate KPIs regarding the network performance as described in III-C. The results of this analysis were used for the design and validation of realistic simulations of the IWT network traffic, which in turn can allow operators to optimize the performance and the quality of their services.

The main concept, upon which our solution is built, is the observation that AoIs are generally locations where vessels come to a stop. Thus, the problem is equivalent to that of locating areas inside of which, instances of messages have been sent reporting zero speed. As we expect vessels to have spent significant time within these AoIs, it is natural to receive an extremely high number of AIS messages that has originated from non-moving vessels operating there. As a result, in order to discover the potential Areas of Interest in the network, we search the data-set for clusters of data points that correspond to AIS messages reporting zero speed. In Figure 7 we present a heatmap of all the AIS messages

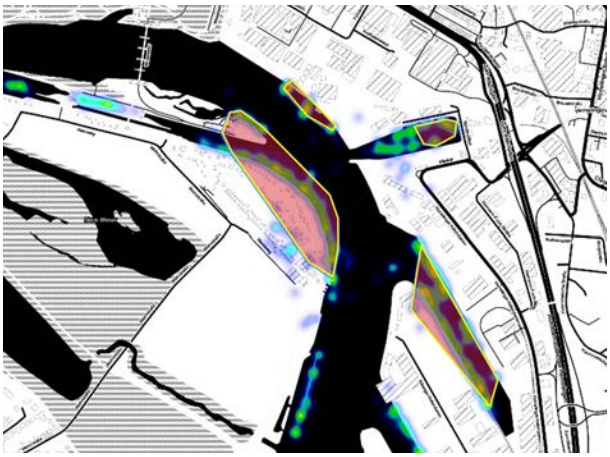


FIGURE 8. Output of the automated discovery of areas of interest in inland waterway networks using clustering - the clusters identified in bright yellow frames.

corresponding to non-moving vessels. It is obvious that certain parts of the corridor contain many more than others. However, the clustering problem does not have an equally obvious solution.

2) SUGGESTED SOLUTION

Taking into consideration the characteristics of the problem at hand, where the number and shape of clusters is a priori unknown, using the DBSCAN algorithm is a choice that makes much sense. DBSCAN [32] (Density-Based Spatial Clustering of Applications with Noise) is a clustering algorithm that groups data points that are closely packed together in high-density regions. The algorithm works by first selecting a random unvisited data point, and then finding all of the neighbouring points within a specified radius. If the number of neighbours exceeds a specified threshold, the data point is labelled as a core point. Otherwise, it is labelled as a border point. Once all core and border points have been identified, the algorithm proceeds to form clusters by connecting core points with their nearby neighbours. Data points that are not assigned to any cluster are labelled as noise.

Given the fact that a containerised Apache Spark cluster backs the infrastructure we used in the context of the IW-NET project, we selected to use GEOSCAN [33] to implement the DBSCAN clustering algorithm. GEOSCAN is an open-source implementation of DBSCAN created and maintained by Databricks, the company that commercializes Apache Spark. GEOSCAN leverages the H3 [34] geospatial indexing system and the GraphX [35] graph processing framework, built on top of Spark. The H3 geospatial indexing system is a hierarchical hexagonal grid system developed by Uber Technologies. It provides a way to divide the Earth's surface into hexagonal cells of different resolutions, allowing for efficient indexing, analysis, and processing of geospatial data, whereas GraphX is Apache Spark's built-in API for graphs and graph-parallel computation.

GEOSCAN [33] can perform efficient distributed processing. The core of the algorithm relies on GraphX to detect points having more than a minimum number of neighbours in a distance below a user-set threshold. The results of this analysis can be stored in the widely adopted GeoJSON format [36]. Figure 8 displays the output of this analytics task executed for terminal detection depicted on a map. The heatmap for locations where vessels have reported zero speed is also visible for reference.

C. LIVE DATA ANALYTICS USING AIS DATA STREAMS

1) PROBLEM DESCRIPTION

The purpose of this second scenario is to utilise a data-set describing an Inland Waterway network's Areas of Interest (AoI), combined with an incoming stream of AIS messages from vessels navigating it to generate live data analytics which can be helpful and provide visibility, transparency and insights on the operational level of the logistics business. A final processing stage can be executed on the resulting condensed logs for the calculation of KPIs and analytics in time windows selected by the users. Similarly with the previous use case, we have shared the results of our analysis with stakeholders in the business world who drove the implementation of high-level solutions to the issues which were revealed.

2) SUGGESTED SOLUTION

The main concept around which our proposed solution for the live data analytics is built, involves the creation and regular updates of condensed event logs in real-time. Having access to a live stream of AIS data allows us to track the location of vessels that sail in the network. However, according to the AIS communication protocol standards, vessels can produce an AIS message every few seconds. The rapid rate at which these messages arrive results in an enormous amount of data points which is challenging to process quickly and efficiently. To mitigate this effect, we introduce the concept of a condensed event log, per vessel - which we call vessel STATE LOG, and per AoI - which we call AoI ACTIVITY LOG.

Combining information about the vessels' locations with geo-fencing data that indicate Areas of Interest in the network enables us to create a vessel STATE LOG for each one of the vehicles, where we record entry and exit times into/from these Areas of Interest every time the vessels visit them. Moreover, we additionally create and regularly update a AoI ACTIVITY LOG for each one of the networks Areas of Interest, where all entries and exits the all vessels into/from the specific Point of Interest are registered. Table 1 contains the attributes of the STATE and ACTIVITY LOGS.

A motivating example of how our approach reduces the data size to be processed is presented with the help of Figure 9. We can see that after the vessel enters the AoI indicated by the blue overlay and frame, it sends numerous AIS before it exits its borders. For the needs of our analysis,

TABLE 1. Attributes of the (vessel) STATE LOG and the (AoI) ACTIVITY LOG.

Attribute	Description
`VesselID`	Unique Identifier corresponding to the vessel
`AoIID`	Unique Identifier corresponding to the AoI
`EntranceTimestamp`	Timestamp of the first message indicating the vessel with `ID = VesselID` entered the AoI with `ID = AoIID`
`ExitTimestamp`	Timestamp of the first message indicating the vessel with `ID = VesselID` exited the AoI with `ID = AoIID`

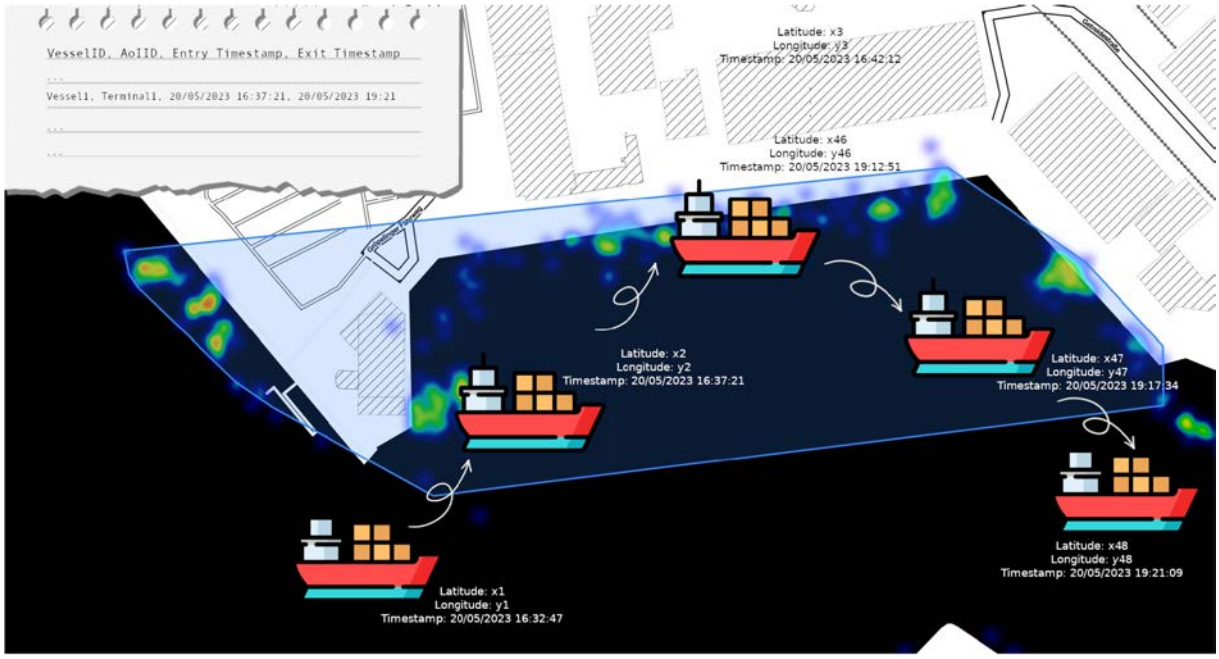


FIGURE 9. Attributes of the (vessel) STATE LOG and the (AoI) ACTIVITY LOG.

we can ignore any messages that do not cause a change of state for the vessel (i.e., in the context of our use case do not indicate it has left the area). Therefore, moving from the AIS data stream to the condensed Event stream that implements the STATE LOG, reduces our data-set from 48 rows (equal to the number of AIS messages sent by Vessel1) to only one containing the `VesselID`, `AoIID` and `EntranceTimestamp`/`ExitTimestamp`.

IV. WATER LEVEL PREDICTION USING NEURAL NETWORKS

This section will delve into forecasting water levels for a few stations in the Danube River using machine learning techniques. The impact of the water levels on the navigability and efficiency of Inland Waterway networks is a topic well-documented in existing literature [37], [38]. Christodoulou et al. [38] discuss the impact of water levels on the capacity of vessels with respect to their classification and size. At the same time, significant effort is being made to create tools that will allow captains and inland waterway vessel crews to take information such as water levels into account in their trip planning process, as evidenced in the work by Prandtstetter et al. [39]. The importance of being able to forecast the water levels in waterway corridors cannot be overstated, as it heavily influences the capacity of the vessels and therefore the planning and operational stages of the logistics business. In our case, our objective is to provide forecasts as accurate as possible for an automated decision-making process to take into consideration when suggesting optimal routing in the planning/scheduling phase.

Our investigation involves the evaluation of multi-day ahead forecasts, as well as daily predictions. By integrating weather data into our predictive models, we aim to uncover patterns and trends that contribute to more nuanced and precise predictions. Although employed in the current setting as a proof-of-concept, the main principles of our approach can be generalized and utilized in any similar scenario. The source code for this work can be found at GitHub.[6]

A. DATA COLLECTION

The training data for this study was derived from daily measurements of water levels obtained from three distinct stations situated in Austria, namely Kienstock, Pfelling, and Wildungsmauer. This data is publicly available online through the respective River Information Systems (RIS)

[6]https://github.com/iwnet/water-level-forecasting

View	Query	SQL
Vessel	Time spent in any specific AoI per visit (in hours) (last 30 days window)	SELECT AoIID, AVG(DATEDIFF(ExitTimestamp, EntranceTimestamp))/24 AS AverageTimeInAoI FROM STATELog WHERE EntranceTimestamp >= DATE_SUB(current_date(), 30) GROUP BY AoIID;
	Average number of visits to any specific AoI per week (last 30 days window)	SELECT AoIID, COUNT(*) / 4 AS AverageVisitsPerWeek FROM STATELog WHERE EntranceTimestamp >= DATE_SUB(current_date(), 30) GROUP BY AoIID;
Area of Interest	Average total weekly number of vessel visits by any vessel (last 30 days window)	SELECT (COUNT(*) / 4) AS AvgVisitsPerWeek FROM ACTIVITYLog WHERE EntranceTimestamp >= DATE_SUB (current_date(), 30);
	Total number of vessel visits for the current week	SELECT COUNT(*) AS TotalVisitsToAoIX FROM ACTIVITYLog WHERE ExitTimestamp >= DATE_SUB(current_date(), 7);

FIGURE 10. Queries.

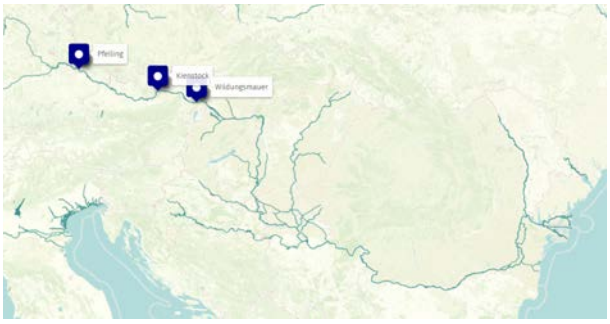


FIGURE 11. Locations of the 3 measurement stations.

TABLE 2. Dataset statistics.

Stations	Mean	Standard Deviation	Variance
Kienstock	312.58	87.54	7663.42
Pfelling	223.32	78.55	6171.16
WM	284.02	83.65	6998.46
Kruskal-Wallis test			
Test Statistic	823.28	P-value	1.68e-179

portals. The temporal scope of the data-set encompasses a four-year period spanning from 2017 to 2020. Table 2 demonstrates a statistical analysis and comparison of the data from each station. We present mean, standard deviation and variance for each time series and in addition we include the results of the Kruskal-Wallis [40] test to examine any similarities between the datasets.

We use the data from each station separately, focusing on predicting values for a given station rather than training models to be used across all different locations. This approach helps keep the complexity of each model low and make more accurate predictions at the same time. In addition, we do not train the models using data across the four-year span of the whole data-set, but we create time frames of 6-month length each. Each time frame is used as a training set and the 15 days following the training set are used as a test set, meaning a ratio of approximately 90% for training and 10% for testing, which is consistent with the literature and prior work on predicting water levels [41].

Daily weather data was collected for each of the aforementioned stations. We collected daily measurements for various weather metrics using the services of Visual Crossing [42]. Daily weather data served as pivotal future co-variates in our predictive models and specifically precipitation coverage emerged as a key focus, acknowledging its influence on water level dynamics. For each training set, the weather data exhibited a temporal length, same as the same time frame as the water level measurements. Additionally, an extra buffer of 15 days was appended to the weather data, aligning with the duration for which predictions were sought. In practical application scenarios, these additional 15 days would be supplied as expected weather predictions.

B. MODEL TRAINING

The prediction of water levels involves a temporal aspect, where each data point corresponds to a specific moment in time. To address the sequential nature of this task,

TABLE 3. Model selection.

Model type	*LSTM, RNN, GRU*
Hidden dimensions	*32, 128, 1024, 2048*
N° layers	*1, 2*
Input length	*10, 15, 21, 32*
Output length	*1, 5, 10*

TABLE 4. Models evaluated for each station.

Station	Model configurations				
	Model Type	***Hidden Dimension***	***N° layers***	Input Length	***Output Length***
Kienstock	GRU	32/2048	1	10/15	1/5
Pfelling	GRU	128/2048	1	10/15	1/5
WM	GRU	32/2048	1	10/15	1/5

recurrent neural networks (RNNs) are generally considered suitable [43]. Specifically, long short-term memory networks (LSTMs) [44], standard RNNs [45], and gated recurrent units (GRUs) [46] were identified as potential candidates.

We conducted a comprehensive exploration of various model configurations to optimize predictive performance. Table 3 provides an overview of the diverse configurations we experimented with, encompassing different architectures and hyper-parameter settings. Our experimentation involved the manipulation of key parameters, including the number of hidden cells, the architecture's depth (single or double layers), and variations in input/output sequence lengths.

Our models were trained using the Darts [47] library. Darts is a Python library for user-friendly forecasting and anomaly detection on time series. It provides many statistical models as well as common machine learning models, such as the ones we used in our work (LSTMs, GRUs and RNNs). Darts simplifies the training process by simply requiring one, or more, time series which are used to train models and make forecasts. The time series are distinguished into two different categories: the *target series*, which are the series we want to forecast, and the *co-variate series*, which can potentially help forecast the target. In our work the daily water level measurements are the target series and the weather data are the co-variate ones. Darts' models are initialized with an `input_chunk_length`, the input length provided to the model, and an `output_chunk_length`, used by the library to calculate the training loss and adjust the model's parameters. As shown on Table 3 we experiment with various input sizes and we conclude that models forecast best when using the last 10 to 15 days.

During training, we pass to Darts a single target time-series, the 6-month timeframe we are currently training for, and it automatically creates training samples based on the `input_chunk_length` and `output_chunk_length`. The Darts library is responsible for creating the training data-set from a single time series input, provided an input and output length. There are several possible ways to slice the series to produce training samples, and Darts provides a few data-sets. We use the default `Sequentialdata-set`, which simply builds all the consecutive pairs of input/output sub-series (of lengths `input_chunk_length` and `output_chunk_length`) existing in the series, as shown in Figure 13.

An essential consideration in our approach is the efficient use of computational resources. Despite employing a basic CPU processor, the training of our models is swift. This expedited training is attributable to the relatively modest size and simplicity of the models we employ. Consequently, there is no imperative need for GPU acceleration. This efficiency is particularly advantageous, given our recurrent need to re-train models regularly to accommodate the latest data and make predictions for future periods. The learning rate for training is chosen using PyTorch Lightning's *Tuner.lr_find* which performs a range test of good initial learning rates, to reduce the amount of guesswork, and we choose Mean Squared Error (MSE) as the loss function.

C. RESULTS

To ascertain the models/configurations that excel in forecasting, the Root Mean Squared Error (RMSE) function is employed. We train every configuration across various (but not all) time windows. The RMSE is then calculated and models with the lowest cumulative RMSE across all time windows are identified as the most effective. Table 4 presents a summary, delineating the optimal model configurations for each station based on their superior performance in minimizing RMSE. In this section, we will focus on these models and their evaluation.

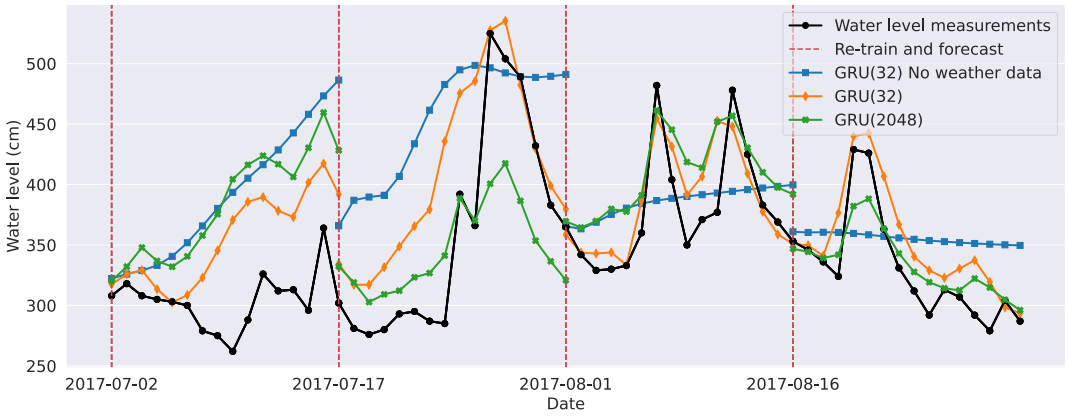
After selecting the best-performing models, we retrain them to all consecutive time windows, across the history data for each station, in order to make predictions for the whole data-set. We consider two primary approaches to comprehensively gauge the models' performance: multi-day ahead forecasting and sequential one-day forecasting with iterative updates.

Moreover, we compare the performance of each model, when removing the weather co-variate, evaluating the impact of incorporating weather data into our models.

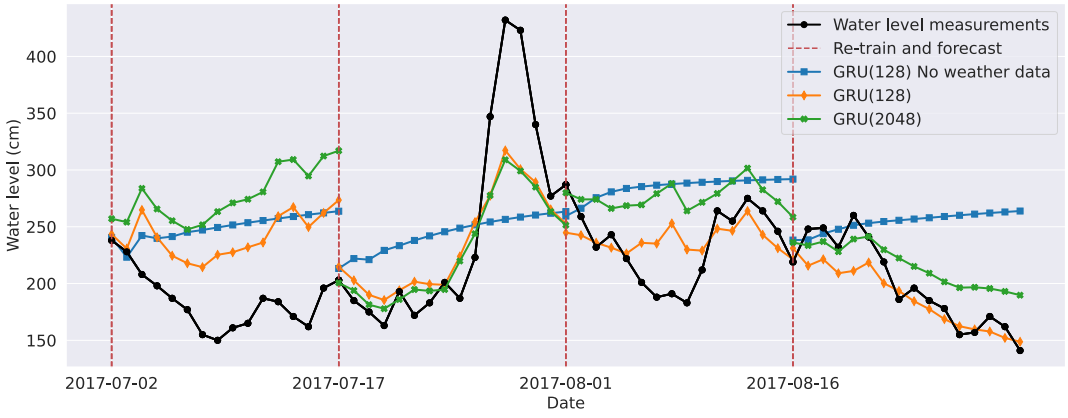
1) MULTI-DAY AHEAD FORECASTING

In the first evaluation scenario, the models are challenged to predict water levels multiple days into the future. This method is representative of a real-world application where users may require an extended forecast to inform decision-making processes. To conduct this evaluation, the trained models are provided with the whole 6-month historical data, which was used to train them, and their performance is then measured against the true water levels for each subsequent day over the forecast horizon.

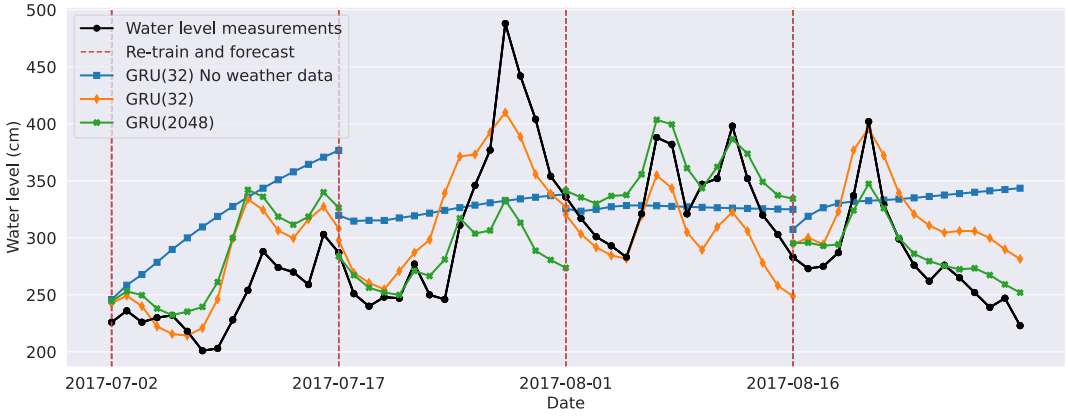
It is evident that the models exhibit commendable performance in capturing the overall trend of river water levels. Despite occasional challenges in predicting sudden spikes, the models consistently showcase proficiency in forecasting fluctuations even up to a 16-day horizon. This implies a robust



(a) Kienstock



(b) Pfelling



(c) Wildungsmauer

FIGURE 12. **Multi day forecasts.**

ability to provide stakeholders with valuable insights into the general trajectory of river water levels over an extended period.

In Fig. 12, we showcase results for each station for a few consecutive time windows. Each vertical dotted line represents a point in time where we re-trained the models in the latest 6-month period before forecasting the following 15-day window. In addition, Figures 15 display violin plots which summarise the absolute error of each prediction day across all successive time windows. Finally, tables 6, 5 and 7 showcase the average RRMSE, RMSE, MAE and SMAPE, of each model for long forecasting, for each station, for the most promising model configurations. In these tables we also include two additional forecasting methods, two linear

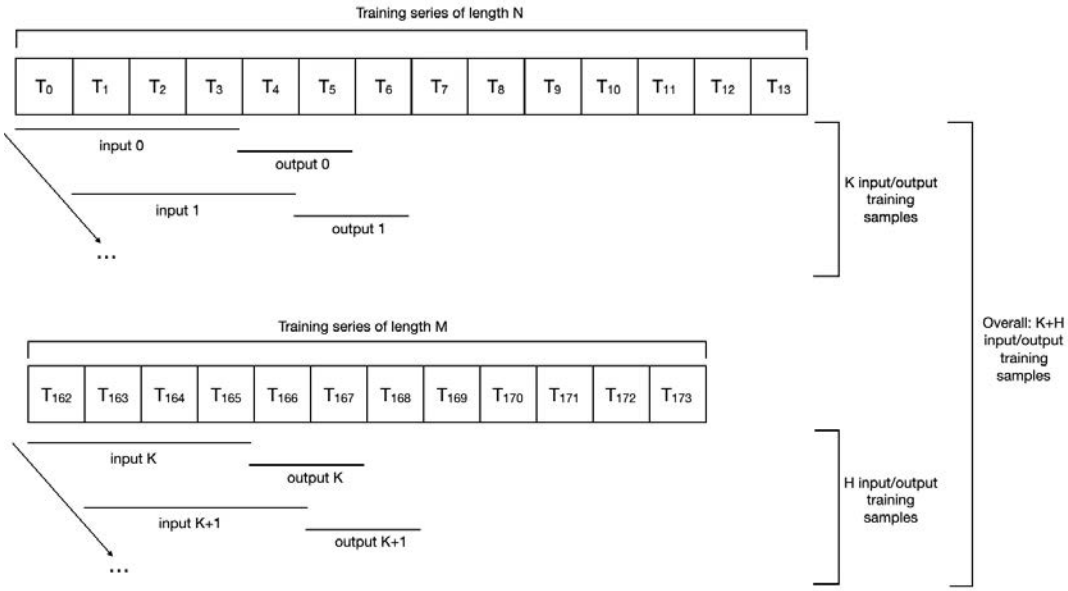


FIGURE 13. The slicing of two target time series to produce some input/output training samples, in the case of a Sequential data-set (no co-variate in this example).

regression models (with different input sizes) and a naive forecasting approach which simple forecasts the last available measurement.

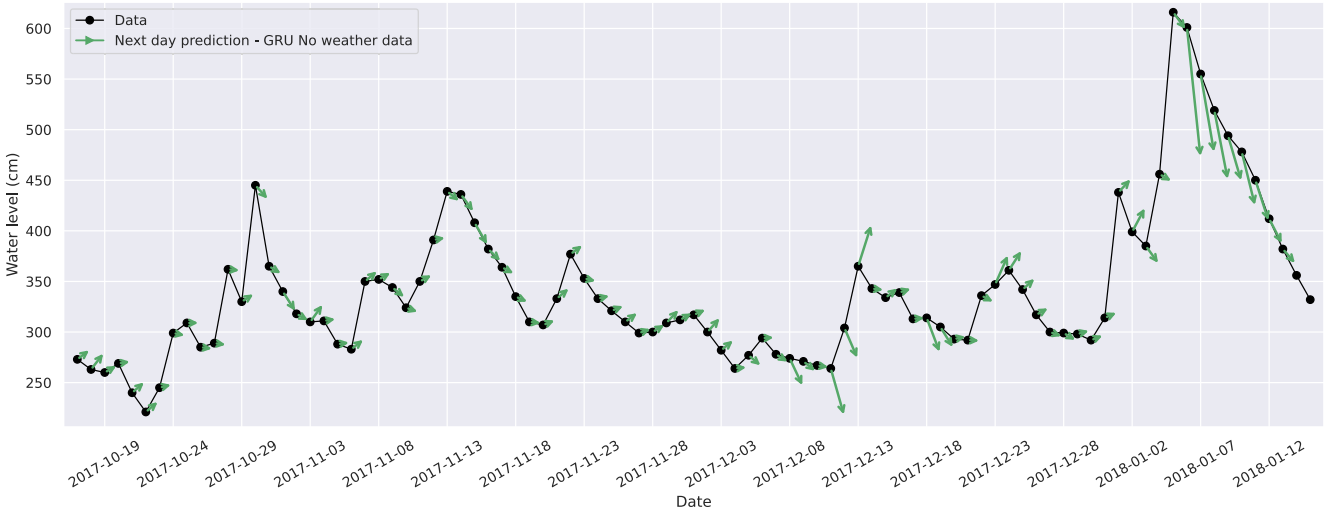
2) SEQUENTIAL ONE-DAY FORECASTING WITH ITERATIVE UPDATES

In the second evaluation approach, the models undergo scrutiny regarding their proficiency in delivering precise daily predictions. This evaluation method involves augmenting the input data with the actual value of each day, before forecasting the next one, mimicking a realistic forecasting scenario. This iterative process mirrors the dynamic nature of real-time data, progressively providing the models with the latest data. At each step, a prediction is computed, revealing the evolving accuracy of the model as it adapts to the fluctuating conditions of river water levels. Note that we still re-train the model every 15 days with the latest 6-month time window. For this evaluation, we prefer to train using an output length size of 1, since we only want the model to focus on the next day forecast.

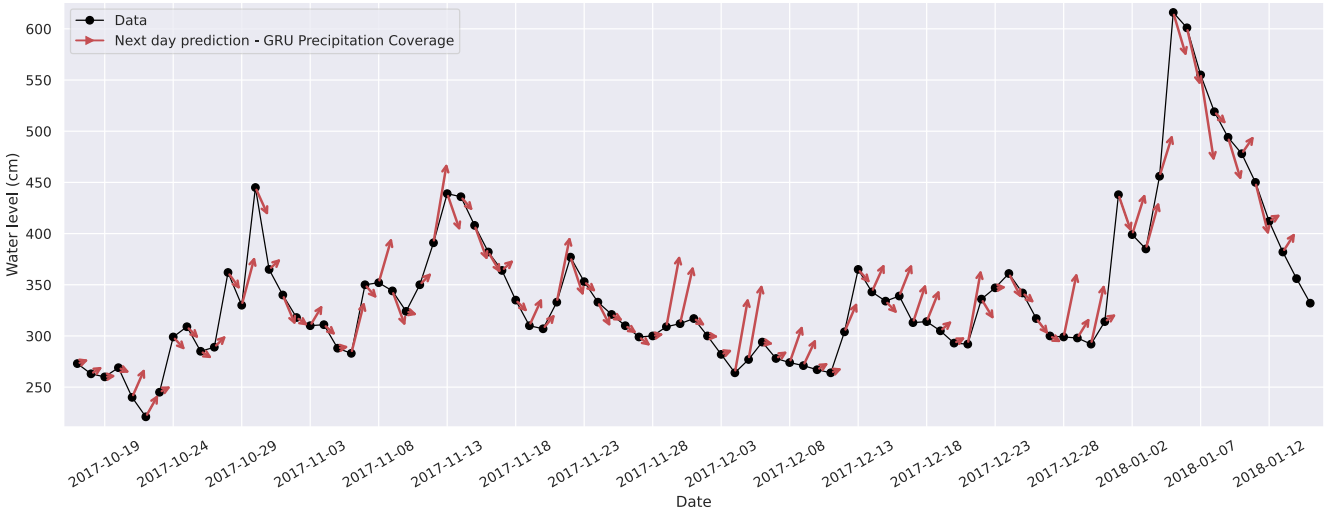
This approach unveils the models' heightened ability to generate more refined predictions when granted the opportunity to dynamically adjust to evolving data. The iterative incorporation of true values from subsequent days enables the models to responsively adapt to changing river conditions, resulting in predictions that closely align with observed values. Consequently, the day-to-day forecasting method emerges as a notably more reliable and accurate approach for short-term predictions, showcasing the models' adaptability and precision in capturing the intricacies of daily variations in river water levels. Tables 8, 9 and 10 showcase the average RRMSE, RMSE, MAE and SMAPE of each model for daily forecasts, for each station. Similarly to the long forecasting case, we include two additional forecasting methods, one using a linear regression model and one naive

TABLE 5. Models' performance for Pfelling, long 15-day forecasting.

Model	RRMSE	RMSE	MAE	SMAPE
GRU /w weather (dim:32,in:10,out:1)	19.55%	55.13	47.42	17.411%
GRU /w weather (dim:32,in:15,out:1)	19.73%	55.47	47.93	17.48%
GRU /w weather (dim:128,in:10,out:5)	19.98%	56.58	47.52	17.534%
GRU /w weather (dim:128,in:15,out:5)	20.11%	56.72	48.73	18.008%
Linear Reg. (in:15,out:1)	21.12%	58.12	48.47	17.533%
GRU /w weather (dim:32,in:10,out:5)	20.93%	58.49	51.15	18.88%
Last value (naive)	21.24%	58.54	48.45	17.164%
GRU /w weather (dim:32,in:15,out:5)	20.48%	58.59	50.33	18.377%
Linear Reg. (in:10,out:1)	21.63%	60.59	51.03	18.506%
GRU /w weather (dim:128,in:10,out:1)	21.47%	60.84	52.4	19.349%
GRU /w weather (dim:128,in:15,out:1)	21.56%	61.35	53.59	19.345%
GRU (dim:128,in:15,out:1)	23.5%	65.52	55.91	20.215%
GRU (dim:32,in:10,out:1)	23.74%	66.88	57.68	20.926%
GRU (dim:128,in:10,out:1)	24.43%	68.09	58.8	21.588%
GRU (dim:32,in:15,out:5)	24.34%	69.46	60.64	21.936%
GRU (dim:128,in:15,out:5)	24.96%	71.16	61.81	22.543%
GRU (dim:128,in:10,out:5)	25.71%	72.45	63.04	23.102%
GRU (dim:32,in:15,out:1)	26.61%	74.37	64.92	23.213%
GRU (dim:32,in:10,out:5)	27.07%	75.33	65.02	23.578%

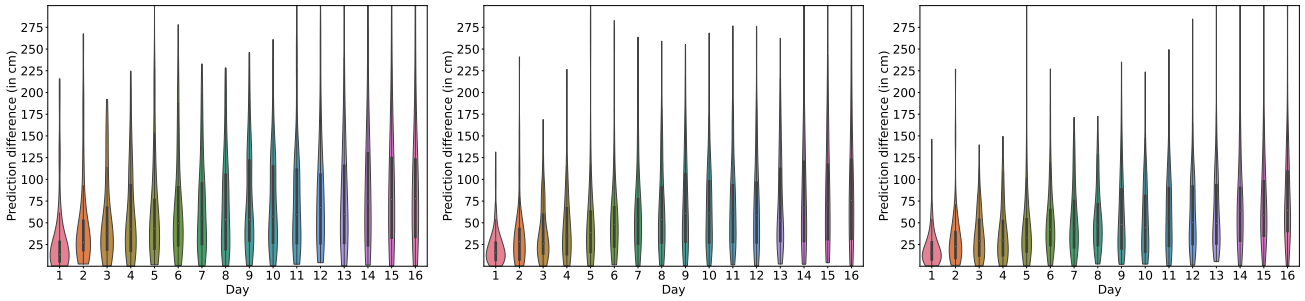


(a) GRU without Precipitation Coverage as co-variate



(b) GRU with Precipitation Coverage as co-variate

FIGURE 14. Next day forecasts: Predicting with the weather as co-variate vs training on water levels - Input is updated every day, the tip of each arrow represents the predicted value. Kienstock station.



(a) GRU(32) - no weather data (b) GRU(32) (c) GRU(2048)

FIGURE 15. Multi day forecasts - Distribution of RMSE for Each Day, Pfelling station.

approach, where we simply forecast the last measurement (no change) for the water level.

For stations Kienstock and Wildungsmauer we see a clear potential in forecasting water levels, especially when we incorporate weather data. While the quantitative metrics may not exhibit a significant difference, a qualitative gain is evident when examining graphical representations. For instance, in graph 14b, our model accurately predicts an

TABLE 6. Models' performance for Kienstock, long 15-day forecasting.

Model	RRMSE	RMSE	MAE	SMAPE
GRU /w weather (dim:32,in:10,out:5)	15.78%	53.66	44.38	12.577%
GRU /w weather (dim:128,in:10,out:1)	15.69%	53.86	45.36	12.953%
GRU (dim:32,in:10,out:1)	16.2%	55.73	47.13	13.471%
Linear Reg. (in:10,out:1)	16.06%	55.75	46.42	13.284%
GRU (dim:128,in:15,out:1)	16.22%	56.12	47.61	13.582%
GRU (dim:128,in:10,out:1)	16.46%	56.56	47.27	13.44%
GRU (dim:128,in:15,out:5)	16.32%	56.66	47.39	13.513%
Linear Reg. (in:15,out:1)	16.4%	56.91	47.97	13.728%
GRU (dim:32,in:15,out:1)	16.6%	57.23	48.83	14.042%
GRU (dim:32,in:15,out:5)	16.75%	57.72	48.88	14.003%
GRU /w weather (dim:128,in:15,out:5)	17.1%	58.76	49.18	14.201%
GRU (dim:128,in:10,out:5)	17.28%	58.98	50.23	14.248%
GRU (dim:32,in:10,out:5)	17.26%	59.1	49.57	14.093%
GRU /w weather (dim:32,in:15,out:1)	17.89%	60.97	50.53	14.312%
GRU /w weather (dim:32,in:15,out:5)	18.67%	62.04	52.67	14.905%
Last value (naive)	18.02%	63.42	52.48	14.749%
GRU /w weather (dim:128,in:10,out:5)	18.93%	64.11	54.81	15.58%
GRU /w weather (dim:128,in:15,out:1)	19.45%	64.98	54.67	16.528%
GRU /w weather (dim:32,in:10,out:1)	20.29%	68.61	59.31	16.528%

TABLE 7. Models' performance for Wildungsmauer, long 15-day forecasting.

Model	RRMSE	RMSE	MAE	SMAPE
GRU (dim:128,in:15,out:1)	16.84%	51.11	42.6	14.032%
GRU (dim:32,in:15,out:5)	17.09%	51.54	42.77	14.087%
GRU /w weather (dim:128,in:15,out:5)	17.04%	51.55	43.47	14.24%
GRU (dim:128,in:10,out:1)	17.07%	51.7	42.84	14.11%
Linear Reg. (in:10,out:1)	17.12%	52.16	43.4	14.291%
GRU /w weather (dim:128,in:10,out:5)	17.55%	52.25	43.99	14.496%
GRU (dim:32,in:10,out:5)	17.43%	52.36	43.74	14.398%
GRU /w weather (dim:128,in:10,out:1)	17.53%	52.73	44.93	14.701%
GRU (dim:128,in:10,out:5)	17.51%	52.84	44.06	14.493%
GRU (dim:32,in:15,out:1)	17.63%	53.02	43.83	14.396%
GRU /w weather (dim:32,in:15,out:1)	17.68%	53.31	46.37	15.281%
GRU /w weather (dim:32,in:10,out:1)	17.83%	53.86	45.33	14.975%
GRU (dim:128,in:15,out:5)	17.95%	54.17	45.71	14.8%
Linear Reg. (in:15,out:1)	17.89%	54.36	45.55	15.011%
GRU /w weather (dim:128,in:15,out:1)	18.18%	54.41	44.84	14.807%
GRU /w weather (dim:32,in:10,out:5)	18.11%	55.15	46.13	14.973%
Last value (naive)	18.18%	55.54	46.34	15.175%
GRU (dim:32,in:10,out:1)	18.68%	55.64	47.17	15.409%
GRU /w weather (dim:32,in:15,out:5)	21.03%	62.84	52.66	16.482%

upcoming increase in water levels (from 2017-12-03 to 2017-12-10), contrasting with a naive model that might seemingly outperform in terms of RMSE. Notably, Pfelling station, a location where linear regression models and naive approaches excel, exhibits the lowest mean and standard deviation, as highlighted in Table 2. This observation justifies why a simplistic approach, such as forecasting the same values each day, could yield better quantitative performance, despite the nuanced insights provided by our model in capturing dynamic patterns.

In Figure 14 we display the performance of daily forecasting; for each daily measurement, we provide a prediction for the next day. Each arrow represents a daily prediction; the base of the arrow is the last measurement provided to the model and the tip is the forecast for the next day.

D. RELATED WORK

The topic of utilizing statistical analytics, machine learning and artificial intelligence techniques to make predictions regarding the evolution of natural phenomena has been common in literature ever since researchers have had large enough data describing them, as well as the processing power to analyze them, readily available. In contrast to our approach, most studies aim at forecasting extreme events which can cause natural disasters due to overflow [48].

In contrast to more traditional approaches, where the physical systems had to be modelled in great detail, these data-driven approaches have allowed researchers to generate accurate predictive models based mainly on observation. More specifically, concerning the topic of water level prediction, a significant number of studies have been published over recent years. In [49], Nguyen et al. utilize a hybrid method that combines ARIMA and neural network techniques such as KNN, SVR, RF, LSTM to forecast water levels of the Red River in Vietnam. In similar fashion, in [50], Gan et al. leverage the predictive capabilities of the LightGBM model to outperform predictions obtained from physics-based models, such as the non-stationary tidal harmonic analysis model (NS_TIDE) for the lower Columbia River, in the USA. In another approach to the problem of predicting water levels of rivers, Ahmed et al. [51] combine historical measurements with rainfall data to train a model that performs accurate predictions for a case study focused on the Durian Tunggal River, in Malaysia. In a slightly different context, Zhu et al. [52] study a system of 69 temperate lakes in Poland and suggest models that can estimate their water

TABLE 8. Models' performance for Kienstock, daily forecasts (sorted by RMSE).

Model	RRMSE	RMSE	MAE	SMAPE
GRU /w weather (dim:128,in:15,out:1)	8.2%	28.21	22.22	6.387%
GRU /w weather (dim:128,in:10,out:1)	8.21%	28.29	22.04	6.263%
GRU /w weather (dim:32,in:10,out:5)	8.21%	28.38	20.96	5.897%
GRU /w weather (dim:32,in:10,out:1)	8.31%	28.67	22.24	6.301%
Linear Reg. (in:10,out:1)	8.25%	28.68	21.28	6.015%
Linear Reg. (in:15,out:1)	8.27%	28.72	21.32	6.046%
GRU /w weather (dim:128,in:10,out:5)	8.38%	28.88	21.95	6.241%
GRU (dim:32,in:15,out:1)	8.32%	28.88	21.62	6.136%
Last value (naive)	8.35%	29.05	20.79	5.753%
GRU /w weather (dim:32,in:15,out:1)	8.37%	29.06	21.71	6.07%
GRU (dim:32,in:15,out:5)	8.41%	29.24	22.03	6.237%
GRU (dim:128,in:10,out:1)	8.44%	29.27	21.67	6.15%
GRU (dim:32,in:10,out:1)	8.46%	29.52	22.36	6.285%
GRU (dim:32,in:10,out:5)	8.51%	29.53	22.71	6.434%
GRU (dim:128,in:15,out:1)	8.55%	29.69	22.64	6.417%
GRU /w weather (dim:128,in:15,out:5)	8.64%	29.77	22.46	6.399%
GRU /w weather (dim:32,in:15,out:5)	8.76%	29.92	22.96	6.57%
GRU (dim:128,in:10,out:5)	8.64%	30.04	23.04	6.526%
GRU (dim:128,in:15,out:5)	8.71%	30.06	22.75	6.489%

TABLE 9. Models' performance for Pfelling, daily forecasts (sorted by RMSE).

Model	RRMSE	RMSE	MAE	SMAPE
Last value (naive)	8.76%	23.34	17.82	6.526%
Linear Reg. (in:10,out:1)	8.94%	23.82	17.83	6.672%
Linear Reg. (in:15,out:1)	9.02%	23.93	17.94	6.733%
GRU /w weather (dim:32,in:10,out:1)	8.76%	24.08	19.31	6.962%
GRU /w weather (dim:32,in:15,out:1)	8.97%	24.47	19.6	7.092%
GRU /w weather (dim:32,in:15,out:5)	9.3%	25.6	20.46	7.33%
GRU /w weather (dim:128,in:10,out:5)	9.38%	25.75	20.62	7.405%
GRU /w weather (dim:128,in:10,out:1)	9.44%	25.95	20.9	7.524%
GRU (dim:32,in:10,out:1)	9.55%	26.4	20.36	7.251%
GRU (dim:128,in:10,out:1)	9.6%	26.44	20.55	7.372%
GRU /w weather (dim:128,in:15,out:5)	9.61%	26.53	21.45	7.666%
GRU /w weather (dim:32,in:10,out:5)	9.69%	26.84	21.48	7.727%
GRU (dim:32,in:15,out:5)	9.66%	26.92	21.03	7.494%
GRU (dim:128,in:10,out:5)	9.89%	27.26	21.11	7.556%
GRU (dim:32,in:10,out:5)	9.89%	27.59	21.5	7.589%
GRU (dim:128,in:15,out:5)	10.05%	28.07	22.13	7.835%
GRU (dim:32,in:15,out:1)	10.06%	28.41	22.25	7.806%
GRU (dim:128,in:15,out:1)	10.21%	28.69	22.08	7.733%
GRU /w weather (dim:128,in:15,out:1)	10.44%	29.62	23.78	8.244%

levels based on the Feed-Forward Neural Network and Deep Learning techniques. Finally, in [53], Wee et al. present a comprehensive survey of papers that tackle the problem of water level forecasting based on machine learning and report the details of their findings.

E. OBSERVATIONS

We recognize significant potential in leveraging machine learning for forecasting water levels. The weather co-variate plays a pivotal role, proving essential for ensuring precise predictions. In our study, we exclusively utilized weather data from the specific location corresponding to the prediction station in each case. Nevertheless, augmenting the models with weather data from surrounding areas, influencing the river, holds the promise of yielding even more superior results.

In general we see that the complexity and size of the model does not play an important role. Using a bigger GRU model with more hidden dimensions does not offer any improved performance. Using smaller models gives us a similar performance with the added benefit of faster training, which is crucial since we re train the models every 15 days.

Concerning multi-day forecasting, our models excel in capturing fluctuation patterns, though there exists some variance from the actual values. It is noteworthy that as we extend our forecasting horizon, the disparity increases. Despite this, we successfully maintain an accurate depiction of the general trend in water levels. Figures 12 and 15 underscore the pivotal role of reliable weather predictions in capturing these trends. In practical applications, obtaining precise weather forecasts for periods beyond 5 days proves challenging. However, our models only necessitate precipitation coverage as an input, which may be more predictable in the near future, even surpassing the 5-day mark.

The daily step forecast emerges as a markedly superior method for predicting water levels. Ideally, in Figure 14, each arrow tip should align with the corresponding water level value. While this alignment is not consistently achieved, especially during significant spikes and shallows, the model demonstrates a remarkable ability to accurately predict trends and forecast toward the correct values. Notably, in instances of sudden spikes in water levels, our model exhibits a steep increase in predictions 2-3 days prior to the actual rise. This observation underscores the clear potential of employing machine learning for water level prediction.

Overall, machine learning for water level forecasting has yielded promising qualitative outcomes, showcasing the

TABLE 10. Models' performance for Wildungsmauer, daily forecasts (sorted by RMSE).

Model	RRMSE	RMSE	MAE	SMAPE
GRU /w weather (dim:128,in:10,out:1)	8.21%	24.96	19.22	6.248%
GRU /w weather (dim:32,in:15,out:1)	8.29%	25.09	18.51	6.004%
GRU /w weather (dim:128,in:10,out:5)	8.35%	25.28	19.22	6.31%
GRU /w weather (dim:32,in:10,out:1)	8.37%	25.36	18.94	6.143%
GRU /w weather (dim:128,in:15,out:5)	8.43%	25.58	19.3	6.312%
Linear Reg. (in:10,out:1)	8.43%	25.69	19.11	6.184%
Linear Reg. (in:15,out:1)	8.51%	25.91	19.46	6.288%
GRU /w weather (dim:128,in:15,out:1)	8.54%	25.96	19.42	6.343%
GRU (dim:32,in:15,out:5)	8.58%	25.99	19.58	6.377%
GRU /w weather (dim:32,in:15,out:5)	8.71%	26.18	19.8	6.451%
GRU (dim:32,in:10,out:1)	8.64%	26.2	19.83	6.456%
Last value (naive)	8.6%	26.25	19.52	6.192%
GRU (dim:128,in:10,out:1)	8.7%	26.49	19.66	6.396%
GRU (dim:128,in:10,out:5)	8.77%	26.61	20.06	6.561%
GRU (dim:32,in:10,out:5)	8.79%	26.64	20.01	6.506%
GRU (dim:32,in:15,out:1)	8.88%	26.83	19.88	6.488%
GRU /w weather (dim:32,in:10,out:5)	8.86%	26.84	19.97	6.511%
GRU (dim:128,in:15,out:5)	8.89%	27.02	20.61	6.746%
GRU (dim:128,in:15,out:1)	9.02%	27.53	20.27	6.644%

potential of accurate predictions. However, we acknowledge certain concerns in the quantitative performance of the models and that simpler approaches occasionally outperform more complex models. That said, our emphasis in this study leans towards the IW-NET system, considering machine learning as a complementary proof-of-concept rather than a definitive forecasting solution. Moving forward, further research should address the constraints imposed by data availability and model complexity to enhance the robustness and reliability of water level predictions.

F. LIMITATIONS

It is clear that our machine learning models demonstrate promising qualitative outcomes, however, there are limitations in their quantitative performance. Metrics such as RMSE do not show a substantial improvement, and in certain instances, such as the Pfelling station in Table 9, the naive approach and linear regression models outperform the GRU models. It is crucial to acknowledge that the current challenge lies in the limited availability of data for proper training and experimentation with various machine learning models. Our datasets consists of only four years of daily measurements and is relatively small compared to the extensive datasets utilized in the literature [52]. Limited dataset size also affects more complex models, like deep GRUs, that require more data for successful training. Finally, a station like Pfelling showcases less fluctuations as we can see at Table 2. The mean and the standard deviation is much less compared to the other stations, meaning not only that the models have less samples to train, but also that naive approaches tend to be more accurate since the water level does not fluctuate a lot throughout the year.

V. CONCLUSION AND FUTURE WORK

In this work, we presented IW-NET BDA, an open-source Big-Data enabled distributed processing framework that is used to collect and process data from the Inland Waterway Transport Logistics domain. We present IW-NET BDA's architectural components and the deployment scheme of the infrastructure. We finally describe two different actual business cases where IW-NET BDA was used to perform i) automatic Areas of Interest identification and geo-temporal analytics and ii) water level forecasting using neural networks. In both cases, IW-NET BDA successfully tackles each problem in hand. A geo-fencing method using DBSCAN algorithm correctly identified existing point of interests in the first case, whereas in the second case, Gated Recurrent Units (GRU), a special category of Recurrent Neural Networks, could efficiently predict water levels with a relatively small error.

Our plans for future work involve expanding the supported execution environments that the BDA can support out-of-the-box by automating the configuration process, which for the average distributed system is quite complex and often presents a daunting challenge for system administrators. By providing a ready-to-deploy environment we hope that we can help developers experiment with new technologies and test their code with minimal effort. Other improvements involve supporting more complex workflows and improvements on the way that stream processing is implemented.

Regarding water level forecasting, we see that utilizing machine learning models becomes evident. Our current research is constrained by the limited size of our dataset (the historical water level measurements), as well as the available weather data. Future work should focus on investigating more sophisticated models trained over extended periods to comprehensively analyze annual patterns and provide the models with a greater number of samples and diverse examples. Additionally, it is worthwhile to explore alternative weather metrics beyond precipitation, such as snow coverage or tide phase. While our study concentrates on weather data at a specific station, it is reasonable to presume that weather conditions in different areas may significantly impact the water level at a particular station. Expanding our analysis to encompass a broader range of weather variables and geographic locations could enhance the accuracy and applicability of our forecasting models.

Finally, a comprehensive environmental impact assessment of the innovations introduced would be a very intuitive

way to express the effectiveness of our suggested solutions. However, such a study has not been performed in the context of the project. Most of the developed software and hardware are prototypes, some of which are in the process of being introduced in a production environment. Their impact can be assessed after a short period of production-level operation.

REFERENCES

[1] K.-D. Ruske, P. Kauschke, J. Reuter, H. von der Gracht, T. Gnatzy, and I.-L. Darkow, "Transportation & logistics 2030—volume 3: Emerging markets new hubs, new spokes, new industry leaders?" SMI (Supply Chain Manag. Inst.), Price Waterhouse Cooper, Tech. Rep., Oct. 2010.

[2] G. Wang, J. Koshy, S. Subramanian, K. Paramasivam, M. Zadeh, N. Narkhede, J. Rao, J. Kreps, and J. Stein, "Building a replicated logging system with apache kafka," *Proc. VLDB Endowment*, vol. 8, no. 12, pp. 1654–1655, Aug. 2015, doi: 10.14778/2824032.2824063.

[3] I. Indu, P. M. R. Anand, and V. Bhaskar, "Identity and access management in cloud environment: Mechanisms and challenges," *Eng. Sci. Technol., Int. J.*, vol. 21, no. 4, pp. 574–588, Aug. 2018. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2215098617316750

[4] S. Thorgersen and P. I. Silva, *Keycloak-Identity and Access Management for Modern Applications: Harness the Power of Keycloak, OpenID Connect, and OAuth 2.0 Protocols to Secure Applications*. Birmingham, U.K.: Packt Publishing, 2021.

[5] *GS1 General Specifications—Standards | GS1*. Accessed: Feb. 15, 2024. [Online]. Available: https://www.gs1.org/genspecs

[6] *Streamlined Presentation of UN/CEFACT Standards | UNECE*. Accessed: Feb. 15, 2024. [Online]. Available: https://unece.org/trade/uncefact/mainstandards

[7] K. Black, *Business Statistics: For Contemporary Decision Making*. Hoboken, NJ, USA: Wiley, 2023.

[8] A. Merendino, S. Dibb, M. Meadows, L. Quinn, D. Wilson, L. Simkin, and A. Canhoto, "Big data, big decisions: The impact of big data on board level decision-making," *J. Bus. Res.*, vol. 93, pp. 67–78, Dec. 2018. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0148296318304132

[9] A. Kumari, S. Tanwar, S. Tyagi, and N. Kumar, "Verification and validation techniques for streaming big data analytics in Internet of Things environment," *IET Netw.*, vol. 8, no. 3, pp. 155–163, May 2019.

[10] J. Guo, Z. Liu, S. Tian, F. Huang, J. Li, X. Li, K. K. Igorevich, and J. Ma, "TFL-DT: A trust evaluation scheme for federated learning in digital twin for mobile networks," *IEEE J. Sel. Areas Commun.*, vol. 41, no. 11, pp. 3548–3560, Nov. 2023.

[11] Z. Liu, L. Wan, J. Guo, F. Huang, X. Feng, L. Wang, and J. Ma, "PPRU: A privacy-preserving reputation updating scheme for cloud-assisted vehicular networks," *IEEE Trans. Veh. Technol.*, vol. 1, no 1, pp. 1–16, Oct. 2023.

[12] S. Salloum, R. Dautov, X. Chen, P. X. Peng, and J. Z. Huang, "Big data analytics on Apache Spark," *Int. J. Data Sci. Anal.*, vol. 1, no. 3, pp. 145–164, Jun. 2016.

[13] K. Shvachko, H. Kuang, S. Radia, and R. Chansler, "The Hadoop distributed file system," in *Proc. IEEE 26th Symp. Mass Storage Syst. Technol. (MSST)*, May 2010, pp. 1–10.

[14] M. Nalin Vora, "Hadoop-HBase for large-scale data," in *Proc. Int. Conf. Comput. Sci. Netw. Technol.*, vol. 1, Dec. 2011, pp. 601–605.

[15] M. Stonebraker and L. A. Rowe, "The design of POSTGRES," *ACM SIGMOD Rec.*, vol. 15, no. 2, pp. 340–355, Jun. 1986.

[16] M. Kiran, P. Murphy, I. Monga, J. Dugan, and S. S. Baveja, "Lambda architecture for cost-effective batch and speed big data processing," in *Proc. IEEE Int. Conf. Big Data (Big Data)*, Oct. 2015, pp. 2785–2792.

[17] T. Dierks and E. Rescorla, "The transport layer security (TLS) protocol version 1.2," RFC 5246 (Proposed Standard), Internet Eng. Task Force, Aug. 2008.

[18] R. P. Padhy, M. Ranjan, P. L. Suresh, C. Satapathy, and O. I. Pvt. (2011). *RDBMS To NoSQL: Reviewing Some Next-Generation Non-Relational Database's*. [Online]. Available: https://api.semanticscholar.org/CorpusID

[19] P. Hunt, M. Konar, F. P. Junqueira, and B. Reed, "Zookeeper: Wait-free coordination for internet-scale systems," in *Proc. USENIX Annu. Tech. Conf. (USENIX ATC)*, 2010, pp. 1–14.

[20] Z. Böszörményi and H.-J. Schönig, *PostgreSQL Replication*. Birmingham, U.K.: Packt Publishing, 2013.

[21] *Pgpool Wiki*. Accessed: Feb. 15, 2024. [Online]. Available: https://www.pgpool.net

[22] E Foundation. *Eclipse Jetty*. Accessed: Jan. 30, 2024. [Online]. Available: https://eclipse.dev/jetty/

[23] HAProxy. *Haproxy—The Reliable, High Performance Tcp/Http Load Balancer*. Accessed: Feb. 15, 2024. [Online]. Available: https://www.haproxy.org/

[24] Keepalived. *Keepalived for Linux*. Accessed: Jan. 30, 2024. [Online]. Available: https://www.keepalived.org/index.html

[25] T. Dierks and C. Allen, "The TLS protocol version 1.0," RFC 2246, Internet Eng. Task Force, Jan. 1999.

[26] Apache. *Apache Livy*. Accessed: Jan. 30, 2024. [Online]. Available: https://livy.apache.org/

[27] V. K. Vavilapalli, A. C. Murthy, C. Douglas, S. Agarwal, M. Konar, R. Evans, T. Graves, J. Lowe, H. Shah, and S. Seth, "Apache Hadoop yarn: Yet another resource negotiator," in *Proc. 4th Annu. Symp. Cloud Comput.*, 2013, pp. 1–16.

[28] M. S. Keller, "Take command: Cron: Job scheduler," *Linux J.*, vol. 1999, no. 65es, p. 15, 1999.

[29] Apache. *Apache Zookeeper*. Accessed: Dec. 14, 2023. [Online]. Available: https://github.com/apache/zookeeper

[30] A. Harati-Mokhtari, A. Wall, P. Brooks, and J. Wang, "Automatic identification system (AIS): Data reliability and human error implications," *J. Navigat.*, vol. 60, no. 3, pp. 373–389, Sep. 2007.

[31] B. Wiegmans and R. Konings, *Inland Waterway Transport: Challenges and Prospects*. New York, NY, USA: Routledge, 2016.

[32] M. Ester, H.-P. Kriegel, J. Sander, and X. Xu, "A density-based algorithm for discovering clusters in large spatial databases with noise," in *Proc. KDD*, 1996, vol. 96, no. 34, pp. 226–231.

[33] A. Amend. *Geoscan*. Accessed: Dec. 14, 2023. [Online]. Available: https://github.com/databrickslabs/geoscan

[34] Uber. *H3*. Accessed: Dec. 12, 2023. [Online]. Available: https://github.com/uber/h3

[35] J. E. Gonzalez, R. S. Xin, A. Dave, D. Crankshaw, M. J. Franklin, and I. Stoica, "$GraphX$: Graph processing in a distributed dataflow framework," in *Proc. 11th USENIX Symp. Operating Syst. Design Implement. (OSDI)*, 2014, pp. 599–613.

[36] H. Butler, M. Daly, A. Doyle, S. Gillies, S. Hagen, and T. Schaub, "The GeoJSON format," RFC 7946, Internet Eng. Task Force, Aug. 2016.

[37] F. Nur, M. Marufuzzaman, and S. M. Puryear, "Optimizing inland waterway port management decisions considering water level fluctuations," *Comput. Ind. Eng.*, vol. 140, Feb. 2020, Art. no. 106210. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0360835219306795

[38] A. Christodoulou, P. Christidis, and B. Bisselink, "Forecasting the impacts of climate change on inland waterways," *Transp. Res. D, Transp. Environ.*, vol. 82, May 2020, Art. no. 102159.

[39] M. Prandtstetter, P. Widhalm, H. Leitner, and I. Czege, "A novel visualisation tool for reliable inland navigation," *Transp. Res. Proc.*, vol. 72, pp. 3537–3544, 2023.

[40] P. E. McKight and J. Najab, "Kruskal-wallis test," in *The Corsini Encyclopedia of Psychology*. Hoboken, NJ, USA: Wiley, 2010, p. 1.

[41] E. S. K. Tiu, Y. F. Huang, J. L. Ng, N. AlDahoul, A. N. Ahmed, and A. Elshafie, "An evaluation of various data pre-processing techniques with machine learning models for water level prediction," *Natural Hazards*, vol. 110, no. 1, pp. 121–153, Jan. 2022.

[42] Visual Crossing Corporation. (2023). *Visual Crossing Weather (2017–2020)*. [Data Service]. [Online]. Available: https://www.visualcrossing.com/

[43] S. Zargar, *Introduction To Sequence Learn. Models: RNN, LSTM, GRU*. Raleigh, NC, USA: Department of Mechanical and Aerospace Engineering, North Carolina State Univ., 2021.

[44] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Comput.*, vol. 9, no. 8, pp. 1735–1780, Nov. 1997.

[45] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning internal representations by error propagation," in *Parallel Distributed Processing Explorations in the Microstructure of Cognition*. Cambridge, MA, USA: MIT Press, 1986, ch. 8, pp. 318–362.

[46] K. Cho, B. van Merrienboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio, "Learning phrase representations using RNN encoder–decoder for statistical machine translation," 2014, *arXiv:1406.1078*.

[47] J. Herzen, F. Lässig, S. G. Piazzetta, T. Neuer, L. Tafti, G. Raille, T. Van Pottelbergh, M. Pasieka, A. Skrodzki, and N. Huguenin, "Darts: User-friendly modern machine learning for time series," *J. Mach. Learn. Res.*, vol. 23, no. 1, pp. 5442–5447, 2022.
[48] F.-J. Chang, P.-A. Chen, Y.-R. Lu, E. Huang, and K.-Y. Chang, "Real-time multi-step-ahead water level forecasting by recurrent neural networks for urban flood control," *J. Hydrol.*, vol. 517, pp. 836–846, Sep. 2014. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0022169414004739
[49] T.-T.-H. Phan and X. H. Nguyen, "Combining statistical machine learning models with ARIMA for water level forecasting: The case of the red river," *Adv. Water Resour.*, vol. 142, Aug. 2020, Art. no. 103656.
[50] M. Gan, S. Pan, Y. Chen, C. Cheng, H. Pan, and X. Zhu, "Application of the machine learning LightGBM model to the prediction of the water levels of the lower Columbia river," *J. Mar. Sci. Eng.*, vol. 9, no. 5, p. 496, May 2021.
[51] A. N. Ahmed, A. Yafouz, A. H. Birima, O. Kisi, Y. F. Huang, M. Sherif, A. Sefelnasr, and A. El-Shafie, "Water level prediction using various machine learning algorithms: A case study of durian Tunggal river, Malaysia," *Eng. Appl. Comput. Fluid Mech.*, vol. 16, no. 1, pp. 422–440, Dec. 2022.
[52] S. Zhu, B. Hrnjica, M. Ptak, A. Choiński, and B. Sivakumar, "Forecasting of water level in multiple temperate lakes using machine learning models," *J. Hydrol.*, vol. 585, Jun. 2020, Art. no. 124819.
[53] W. J. Wee, N. B. Zaini, A. N. Ahmed, and A. El-Shafie, "A review of models for water level forecasting based on machine learning," *Earth Sci. Informat.*, vol. 14, no. 4, pp. 1707–1728, Dec. 2021.

NIKOLAOS CHALVANTZIS (Graduate Student Member, IEEE) is currently pursuing the Ph.D. degree with the Computing Systems Laboratory, School of Electrical and Computer Engineering, National Technical University of Athens (NTUA). His research interest includes distributed and stream processing systems.

ARISTOTELIS VONTZALIDIS received the Master of Engineering degree in electrical and computer engineering from the National Technical University of Athens (NTUA), in 2021. He is a Research Engineer with the Computing Systems Laboratory, NTUA.

EVDOKIA KASSELA received the Master of Engineering degree in electrical and computer engineering and the Master of Science degree in data science and machine learning from the National Technical University of Athens (NTUA), in 2013 and 2023, respectively, where she is currently pursuing the Ph.D. degree with the Computing Systems Laboratory. She is also a Senior-Level Data Engineer. She was a Teaching Assistant in various big data related courses offered by the School of Electrical and Computer Engineering, NTUA. Her research interests include big data topics and relational processing.

ARIS SPYROU received the Master of Science degree in data science and machine learning from the National Technical University of Athens (NTUA), in 2022. He is a Research Assistant with the Computing Systems Laboratory, NTUA. He is also with Procter & Gamble, as a Data Scientist for supply chain topics. His research interests include machine learning and forecasting topics.

NIKOLAOS NIKITAS received the Master of Engineering degree in electrical and computer engineering from the University of Patras (UoP), in 2019, and the Master of Science degree in data science and machine learning from the National Technical University of Athens (NTUA), in 2021. He was a Teaching Assistant of large-scale data management courses in the Data Science M.Sc. Program, Athens University of Economics and Business (AUEB). He is a Senior Platform Engineer. His professional interests include DevOps practices, GitOps principles, and infrastructure as code, demonstrating a keen enthusiasm for constructing resilient distributed systems.

NIKODIMOS PROVATAS received the Master of Engineering degree in electrical and computer engineering and the Master of Science degree in data science and machine learning from the National Technical University of Athens (NTUA), in 2016 and 2020, respectively, where he is currently pursuing the Ph.D. degree with the Computing Systems Laboratory. He is also a Senior-Level Data Engineer. He was a Teaching Assistant in various big data related courses offered by the School of Electrical and Computer Engineering, NTUA. His research interests include machine learning and big data topics.

IOANNIS KONSTANTINOU received the Diploma degree in electrical and computer engineering and the M.Sc. and Ph.D. degrees in techno-economic systems from the National Technical University of Athens (NTUA), in 2004, 2007, and 2011, respectively. He is an Assistant Professor with the Informatics and Telecommunications Department, University of Thessaly, where he regularly teaches operating systems and programming courses. He is also a Senior Researcher with the Computing Systems Laboratory, NTUA, where he also teaches advanced topics in databases. His research interests include large-scale distributed data management systems (cloud computing and big-data systems). He serves as a member for the Board of Directors of KTP SA. He was a recipient of the one Best Paper Award (IEEE CCGRID 2013) and the one Best Paper Award Nomination (IEEE CCGRID 2015) for his work on large-scale distributed systems.

NECTARIOS KOZIRIS (Member, IEEE) is a Professor of computer science and the Dean of the School of Electrical and Computer Engineering, National Technical University of Athens. Since 1998, he has been involved in the organization of many international scientific conferences, including IPDPS, ICPP, SC, and SPAA. He has given many invited talks in conferences and universities. He has participated as a partner or a consortium coordinator in several EU projects involving large-scale systems (ACTiCLOUD, EuroEXA, SELIS, CELAR, ASAP, EGI, PRACE, GREDIA, GRID4ALL, and ARCOMEM). His research interests include parallel and distributed systems, interaction between compilers, OS and architectures, datacenter hyperconvergence, scalable data management, and large-scale storage systems. He is a member of the IEEE Computer Society, a Senior Member of ACM, and the Elected Chair of the IEEE Greece Section and started the IEEE Computer Society Greece. He was a recipient of two Best Paper Awards for his research in parallel and distributed computing (IEEE/ACM IPDPS 2001 and CCGRID 2013). In 2015, he received honorary recognition from Intel for his research and insightful contributions in transactional memory (TSX synchronization extensions). To promote the open source software in Greece, he has co-founded the Greek Free/Open Source Software Society (GFOSS-www.ellak.gr) in 2008, with members 29 Greek universities and research centers, where he is serving as the Vice-Chair of the Board of Directors.