



Article

Engineering Resource-Efficient Data Management for Smart Cities with Apache Kafka [†]

Theofanis P. Raptis ^{1,*}, Claudio Cicconetti ¹, Manolis Falelakis ², Grigorios Kalogiannis ³, Tassos Kanellos ⁴ and Tomás Pariente Lobo ⁵

¹ Institute of Informatics and Telematics, National Research Council, 56124 Pisa, Italy

² Netcompany-Intrasoft, 190 02 Athens, Greece

³ Sphynx Technologies Solution AG, 6300 Zug, Switzerland

⁴ ITML, 115 25 Athens, Greece

⁵ Atos Spain, 28037 Madrid, Spain

* Correspondence: theofanis.raptis@iit.cnr.it

[†] This paper is an extended version of our paper published in the Proceedings of the IEEE International Smart Cities Conference (ISC2) 2022, Pafos, Cyprus, 26–29 September 2022.

Abstract: In terms of the calibre and variety of services offered to end users, smart city management is undergoing a dramatic transformation. The parties involved in delivering pervasive applications can now solve key issues in the big data value chain, including data gathering, analysis, and processing, storage, curation, and real-world data visualisation. This trend is being driven by Industry 4.0, which calls for the servitisation of data and products across all industries, including the field of smart cities, where people, sensors, and technology work closely together. In order to implement reactive services such as situational awareness, video surveillance, and geo-localisation while constantly preserving the safety and privacy of affected persons, the data generated by omnipresent devices needs to be processed fast. This paper proposes a modular architecture to (i) leverage cutting-edge technologies for data acquisition, management, and distribution (such as Apache Kafka and Apache NiFi); (ii) develop a multi-layer engineering solution for revealing valuable and hidden societal knowledge in the context of smart cities processing multi-modal, real-time, and heterogeneous data flows; and (iii) address the key challenges in tasks involving complex data flows and offer general guidelines to solve them. In order to create an effective system for the monitoring and servitisation of smart city assets with a scalable platform that proves its usefulness in numerous smart city use cases with various needs, we deduced some guidelines from an experimental setting performed in collaboration with leading industrial technical departments. Ultimately, when deployed in production, the proposed data platform will contribute toward the goal of revealing valuable and hidden societal knowledge in the context of smart cities.

Keywords: smart cities; Apache Kafka; Apache NiFi; data management; Industry 4.0



Citation: Raptis, T.P.; Cicconetti, C.; Falelakis, M.; Kalogiannis, G.; Kanellos, T.; Lobo, T.P. Engineering Resource-Efficient Data Management for Smart Cities with Apache Kafka. *Future Internet* **2023**, *15*, 43. <https://doi.org/10.3390/fi15020043>

Academic Editor: Konstantinos Oikonomou, Georgios Tsoumanis, Athanasios Tsipis

Received: 29 November 2022

Revised: 14 January 2023

Accepted: 17 January 2023

Published: 22 January 2023



Copyright: © 2023 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<https://creativecommons.org/licenses/by/4.0/>).

1. Introduction

During the last 10–15 years, there has been an explosion of enabling technologies for the realisation of the Internet of Things (IoT), including sensors, actuators, embedded devices with computation capabilities, software platforms, and communication protocols [1,2]. This phenomenon was driven initially by the huge potential foreseen in the automation and digitisation of industrial applications [3] and personal health systems [4], but it benefited many other segments through spillover effects. One of the most important outlets of the growing IoT ecosystem has been the smart city market [5], which has the potential to incorporate new technologies to supply citizens, as well as city councils, with new services or more efficient realisations of existing ones.

In the early developments of smart cities, each service relied on its own devices that could operate only with a dedicated proprietary platform in a vertical manner [6]. Typically,

the platforms offered Application Programming Interfaces (APIs) for the consolidation of data across multiple services in the cloud, e.g., for integrated user dashboards or big data analysis of historical data [7]. Indeed, many studies have focused on supporting the semantic interoperability of data only *after* they have been safely stored in a common repository (currently referred to as “data lake”) [8,9]. However, such a compartmentalised structure had limitations, especially in terms of redundant deployed resources and inefficient management. Therefore, the community has moved towards a horizontal approach, where a common platform is able to communicate with all kinds of devices, which are most often sensors in smart city applications [10].

This evolution is illustrated in Figure 1, which also shows the high-level architecture of the Data Management Platform (DMP)-defined H2020 MARVEL project, which aspires to define a comprehensive solution for multi-modal real-time analytics applications. The H2020 MARVEL project aims to support efficient and informed decision-making for smart city administration and public authorities by adopting an IoT approach for analysing and fusing multimodal data from sensors installed at a city level.

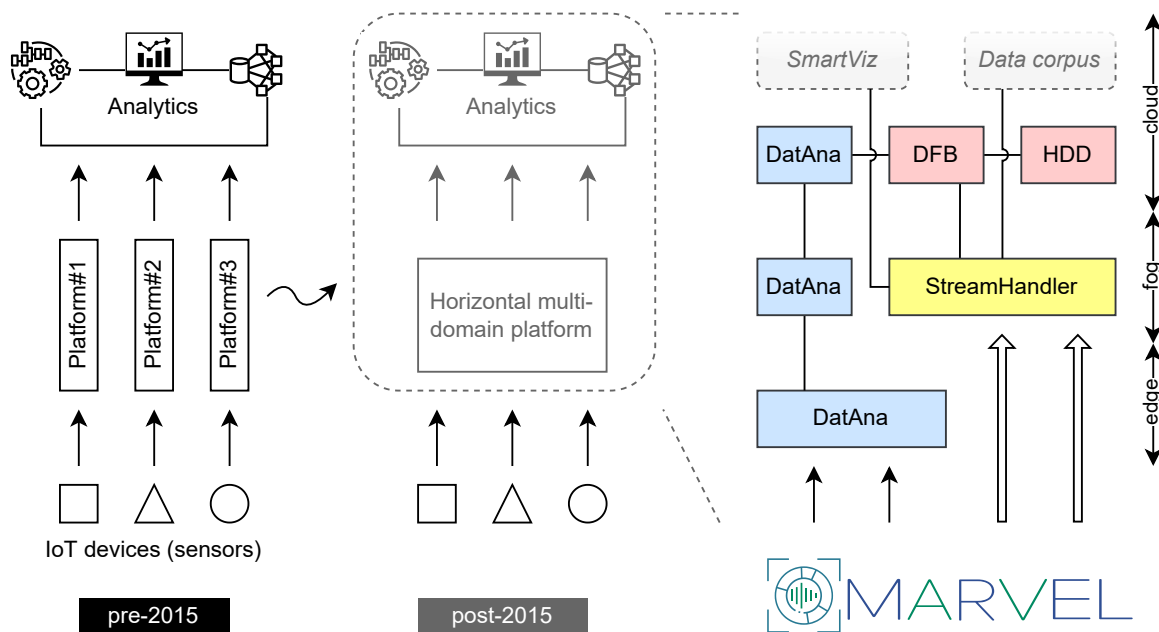


Figure 1. Transition of smart city data management and distribution systems from vertical silos to horizontal (cloud-based) platforms and positioning of the H2020 MARVEL project’s proposition for multi-layer Edge to Fog to Cloud (E2F2C) environments.

Such applications derive from the analysis of the requirements and expectations in several use cases of practical interest and the high impact on the qualities of life of citizens; they will be validated in three small-scale field trials across Europe, i.e., in Malta, Serbia, and Italy [11]. In the project, we exploit the recent trend of breaking down the computation elements of the system into three layers in a hierarchy [12,13]: the *edge* layer is closest to the sensing and embedded computation devices, but it consists of devices with modest capabilities in terms of computation, connectivity, and storage; the *fog* layer has more powerful capabilities and we treat it similar to a small private cloud, which is, however, under the control of the end user; and, finally, the *cloud* layer is hosted on public remote data centers, which have virtually infinite capacity but incur a high latency and usage costs.

The H2020 MARVEL project covers all aspects, from the development of new sensors, e.g., directional microphone arrays, to the efficient training of Artificial Intelligence (AI)/Machine Learning (ML) models on devices with limited capabilities and the ethics of data collection and analysis. However, in this paper, we focus only on the DMP, which is the core of the project’s software architecture and the main contribution of this paper. The

DMP attempts to solve the problem of big data management in the cases of Smart City IoT applications by addressing the following aspects:

- Maximise the adaptability of the overall framework according to smart city end-user needs/use cases. The DMP adopts loosely coupled interfaces (e.g., MQTT and Kafka message brokers) and standardised data models (Smart Data Models initiative) that allow the interoperability of diverse AI components, which are selected to be applied on a case by case basis.
- Allow the extensibility of the framework to be able to accommodate future needs by adopting new sensing technologies and analytics mechanisms. The standardised RTSP protocol allows the reception of AV streams from most commercial IP cameras and microphones. The loosely coupled interfaces and standardised data models allow the ingestion of data from any other potential future source (e.g., new AI component).
- Remain fully scalable to be able to accommodate any data/source volume requirement. This is achieved by employing fully scalable data ingestion and processing mechanisms at the edge and fog layer (DatAna and StreamHandler), which can be deployed on virtually any additional infrastructure edge and fog node that become available. At the same time, the big data aggregation mechanism at the cloud layer (DFB) is based on fully scalable industry-standard technologies (Kafka, Elastic Search) and its operation is further optimised in cases of significant loads by the HDD.

The DMP offers a full-featured, adaptable design solution for transferring, fusing, processing, and providing persistence to inference results generated by various AI components as well as raw audiovisual data. The DMP consists of the following components:

- **DatAna**, for the processing and transmission of structured data produced by AI components to all the layers in the inference pipeline;
- **DFB** (Data Fusion Bus), which is in charge of managing heterogeneous data across multiple components in the cloud;
- **StreamHandler**, which processes, stores, and delivers real-time AV data at the fog layer towards the processing servers and the Data corpus;
- **HDD** (Hierarchical Data Distribution), which can optimize the data management based on the available resources and current workload.

Those core components work in close connection to the **Data corpus**, which is the repository of the data collected from the sensors, mainly consisting of microphones and cameras, for visualisation and augmentation, and the **SmartViz**, which provides the Human Machine Interface (HMI) for visualisation and analysis.

As the main contribution of this paper, expanding our previous work [14], we distil the collaborative process for the design and implementation of the DMP, which was carried out over the first two years of the H2020 MARVEL project, into a *reference software architecture*. The architecture is modular, which allows for a smooth integration/update of best-of-breed technologies in the market to undergo the individual tasks. The paper benefits the research community through a set of lessons learnt and helpful guidelines for the design, development, and deployment of production-ready analytics platforms for smart city environments.

The rest of the paper is structured as follows. First, we introduce the background and foundations of the H2020 MARVEL project, which are needed to understand the concepts and terminology used in the rest of the paper, in Section 2. This section also includes an overview of the essential state of the art on data/resource management in edge/fog systems. In Section 3, we then illustrate all the main components of the H2020 MARVEL project's DMP and the Data-corpus-as-a-service in Section 4. The preliminary results obtained during the mid-project integration tests are reported in Section 5, while Section 6 presents some useful design guidelines that directly reflect our experience. Furthermore, in Section 7, we discuss the main limitations of our approach, together with potential solutions to overcome them: to this aim, we also provide in Section 8 an outlook on a long-term

evolution of the DMP through the migration to serverless computing. Finally, Section 9 concludes the paper. The flowchart of our methodology is displayed in Figure 2.

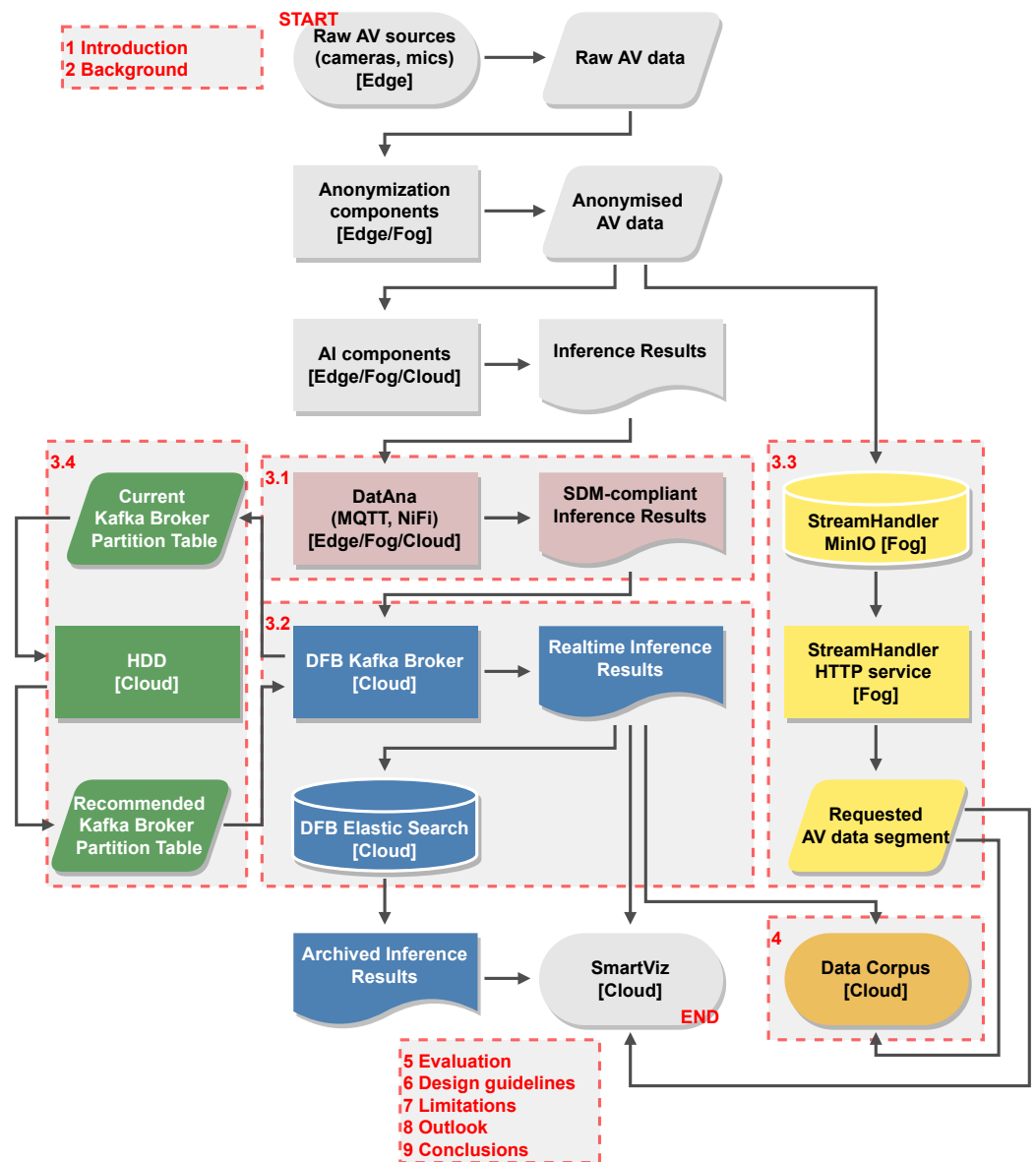


Figure 2. The flowchart of our methodology.

2. Background

In this section, we provide an overview of the aspects of the H2020 MARVEL project that are relevant to the design and implementation of the DMP, which is the main subject of this work.

A layered view of the physical deployment of the project’s pilots is illustrated in Figure 3. The lowest layer is the *edge*, which contains the sensors and embedded devices to perform on-site operations. We have two types of sensors: microphone arrays, producing audio streams, and cameras producing AV streams. All the sensors are expected to operate continuously during the service lifetime. The embedded devices, i.e., Raspberry Pis and NVIDIA Jetson boards, perform operations directly at the edge, which include the anonymisation of the AV streams and basic inference operations. The intermediate layer is called the *fog* (we note that, in the scientific literature and in the market press, the terms “edge” and “fog” do not have universally accepted meanings. Sometimes they are even used interchangeably), which includes more powerful computation resources

that are shared by multiple edge sites, e.g., workstations and rack-mountable servers with Graphics Processing Units (GPUs), which are suitable for not only managing the different data streams but also performing more advanced inference tasks, as well as training AI models. Finally, the *cloud* is hosted in an infrastructure provided by a project partner and it is common for all the pilots. The cloud provides the long-term storage of data and all the services for HMI, i.e., visualisation and real-time (on-demand) analysis.

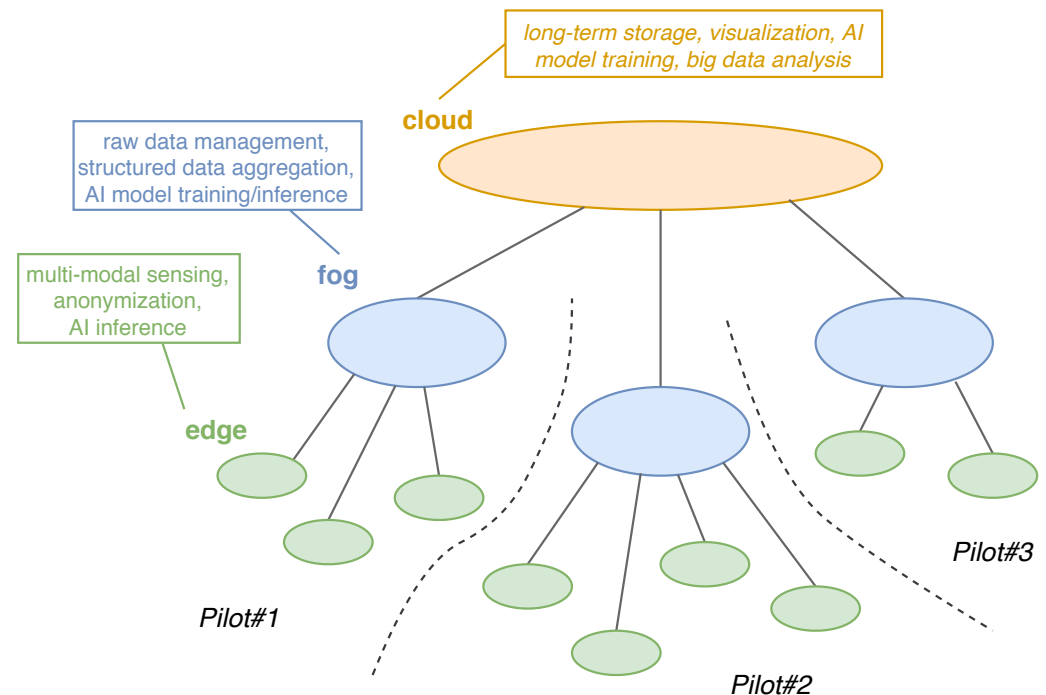


Figure 3. E2F2C view of the H2020 MARVEL project's pilots.

In Figure 3, we distinguish between raw data vs. structured data:

- The **raw data** are the AV streams generated by the sensors, irrespective of whether they have been anonymised or not. Their formats and characteristics are heterogeneous because they depend on the physical devices installed (e.g., may use different codec or sample AV at different rates). In any case, the throughput is generally high, especially for video, which requires carefully provisioned bandwidth, long-term storage, and configuration of the data distribution services.
- The **structured data**, instead, are data generated by the analytics applications, i.e., the inference components. In the project, there are different data models for each AI component, but all of them have been based on the fully Smart-Data-Model (SDM)-compliant (<https://smartdatamodels.org/>, accessed on 16 January 2023) data models that DatAna are producing. Furthermore, their throughput is generally much lower than that of raw data.

A schematic of a typical multi-modal application in the H2020 MARVEL project is illustrated in Figure 4. We have used “thick” arrows to represent the raw data flows, while “thin” arrows refer to structured data flows; this reflects, in a visual manner, their different bandwidth requirements. The diagram is intended to provide the reader with a sketch of the components that are involved in the deployment of a service, which can differ in practice depending on the specific use case and pilot. The services we support in the project include surveillance applications (e.g., detecting anomalies in public spaces), emotion recognition (with aerial images taken by drones), and road safety (e.g., monitoring junctions with mixed cars and bike traffic).

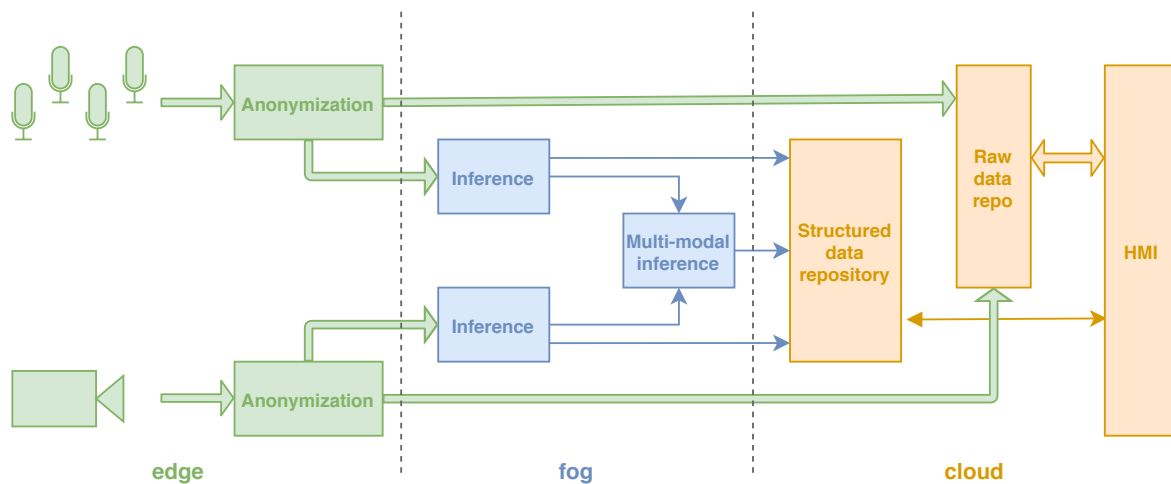


Figure 4. Schematic of a typical multi-modal, i.e., audio and video, real-time analytics application in H2020 MARVEL project. The mapping of components to the edge/fog/cloud layers is only indicative: in real applications, this may depend on physical, environmental, and administrative constraints.

Related Work

The topic of the efficient allocation of resources in E2F2C has been extensively studied in the scientific literature (e.g., [15]). The vast majority of the works propose highly simplified mathematical models, which are then solved to maximise a given objective function, but they do not linger on the practical implementation of their solution using industry-grade tools. On the other hand, in the H2020 MARVEL project, our goal was to realise a solution based on widely-used, reliable, and open source software. Furthermore, some works have focused on the specific aspect of efficient data distribution. In [16], the authors have proposed Peer-to-Peer (P2P) as a means to distribute data in a robust and decentralised manner among agents at the edge. Such an approach only partially covers the needs of the H2020 MARVEL project, since, due to its use case requirements, the H2020 MARVEL project AV real-time streams cannot be stored efficiently in a P2P overlay. An in-network storage management is put forward in [17], to place both raw and structured data on distributed resources using the Google File System, which is, however, proprietary. Another perspective is adopted in [18], which builds on top of the distributed Apache Cassandra (<https://cassandra.apache.org/>, accessed on 16 January 2023) database by adding geolocalisation tags that help distribute the load across edge/fog nodes; we have not considered this feature since, at least in the project pilots, the edge/fog nodes are all deployed in close proximity. The solutions mentioned can be gradually incorporated in the DMP at a later stage of the project, which is still ongoing with an expected completion date at the end of 2023.

More specifically, Apache Kafka has also been widely used in the streaming applications domain. In [19], the authors replicate Apache Kafka logs for various distributed data-driven systems of LinkedIn, including source-of-truth data storage and stream processing. In [20], the authors design a distributed cluster processing model based on Apache Kafka data queues to optimise the inbound efficiency of seismic waveform data. In [21], the authors extend Apache Kafka by building an in-memory distributed complex event recognition engine built on top of Apache Kafka streams. In [22], the authors design a simulation platform enabling evaluations of future mobility scenarios, based on an Apache Kafka architecture. In [23], the authors break the streaming pipeline into two distinct phases and evaluate the percentile latencies for two different networks, namely 40 GbE and InfiniBand EDR (100 Gbps), to determine if a typical streaming application is network intensive enough to benefit from a faster interconnect. Moreover, they explore whether the volume of an input data stream has any effect on the latency characteristics of the streaming pipeline and, if so, how it compares for different stages in the streaming pipeline and different network interconnections. In [24], the authors propose a distributed framework for the application of stream processing on heterogeneous environmental data, which addresses

the challenges of data heterogeneity from heterogeneous systems and offers real-time processing of huge environmental datasets through a publish/subscribe method via a unified data pipeline with the application of Apache Kafka for real-time analytics. In [25], the authors find that filtering on large datasets is best performed in a common upstream point instead of being pushed to, and repeated, in downstream components. To demonstrate the advantages of such an approach, they modify Apache Kafka to perform limited native data transformation and filtering, relieving the downstream Spark application from doing this. Their approach outperforms four prevalent analytics pipeline architectures with negligible overhead compared to standard Kafka. In the next sections, we illustrate the high-level design and current status of development of H2020 MARVEL project's DMP.

3. DMP Software Architecture Design

The main components of the DMP are introduced briefly in Section 1 (see also Figure 1). Below, we provide an overview, followed by component details in dedicated sub-sections.

DatAna is responsible for collecting the inference results from all AI components from all layers through its instances residing at each layer, transforming them into SDM-compliant counterparts and then transferring them to higher layers in the E2F2C continuum. The DatAna cloud layer thus aggregates all the transformed inference results and relays them to the DFB, which also resides in the cloud. **DFB** persistently stores all the SDM-compliant inference results it receives, but also ensures they are available in real time to SmartViz and Data Corpus. DFB also exposes a REST API to SmartViz to allow it to access all the archived inference results in its Elasticsearch (<https://www.elastic.co/>, accessed on 16 January 2023) database. In addition, the DFB receives user-generated inference result verification messages from SmartViz and use this information to update the corresponding inference results stored in its database. **HDD** interacts exclusively with the DFB to receive information on current Kafka topic partitioning and associated performance metrics and to send updated, optimised Kafka topic-partitioning recommendations. In parallel, **StreamHandler** receives information on active AV sources after requesting it from a component called *AV Registry* via a REST call and it uses that information to connect to all the active AV sources and receive their AV data streams to segment them and store them persistently. StreamHandler also exposes a REST API that is accessed by SmartViz to request archived AV data from specific sources and points in time. The Data Corpus resides at the cloud and is subscribed to all the DFB Kafka topics where DatAna publishes SDM-compliant inference results to receive them in real time and archive them internally to ensure they are available for further AI training purposes along with the associated AV data it collects from StreamHandler. The Data Corpus is also subscribed to the DFB Kafka topic that is used by SmartViz to publish user-generated inference result verifications to receive them in real time and update the corresponding archived inference results accordingly. The Data Corpus is also connected to StreamHandler, from which it receives AV data as binary files that are a result of AV stream segmentation.

3.1. DatAna

DatAna is a component distributed across all three E2F2C layers, with a separate instance deployed at each infrastructure node. DatAna is complemented by an MQTT message broker, which is also deployed at each infrastructure node, alongside DatAna. Each instance of the MQTT (<https://mqtt.org/>, accessed on 16 January 2023) message broker is responsible for collecting structured data, i.e., inference results, from the AI components residing on the same layer as the respective MQTT instance. Specifically, AI components publish their raw inference results to dedicated MQTT topics in real time as they are being produced through the analysis of the AV data streams they receive. The input inference results of each AI component are formatted as JSON documents according to a dedicated distinct data model that fits the requirements of each AI component. The following are the most notable fields in these data models:

- AV source ID. The ID of the AV source that produced the stream that was analysed to produce the inference result.
- Inference result ID. A unique identifier for the inference result.
- Timestamps. In case the inference result refers to an instant in time, a single timestamp is provided. In case the inference result refers to a period in time, two timestamps are provided, corresponding to the start and end of the time period of the result. All the time information is absolute and following the ISO 8601 UTC format.

Besides the above, the raw inference results contain other fields that are specific to the needs of each AI component.

Each DatAna instance residing on the same infrastructure node as an MQTT broker subscribes to the broker's topics to receive all the incoming input AI inference results. Subsequently, DatAna transforms the input inference results into SDM-compliant counterparts. Three data models that belong in the collection of smart data models of the SDM standard have been identified to be relevant to H2020 MARVEL project, which have been modified by adding additional fields to account for the project's needs:

- **MediaEvent**: to describe general AI inference results.
- **Alert**: to describe AI inference results that should be perceived as alerts.
- **Anomaly**: to describe AI inference results that should be perceived as detected anomalies.

DatAna autonomously selects the most appropriate data model to perform the transformation, whose output is then relayed to higher-level layers.

Specifically, the SDM-compliant inference results produced by DatAna at the edge layer are relayed to DatAna at the fog layer and the SDM-compliant inference results produced by DatAna at the fog layer are relayed to DatAna at the cloud layer. The DatAna instance at the cloud layer is responsible for relaying the SDM-compliant inference results it collects from all the layers to the DFB by publishing them to the appropriate DFB Kafka topics.

3.2. DFB

The DFB resides at the cloud and receives all the SDM-compliant inference results published by the DatAna cloud instance and stores them persistently in its Elasticsearch database. The DFB also exposes a REST API to SmartViz to allow it to access all the archived inference results in its Elasticsearch database. The DFB receives user-generated verifications of the inference results from SmartViz when they are published to a dedicated DFB Kafka topic and uses them to update the respective archived inference result entries accordingly. The DFB also accesses a REST API at the HDD for dispatching the currently applied Kafka topic partition information along with the associated performance measurements to it. Using the same REST API, the DFB can also receive the updated Kafka topic partition allocation that is recommended by the HDD. SmartViz is subscribed to all the DFB Kafka topics where DatAna publishes SDM-compliant inference results to receive them in real time and present them to the user. SmartViz also allows users to verify the inference results they are presented with. SmartViz transmits these user-generated verifications to the DFB by publishing them to a dedicated Kafka topic available at the DFB.

3.3. StreamHandler

StreamHandler resides at the fog and receives AV data streams from all the active AV sources (CCTV cameras, network-enabled microphones, and AV anonymisation instances) via RTSP. During initialisation, StreamHandler accesses the REST API of the AVRegistry to discover the active AV sources and their details. During operation, StreamHandler consumes the AV RTSP streams and segments them according to a pre-specified time intervals to generate binary documents, suitable for persistent storage. StreamHandler archives the generated AV data files and also exposes a REST API to accept requests from SmartViz about the transmission of AV data from specific AV sources (reference to AV Source id) and from specific points in time. Upon such requests, StreamHandler retrieves

the necessary binary files, compiles a unified/edited version of the stream that corresponds to the timeframe requested and generates a link to the said binary file that is to be consumed by SmartViz.

3.4. HDD

The HDD exposes a REST API to allow for the reception of the currently applied DFB Kafka topic partition information along with associated performance measurements from the DFB. The HDD uses this information as an input to calculate an optimised Kafka topic partition allocation and subsequently ensures it is available to the DFB via its REST API. The exact optimisation method that is implemented by the HDD can be found in [26].

The DMP has been applied in five use cases defined for the needs of the initial version of the H2020 MARVEL project Integrated framework.

4. Data Corpus-as-a-Service

In the context of the H2020 MARVEL project framework design activities, certain similarities and overlaps were identified between the functionalities of StreamHandler and those of DFB and DatAna with regards to big data management. However, following an in-depth analysis of the H2020 MARVEL project framework requirements that the DMP should satisfy, a gap was identified that could not be covered by the DFB and DatAna solutions. This gap was related to the management of audio–visual data. More specifically, the following requirements were established:

- Receive and efficiently archive live streams of audiovisual binary data from all the relevant H2020 MARVEL project sensors, devices, and components during system operation.
- The persistent storage of archived AV data should comply with high data security standards and data privacy requirements.
- Provide access to archived audiovisual binary data to the H2020 MARVEL project UI (SmartViz) by streaming requested archived audiovisual data upon demand in order to present them to the end-user and in association with relevant inference results produced by H2020 MARVEL project AI components.
- Support the expansion of the data set of the Data Corpus by relaying selected archived audiovisual data to it.

StreamHandler was found to be in a position to be able to satisfy these requirements and fill the gap by extending its supported data source types, its connectors, and data storage capabilities. This course of action was aligned with INTRA's strategic plan to expand the StreamHandler platform in the direction of audiovisual data management for increased interoperability in order to address additional business cases and reinforce its position in big data management and smart city domains.

The H2020 MARVEL project Data Corpus-as-a-Service has been one of the major outcomes of the H2020 MARVEL project trying to address the lack of extremely large public sets of annotated audio–visual recordings, which has been an obstacle for the scientific and industrial community to enhance audio–visual analytics. The H2020 MARVEL project Data Corpus is an extreme-scale public corpus of processed multimodal audio–visual open data, obtained free of charge and released as a service under the aegis of a specific service level agreement (SLA). It addresses all aspects that are related to the efficient sharing of heterogeneous data pools such as accessibility, operability, managing streaming and network, legal considerations, security, privacy, and technical concerns.

The H2020 MARVEL project Data Corpus-as-a-Service has been the user's endpoint for accessing, through respective queries, the vast processed audio–visual data that is being stored in the H2020 MARVEL project infrastructure. Enriching and sharing the Data Corpus drives research and innovation in multimodal processing and analytics, which contribute to the process of benchmarking edge, fog, cloud, and AI technologies. Furthermore, besides its contribution to the areas of open science and open data, the H2020 MARVEL project Data Corpus-as-a-Service empowers smart city authorities to better support their

societies, deriving new and advancing the existing knowledge; it enables them to build and deploy innovative applications that are based on multimodal perception and intelligence. Furthermore, the H2020 MARVEL project Data Corpus-as-a-Service serves as a key source of interaction with developers and researchers. It serves as a pool of data available to the public, enabling the scientific and industrial community to enhance audio–visual analytics and enforcing the foundations for smart city services that improve the security and well-being of their citizens. In addition, the H2020 MARVEL project Data Corpus provides the possibility for SMEs and start-ups to build on top of these data assets and create new business by exploring extreme-scale multimodal analytics. Furthermore, by adopting an SLA-enabled Big Data analytics framework, it is expected to maximise the impact that the H2020 MARVEL project corpus has on the international scientific and research community.

The H2020 MARVEL project’s Data Corpus enables smart cities to build and deploy innovative applications that are based on multimodal perception and intelligence and it is freely available for all societal EU data marketplaces. Furthermore, the H2020 MARVEL project Data Corpus-as-a-Service enables the possibility for SMEs and start-ups to build on top of these data assets and create new businesses by exploring extreme-scale multimodal analytics. From a technical point of view, the H2020 MARVEL project Corpus component consists of four subcomponents:

- An extreme-scale audio–visual recording repository where processed data are stored.
- A database for storing the respective metadata where the folder structure might be different from a physical file directory structure.
- A management application.
- A series of application programming interfaces (APIs) that allows accessing and querying recordings stored in the repository.

4.1. Data Corpus Architecture

Figure 5 represents a high-level architecture of the H2020 MARVEL project Data Corpus-as-a-Service including the previous subcomponents. The extreme-scale audio–visual recording repository where processed data are stored adapts the HBase distributed database, which is an open source non-relational distributed database modelled after Google’s Bigtable that is being built and run on top of the Hadoop Distributed File System (HDFS).

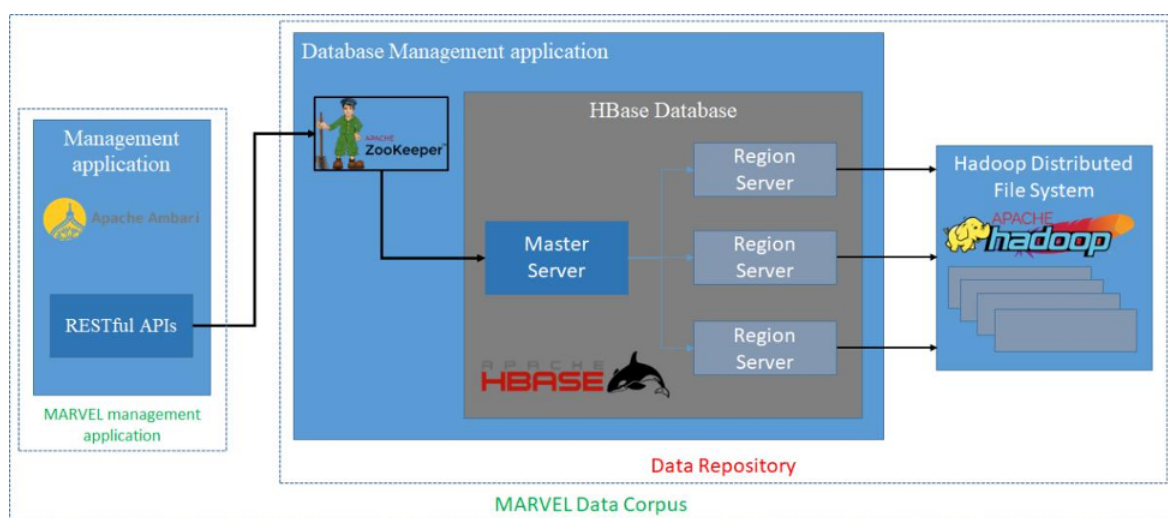


Figure 5. H2020 MARVEL project’s Data Corpus architecture.

Based on Figure 5, the management application of the H2020 MARVEL project database is based on Zookeeper, an open source project that provides services such as maintaining configuration information and naming and providing distributed synchronisation. The database that stores the respective metadata is based on an HBase distributed database in which a master server assigns regions to the region servers and handles load balancing of the regions across region servers by unloading the busy servers and shifting the regions to less occupied servers. In HBase, a table is both spread across a number of region servers as well as consisting of individual regions. As the tables are split, the splits become regions. The regions store a range of key-value pairs and each region server manages a configurable number of regions. The management application of the H2020 MARVEL project Corpus has been based on the Ambari project that is aimed at simplifying Hadoop management by developing software for provisioning, managing, and monitoring Apache Hadoop clusters. Ambari provides an intuitive, easy-to-use Hadoop management web user interface backed by its RESTful APIs.

The storing of the audio/video files themselves is performed using the Hadoop Distributed File System (HDFS). It saves files on commodity machines, providing high-aggregate bandwidth across the Corpus cluster. The administration of this repository is supported by the HBase distributed database (an open source non-relational distributed database). The management application of this database is implemented by the Zookeeper. It is an open source module that provides administrative services, such as maintaining the naming and configuration information and providing distributed synchronisation, etc. Thereupon, queries (accessing) are performed via the web interface of Ambar that offers an intuitive, easy-to-use Hadoop management solution with Web UI and RESTful APIs. The storing of the audio/video files themselves is performed by the Hadoop Distributed File System (HDFS). It saves files on commodity machines, providing high aggregate bandwidth across the Corpus cluster. The administration of this repository is supported by the HBase distributed database; the management application of this database is implemented by the Zookeeper while the queries (accessing) of the HDFS are performed via the web interface of Ambari.

The H2020 MARVEL project data corpus data storage is based on the Hadoop Distributed File System (HDFS) fault-tolerant data storage file system [27]. There are various tools and solutions available to secure the HDFS environment and each of them has different features and effectiveness under various contexts. The security tools and solutions can be divided into four categories [28,29], which are (i) encryption, (ii) authentication, (iii) authorization, and (iv) auditing.

4.2. User Authentication

Authentication refers to verification of the system or user identity for accessing the system or, in other words, it is the procedure of confirming whether the user is the person they claimed to be. Two common authentication technologies are: (i) Lightweight Directory Access Protocol (LDAP) for the directory, identity, and other services and (ii) Kerberos. Authorisation is the process of determining the access rights of the user, specifying what they can do with the system. As Hadoop mixed various systems in its environment, it required numerous authorisation controls with different granularities. In Hadoop, the process of setup and maintenance of the authorisation control is simplified and can be performed by dividing users into groups by specifying them in the existing LDAP or Active Directory (AD). Other than that, the authorisation can also be set up by providing role-based access control for connection methods that are alike. The most popular tool for authorisation control is Apache Sentry. Currently, user authentication for the H2020 MARVEL project data corpus is performed through MARVdash, thus allowing for an external, secure access to the Data Corpus API services by integrating a web proxy service. The transmission of data is secured with HTTPS and the underlying TLS. The authorisation level is defined during the registration of the user account (e.g., simple user, administrator, etc.).

HBase can be configured to provide User Authentication, which ensures that only authorised users can communicate with HBase. The authorisation system is implemented at the RPC level and is based on the Simple Authentication and Security Layer (SASL) [6], which supports (among other authentication mechanisms) Kerberos. SASL allows authentication, encryption, negotiation, and/or message integrity verification on a per connection basis. The most popular mechanism for authentication is Kerberos, which is also the primary authentication for Hadoop developed by MIT [30]. The Kerberos protocol provides secure communications over a non-secure network by using secret-key cryptography. The protocol of Kerberos is shown as below:

- The client first needs to request Ticket Grant Ticket (TGT) from the Authentication Server (AS) of the Key Distribution Centre (KDC).
- After the client receives the TGT, the client must request a Service Ticket (ST) from the Ticket Grant Server (TGS), which comes from the Key Distribution Centre (KDC) (the Authentication Server and Ticket Grant Server are the components of the Key Distribution Centre).
- The client can use the ST to authenticate a Name Node. The TGT and ST are renewed after a long run of jobs. The greatest benefit of Kerberos is that the ticket cannot be renewed if it was stolen (ref). Kerberos provides powerful authentication for Hadoop. Instead of using a password alone, the cryptographic mechanism is used when requesting services [31,32].

4.3. Privacy Assessments for Data Corpus

In general, the datasets that are stored in the Data Corpus are meant to become public. Therefore, those data have been anonymised by the rest of the H2020 MARVEL project platform components (e.g., VideoAnony). Nevertheless, there can exist datasets that are in progress and/or the data owner (pilot) has not yet realised/authorised their public use. Three dissemination levels are considered. The public datasets can be viewed by everyone (including external users). The private datasets can be viewed only by the H2020 MARVEL project partners. The in-progress datasets may not have been anonymised yet and are visible only among the consortium members. The data owners (pilots) can set the visibility scope as they are processing their datasets. Next, upon user registration and login in, the platform, the authentication/authorisation policy for a user type is implemented by the above-mentioned security mechanisms. Apart from those built-in controls, the Assurance Platform [33] has been deployed in the H2020 MARVEL project backend. This platform can additionally perform a security/risk analysis for the Corpus components, as well as penetrate testing elements. Moreover, the Monitoring Assessment Profiles can be developed in order to continuously monitor the real-time operation of the system in terms of security and/or privacy. For each profile, a set of Event Captors gather information from the system, such as event logs. The main capturing mechanisms are implemented with Elastic Beats, while customised events captors are also supported. These sensing mechanisms are sending information back to the platform and its Monitor subcomponent, which analyses the events with a rule-based logic (implemented in the Drools rule engine). Machine Learning (ML) evaluations can also be supported for cases that require more complex analysis. Henceforth, security and/or privacy assessments can be performed. Two assessment examples are considered so far:

- Availability of service: Event captors are periodically checking wherever a service is up or down. In the later case, the monitoring module can calculate metrics, such as the average downtime time, the mean time to repair/restore, or the mean time to respond to an incident. Thus, the Assurance Platform can assess the availability of the internal Hadoop/HBase services as well as the web-based Graphical User Interface (GUI) that is offered to the external users.
- User access and Data Ownership: Event captors can collect the user-related actions from logfiles. Therefore, the Assurance Platform can assess the applicability of privacy controls for accessing the data. For example, the platform can recognise wherever

private or in-progress datasets have been viewed/downloaded by an unauthorised external user (an attacker that gained access to the system) and raise an alert to the system administrator. The platform can also measure the number of violations in the case of such an attack, which may also reflect the time that the administrator required to fix/mitigate the problem (i.e., block the malicious or compromised user account). Similarly, the platform can audit the processing of aspects for the datasets, i.e., whenever an unauthorised user account managed to ingest new data or update/delete existing datasets.

The rulesets for the deployed assessment profiles form the basis for the Service Level Agreement (SLA) driven verification of the system. The aforementioned metrics (e.g., mean time to respond, number of violations, etc.) can be utilised with thresholds specifying the fair use of the system.

4.4. The Graphical User Interface

The user (either H2020 MARVEL project internal or external) can access the Corpus via a graphical user interface (GUI). This GUI is developed in Angular 12 and can be utilised by users to review and retrieve the ingested datasets (Figure 6). A user can also perform queries and search for datasets/snippets (e.g., pilot name, based on keywords, etc.). Underneath, such queries are facilitated by the Elasticsearch–Logstash–Kibana (ELK) stack and the user performs them from the GUI. In the context of the H2020 MARVEL project Data Corpus UI prototyping process, several different user stories have been considered, such as viewing, adding, and deleting the data. However, the ultimate goal to perfect a given user journey is to have the best UX and via a few clicks to be able to process the data of the Corpus.

An indicative user flow starts with the main front page of the UI where the H2020 MARVEL project Data Corpus user can have an overall overview of the current status of the uploaded datasets and snippets. Furthermore, the latter works as a starting point for the user to add new dataset, view, or update existing ones. The important feature of this flow is that the user can have a single page, a complete synopsis of the uploaded data inside the Corpus, and the corresponding actions over them.

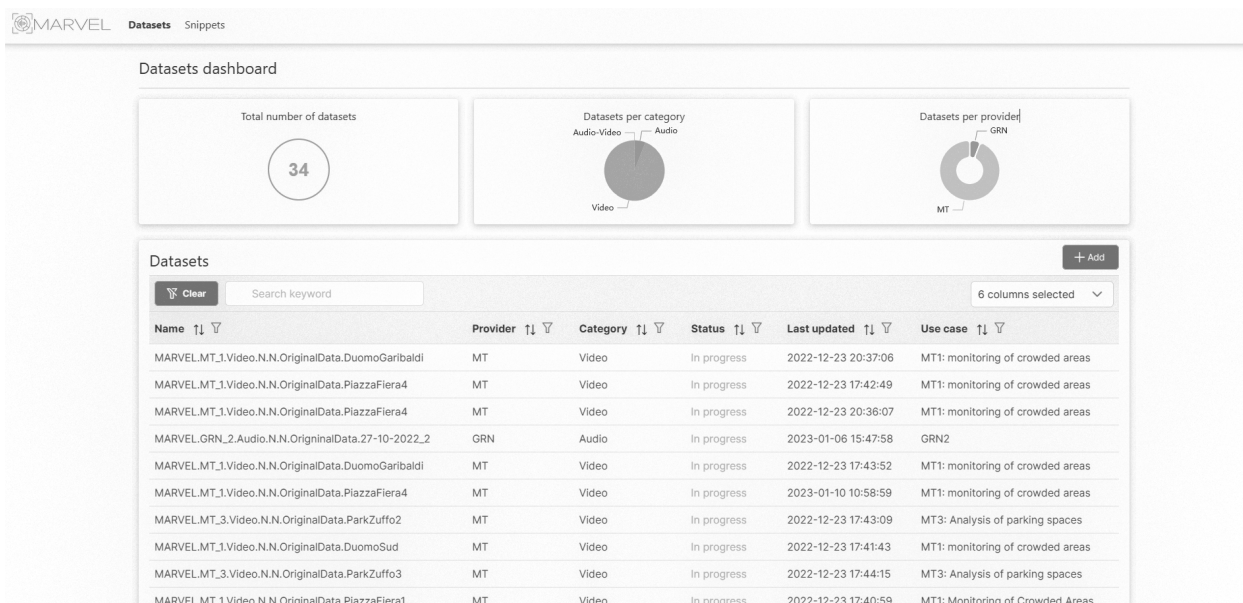


Figure 6. GUI: main dashboard (mock-up data).

From this point, the internal H2020 MARVEL project user can add, edit, view, or delete the selected dataset with a simple click since the main page of the UI redirects him/her to the corresponding page of the interface. When it comes to adding data to the Corpus, the interface

guides her via a single page where a series of related fields must be filled, in accordance with the unified data model of the dataset entries in the Corpus repository (Figure 7).

The screenshot shows the 'Add new dataset' form in the MARVEL interface. The form is titled 'Add new dataset' and contains the following fields:

- Name:** MARVEL_11-01-2023_0
- Data provider *:** (empty text input)
- Use case *:** (empty text input)
- Keyword(s):** (empty text input with a note: '(Add keyword by pressing 'enter' or 'tab' or 'click out of form)')
- Description:** (empty text area)
- Status *:** (empty text input)
- Category *:** (dropdown menu with options: Audio, Video, Audio-Video)

Figure 7. GUI: main dashboard (add new dataset).

Editing a dataset is as simple as it can be and can be performed through a single page also. Since the relative correlated information needed to be filled by the user is quite a lot, the UI of the Corpus, via a uniform view, provides the ability to have an overall control of their entries (Figure 8).

The screenshot shows the 'Edit dataset' form in the MARVEL interface. The form is titled 'Edit dataset' and contains the following fields:

- Name:** MARVEL_MT_duomo-garibaldi.Video.N.Y.OriginalData.11-01-2023_1
- Data provider *:** MT
- Use case *:** MT1: monitoring of crowded areas
- Keyword(s):** monitoring of crowds in open spaces (with a note: '(Add keyword by pressing 'enter' or 'tab' or 'click out of form)')
- Description:** test
- Status *:** In progress
- Category *:** Video
- Metadata:**
 - Video resolution *:** 1600x1200
 - Video fps *:** 12.5
 - Annotation software *:** -
 - Annotation ontology *:** -
 - Device id *:** duomo-garibaldi
 - Duration (sec) *:** 60
 - Location latitude *:** N/A
 - Location longitude *:** N/A

Figure 8. GUI: main Dashboard (edit dataset).

Last but not least, the H2020 MARVEL project Data Corpus user can view and delete a specific dataset by just selecting it and performing the relative action. Upon successful deletion, the dataset list presented on the front page of the UI is automatically refreshed. The overall Corpus and its GUI functionality are being designed to achieve maximum interaction with all the pilots and their end-to-end demonstrations, subject to the cloud data sharing restrictions. There might be access restrictions for some piloting data that cannot be offered for use by the general public outside the consortium members.

5. Evaluation of the Setup

In this section, we summarise the results obtained during the first system integration tests of the H2020 MARVEL project.

5.1. *DatAna*

During the tests performed during the MVP, the performance metrics of a single NiFi instance were measured. Table 1 summarises the collected measurements for the specified metrics.

Table 1. Results of the measurements for DatAna component.

Metric	Value
Data loss rate	0
Service availability-failed request	100% availability
Data access restriction	None
Data throughput	1.1 MB/s
Response time	47.1 ms
Number of cluster nodes	1

For a more in-depth analysis of the performance metrics, there is this Cloudera study [34], which reports how NiFi behaves in terms of scalability and performance (data rates) using very demanding workloads.

5.2. *DFB*

For DFB, the following high-level performance indicators were considered:

- **Data Integrity:** to confirm that advanced encryption mechanisms over end-to-end data transfer guarantee data integrity. Metric: Data loss rate.
- **Availability:** to verify that DFB resources are available and discoverable. Metrics: Service availability-failed request, data access restriction.
- **Performance** (for high volume, heterogeneous data streams): to measure different performance metrics under different execution conditions. Metrics: Data transfer latency, data throughput, response time, and number of cluster nodes.

Table 2 summarises the collected measurements for the specified metrics.

Table 2. Results of the measurements for DFB component.

Metric	Value
Data loss rate	0
Service availability-failed request	100% availability
Data access restriction	None
Data transfer latency	5 ms (200 MB/s load)
Data throughput	605 MB/s
Response time	5 ms (200 MB/s load)
Number of cluster nodes	3

5.3. *StreamHandler*

The preliminary testing has indicated that StreamHandler is capable of processing at least three Full HD AV data streams in parallel with no performance lag when deployed on an infrastructure with two CPU cores allocated.

5.4. HDD

For the purposes of evaluating the efficiency of HDD, we took into account the industrial best practices in the related application sectors. We identified Kafka setup guidelines used by credible industrial service providers. For example, Microsoft recommends that it would be better to constrain the existing partitions per broker (including replicas) to a number not more than 1000. In another example, Confluent recommends setting the number of partitions per broker to at least $100 \cdot B$. Consequently, combining the essence of these configuration recommendations, we arrive at the following benchmark method, called MS-CNFL: $P = \min\left(P \in_R \left[1 \dots \frac{1,000 \cdot B}{r}\right], P \in_R [1 \dots 100 \cdot B]\right)$ and $b \in_R [1 \dots B]$, where \in_R denotes uniformly random selection. We measure the system throughput, captured by the ultimate number of partitions selected by each algorithm (our algorithms being BroMin and BroMax of [26]); the replication latency, captured by the amount of time that is needed to process each message, in the sense of time required for data to be stored or retrieved; the numbers or costs of the application’s infrastructure, captured by the number of brokers used in the Apache Kafka cluster; and the OS load metric via the open file handles and the unavailability metric via the unavailability time. We perform the measurements for variable numbers of consumers. Indicatively, we display the performance in terms of throughput (number of partitions) and replication latency, in Figure 9. We can see that the HDD maintains equivalent numbers of partitions (and therefore throughput), but, at the same time, does not violate the latency constraint (as the benchmark is doing). The complexity analysis details can be found in [26].

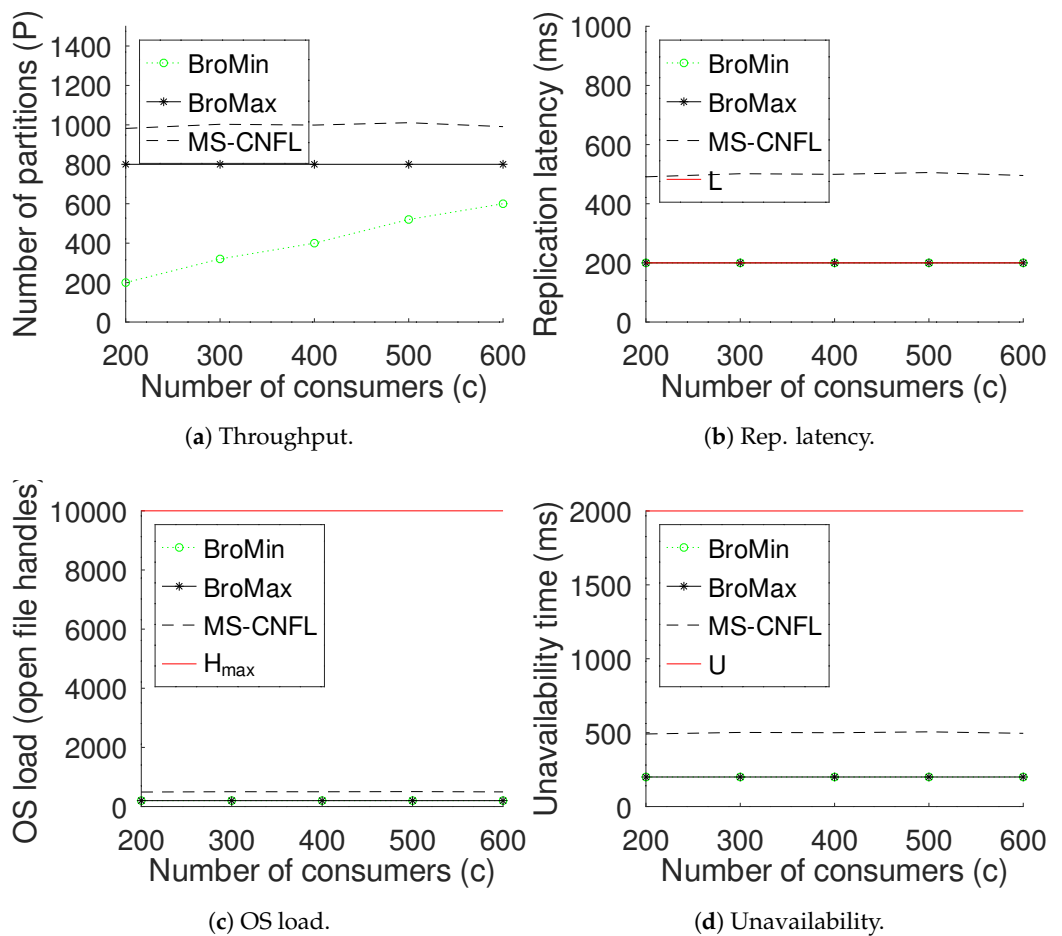


Figure 9. Results of the simulated measurements for HDD component.

6. Design Guidelines

In this Section, we report some useful guidelines that reflect the experiences that we had when building our platform.

- Distil the data exchange requirements of the involved components to consolidate the necessary I/O interfaces as much as possible and consequently reduce the integration complexity.
- Decouple as much as possible the direct data exchange between pairs of individual component instances to reduce the integration complexity, i.e., avoid the use of REST APIs wherever possible and promote the use of pub/sub distributed messaging systems.
- Implement open, industry-standard approaches for increased interoperability, scalability, and expandability.
- Align the data models used for handling and storing the inference results with the SDM standard in order to improve the visibility and acceptance of the envisioned results.
- Achieve a versatile, yet consistent and coherent solution that can support a multitude of different use cases and scenarios and operate on different infrastructure configurations. In our case, this is achieved through the design of the DMP and the specification of an adaptive reference “AI Inference Pipeline” architecture. The DMP is fully scalable and interoperable as it can be adapted to incorporate virtually any number of edge and fog nodes, while it can handle data emerging from any H2020 MARVEL project component (e.g., anonymisation components, AI components, etc.) at any layer of the E2F2C continuum.
- Handle multimodal raw (AV) and structured (inference results) data by collecting from and distributing among multiple endpoints both in real time and asynchronously via persistent storage mechanisms.
- Maintain an up-to-date comprehensive documentation of the specifications for all the implemented I/O interfaces and data models using version control. In our case, a GitLab repository was used for this purpose.

7. Limitations

The DMP has been designed and realised within an international collaborative R&D project, which has allowed us to freely experiment with cutting-edge technologies following a revolutionary, rather than evolutionary, approach in a heterogeneous cross-disciplinary environment. While this setting is ideal to inspire and foster new ideas compared to a more traditional compartmentalised scenario typical of (especially large) industries, with hindsight we can identify some possible weaknesses of our framework when deployed in a large-scale production system related to: universality, security, reliability, and scalability. We discuss each item separately in the following.

Universality. First, we have designed the DMP with the system and technical requirements of the project’s use cases in mind. The latter have been carefully selected to cover a broad selection of applications currently of high impact in smart cities, with the help of end users involved in the project. However, it is evident that the landscape of Information and Communication Technologies (ICT) for smart cities already today is extremely fragmented and heterogeneous, due to the high variability of size/density factors and geographical, historical, and cultural differences, even within the European Union only. Therefore, we could not have aspired to design a one-size-fits-all solution. Instead, we have defined a modular structure, with the goal of supporting a swift replacement of individual components that might be needed to address a specific unforeseen need in a future deployment. Nonetheless, whether the DMP is the ideal solution for a given analytics application in a specific smart city deployment remains to be evaluated on a case-by-case manner.

Security. As it is known by whomever is familiar with security in ICT, a system is as weak as its weakest component. Therefore, we have identified the HMI and, in particular, the access to the data corpus, as the most critical component security-wise. Accordingly, the state-of-the-art solutions for encryption and user authentication have been put in place, as described in Section 4. However, emerging technologies invariably carry new threats: this is true, in general, for smart city technologies [35] and, in particular, for edge computing environments [36]. The design of the DMP adopts a neutral perspective towards such

issues, since we assume that the best practices are being followed at every step of the deployment of the components involved, both internally and externally. Such an approach may not be sufficient for all use cases depending on the threat models and local regulations, hence it may have to be revisited before going to production.

Reliability. In practice, most smart city services run for a very long time (years) after their initial deployment. Therefore, it is very likely that during their lifetime they will experience all kinds of difficulties, from hardware failures to network disconnections, from power outages to system-wide upgrades. Most individual components of the DMP natively support redundancy and are very resilient to sudden unexpected behaviours because they are based on mature and industry-grade software. However, guaranteeing availability was not one of the objectives of the H2020 MARVEL project, which aimed at achieving a prototype-level technology readiness. Long-term evolutions of the DMP based on serverless computing can help increasing the reliability of the framework as a whole; we elaborate more on this in Section 8 below.

Scalability. A common trait of most smart city developments is that it happens incrementally over time: except for those rare occasions where a new city is founded, the city councils and other local administrators usually allocate a (variable) amount of budget to be spent over a time span of many years for the installation of new services. Often, it is difficult to predict at Day 0 the final amount of resources that will be employed, thus scalability by design is important to allow for frictionless and organic growth. As for reliability, most of the DMP components, especially those in the fog and cloud layer, are highly scalable both vertically, i.e., when using a more powerful computation element (e.g., increasing the virtual resources of a VM) and horizontally, i.e., when adding more replications of the same software element (e.g., in a K8s (<https://kubernetes.io/>, accessed on 16 January 2023) cluster). However, the components in the edge layer may suffer from scalability issues, in terms of processing power, energy demands, and network bandwidth, which are all very limited resources in this tier. We foresee that a transition to serverless computing may ease this weakness, as discussed in the next section.

8. Outlook

The MARVEL framework relies on distributed computation across the edge–fog–cloud continuum. To achieve this, enough resources must be available for the required usage of the components. It is worth noting that each of the nodes (at the cloud, fog, or edge) should be able to accommodate at least an MQTT broker and NiFi/MiNiFi and one inference model instance per camera/microphone stream used in the node. The current solution relies on deploying the anonymisation components also at the edge for privacy concerns, as well as a RTSP server to stream the AV data to the rest of the components of the framework after that. This imposes some minimal computational constraints, especially at the edge nodes.

On the other hand, the framework is extensive enough to incorporate new AV streams by adding new pipelines composed of anonymisation components, RSTS streams, and inference models that output their results to the existing or new MQTT brokers (inference data) or to StreamHandler (AV data). Another point of extensibility is the possibility of adding new devices (e.g., fog or edge nodes) to the Kubernetes network, using MARVDash to facilitate the deployment of the MARVEL components in the new devices. The addition of inference models requires a more work, as appropriate pipelines should be set in DatAna to transform the outputs of the models to the Smart Data Models adopted in MARVEL, new topics for the models should be added to Kafka, and the DFB needs to store the new data into the Elastic Search. In this last case, new visualisations might need to be designed if the new data require it.

The framework is based on microservices, i.e., containerised applications that are deployed across the computational resources in the E2F2C tiers. In this section, we provide a perspective on the long-term evolution of the framework by discussing a possible alternative software architecture, based on *serverless computing*, which could be used for data management in smart city applications. Serverless computing is a mature and trend-

ing technology in cloud computing [37], which greatly simplifies the orchestration and management process by providing the developers with a neat and simple abstraction: everything is a function. The developers code elementary functions, which are invoked by client applications or other functions, typically using HTTP methods. For this reason, the programming model of serverless computing is known as Function as a Service (FaaS). The number of instances deployed for a given function is dynamically adapted using autoscaling features so that only the resources actually needed at a given time are used; it is even common to enable so-called *scale-to-zero*, which means that if a function is not invoked for some time then all its instances are evicted from the run-time execution platform, which is very energy-efficient [38] even though it can lead to high-tail latencies due to cold-start effects [39]. Finally, the functions can be composed at run-time to produce the intended behaviour and each function can be substituted dynamically by onboarding a new version on the serverless platform, without the need to stop or restart the overall service: the resulting system is highly scalable and reliable.

Serverless computing is efficient in the cloud for both stateless and stateful functions. A stateless function produces an output (a return value) that only depends on the input (arguments) provided, while a stateful one also has some application- or session-specific state that contributes to how the logic of the function behaves. In the cloud, the state is typically kept in an in-memory or storage service, which can be accessed by stateful functions as needed to retrieve, and update, the state upon each function invocation. The process is efficient due to the logical proximity of the serverless platforms and state management services and their fast interconnection in data centres. However, at the edge and fog layer, which are common deployments for smart city applications, the situation can be very different: reaching remote state management services can be costly, in terms of access fees and latencies, and cumbersome, due to the limited bandwidth available to reach the Internet and public cloud services. Think for instance of the archetypal multi-modal analytics application illustrated in Figure 4: the inference components may need a semi-persistent state to process the audio-video across a time window and the whole neural network model can be considered an internal state during the training phase of the analytics applications. Research is ongoing on how to optimise state management in serverless platforms in the edge and fog layers: for instance, Cicconetti et al. proposed different models to achieve this goal in [40], but the studies in the literature are still in their infancy compared to the maturity level of the serverless-in-the-cloud technology.

In the long term, with specific reference to the multi-modal smart city analytics applications in the H2020 MARVEL project a possible deployment scenario could be the following:

- **Edge tier:** only deploy stateless functions, so that there is no latency due to accessing state management services; the orchestration platform has maximum flexibility to autoscale function instances, which leads to the highest gains in terms of the reduced overprovisioning of (costly) resources and energy consumption.
- **Fog tier:** deploy stateless and stateful functions as needed. The stateful functions can retrieve/update the state residing in a local service at the fog layer itself. If the state of an application is *ephemeral*, i.e., it is only relevant for a limited amount of time and a given location, then there is no need to synchronise it periodically with a global state of the system in the cloud, but such an operation is required in general for a persistent state (e.g., a neural network model under training).
- **Cloud tier:** state-of-the-art serverless platforms can be used to efficiently deploy and run all kinds of functions.

A further optimisation to be considered is that the stateless vs. stateful behaviour of a function can be temporary. For instance, with *continual learning* applications, which are of interest for smart cities when the analytics function has to be adapted to the environment over time, a component can be stateless at times (during the inference phases) and stateful at other times (when there is a need to retrain the model). In this situation, a smart orches-

tration can obtain full benefits in terms of performance while keeping the overprovisioning of the resources low.

9. Conclusions

In this paper, we have illustrated the main design and implementation features of two core components of the H2020 MARVEL project: the DMP, which is a framework for the management of real-time multi-modal data flows across the edge–fog–cloud layers, and the Data Corpus-as-a-Service component, which is a public repository of processed audio–visual open data acquired through the DMP in the project’s pilots. The structure of both entities is modular and flexible and they consist mainly of sub-components relying on mature open source technologies. A succinct report of the validation phase has been included, together with a high-level description of the technical limitations, in terms of universality, security, reliability, and scalability; furthermore, we have provided design guidelines derived from our lessons learnt along a two-year journey in the project.

At the time of writing, the work in the H2020 MARVEL project is still ongoing, but we foresee that many research directions will remain open even after its completion. These include: smooth integration of the user in the design of applications through low-code or no-code approaches, following a human-centric approach; efficient coexistence of heterogeneous flows with different bit-rates (from few kb/s for audio to many Mb/s for video) and characteristics (sporadic short messages for alert application vs. continuous streams for raw sensed data); arbitrary scalability and reliability for long-running systems; and identification of smart city specific security threats that are applicable to the data distribution and management. A promising research direction is the migration to a serverless computing paradigm, which we have briefly discussed in the paper.

Author Contributions: Conceptualization, T.P.R., C.C., M.F., G.K., T.K. and T.P.L.; Methodology, T.P.R., C.C., G.K., T.K. and T.P.L.; Software, T.P.R., M.F., G.K. and T.P.L.; Validation, T.P.R. and T.P.L.; Investigation, M.F. and T.K.; Data curation, G.K.; Writing—original draft, T.P.R., C.C., G.K. and T.K.; Writing—review & editing, T.P.R., C.C., M.F., G.K., T.K. and T.P.L.; Visualization, C.C. and G.K.; Supervision, T.P.R. All authors have read and agreed to the published version of the manuscript.

Funding: This work was funded by the European Union’s Horizon 2020 research and innovation programme MARVEL under grant agreement No 957337. This publication reflects the authors views only. The European Commission is not responsible for any use that may be made of the information it contains.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Al-Fuqaha, A.; Guizani, M.; Mohammadi, M.; Aledhari, M.; Ayyash, M. Internet of Things: A Survey on Enabling Technologies, Protocols, and Applications. *IEEE Commun. Surv. Tutor.* **2015**, *17*, 2347–2376. [[CrossRef](#)]
2. Lin, J.; Yu, W.; Zhang, N.; Yang, X.; Zhang, H.; Zhao, W. A Survey on Internet of Things: Architecture, Enabling Technologies, Security and Privacy, and Applications. *IEEE Internet Things J.* **2017**, *4*, 1125–1142. [[CrossRef](#)]
3. Xu, L.D.; He, W.; Li, S. Internet of Things in Industries: A Survey. *IEEE Trans. Ind. Inform.* **2014**, *10*, 2233–2243. [[CrossRef](#)]
4. Islam, S.M.R.; Kwak, D.; Kabir, M.H.; Hossain, M.; Kwak, K.S. The Internet of Things for Health Care: A Comprehensive Survey. *IEEE Access* **2015**, *3*, 678–708. [[CrossRef](#)]
5. Zanella, A.; Bui, N.; Castellani, A.; Vangelista, L.; Zorzi, M. Internet of Things for Smart Cities. *IEEE Internet Things J.* **2014**, *1*, 22–32. [[CrossRef](#)]
6. Duygan, M.; Fischer, M.; Pärli, R.; Ingold, K. Where do Smart Cities grow? The spatial and socio-economic configurations of smart city development. *Sustain. Cities Soc.* **2022**, *77*, 103578. [[CrossRef](#)]
7. Javed, A.R.; Shahzad, F.; ur Rehman, S.; Zikria, Y.B.; Razzak, I.; Jalil, Z.; Xu, G. Future smart cities: Requirements, emerging technologies, applications, challenges, and future aspects. *Cities* **2022**, *129*, 103794. [[CrossRef](#)]
8. Perera, C.; Zaslavsky, A.; Liu, C.H.; Compton, M.; Christen, P.; Georgakopoulos, D. Sensor Search Techniques for Sensing as a Service Architecture for the Internet of Things. *IEEE Sens. J.* **2014**, *14*, 406–420. [[CrossRef](#)]
9. Desai, P.; Sheth, A.; Anantharam, P. Semantic Gateway as a Service Architecture for IoT Interoperability. In Proceedings of the 2015 IEEE International Conference on Mobile Services, New York, NY, USA, 27 June–2 July 2015; pp. 313–319. [[CrossRef](#)]

10. Haque, A.K.M.B.; Bhushan, B.; Dhiman, G. Conceptualizing smart city applications: Requirements, architecture, security issues, and emerging trends. *Expert Syst.* **2022**, *39*, e12753.
11. Bajovic, D.; Bakhtiarnia, A.; Bravos, G.; Brutti, A.; Burkhardt, F.; Cauchi, D.; Chazapis, A.; Cianco, C.; Dall'Asen, N.; Delic, V.; et al. MARVEL: Multimodal Extreme Scale Data Analytics for Smart Cities Environments. In Proceedings of the 2021 International Balkan Conference on Communications and Networking (BalkanCom), Novi Sad, Serbia, 20–22 September 2021; pp. 143–147.
12. Ahmed, E.; Ahmed, A.; Yaqoob, I.; Shuja, J.; Gani, A.; Imran, M.; Shoaib, M. Bringing Computation Closer toward the User Network: Is Edge Computing the Solution? *IEEE Commun. Mag.* **2017**, *55*, 138–144. [[CrossRef](#)]
13. Porambage, P.; Okwuibe, J.; Liyanage, M.; Ylianttila, M.; Taleb, T. Survey on Multi-Access Edge Computing for Internet of Things Realization. *IEEE Commun. Surv. Tutor.* **2018**, *20*, 2961–2991. [[CrossRef](#)]
14. Raptis, T.P.; Cicconetti, C.; Falelakis, M.; Kanellios, T.; Lobo, T.P. Design Guidelines for Apache Kafka Driven Data Management and Distribution in Smart Cities. In Proceedings of the 2022 IEEE International Smart Cities Conference (ISC2), Pafos, Cyprus, 26–29 September 2022; pp. 1–7. [[CrossRef](#)]
15. Wang, P.; Zheng, Z.; Di, B.; Song, L. HetMEC: Latency-optimal Task Assignment and Resource Allocation for Heterogeneous Multi-layer Mobile Edge Computing. *IEEE Trans. Wirel. Commun.* **2019**, *18*, 4942–4956. [[CrossRef](#)]
16. Becker, S.; Schmidt, F.; Kao, O. EdgePier: P2P-based Container Image Distribution in Edge Computing Environments. In Proceedings of the 2021 IEEE International Performance, Computing, and Communications Conference (IPCCC), Austin, TX, USA, 28–30 October 2021; IEEE: Piscataway, NJ, USA, 2021; pp. 1–8. [[CrossRef](#)]
17. Nicolaescu, A.C.; Mastorakis, S.; Psaras, I. Store edge networked data (SEND): A data and performance driven edge storage framework. In Proceedings of the IEEE INFOCOM 2021—IEEE Conference on Computer Communications, Vancouver, BC, Canada, 10–13 May 2021. [[CrossRef](#)]
18. Gupta, H.; Xu, Z.; Ramachandran, U. DataFog: Towards a Holistic Data Management Platform for the IoT Age at the Network Edge. In Proceedings of the 2018 USENIX Annual Technical Conference, Boston, MA, USA, 11–13 July 2018; USENIX Workshop on Hot Topics in Edge Computing, HotEdge 2018, co-located with USENIX ATC 2018; pp. 1–6.
19. Wang, G.; Koshy, J.; Subramanian, S.; Paramasivam, K.; Zadeh, M.; Narkhede, N.; Rao, J.; Kreps, J.; Stein, J. Building a Replicated Logging System with Apache Kafka. *Proc. VLDB Endow.* **2015**, *8*, 1654–1655. [[CrossRef](#)]
20. Chai, X.C.; Wang, Q.L.; Chen, W.S.; Wang, W.Q.; Wang, D.N.; Li, Y. Research on a Distributed Processing Model Based on Kafka for Large-Scale Seismic Waveform Data. *IEEE Access* **2020**, *8*, 39971–39981.
21. Langhi, S.; Tommasini, R.; Valle, E.D. Extending Kafka Streams for Complex Event Recognition. In Proceedings of the 2020 IEEE International Conference on Big Data (Big Data), Atlanta, GA, USA, 10–13 December 2020; pp. 2190–2197.
22. Gütlein, M.; Djanatljev, A. On-demand Simulation of Future Mobility Based on Apache Kafka. In *Simulation and Modeling Methodologies, Technologies and Applications*; Obaidat, M.S., Oren, T., Rango, F.D., Eds.; Springer International Publishing: Cham, Switzerland, 2022; pp. 18–41.
23. Javed, M.H.; Lu, X.; Panda, D.K.D. Characterization of Big Data Stream Processing Pipeline: A Case Study Using Flink and Kafka. In Proceedings of the Fourth IEEE/ACM International Conference on Big Data Computing, Applications and Technologies, Austin, TX, USA, 5–8 December 2017; Association for Computing Machinery: New York, NY, USA, 2017; BDCAT '17; pp. 1–10.
24. Akanbi, A. ESTemd: A Distributed Processing Framework for Environmental Monitoring Based on Apache Kafka Streaming Engine. In Proceedings of the 2020 the 4th International Conference on Big Data Research (ICBDR'20), Tokyo, Japan, 27–29 November 2020; ACM: New York, NY, USA, 2020; ICBDR 2020; pp. 18–25.
25. Falk, E.; Gurbani, V.K.; State, R. Query-Able Kafka: An Agile Data Analytics Pipeline for Mobile Wireless Networks. *Proc. VLDB Endow.* **2017**, *10*, 1646–1657. [[CrossRef](#)]
26. Raptis, T.P.; Passarella, A. On Efficiently Partitioning a Topic in Apache Kafka. In Proceedings of the 2022 International Conference on Computer, Information and Telecommunication Systems (CITS), Athens, Greece, 13–15 July 2022; IEEE: New York, NY, USA, 2022.
27. Sivaraman, E.; Manickachezian, R. High Performance and Fault Tolerant Distributed File System for Big Data Storage and Processing Using Hadoop. In Proceedings of the 2014 International Conference on Intelligent Computing Applications, Coimbatore, India, 6–7 March 2014; pp. 32–36. [[CrossRef](#)]
28. Saraladevi, B.; Pazhaniraja, N.; Paul, P.V.; Basha, M.S.; Dhavachelvan, P. Big Data and Hadoop—A Study in Security Perspective. *Procedia Comput. Sci.* **2015**, *50*, 596–601. [[CrossRef](#)]
29. Park, S.; Lee, Y. Secure Hadoop with Encrypted HDFS. In *Grid and Pervasive Computing*; Park, J.J.H., Arabnia, H.R., Kim, C., Shi, W., Gil, J.M., Eds.; Springer: Berlin/Heidelberg, Germany, 2013; pp. 134–141.
30. Neuman, B.; Ts'o, T. Kerberos: An authentication service for computer networks. *IEEE Commun. Mag.* **1994**, *32*, 33–38. [[CrossRef](#)]
31. Algaradi, T.; B, R. Static Knowledge-Based Authentication Mechanism for Hadoop Distributed Platform using Kerberos. *Int. J. Adv. Sci. Eng. Inf. Technol.* **2019**, *9*, 772. [[CrossRef](#)]
32. Kanyeba, M.; Yu, L. Securing Authentication Within Hadoop. In Proceedings of the 2016 International Conference on Electrical, Mechanical and Industrial Engineering, Phuket, Thailand, 24–25 April 2016. [[CrossRef](#)]
33. Smyrlis, M.; Somarakis, I.; Spanoudakis, G.; Hatzivasilis, G.; Ioannidis, S. CYRA: A Model-Driven CYber Range Assurance Platform. *Appl. Sci.* **2021**, *11*, 5165. [[CrossRef](#)]
34. Payne, M. Processing One Billion Events per Second with NiFi. 2020. Available online: <https://blog.cloudera.com/benchmarking-nifi-performance-and-scalability/> (accessed on 21 June 2022).

35. Ahmad, K.; Maabreh, M.; Ghaly, M.; Khan, K.; Qadir, J.; Al-Fuqaha, A. Developing future human-centered smart cities: Critical analysis of smart city security, Data management, and Ethical challenges. *Comput. Sci. Rev.* **2022**, *43*, 100452. [[CrossRef](#)]
36. Xiao, Y.; Jia, Y.; Liu, C.; Cheng, X.; Yu, J.; Lv, W. Edge Computing Security: State of the Art and Challenges. *Proc. IEEE* **2019**, *107*, 1608–1631. [[CrossRef](#)]
37. Taibi, D.; Spillner, J.; Wawruch, K. Serverless Computing-Where Are We Now, and Where Are We Heading? *IEEE Softw.* **2021**, *38*, 25–31. [[CrossRef](#)]
38. Patros, P.; Spillner, J.; Papadopoulos, A.V.; Varghese, B.; Rana, O.; Dustdar, S. Toward Sustainable Serverless Computing. *IEEE Internet Comput.* **2021**, *25*, 42–50. [[CrossRef](#)]
39. Vahidinia, P.; Farahani, B.; Aliee, F.S. Cold Start in Serverless Computing: Current Trends and Mitigation Strategies. In Proceedings of the 2020 International Conference on Omni-layer Intelligent Systems (COINS), Barcelona, Spain, 31 August–2 September 2020; pp. 1–7. [[CrossRef](#)]
40. Cicconetti, C.; Conti, M.; Passarella, A. FaaS execution models for edge applications. *Pervasive Mob. Comput.* **2022**, *86*, 101689. [[CrossRef](#)]

Disclaimer/Publisher’s Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.