Discover Applied Sciences

Research

Building a modern data platform based on the data lakehouse architecture and cloud-native ecosystem

Ahmed AbouZaid[1] · Peter J. Barclay[1] · Christos Chrysoulas[2] · Nikolaos Pitropakis[1]

Received: 30 July 2024 / Accepted: 3 February 2025
Published online: 22 February 2025

Abstract
In today's Big Data world, organisations can gain a competitive edge by adopting data-driven decision-making. However, a modern data platform that is portable, resilient, and efficient is required to manage organisations' data and support their growth. Furthermore, the change in the data management architectures has been accompanied by changes in storage formats, particularly open standard formats like Apache Hudi, Apache Iceberg, and Delta Lake. With many alternatives, organisations are unclear on how to combine these into an effective platform. Our work investigates capabilities provided by Kubernetes and other Cloud-Native software, using DataOps methodologies to build a generic data platform that follows the Data Lakehouse architecture. We define the data platform specification, architecture, and core components to build a proof of concept system. Moreover, we provide a clear implementation methodology by developing the core of the proposed platform, which are infrastructure (Kubernetes), ingestion and transport (Argo Workflows), storage (MinIO), and finally, query and processing (Dremio). We then conducted performance benchmarks using an industry-standard benchmark suite to compare cold/warm start scenarios and assess Dremio's caching capabilities, demonstrating a 12% median enhancement of query duration with caching.

Article Highlights

- This research creates a blueprint using DataOps, Kubernetes, and Cloud-Native ecosystem to build a resilient Big Data platform following the Data Lakehouse architecture, the base for Machine Learning (MLOps) and Artificial Intelligence (AIOps).
- Using an iterative approach, we architectured and implemented the core of the platform, which is composable and cloud-agnostic, avoiding vendor lock-in, which could run on any Cloud provider or on-premises.
- The initial benchmarking showed that the platform could efficiently handle millions of records, benefiting from Apache Iceberg features.

Keywords Data Lakehouse · Kubernetes · DataOps · Cloud-Native · Big Data · Artificial Intelligence

✉ Ahmed AbouZaid, Ahmed@aabouzaid.com; Peter J. Barclay, P.Barclay@napier.ac.uk; Christos Chrysoulas, C.Chrysoulas@hw.ac.uk; Nikolaos Pitropakis, N.Pitropakis@napier.ac.uk | [1]School of Computing, Engineering & The Built Environment, Edinburgh Napier University, Edinburgh, Scotland, UK. [2]School of Mathematical & Computer Sciences, Heriot-Watt University Edinburgh, Scotland, Edinburgh, Scotland, UK.

عِندَ الصَّباحِ يَحمَدُ القومُ السُّرَى. *(In the morning, the people praise the night's journey) - Arabic Proverb*
"Αρχή ήμισυ παντός". (The beginning is half of everything) - Greek Proverb
"Is obair latha tòiseachadh". (Beginning is a day's work) - Scottish Gaelic Proverb

1 Introduction

Undoubtedly, we live in the "Big Data" era, where data size has multiplied several times in the last twenty years ([7], p. 5), and new data types and formats have emerged from different sources. This exponential increase brought many new challenges where old methods could not cope with that kind of data, which is different in quantity and quality, but also all of its properties, or what is known as the "4 Vs", which are Volume, Velocity, Variety, and Veracity ([16], p. 148).

Furthermore, utilising external services like the public Cloud to handle such data could increase the risk of vendor lock-in ([27], p. 2). Thus, old methods might be unable to deal with that change, especially for data management, which is the first gate to all other data-driven activities like artificial intelligence, machine learning, and business intelligence. For example, the 2018 Kaggle Machine Learning & Data Science survey showed that data professionals spent most of their time cleaning data (23%) [22]. As a result, investing in building a modern data platform and defining its characteristics, such as its architecture, technologies, and methodologies, are crucial for many organisations.

A Data Management Platform (DMP) is a central place to manage data from one or more sources. The definition of the DMP varies, depending on the domain ([4], pp. 16–17). Generally speaking, DMPs could be specialised, like a DMP to manage research data, or could be generic, such as managing different kinds of data (typically, data associated with the business, regardless of its type). The current research focuses on the generic DMP, which ideally should work with structured, semi-structured, and unstructured data. As the data has grown, the DMP architecture evolved over time to provide added value to businesses, namely Data Warehouse, Data Lake, and finally, Data Lakehouse.

The Data Warehouse (DWH) started in the 1980s and has been used as a suitable location to store analytical data in a predefined format ([24], p. 2). As specified by Ralph Kimball, who has made significant contributions to defining and polishing the DWH foundation, a Data Warehouse is "a system that extracts, cleans, conforms, and delivers source data into a dimensional data store and then supports and implements querying and analysis for decision making" ([26], p. 22).

DWHs have been used for decades, so they are battle-tested and have had the best optimisations for a long time. Hence, they provide many advantages working as a source of truth, high-quality data, effective querying, and are paramount supporting businesses to embrace data-driven decision-making ([44], p. 492). On the other hand, DWH also has downsides. Firstly, it works only with structured data, which is insufficient to handle the increase in data types and sizes. Also, operational-wise, DWHs are very expensive; in addition to proprietary software licensing, the data preparation and clean-up processes require a lot of time and resources. These resources are historically used to run on on-premise hardware, adding many operational and maintenance overheads. For these reasons, many enterprises moved to the next generation to deal with these disadvantages, which is Data Lake architecture.

The Data Lake (DL) started around 2010; the term was used for the first time by James Dixon, who described it as "large amounts of heterogeneous data are added from a single source, and users can access them for a variety of analytical use cases" ([19], p. 180). Furthermore, Fang ([17], p. 820) defined the term as "a methodology enabled by a massive data repository based on low-cost technologies that improve the capture, refinement, archival, and exploration of raw data within an enterprise. A data lake contains the mess of raw unstructured or multi-structured data that for the most part have unrecognised value for the firm".

With the rise of Cloud Computing and cheap commodity storage solutions, DL was able to handle the new Big Data cases (where data comes in different types and sizes); it was also able to store structured and semi-structured formats, and at the same time, put the cost under control. Furthermore, business-wise, another advantage of DL is that it can store more data without knowing the use-cases in advance. At the same time, it provides an easy way to mine the data to find new business cases since it is in a raw format. Nevertheless, when talking about DL disadvantages, and because of its ability to store different types of data, it is easily turned from a "lake" to a "swamp" where the DL is more or less used as a place to dump data. Moreover, due to the variety of stored data and the lack of a unified way to create a metadata layer, it is hard to govern and provide fine-grained access to DL resources. Finally, the DL performance is not unified because the underlying data vary in size, type, and format ([19], p. 180). For these reasons, a new generation emerged to overcome the DL disadvantages, which is Data Lakehouse architecture.

The Data Lakehouse (DLH) is a relatively new hybrid architecture shown for the first time in 2020 which combines the capabilities of a DL and a DLH simultaneously. It is designed to handle both structured and unstructured data in a

scalable, flexible, and cost-effective way. The research by Armbrust et al. [1] defines DLH as "a data management system based on low-cost and directly-accessible storage that also provides traditional analytical DBMS management and performance features such as ACID transactions, data versioning, auditing, indexing, caching, and query optimisation. Lakehouses thus combine the key benefits of data lakes and data warehouses: low-cost storage in an open format accessible by various systems from the former, and powerful management and optimisation features from the latter" ([1], p. 3). By inspecting that definition, it shows how DLH architecture promises to fix DL weakness while providing all DWH strengths as described in the previous sections.

Additionally, the change in the data management architectures was accompanied by changes in storage systems since the traditional Big Data storage formats like Parquet and ORC are optimised for read-heavy workloads and lack support for updates and deletes. Thus, there was a need for a layer on top of them to maintain data quality and consistency. Hence, the new DLH architectures "centre around open storage formats such as Delta Lake, Apache Hudi and Apache Iceberg that implement transactions, indexing and other DBMS functionality on top of low-cost data lake storage (e.g., Amazon S3) and are directly readable from any processing engine. Lakehouse systems are quickly replacing traditional data lakes" ([25], p. 1).

Open table formats like Apache Hudi, Apache Iceberg, and Delta Lake are designed to improve data management for Big Data workloads where these formats provide a separation between computing and data. Overall, these open table formats provide organisations with advanced data management capabilities that help improve data quality and consistency, increase productivity, and reduce costs. To deal with Big Data workloads, the new formats provide fundamental advantages like transactional processing, schema evolution, data versioning, and time travel [1, 25, 42]. As of early 2023, all three open table formats (Apache Hudi, Apache Iceberg, and Delta Lake) have seen significant usage in production environments and adoption across the majority of vendors like AWS, Google, Azure, Snowflake, Databricks, Dremio, and Cloudera, which is a promising indicator to widely support the new open formats [8]. However, it is worth pointing out that even though the open table formats evolved and provided approximately the same features, one of the latest researches about DLH storage systems by Jain et al. [25] showed that open table formats vary simultaneously with different factors. Hence, it is essential to pay attention to the properties of each format and how suitable they are to the nature of the project or the use case.

Our work aims to investigate the current Big Data challenges and present a modern data platform using Kubernetes and Cloud-Native software in harmony with DataOps methodologies. The significant change in the data management landscape in recent years, influenced by Big Data, requires an organisational transformation that affects the "Golden Triangle": People, Process, and Technology [41]. The contributions of our work can be summarised as follows:

- We focus on two sides of the Golden Triangle, process and technology, to build a blueprint and proof-of-concept for a portable, resilient, and efficient modern data platform using DataOps guidelines, Kubernetes, and Cloud-Native software.
- We compare tools to make a selection for the platform's core components, such as ingestion and transport, storage, and query engine.
- We verify and validate against the defined focus areas using real-world data for COVID-19 cases between 2020 and 2022.
- We assess the preliminary platform's performance using the industry-standard benchmark suite Transaction Processing Performance Council for Decision Support (TPC-DS).

The rest of the paper can be summarised as follows. Section 2 builds the background on manipulating Big Data, DataOps and the latest data management architectures, including Cloud solutions, while outlining related approaches. Section 3 briefly presents our approach and the detailed architecture, while Sect. 4 presents the benchmarking implementation and evaluation. Sect. 5 discusses the results. Finally, Section 6 concludes by giving some pointers for future work.

2 Background and related work

2.1 Big data

A variety of data types has grown rapidly due to increased data sources such as social networks, semantic webs, Internet of Things sensors, and location-based services. Their characteristics are the "4 Vs", Volume, Velocity, Variety, and Veracity

([16], p. 148). Data management is crucial because it is the first step to other data-driven activities. However, there are obstacles to finding an efficient way to manage it ([46], p. 237). That is why data management platforms have evolved to deal with the changes qualitatively and quantitatively. For example, data quality could severely affect the opportunities of Big Data ([9], pp. 78–80), and high quality requires a solid process to handle it. This situation led to the emergence of the "DataOps" approach.

2.2 DataOps

Following the software development domain, where DevOps is the standard approach to operating code, DataOps is an emerging approach advocated by practitioners to tackle data management challenges for analytics ([31], p. 61). DataOps provides a process-oriented approach on top of regular data pipelines, a crucial element in handling Big Data ([34], p. 182). It is often seen as a way to bridge the gap between data engineers, who are responsible for building and maintaining the infrastructure and pipelines that handle data, and data scientists, who are responsible for analysing and interpreting the data to generate insights and predictions ([31], pp. 64–65). In terms of its adoption, DataOps is considered a highly intuitive approach to building a data environment, which motivates newer companies to adopt DataOps with its ease and more remarkable speed compared to established ones, which introduce a significant overhead to their existing data practices ([43], Chapter 1).

2.3 Data management architecture

A Data Management Platform (DMP) is a central place to manage data from one or more sources. The definition of the DMP varies, depending on the domain ([4], pp. 16–17). Generally speaking, DMPs could be specialised for a particular type of data or generic to manage different kinds of data (typically, data associated with the business, regardless of its type). Currently, there are three well-known data management architectures in the market: Data Warehouse (DWH), Data Lake (DL), and Data Lakehouse (DLH).

Data Warehouse started in the 1980s and has been used as a suitable location to store analytical data in a predefined format ([24], p. 2). As specified by Ralph Kimball, a Data Warehouse is "a system that extracts, cleans, conforms, and delivers source data into a dimensional data store and then supports and implements querying and analysis for decision making" ([26], p. 22). Data Lake started around 2010; James Dixon described it as "large amounts of heterogeneous data are added from a single source, and users can access them for a variety of analytical use cases" ([19], p. 180). Furthermore, Fang ([17], p. 820) defined the term as "a methodology enabled by a massive data repository based on low-cost technologies that improve the capture, refinement, archival, and exploration of raw data within an enterprise. A data lake contains the mess of raw unstructured or multi-structured data that for the most part have unrecognised value for the firm". Data Lakehouse is a relatively new hybrid architecture that was shown for the first time in 2020 and combines the capabilities of a DL and a DLH simultaneously. The research by Armbrust et al. [1] defines DLH as "a data management system based on low-cost and directly-accessible storage that also provides traditional analytical DBMS management and performance features such as ACID transactions, data versioning, auditing, indexing, caching, and query optimisation. Lakehouses thus combine the key benefits of data lakes and data warehouses: low-cost storage in an open format accessible by various systems from the former, and powerful management and optimisation features from the latter" ([1], p. 3). Inspecting that definition shows how DLH architecture promises to fix DL weakness while providing all DWH strengths as described in the previous sections.

2.4 Cloud computing

Cloud computing refers to a service delivery model for information technology in which resources such as software, storage, and computing power are made available over the internet to users on demand. In this model, users do not own or manage the physical infrastructure that supports these resources but instead access them as a utility provided by a third-party service provider. Therefore, Cloud computing allows users to access and use these resources on demand rather than purchasing and maintaining their physical hardware and software ([40], pp. 1–4). Cloud computing could be categorised into three categories ([40], pp. 30–33):

- *Public Cloud* is a computing environment owned and operated by a third-party Cloud services provider, such as Amazon Web Services (AWS), Microsoft Azure, or Google Cloud Platform. Public Clouds are designed to be accessed

over the internet and are generally available to any organisation or individual that wants to use them. They offer a variety of services, including Infrastructure as a Service (IaaS), Platform as a Service (PaaS), and Software as a Service (SaaS). Public Clouds are generally more cost-effective and easier to use than private Clouds, but they may not offer the same level of security or control.

- *Private Cloud* is a computing environment owned and operated by a single organisation for exclusive use. Private Clouds can be implemented on-premises or hosted by a third-party provider. They offer the benefits of Cloud computing, such as scalability and flexibility, but with the added security and control of an on-premises environment. Private Clouds are generally more expensive and require more resources than public Clouds, but they may be necessary for organisations with strict security or compliance requirements.
- *Hybrid Cloud* is a computing environment that combines both public and private Clouds. It allows organisations to use the best of both worlds by using public Clouds for certain workloads and private Clouds for others. For example, an organisation might use a public Cloud for non-critical workloads that do not require high security and a private Cloud for sensitive workloads that need to be kept in-house. Hybrid Clouds offer the benefits of both public and private Clouds, but they can be more complex to manage and require careful planning to ensure that workloads are placed in the appropriate environment.

Because of all the benefits of the Cloud, the adoption rate has soared over time, especially in recent years [33]. Nevertheless, many enterprises have challenges and obstacles using Cloud services, especially data-related ones. According to research conducted on UK organisations, it showed that "data portability and interoperability concerns were the most discussed theme in relation to vendor lock-in" ([35], p. 4) and "while security and governance concerns often can be answered by encryption, and cost concerns can be answered by cost-based decision-making models, vendor lock-in problems stay" ([27], p. 2). The same research from Opara-Martins et al. ([35], p. 10) showed how using a data architecture that relies on standard open data formats could play a key role in avoiding vendor lock-in where "overall, the results indicate that these challenges closely relate to interoperability and data portability issues prevalent in the Cloud environment".

2.5 Cloud-native software

Cloud-Native refers to the architecture and design of an application built specifically to take advantage of the Cloud computing model. In other words, a Cloud-Native application is designed to be scalable and resilient and take advantage of the automatic provisioning of resources provided by Cloud platforms ([14], p. 16). Thus, Cloud-Native applications are often designed to be run on a distributed system and are built using microservices and containers, which allow them to be easily deployed and scaled ([14], p. 17). Cloud-Native software generally works best on Cloud-Native platforms like Kubernetes, an open-source platform for automating containerised application deployment, scaling, and management. Kubernetes provides a way to deploy predictable and scalable applications, making managing and maintaining large and complex applications more accessible. Moreover, Kubernetes has become the de facto standard for container orchestration and is widely adopted in private and public sectors. According to a survey by the Cloud Native Computing Foundation (CNCF), Kubernetes is used by more than 50% of organisations that deploy containerised applications in production ([9], p. 5).

2.6 Related work

There have been different research attempts around data platforms that can cope with Big Data and fit different workloads and use cases. However, since many data technologies are relatively new, like Data Lakehouse, a hybrid architecture described for the first time in 2020, reviewing MDP research results has been limited to the years between 2020 and 2023. Using a mix-and-match technique with keywords like "modern", "data", "platform", "management", "lakehouse", and "architecture", the academic search engine showed that most of the related work are conference papers that cover an overview of the Data Lakehouse architecture without detailing the actual implementation methodology, which is a clear gap in the academic literature.

Harby and Zulkernine [21] provided a comparative review of Data Warehouses and Data Lakehouses architectures, highlighting the advantages and disadvantages of each architecture, as well as their suitability for different types of data and use cases. They found that despite its advantages over Data Warehouses, Data Lakehouse could be cumbersome regarding its management because of its many moving parts, which is not optimally suited for small-scale applications.

Orescanin and Hlupic [37] covered the concept of a Data Lakehouse, proposing a high-level architecture of a Data Lakehouse and the advantages of each component, such as the ability to handle real-time data ingestion and processing, support for Big Data analytics, and improved data governance. These authors argued that Data Lakehouses are a novel step in analytics architecture that can provide significant benefits for organisations seeking to leverage their data for competitive advantage.

Janssen [15] explored data storage architectures in-depth and investigated the value of Data Lakehouse architecture. He included a case study of a company that has implemented a Data Lake and had evaluated moving to a Data Lakehouse architecture. The study concluded that the company would likely experience significant benefits from the Data Lakehouse architecture, including improved data quality, faster data processing, and increased team collaboration. However, it did not address the implementation phase nor use Kubernetes as a platform for the architecture.

Barron-Lugo et al. [3] propose Xel, which shares the same goals as our work and uses state-of-the-art tools and applications to build a portable data platform for data science services. According to the authors, the Xel project could benefit from the MDP as infrastructure for data science services; however, the Xel project only focuses on the application part of the platform and does not propose an end-to-end architecture which should include the infrastructure.

Based on the reviewed related work, all of them only covered high-level information and to the best of our knowledge, no study covered building a generic data platform using DataOps, Kubernetes, and a Cloud-Native ecosystem, especially a platform focusing on specific aspects like openness, portability, and averting vendor lock-in (cloud-agnosticism). Hence, our work tries to fill this gap by proposing and evaluating a data platform that leverages modern practices and technology and helps to create a data-centric business where different personas smoothly interact with the data platform while managing and benefiting from Big Data. Hence, we are architecting and building a resilient infrastructure using Kubernetes and Cloud-Native ecosystem for the data platform, applying at least one of the DataOps principles, and implementing the core of the DLH architecture.

3 Methods

3.1 Specifications

This section aims to define the specifications related to a modern data platform, regarding how it should work, the core requirements, and the focal areas for the initial implementation. In order to do that, we first need to shed light on the high-level goals of such a platform. These are commonly driven by various business objectives, including enhancing decision-making, uncovering customer preferences and patterns, and increasing operational efficiency. According to LaPlante and Safari [28], 97% of businesses have established data strategies and acknowledge the importance of data strategies. However, only 31% have transitioned to data-driven enterprises, and just 28% have established a data culture. 52.4% of businesses do not compete on data and analytics, and more than half (53.1%) do not view data as a business asset. Remarkably, 71.7% do not have a data culture, and 69% have not established a data-driven culture [28].

Adopting a data-driven approach requires organisational transformation and change management. There are multiple frameworks to handle such a change; one is the "Golden Triangle" framework, which Harold Leavitt proposed in the 1960s. The Golden Triangle, or the People, Process, Technology (PPT) framework, highlights the importance of balancing all three components to achieve success and optimal outcomes. Even though the PPT framework is a popular model used in business management and IT, it can be applied to any industry to help organisations improve their business processes and achieve their strategic objectives [41].

This research's primary focus is two sides of the framework, which are Technology (Kubernetes and Cloud-Native software) and the related Process (DataOps), to build a proof-of-concept of a modern data platform according to the typical business drivers, and at the same time overcomes the concerns from previous research regarding Cloud adoption which are rooted into security and privacy ([35], p. 3), availability and reliability ([35], p. 6), compliance and legal Issues ([35], p. 7), data interoperability and portability ([35], p. 9). Also, another critical concern is vendor lock-in, which is a major concern for many industries ([35], pp. 9–10; [27], pp. 1–2). Finally, data management processes challenges such as establishing formal processes, orchestrating code and data across tools, staff capacity, and monitoring the end-to-end environment ([20], p. 29).

3.2 Prioritisation

Taking the high-level goals and requirements of the data platform as a starting point, the focus of this work will be:

a) Architecting and building a resilient infrastructure for the data platform,
b) applying at least one of the DataOps principles, and
c) implementing the core of the DLH architecture.

Accordingly, an iterative approach framework will be used to build a Minimal Viable Product (MVP) to have better quality and reduce the risk of failure. This approach involves developing the system's essential features first, and then gradually adding additional functionalities in subsequent iterations.

The "Must, Should, Could, and Would haves" (MoSCoW) prioritisation method was used to prioritise which features to include in each iteration. This method categorises features into four groups: Must-haves, Should-haves, Could-haves, and Won't- haves (also known as Will-not-haves). Must-haves are essential features for the system to function, while Should-haves are necessary but not essential. Could-haves are desirable but not vital, and Would-haves are features that are needed but explicitly not included in the current iteration ([12], p. 171).

The MoSCoW prioritisation method allows for a more focused and efficient development process while ensuring that the MVP meets the minimum requirements for the system to be usable. As a result, the following are features of the initial version of the data platform in keeping with the MoSCoW method listed in Table 1.

The following section will cover architecting the prioritised specifications.

4 Architecture and implementation

Related work addressed the evolution of data management architectures like DWH, DL, and finally, DLH, where the DLH architecture was considered the new recommended setup to cope with Big Data. In the manner of the following criteria to architect the initial version of the data platform, the architecture should be (a) provider agnostic, (b) generic or cover different use cases, (c) detailed, and (d) up-to-date. A review of six architectures of data platforms created between the years 2020 and 2023 [5, 11, 13, 30, 32, 36] showed that the "Unified Data Infrastructure v2.0" (UDI v2.0) architecture by Bornstein et al. [5] is the best match of the defined criteria.

The following components should be implemented to have an MVP for the data platform based on the Unified Data Infrastructure v2.0, and more components could be added later. The core components covered in this section are infrastructure, data ingestion, storage, and processing.

As noted, Kubernetes is a container orchestration platform that automates the installation, expansion, and administration of containerised applications. Given the fact that it is used by more than 50% of organisations that deploy

Table 1 Lists the focus areas using the MoSCoW prioritisation method

Group	Priority
Must Have	MH1: Cloud-Native architecture
	MH2: Scalable and Cloud agnostic infrastructure orchestration system
	MH3: Open-source software and open standard formats
	MH4: Data Lakehouse solution as a core of the data platform
Should Have	SH1: A declarative approach for configuration management
	SH2: A data pipeline to ingest data from an external source into the platform in plain text formats like JSON or CSV
	SH3: Self-service capabilities
Could Have	CH1: An easy way to rebuild the system with minimal manual actions
	CH2: An open format table like Apache Iceberg from one of the ingested JSON/CSV files
Will not Have	WH1: Production grade quality like high availability, security hardening, or data quality validation
	WH2: The reset of the components not directly related to the core of the Data Lakehouse

Table 2 Comparison between different ingestion and transport solutions based on the inclusion and exclusion criteria

Solutions	Inclusion criteria			Exclusion criteria		
	IC1	IC2	IC3	EC1	EC2	EC3
Apache Airflow	●	●	●	●		●
Argo Workflows	●	●	●			
Dagster	●	●	●			●
Kubeflow Pipelines	●	●	●		●	●
Prefect	●	●	●	●	●	
Tekton	●	●	●			●

containerised applications in production ([9], p. 5), it is considered the most popular container management system. Its features include self-healing abilities, service discovery and load balancing, storage orchestration, configuration and sensitive data management, resource management, and batch execution.

The extensible capabilities offered by Kubernetes can be tailored to meet the unique requirements of the application or organisation, aiming to make managing and scaling containerised applications simple and efficient in distributed environments. Since Kubernetes is Cloud-Native and Cloud-Agnostic by design, it is the ideal infrastructure orchestration system for data platforms because it offers a shared layer of abstraction that hides the underlying details to deploy and manage containerised applications easily. Kubernetes can run in various environments, such as on-premises and Cloud providers, including Amazon Web Services (AWS), Microsoft Azure, Google Cloud Platform (GCP), and many others.

4.1 Architecture aspects

As Kubernetes is used as an infrastructure orchestrator, it offers a unified approach, allowing workloads to leverage their capabilities effectively, making it one of its most prominent features. Namely, the Operator pattern, which encapsulates the operational knowledge as code, simplifies the deployment process, improves resource utilisation, increases flexibility, and enhances the security of the applications [45]. Also, Helm is a package manager for Kubernetes that makes deploying, managing, and upgrading applications on Kubernetes clusters easier by providing a standardised way to package and distribute them instead of creating all Kubernetes manifests from scratch to saving time and effort, avoiding reinventing the wheel. For those reasons, applications that have Helm chart and Kubernetes Operator should be preferred whenever possible.

The following inclusion (IC) and exclusion (EC) criteria were formulated to select a solution for ingestion and transport that matches the goals and requirements:

- IC1: The solution provides modern ETL and workflow capabilities.
- IC2: The solution is Cloud-Agnostic and uses Cloud-Native approaches.
- IC3: The solution is customisable and fits different use cases.
- EC1: The solution is not easy to deploy on Kubernetes.
- EC2: The solution does not use a declarative style.
- EC3: The solution is complex or has a high learning curve.

According to the focus areas in the specifications section and the matching of the inclusion and exclusion criteria in Table 2, Argo Workflows[1] appears to be the best current fit for this project as an ingestion and transport solution. Argo Workflows is an open-source Kubernetes-native workflow engine for orchestrating parallel and distributed tasks. It provides a simple and powerful way to define, manage, and execute complex workflows, including data processing, machine

[1] https://argoproj.github.io/workflows

Table 3 Comparison between different object storage solutions based on the inclusion and exclusion criteria

Solutions	Inclusion criteria			Exclusion criteria		
	IC1	IC2	IC3	EC1	EC2	EC3
Ceph	●	●	●		●	
MinIO	●	●	●			
OpenIO	●	●	●	●		●
Riak CS	●	●	●	●		●
Zenko	●	●	●			●

learning, and continuous integration pipelines. By default, it uses the Kubernetes Operator pattern to define workflows as declarative YAML files, making it easy to follow DataOps principles. Furthermore, it supports programmatically writing workflows via its Software Development Kit (SDK) and built-in templates to define common workflow patterns for flexibility and extensibility.

Compared to Apache Airflow and Prefect, which require more complex configurations to work on Kubernetes, Argo Workflows is simpler to deploy. Unlike Kubeflow Pipelines and Prefect, which rely more on imperative approaches, Argo Workflows takes a purely declarative approach, which makes it more aligned with Kubernetes development. In comparison to Apache Airflow, Dagster, Kubeflow Pipelines, and Tekton, which have steeper learning curves because of their more complex structures or larger feature sets, Argo Workflows is typically more straightforward, which helps to adopt and apply the DataOps methodology.

In order to select the storage solution that matches the goals and requirements, the following inclusion and exclusion criteria were formulated:

- IC1: The solution provides scalable, high-performance, and distributed object storage capabilities.
- IC2: The solution is Cloud-Agnostic and uses Cloud-Native approaches.
- IC3: The solution is compatible with the standard Amazon S3 APIs.
- EC1: The solution is not easy to deploy on Kubernetes.
- EC2: The solution is not easy to manage and maintain.
- EC3: The solution does not have a large community and support.

According to the focus areas in the specifications section and the matching of the inclusion and exclusion criteria in Table 3, MinIO[2] appears to be the best current fit as a storage solution for this project. MinIO provides several key features, making it a popular choice for building object storage servers for Cloud-Native applications and infrastructure. Specifically, it provides a highly scalable and distributed storage system to store massive amounts of structured and unstructured data. Moreover, MinIO is Amazon S3 compatible with and built on top of the Amazon S3 API, which means it can be easily integrated with various applications and tools.

For the modern data platform use case, MinIO is better than Ceph because of its ease of use, speed, and focus on object storage. When it comes to a cloud-agnostic platform like Kubernetes, OpenIO and Riak CS do not have any built-in support for Kubernetes. MinIO is comparatively simple to set up on Kubernetes, packaged with Helm chart and even Kubernetes Operator, reducing operational overhead. Also, OpenIO, Riak CS, and Zenko have a smaller community and less extensive support, making it more challenging to get resources and help; MinIO has a larger, more active community with extensive documentation.

In order to select the query and processing solution that matches the goals and requirements, the following inclusion and exclusion criteria were formulated:

[2] https://min.io

Table 4 Comparison between different query and processing solutions based on the inclusion and exclusion criteria

Solutions	Inclusion criteria			Exclusion criteria		
	IC1	IC2	IC3	EC1	EC2	EC3
Cloudera Data Platform	●	●	●		●	
Dremio	●	●	●			
Presto	●	●	●	●		
Trino	●	●	●	●		

Table 5 The core items of the data platform's initial architecture

Item	Role	Category
DataOps	Practices framework	Process
Kubernetes	Infrastructure orchestration	Technology
Argo Workflows	Data ingestion pipeline	Technology
MinIO	Data object storage	Technology
Dremio	Data Lakehouse platform	Technology

- IC1: The solution provides query and processing capabilities.
- IC2: The solution is Cloud-Agnostic and uses Cloud-Native approaches.
- IC3: The solution has an open-source and enterprise version.
- EC1: The solution does not position itself as a DLH platform.
- EC2: The solution is not easy to deploy on Kubernetes.
- EC3: The solution does not support modern open standards like Apache Hudi, Apache Iceberg, and Delta Lake.

In line with the focus areas in the specifications section and the matching of the inclusion and exclusion criteria in Table 4, Dremio[3] appears to be the best current fit as a query engine solution for this project, and it will be the cornerstone of the DLH. Dremio Cloud-Native architecture building blocks are open-source technologies such as Apache Arrow, Gandiva, Apache Arrow Flight and Apache Iceberg. These technologies provide several key features for building DLH, including a high-performance query engine, self-service data access, built-in data governance and security. Both open-source and enterprise versions of Dremio are offered, with the latter offering more features and functionalities for enterprises. For those reasons, Dremio is a popular choice for organisations looking to build a self-service DLH platform that can provide fast and efficient access to a wide range of data sources.

Dremio is intended to be a complete Data Lakehouse platform, in contrast to Presto and Trino, which are positioned as query engines for distributed data. Presto and Trino concentrate on SQL queries without a unified data platform strategy, whereas Dremio combines data management and processing capabilities to offer a complete solution for querying, governance, and analytics. Additionally, in contrast to Cloudera Data Platform, which requires a more complicated and resource-intensive setup, Dremio provides simple and scalable deployment on Kubernetes. The smooth containerised operations made possible by Dremio's native Kubernetes support increase its flexibility and manageability.

As a result, the following components are selected to be the core of our data platform:

- Kubernetes: Container orchestration platform.
- Argo Workflows: Data pipeline workflow engine.
- Dremio: Data Lakehouse platform.
- MinIO: Object storage.

[3] https://dremio.com

4.2 Initial design

This section covers the initial design after defining the core components that match the goals and focus on traits including portability to avoid vendor lock-in, scalability to deal with massive amounts of structured and unstructured data, and extensibility by using open standards. Kubernetes and Cloud-Native components offer flexibility and scalability, using Cloud technologies and pay-as-you-go pricing as dynamic resource allocation can maximise operational efficiency. However, as mentioned in the background and related work section, "while security and governance concerns often can be answered by encryption, and cost concerns can be answered by cost-based decision-making models, vendor lock-in problems stay" ([27], p. 2); hence, this proof-of-concept focuses on portability and vendor lock-in aspects. Table 5 lists the initial data platform's architecture to be implemented in the next sections. Figure 1 illustrates our initial proposal for the data platform's architecture. More details about the interaction of components are covered in the next section.

4.3 Implementation approach

A proof-of-concept implementation was built using MVP and Agile methodologies, starting simple and iterating to build a better solution more efficiently. This method follows a data-driven approach by focusing on the core features needed to solve a specific problem, releasing those features quickly, and iterating in line with user feedback.

The development was done in a local Kubernetes cluster for faster interactions, but, at the same time, it is implemented to work seamlessly on any Kubernetes Cloud cluster, which will be used for benchmarking in the next section. The following sections show the tools used to implement and develop the infrastructure, such as code to build and run the platform and the Python code that ingests external data into the platform. The data pipeline will use a real dataset which contains the daily subnational 14-day notification rate of new COVID-19 cases between 2020 and 2022 [10]. All code is available in our GitHub repository.

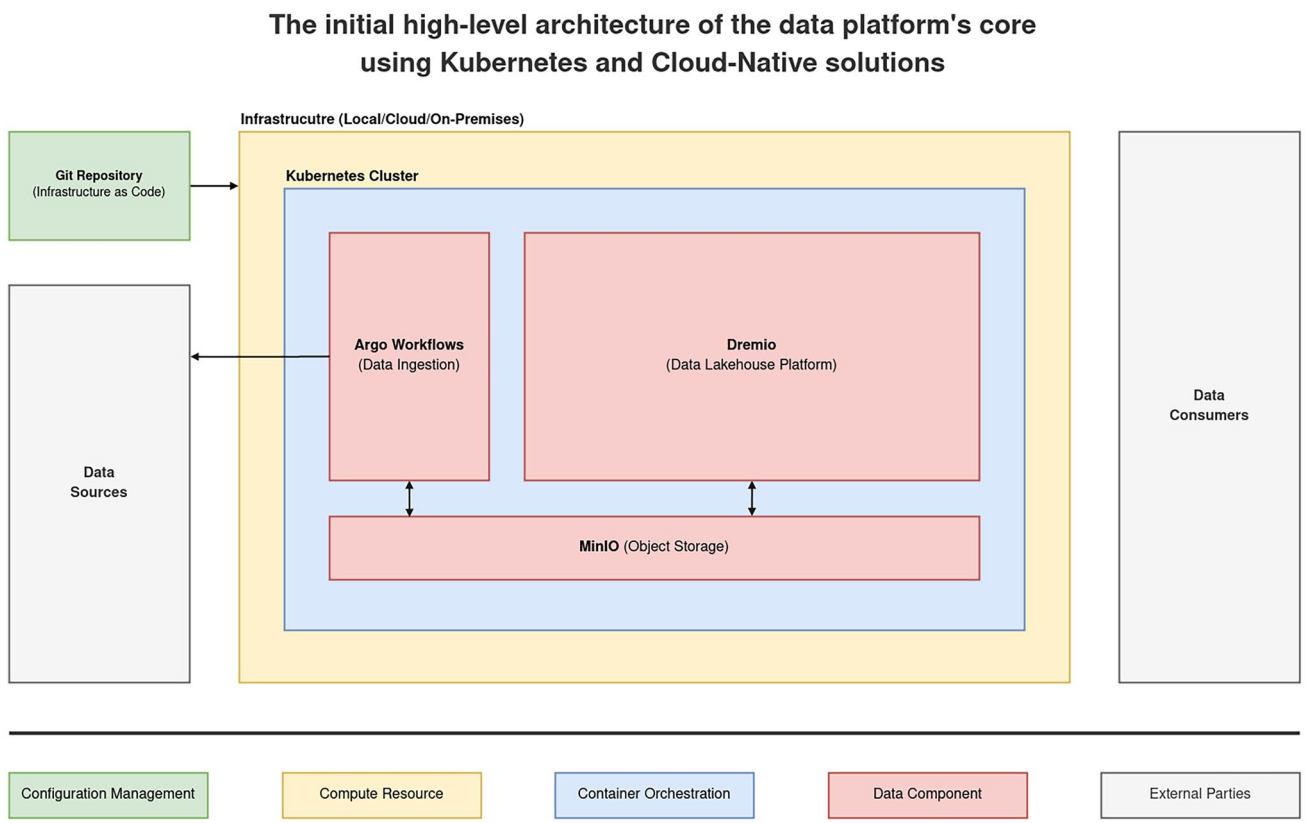

Fig. 1 The initial high-level architecture shows the core parts of the Modern Data Platform using Kubernetes and Cloud-Native solutions

4.4 Initial model

Our proof-of-concept implementation includes the Kubernetes cluster, the data applications (MinIO, Argo Workflows, and Dremio), and finally, a data ingestion pipeline workflow by Argo, in addition to Python. All resources are managed declaratively using Kustomize and stored in a Git repository.

To visualise the interactions of the current implementation, the C4 model will be used. The C4 software architecture model (Context, Containers, Components, and Code) is a visual approach to describe the architecture of a software system developed by Simon Brown in 2011 [6]. It provides a set of hierarchical diagrams that describe the system's different levels of abstraction, from high-level system context diagrams to low-level code diagrams. Figure 2 visualises a high-level platform architecture context, containers, and components (a simplified view where all the abstractions are combined altogether), which will be used in the evaluation section to verify and validate the implementation.

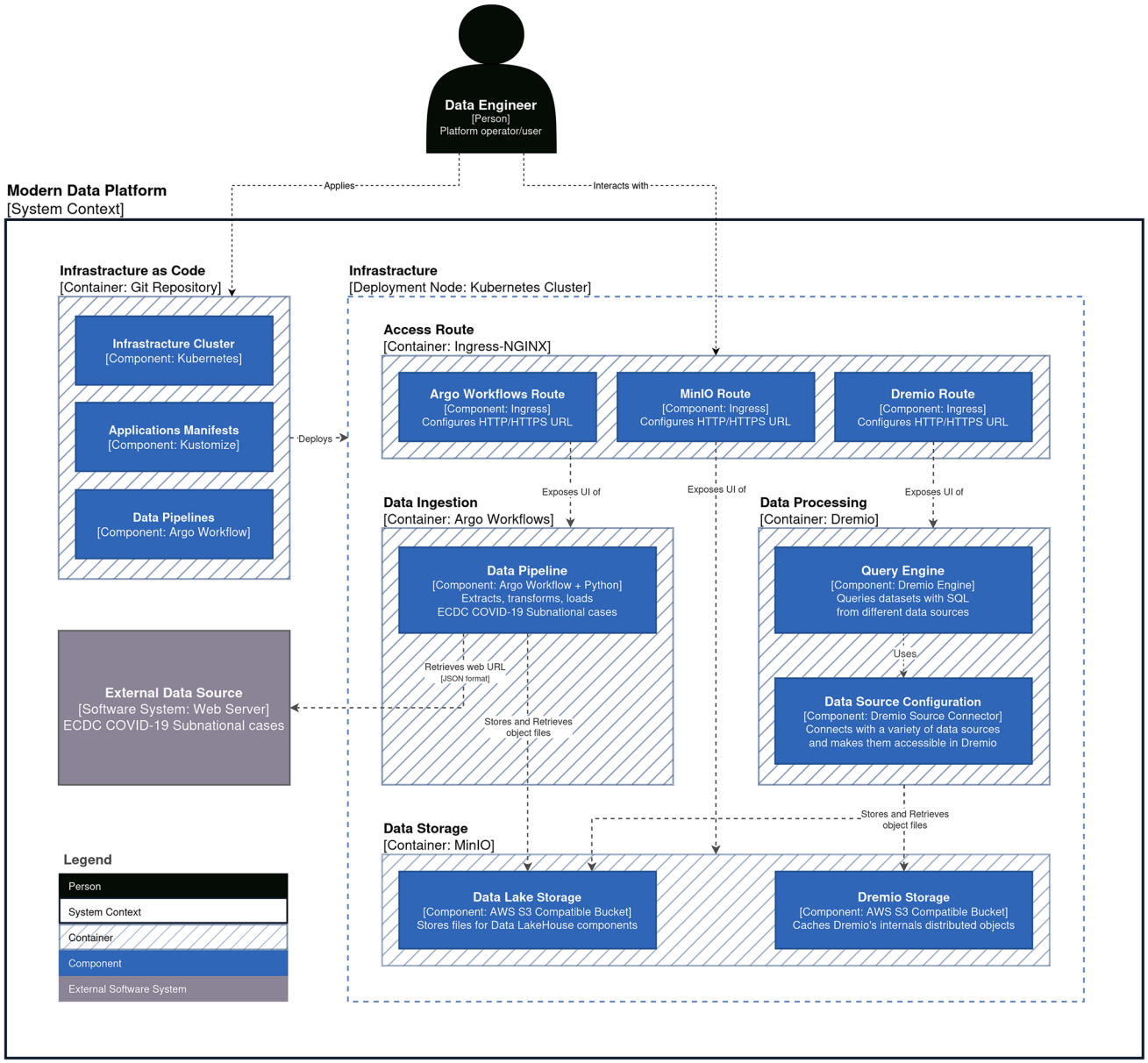

Fig. 2 The data platform's initial model interactions following the C4 model guidelines. (A higher-quality version is available on the project repository)

Based on the initial model architecture (Fig. 2), the Data Engineer plays a key role in orchestrating and configuring the entire data pipeline. As an actor, the Data Engineer defines the data pipeline as code using Argo Workflows and manages the infrastructure resources in a DataOps manner, where all changes are tracked and managed via Git. Hence, the pipeline's configuration and manifests are stored in the Git repository; then, the Data Engineer applies these manifests to the Kubernetes cluster. Once applied, Argo Workflows consumes the pipeline definition and orchestrates the process within Kubernetes.

Figure 3 illustrates the data flow once the data pipeline is configured. The data pipeline is designed to ingest a configurable external data payload (via the Internet), which "ECDC COVID-19 cases subnational" used as an example. Once Argo Workflows retrieves the data, the pipeline processes the data using Python code, transforming and filtering the cases based on the year. The processed data is then stored as JSON files in MinIO, which works as a Data Lake storage.

As the data ingestion and storage phases are done, Dremio connects to MinIO and creates an Apache Iceberg table based on the JSON files, allowing structured queries to be performed efficiently on the data. At this point, the data processing phase is also done, and consumers are ready to query data through Dremio's connectors and integrations. This workflow allows efficient ingestion, processing, storage, and querying of any semi-structured data, empowering users to extract insights in a scalable manner. The technical resources and step-by-step instructions are provided in the project repository.

5 Technical evaluation and benchmarking

5.1 Technical evaluation overview

After finishing the platform specifications, focus areas, architecture, and implementation, the technical evaluation section plays a crucial role in matching the targets and actual implementation. Standard evaluation practices, namely verification and validation, are two essential activities in the software development life cycle to ensure that software meets the required quality standards. Verification is the process of checking whether a software product meets the specifications and requirements of the design. Validation is the process of checking whether the software product meets the needs and expectations of the customer ([18], pp. 3–5). Afterwards, a benchmark will be executed to assess Dremio's performance with real-world scenarios.

5.2 Verification and validation

This section is used to confirm that the core components of the platform have been implemented in line with the targeted architecture Unified Data Infrastructure v2.0, which are Infrastructure (Kubernetes), Ingestion and Transport (Argo Workflows), Storage (MinIO), and finally, Query and Processing (Dremio).

To validate the initial design implementation, the platform was deployed, accessed, and used the same way the end personas would use it. The next sections will review various actions related to different components of the platform (Kubernetes cluster, data applications, and data pipeline) to ensure that the platform meets the needs and expectations of the end users. The code and configuration used in this section come from the project Git repository

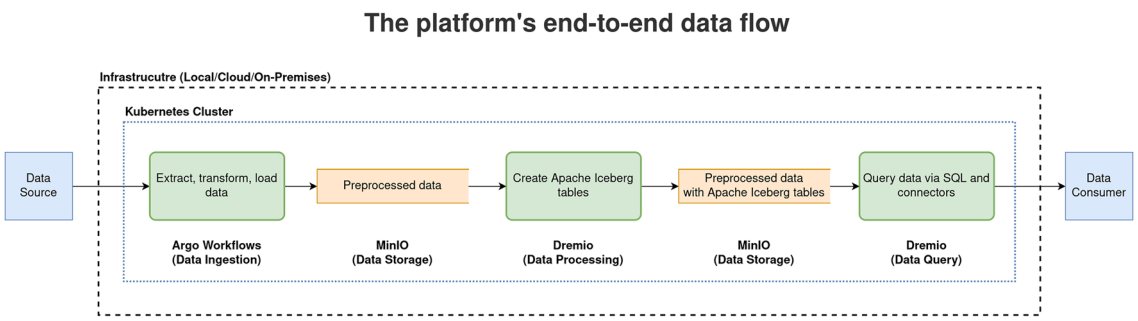

Fig. 3 The data platform's initial model data flow. (A higher-quality version is available on the project repository)

Table 6 Benchmarking compute resources for Dremio and MinIO

Component	CPU	RAM	Disk
Dremio Master	4	8G	10G
Dremio Executor	4	8G	50G
Dremio ZooKeeper	1	1G	1G
MinIO	2	4G	250G
Apache JMeter	4	8G	200G

Table 7 A sample of Dremio's TPC-DS benchmark

Label	Duration cold (s)	Duration warm (s)	Duration diff (%)
q7	42.293	34.041	– 19.51
q71	15.017	13.685	– 8.87
q63	17.296	13.226	– 23.53
q91	6.089	1.396	– 77.07
q84	6.201	4.271	– 31.12

"modern-data-platform-research-paper",[4] which is accessible publicly. After validating all platform functionalities based on the section Initial Model, the next section focuses on evaluating the platform's query performance.

5.3 Benchmarking overview

Performance assessment is an essential part of the evaluation to verify the capabilities of any data solution. Benchmarking against data applications typically involves creating a standardised test environment and running a series of tests against the system using a set of predefined workloads and performance metrics. Here, we review the platform performance, namely, default Dremio caching capabilities. Now, we will turn to the benchmarking process, starting with the framework used for the benchmarking, then preparing the test environment and data generation, then executing the tests, and finally, the benchmark results.

For that use case, one of the recognised industry standard consortia will be used, the Transaction Processing Performance Council (TPC), which develops and sets standards for benchmarks. ([39], p. 10). Namely, the Transaction Processing Performance Performance Council for Decision Support (TPC-DS) will be used to benchmark the platform for this research. The TPC-DS suite is one of the most well-known benchmarking suites in the industry ([39], p. 10). Companies and vendors widely adopt TPC-DS in the industry to demonstrate their competencies to support advanced decision-support systems like data platforms ([2], p. 623). The suite is designed to measure the performance of decision support systems (DSS) that examine large volumes of data and uses a complex schema and model data with various queries that simulate business intelligence queries that might be performed in a real-world scenario. Also, it includes a wide range of query complexities, data sizes, and data types, making it a robust measure of DSS performance ([38], pp. 1138–1139). The TPC-DS tests include ninety-nine queries in four broad groups that characterise most decision support queries: reporting, ad hoc, iterative OLAP, and data mining ([38], p. 1140).

Since Dremio is the benchmarking target, the benchmark guide by Leontiev [29] will be followed for data generation and preparation. As stated in the Leontiev's guide, only fifty-six queries are supported out of ninety-nine queries without making any changes in the queries, and the rest of the queries need some logical query rewrites to work. Even so, for this research project, the benchmark will use the fifty-six out-of-the-box queries. Furthermore, the benchmarking will use a dataset of 100 gigabytes, which is considered a relatively small dataset for Data Lake/Lakehouse. Still, it will provide a baseline and an indicator for Dremio's performance, especially with limited computing resources, given that advanced performance benchmarking is out of the scope of this research as it requires higher computing resources. Table 6 lists the resources used for each component.

Based on the benchmark guide, the data generation was followed by adding CVS headers to the data files, uploading them to the MinIO bucket, loading them in Dremio, and finally creating them as datasets in Apache Iceberg format. For

[4] https://github.com/aabouzaid/modern-data-platform-research-paper

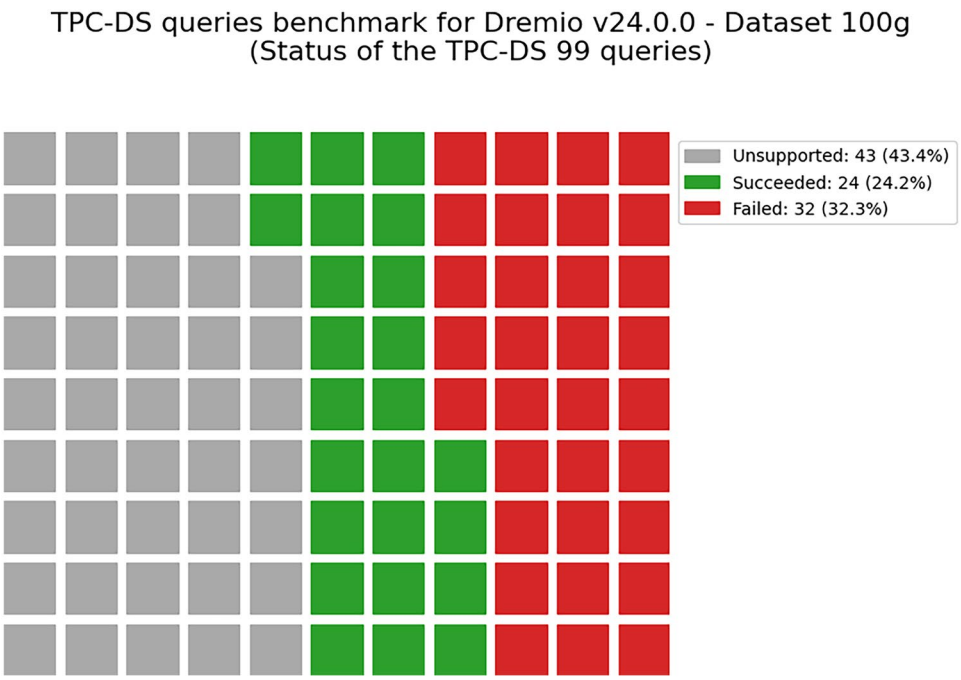

Fig. 4 The status of the TPC-DS 99 queries by Dremio

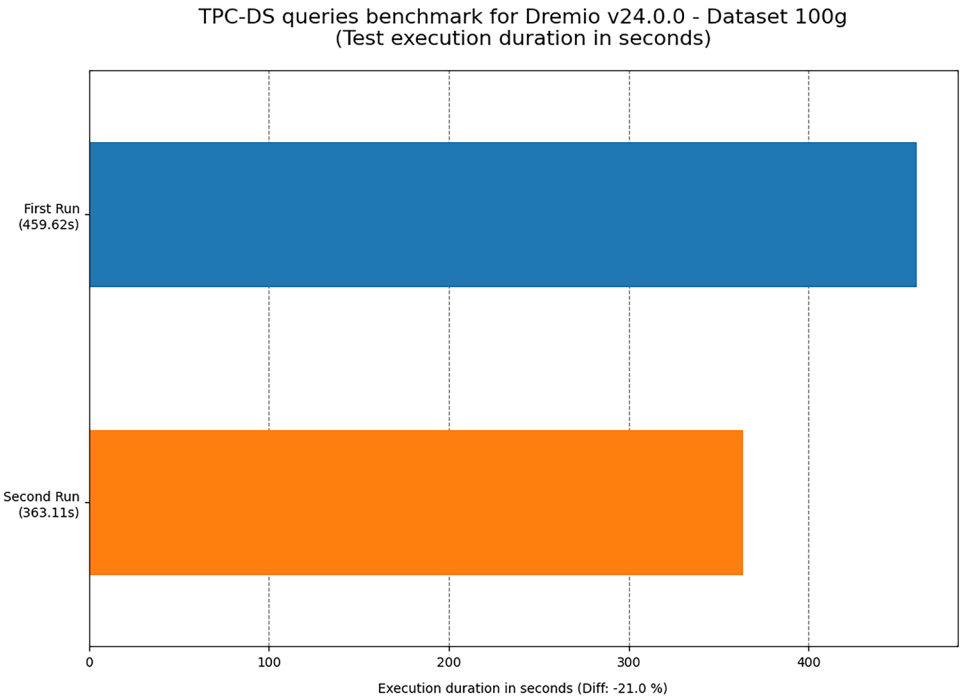

Fig. 5 Test execution time in seconds with a decrease observed in execution duration when the cache is available

the benchmark execution, Apache JMeter v5.5 is used to run the test plan, which will be executed in consecutive runs to record results with or without cache available (cold and warm queries). All the benchmarking details and step-by-step documentation can be found in the project repository.

5.4 Benchmarking outcomes

This section focuses on better understanding TPC-DS benchmark results; thus, a Python Jupyter Notebook was created to plot the query performance using Pandas and Matplotlib. The Python code and the Jupyter Notebook can be found in

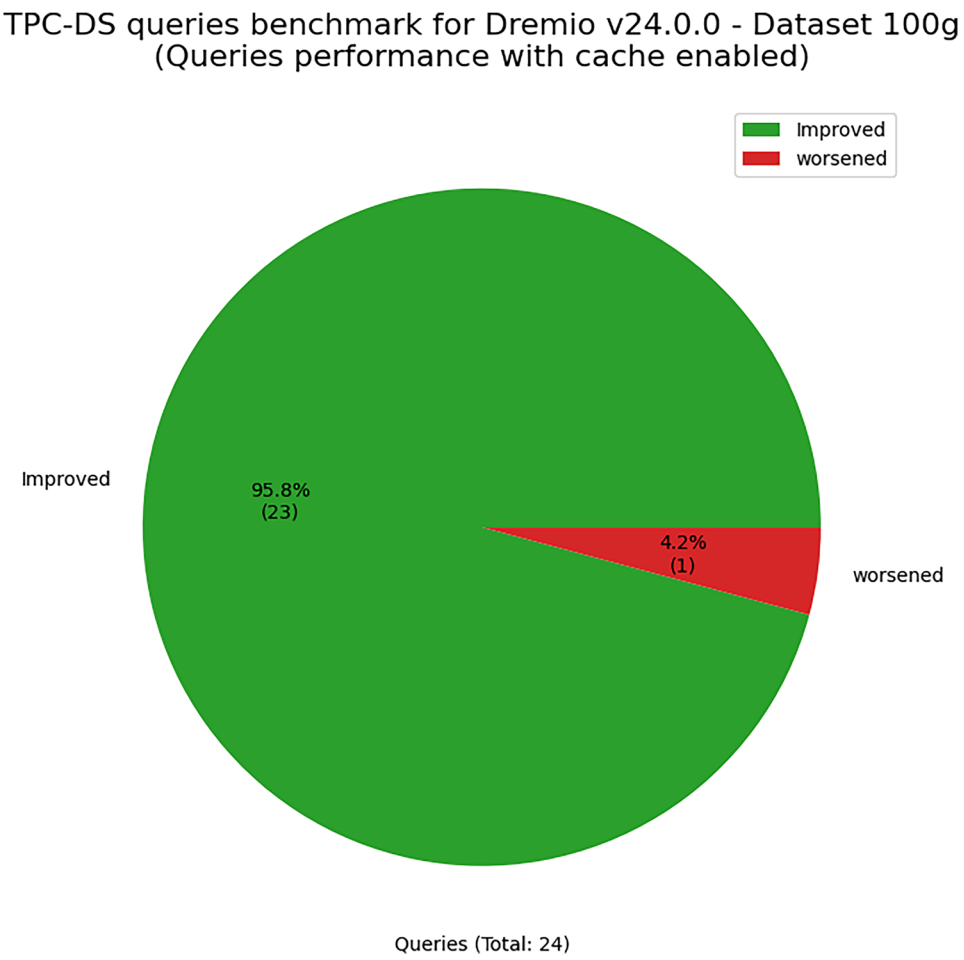

Fig. 6 TPC-DS queries performance with cache enabled

the project repository "modern-data-platform-research-paper" under the "benchmarking" directory[5]. First, Table 7 lists a sample of the data after cleaning and merging the cold and warm query results and calculating the difference percentage between them. Next, various charts give insights into the benchmark results.

The chart in Fig. 4 illustrates the status of the TPC-DS 99 queries, in Dremio, according to the benchmark guide by Leontiev [29], 43 queries are unsupported, and 56 queries are supported. However, during the execution, only 24 queries succeeded, and 32 failed. Based on the initial investigation, most of the queries failed due to the limited resources, namely Dremio executer machine memory.

Turning to the test execution duration, Fig. 5 illustrates a 21% decrease in the duration for the second run where the cache is used.

In general, caching aims to enhance performance, but sometimes, caching introduces overhead and worsens performance. As a result, to validate that, the difference between cold and warm runs was calculated to identify any positive increase in duration as an indicator of worsened performance. Figure 6 shows the performance of the TPC-DS queries where three queries' performance worsened when they ran for the second time and leveraged the cache. With the improvement in the remaining queries, the degraded performance of the three could be accepted as they account for only 4.2% of the total queries. It is worth mentioning that each query engine is different. In real-life scenarios, the queries could be optimised or rewritten based on the engine characteristics for better performance [23].

The performance statistics have been plotted to understand the benchmark performance dynamics better. Figure 7 gives a high-level overview: The best query performance was 77.1% by query ID 91, and the worst was 2% by query ID 55. Moreover, 25% of the queries saw improvements greater than 24%, while 75% experienced enhancements of over 8%. Finally, the median enhancement was approximately 12%. Figure 8 shows the distribution of the queries in more detail.

[5] https://github.com/aabouzaid/modern-data-platform-research-paper/tree/main/benchmarking

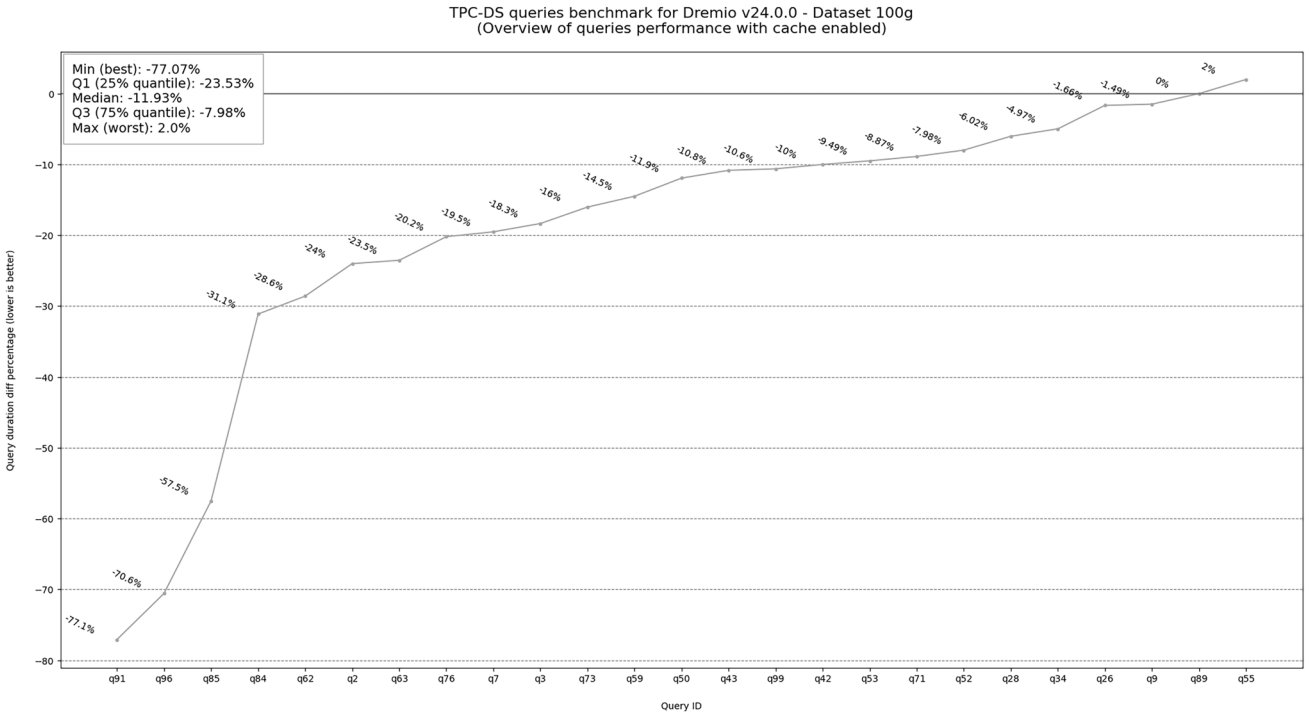

Fig. 7 Overview of query performance with cache enabled

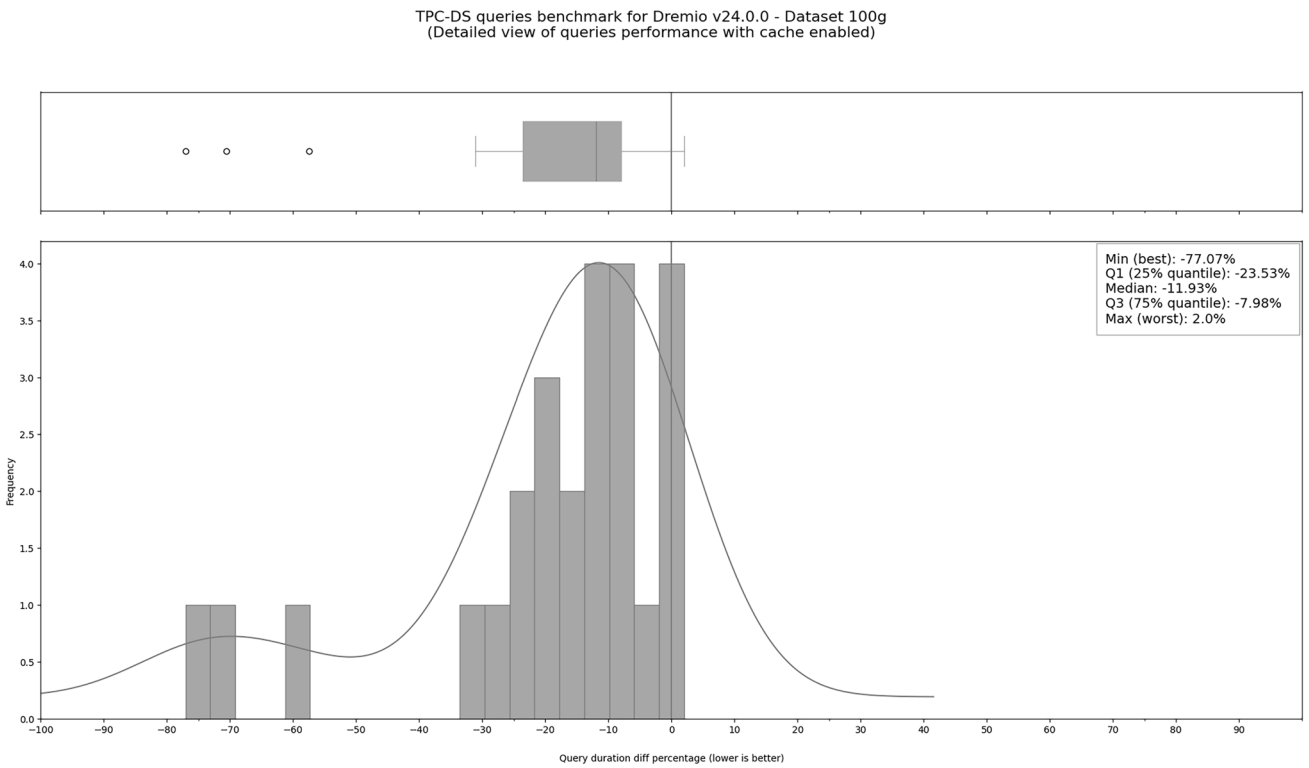

Fig. 8 Detailed view of query performance with cache enabled

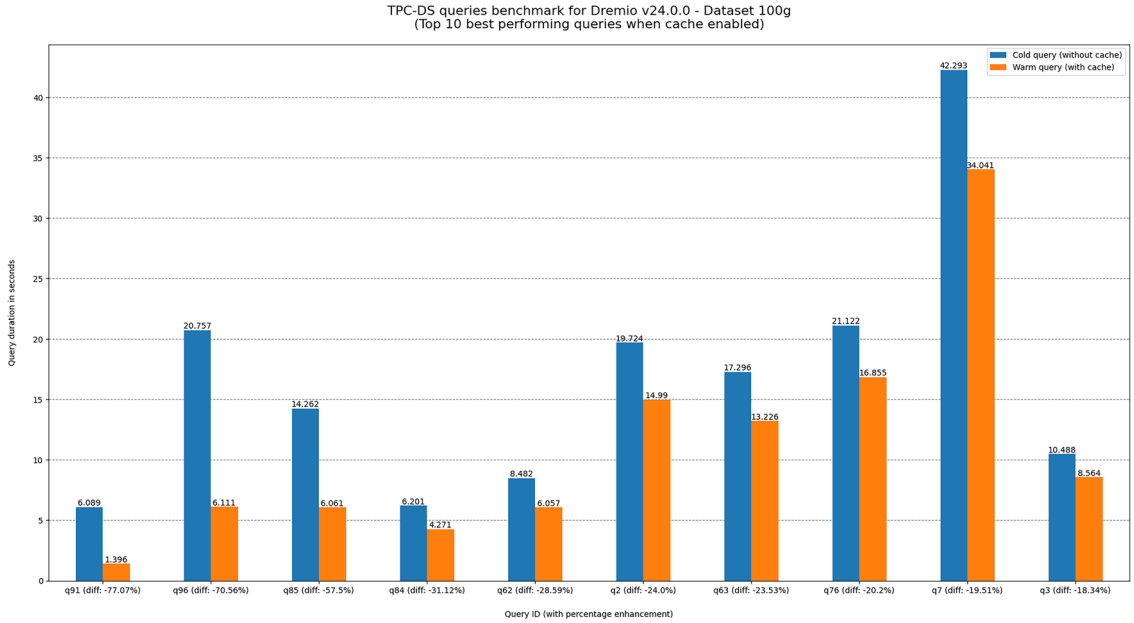

Fig. 9 Top 10 best-performing queries when cache enabled

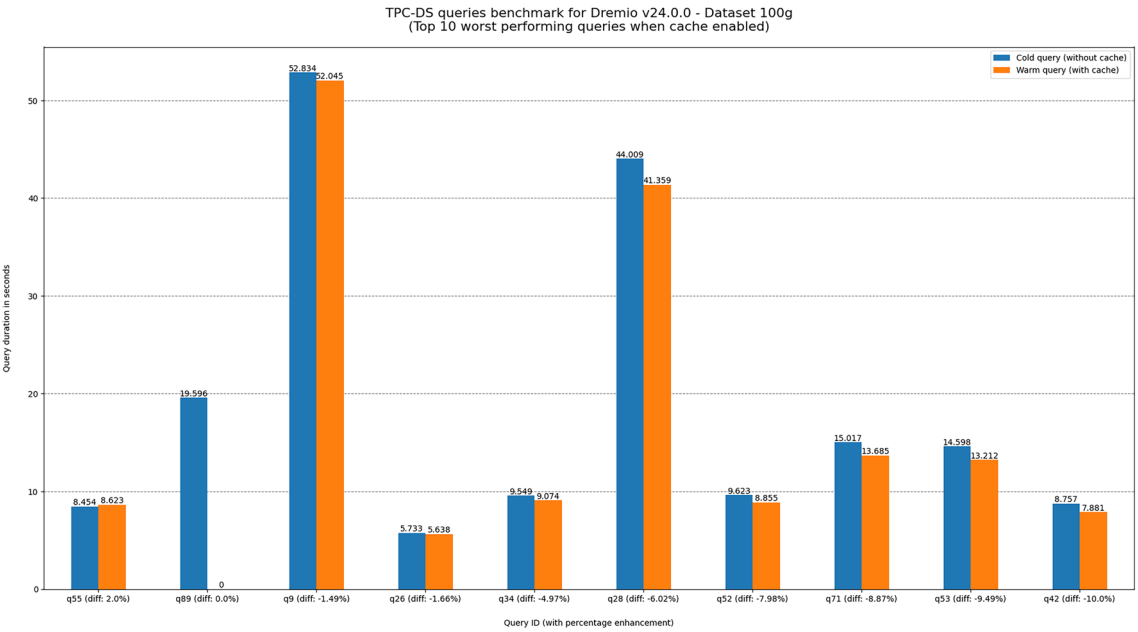

Fig. 10 Top 10 worst-performing queries when cache enabled

It clearly shows that the majority of the queries witness enhancements when the cache is used, but only two queries are outliers with worsened performance.

Finally, diving more into the individual query performance, Figs. 9 and 10 show the best and worst ten query performances in seconds and the difference between the cold and warm queries in percentage.

To conclude this section, the TPC-DS benchmark suite showed Dremio's ability to handle millions of records efficiently, which is a strong indicator that it can handle Big Data challenges and work with millions of records without any special hardware, which makes this solution more straightforward to implement and provides cost-effectiveness. Further, with the cache available, the overall performance enhancement was confirmed by the benchmark execution duration, which is 21% less. In detail, the median query duration was enhanced by 12%, and the best query duration was reduced by

77%, with two outlier queries (4.2% of the total queries) where the cache worsened the performance by about 2%. That enhancement in query duration reflects better resource utilisation, which also saves costs. Nonetheless, there is a space for more advanced benchmarking, such as debugging the ten failed queries, assessing the performance when the data is updated, using different dataset sizes, tuning Dremio's options, or using benchmarking tools tailored for Data Lakehouse like "LHBench"[6].

6 Conclusion

This project used the Data Lakehouse architecture to build a proof-of-concept implementation of a Modern Data Platform using DataOps methodologies for process, Kubernetes as an orchestration platform, and Cloud-Native software data applications. We address a gap in the literature by providing a clear implementation approach. The resulting implementation demonstrated the effectiveness of this combination in creating a resilient data platform. We were able to architect and create a proof-of-concept for a modern data platform using DataOps and Cloud-Native software. We created the core of the data platform using Kubernetes for the container orchestration platform, Argo Workflows for the data pipeline workflow engine, Dremio for the data Lakehouse platform, and MinIO for object storage. That platform has been verified and validated against the defined focus areas, such as using Cloud-Native architecture, scalable and Cloud agnostic infrastructure orchestration system, open-source software and open standard formats, and Data Lakehouse solution as a core of the data platform, which solves modern Big Data challenges like infrastracture resilience, scalability, and portability. Also, this avoids risks such as data interoperability and vendor lock-in. With new architectures, technologies, and practices, the data management landscape will fundamentally transform in the next few years, especially by unifying data formats using open table standards (e.g., Apache Hudi, Apache Iceberg, and Delta Lake). Hence, the boundaries between the Cloud and on-premises could fade over time, providing more flexibility to manage massive amounts of data for organisations of all sizes and use cases.

Before moving to future work, it is essential first to put this project in a broader context and highlight what went well and what could be done better during the project execution. Starting with the approach, using an iterative approach proved to be highly beneficial in achieving the project goals, as it allowed for continuous refinement of the methodologies based on new information or feedback. Further, regularly revisiting the project path and adjusting the methodology accordingly aid in staying on track and progressing towards the project objectives.

On the one hand, by looking back at the related works discussed in the literature review, the project results agree with the previous research about the importance of the Data Lakehouse architecture and how it can provide a flexible and scalable solution for data management compared to previous methods like Data Warehouse and Data Lake. Nevertheless, the current project took a step further to bridge the research gap by architecting and implementing such a platform and going into detail about core components and performance assessment. Moreover, the output of this research project (building a resilient cloud-agnostic data platform based on Data Lakehouse architecture) could be directly used for other projects like the proposed "Xel" project by Barron-Lugo et al. [3], which focuses on the application side only of the data handling. Therefore, this project successfully verified and extended previous research and laid the groundwork for future research. However, it is challenging to directly compare the current research results due to the topic's novelty (since the Data Lakehouse architecture was presented for the first time in 2020) and the research area's originality, making it hard to establish a direct qualitative comparison or reference point.

On the other hand, some of the current project areas could be improved, but they are mainly affected by time and resource constraints. For example, although it was advised in the architecture section to use the Kubernetes Operator whenever possible, the MinIO Operator has not been used in the proof-of-concept due to its complexity. Another concern is the dataset size used in the benchmarking by the TPC-DS suite, as the 100 gigabytes dataset may be insufficient and could potentially lead to inaccurate benchmarking results. It was not possible to use a larger dataset because it requires more time and computing resources than we had available, since the test data generated by the TPC-DS suite should be prepared before using it in Dremio. Lastly, during the benchmarking using the TPC-DS suite, approximately 50% of the test queries could not be executed on Dremio. As a result, it was not possible to compare the benchmarking results with the previous findings of Jain et al. [25]. In future work, the failed and unsupported queries will be reviewed for better benchmarking results.

[6] https://github.com/lhbench/lhbench

Continuing with future work, there are multiple directions, which are split into four categories. First category: Take the same approach as this research, adding more components of the Unified Data Infrastructure v2.0 architecture, for example, implementing a data version control like LakeFS and integrating it with Dremio or introducing another key component like Apache Spark, which provides an analytics engine for data processing. Also, implementing the platform's operational components, like building an operator for Dremio or introducing advanced infrastructure automation via GitOps. Second category: Customise the current generic architecture and implement one of the blueprints suggested by Bornstein et al. [5] like "Modern Business Intelligence", "Multimodal Data Processing", or "Artificial Intelligence and Machine Learning". Third category: Investigate the integration with the public Cloud service since more of the Cloud providers now support the open table formats, which reduces the risk of vendor lock-in and, at the same time, accelerates the development and decreases the maintenance burdens. Fourth category: Use advanced benchmarking to assess the performance of the data platform components in-depth, rewriting the TPC-DS unsupported queries to work on Dremio, comparing Dremio with another solution, benchmarking the platform with refreshed data, using different dataset sizes, tuning Dremio's options, or using benchmarking tools tailored for Data Lakehouse like "LHBench"[7].

Nonetheless, the work to date takes an important step in developing a Modern Data Platform as described.

Author contributions A.A: Conceptualization, Methodology, Software, Validation, Data curation, Writing–original draft, Writing–review & editing. P.J.B: Conceptualization, Methodology, Writing—review & editing, Supervision. C.C: Conceptualization, Writing—review & editing. N.P: Conceptualization, Writing–review & editing.

Funding The authors received no direct funding for this research.

Data availability Data sets used during the current study are publicly available in a repository: The pipeline indentation data input using daily subnational 14-day notification rate of new COVID-19 cases via the European Centre for Disease Prevention and Control (ECDC): https://www.ecdc.europa.eu/en/publications-data/subnational-14-day-notification-rate-covid-19, also a data backup is available on the project GitHub repo: https://github.com/aabouzaid/modern-data-platform-research-paper/tree/main/pipelines/ingestion, the data of the Lakehouse platform performance using the TPC-DS benchmark suite is available on the project GitHub repo: https://github.com/aabouzaid/modern-data-platform-research-paper/tree/main/benchmarking.

Code availability The project repository includes the following: Platform applications manifests (Argo Workflows, Dremio, Ingress Nginx, Minio). Data pipeline manifest (Argo Workflows Pipeline). Dremio Benchmark (Benchmark scripts, Jupyter Notebook).

Declarations

Ethics approval and consent to participate No experiments were conducted on human issues in this research.

Competing interests The authors declare no competing interests.

References

1. Armbrust M, Zaharia M, Ghodsi A, Xin R. Lakehouse: a new generation of open platforms that unify data warehousing and advanced analytics. Conference on Innovative Data Systems Research. 2021. http://cidrdb.org/cidr2021/papers/cidr2021_paper17.pdf
2. Barata M, Bernardino J, Furtado P. An overview of decision support benchmarks: TPC-DS, TPC-H and SSB. In: Advances in intelligent systems and computing. Cham: Springer International Publishing; 2015. p. 619–28. https://doi.org/10.1007/978-3-319-16486-1_61.
3. Barron-Lugo JA, Gonzalez-Compean J, Lopez-Arevalo I, Carretero J, Martinez-Rodriguez JL. Xel: a cloud-agnostic data platform for the design-driven building of high-availability data science services. Futur Gener Comput Syst. 2023;145:87–103. https://doi.org/10.1016/j.future.2023.03.019.

[7] https://github.com/lhbench/lhbench

4. Boch M, Gindl S, Barnett A, Margetis G, Mireles V, Adamakis E, Knoth P. A systematic review of data management platforms. Inf Syst Technol. 2022. https://doi.org/10.1007/978-3-031-04819-7_2.
5. Bornstein M, Li J, Casado M. Emerging architectures for modern data infrastructure. Andreessen Horowitz. 2022. https://a16z.com/2020/10/15/emerging-architectures-for-modern-data-infrastructure/. Accessed 16 Feb 2023.
6. Brown S. The C4 model for visualising software architecture. Leanpub. 2023. https://leanpub.com/visualising-software-architecture/read. Accessed 4 Mar 2024.
7. Cisco Visual Networking Index: Forecast and Trends, 2017–2022. (2019). Cisco Systems, Inc. 2019. https://twiki.cern.ch/twiki/pub/HEPIX/TechwatchNetwork/HtwNetworkDocuments/white-paper-c11-741490.pdf. Accessed 24 Mar 2023.
8. Clarke R. Big data, big risks. Inf Syst J. 2015;26(1):77–90. https://doi.org/10.1111/isj.12088.
9. Cloud Native Survey 2021. In Cloud Native Computing Foundation. 2021. https://www.cncf.io/wp-content/uploads/2022/02/CNCF-AR_FINAL-edits-15.2.21.pdf. Accessed 5 Jan 2023.
10. Data on the daily subnational 14-day notification rate of new COVID-19 cases. European Centre for Disease Prevention and Control (ECDC). 2022. https://www.ecdc.europa.eu/en/publications-data/subnational-14-day-notification-rate-covid-19. Accessed 30 Jul2024
11. DataLakeHouse. DataLakeHouse Reference Architecture. 2020. https://datalakehouse.org/datalakehouse-platform/. Accessed 16 Feb 2023.
12. Del Sagrado J, Del Águila IM. Assisted requirements selection by clustering. Requirements Eng. 2020;26(2):167–84. https://doi.org/10.1007/s00766-020-00341-1.
13. Desai V, Fountaine T, Rowshankish K. How to unlock the full value of data? Manage it like a product. McKinsey & Company. 2022. https://www.mckinsey.com/capabilities/quantumblack/our-insights/how-to-unlock-the-full-value-of-data-manage-it-like-a-product. Accessed 16 Feb 2023.
14. Domingus J, Arundel J. Cloud native DevOps with kubernetes: building, deploying, and scaling modern applications in the Cloud. 2nd ed. Sebastopol: O'Reilly Media; 2022.
15. Janssen NE. The evolution of data storage architectures: examining the value of the data lakehouse [Master Thesis]. University of Twente. 2022.
16. Etzion D, Aragón-Correa JA. Big data, management, and sustainability. Organ Environ. 2016;29(2):147–55. https://doi.org/10.1177/1086026616650437.
17. Fang H. Managing data lakes in big data era: What's a data lake and why has it became popular in data management ecosystem. 2015 IEEE International Conference on Cyber Technology in Automation, Control, and Intelligent Systems (CYBER). 2015. https://doi.org/10.1109/cyber.2015.7288049
18. Fisher MS. Software verification and validation: an engineering and scientific approach. Berlin: Springer Publishing; 2007.
19. Giebler C, Gröger C, Hoos E, Schwarz H, Mitschang B. Leveraging the data lake: current state and challenges. Big Data Analy Knowl Discov. 2019. https://doi.org/10.1007/978-3-030-27520-4_13.
20. Gür I, M. Sc. DataOps for Data Sharing. In B. Otto & J. Rehof (Eds.), ISST Reports (ISSN 0943-1624). 2021. https://ieds-projekt.de/wp-content/uploads/2024/04/DataOps_Fraunhofer-ISST-Report.pdf. Accessed 9 Jan 2023.
21. Harby AA, Zulkernine F. From data warehouse to lakehouse: a comparative review. 2022 IEEE International Conference on Big Data (Big Data). 2022. https://doi.org/10.1109/bigdata55660.2022.10020719
22. Hayes B. How do data professionals spend their time on data science projects? business over Broadway. 2019. https://businessoverbroadway.com/2019/02/19/how-do-data-professionals-spend-their-time-on-data-science-projects/. Accessed 24 Mar 2024.
23. How Semantic Management can reduce your overhead and cut query costs by 50–90%. Single Origin. 2023. https://blog.singleorigin.tech/reduce-overhead-and-improve-performance-with-semantic-management/. Accessed 15 Mar 2023.
24. Inmon WH. Building the data warehouse. 4th ed. Hoboken: Wiley; 2005.
25. Jain P, Kraft P, Power C, Das T, Stoica I, Zaharia M. Analyzing and Comparing Lakehouse Storage Systems. Conference on Innovative Data Systems Research (CIDR '23). 2023. https://www.cidrdb.org/cidr2023/papers/p92-jain.pdf. Accessed 30 Jul2024
26. Kimball R, Caserta J. The data warehouse ETL toolkit: practical techniques for extracting, cleaning, conforming, and delivering data. 1st ed. Hoboken: Wiley; 2004.
27. Kratzke N. Lightweight Virtualization cluster how to overcome cloud vendor lock-in. J Comput Commun. 2014;02(12):1–7. https://doi.org/10.4236/jcc.2014.212001.
28. LaPlante A, Safari AOMC. Building a Unified Data Infrastructure. Van Duuren Media. 2020.
29. Leontiev S. Dremio Benchmarking Methodology - How to Do It Yourself. Dremio. 2020. https://www.dremio.com/blog/dremio-benchmarking-methodology/. Accessed 14 Mar 2023.
30. Ma R, Li W, Ma N, Zhang X, Zhang H. Design and research of big data platform framework for power enterprises. IOP Conf Ser Earth Environ Sci. 2020;529(1):012009. https://doi.org/10.1088/1755-1315/529/1/012009.
31. Mainali K, Ehrlinger L, Himmelbauer J, Matskin M. Discovering DataOps: a comprehensive review of definitions, use cases, and tools. Data Analytics 2021, the Tenth International Conference on Data Analytics, 2021; 61–69.
32. MongoDB. What Is A Data Platform? 2021. https://www.mongodb.com/what-is-a-data-platform. Accessed 16 Feb 2023.
33. More Than Half of Enterprise IT Spending in Key Market Segments Will Shift to the Cloud by 2025. Gartner. 2022. https://www.gartner.com/en/newsroom/press-releases/2022-02-09-gartner-says-more-than-half-of-enterprise-it-spending. Accessed 5 Mar 2023.
34. Munappy AR, Bosch J, Olsson HH. Data pipeline management in practice: challenges and opportunities. In: Product-focused software process improvement. Berlin: Springer; 2020. p. 168–84. https://doi.org/10.1007/978-3-030-64148-1_11.
35. Opara-Martins J, Sahandi R, Tian F. Critical analysis of vendor lock-in and its impact on cloud computing migration: a business perspective. J Cloud Comput. 2016. https://doi.org/10.1186/s13677-016-0054-z.
36. Oppermann A. Architecture for data platforms. Hygraph. 2022. https://hygraph.com/blog/data-platform-architecture. Accessed 17 Feb 2023.
37. Orescanin D, Hlupic T. Data lakehouse—a novel step in analytics architecture. Int Conv Inf Commun Technol, Electron Microelectron. 2021. https://doi.org/10.23919/mipro52101.2021.9597091.

38. Poess M, Nambiar RO, Walrath D. Why you should run TPC-DS: a workload analysis. New York City: ACM Digital Library; 2007. https://doi.org/10.5555/1325851.1325979.
39. Rabl T, Poess M, Baru C, Jacobsen H. Specifying big data benchmarks. In: Lecture notes in computer science. Berlin: Springer; 2013. https://doi.org/10.1007/978-3-642-53974-9.
40. Ruparelia NB. Cloud computing: the MIT Press essential knowledge series. Cambridge: The MIT Press; 2016.
41. Simon B. Complete guide to the people, process, technology framework. Smartsheet. 2019. https://www.smartsheet.com/content/people-process-technology. Accessed 15 Feb 2023.
42. Späti S. Data Lake/Lakehouse Guide: Powered by Data Lake Table Formats (Delta Lake, Iceberg, Hudi). Airbyte. 2022. https://airbyte.com/blog/data-lake-lakehouse-guide-powered-by-table-formats-delta-lake-iceberg-hudi. Accessed 20 Feb 2023.
43. Thusoo A, Sen Sarma J. Creating a data-driven enterprise with DataOps. Sebastopol: O'Reilly Media Inc; 2017.
44. Watson HJ, Goodhue DL, Wixom BH. The benefits of data warehousing: why some organizations realize exceptional payoffs. Info Manag. 2002;39(6):491–502. https://doi.org/10.1016/s0378-7206(01)00120-3.
45. What is a Kubernetes Operator? Red Hat. 2022. https://www.redhat.com/en/topics/containers/what-is-a-kubernetes-operator. Accessed 7 Mar 2023.
46. Yugal L. Business analytics: trends and challenges. Int Conf Intell Emerg Methods Artif Intell Cloud Comput. 2022. https://doi.org/10.1007/978-3-030-92905-3_3.