machines

MDPI

Article

Digital Twin Data Management: Framework and Performance Metrics of Cloud-Based ETL System

Austeja Dapkute *, Vytautas Siozinys, Martynas Jonaitis, Mantas Kaminickas and Milvydas Siozinys

Company "Energy Advice", LT-44175 Kaunas, Lithuania; vytautas.siozinys@energyadvice.lt (V.S.); martynas.jonaitis@energyadvice.lt (M.J.); mantas.kaminickas@energyadvice.lt (M.K.); milvydas.siozinys@energyadvice.lt (M.S.)

* Correspondence: austeja@energyadvice.lt; Tel.: +370-602-30887

Abstract: This study delves into the EA-SAS platform, a digital twin environment developed by our team, with a particular focus on the EA-SAS Cloud Scheduler, our bespoke program designed to optimize ETL (extract, transform, and load) scheduling and thereby enhance automation within industrial systems. We elucidate the architectural intricacies of the EA-SAS Cloud Scheduler, demonstrating its adeptness in efficiently managing computationally heavy tasks, a capability underpinned by our empirical benchmarks. The architecture of the scheduler incorporates Docker to create isolated task environments and leverages RabbitMQ for effective task distribution. Our analysis reveals the EA-SAS Cloud Scheduler's prowess in maintaining minimal overhead times, even in scenarios characterized by high operational loads, underscoring its potential to markedly bolster operational efficiency in industrial settings. While acknowledging the limitations inherent in our current assessment, particularly in simulating real-world industrial complexities, the study also charts potential future research pathways. These include a thorough exploration of the EA-SAS Cloud Scheduler's adaptability across diverse industrial scenarios and an examination of the integration challenges associated with its reliance on specific technological frameworks.

Keywords: digital twin; scheduler; data processing; micro-batch processing; stream processing; ETL queue; task scheduling

Citation: Dapkute, A.; Siozinys, V.; Jonaitis, M.; Kaminickas, M.; Siozinys, M. Digital Twin Data Management: Framework and Performance Metrics of Cloud-Based ETL System. *Machines* **2024**, *12*, 130. https://doi.org/10.3390/machines12020130

Academic Editors: Gianni Campatelli and Kai Cheng

Received: 30 December 2023
Revised: 25 January 2024
Accepted: 7 February 2024
Published: 12 February 2024

1. Introduction

Digital twin technology has emerged as a significant contributor to energy-efficient production and optimization of technological processes in industrial innovation. This technology encompasses a virtual model, which is built using extensive mathematical expressions to simulate processes and equipment for optimal operational control [1]. Given the global emphasis on reducing energy consumption in manufacturing, digital twin technology is increasingly recognized as a vital tool for energy-efficient production lines [2].

With the advent of Industry 4.0, digital twin technology has become integral to enhancing both energy efficiency and process optimization. It necessitates comprehensive data analysis, including mathematical modeling of physical systems and processes, forecasting, and applying statistical algorithms. Digital twin uses fresh monitoring data to represent the real-time state of the system and to estimate the future state. The architecture of digital twin systems is inherently complex due to the need for processing dynamic status changes and providing realistic graphical representations of objects [3].

The components of high-level digital twin architecture include information models, communication mechanisms, and data processing. This architecture integrates various technologies, including internet and interaction technologies, network security, interfaces, and communication protocols [4,5]. The handling of digital twin data, from collection to fusion, requires managing complex datasets, while digital twin services encompass applications, resources, knowledge, and platform services [6].

The International Standardization Organization issued a series of standards (ISO 23247) that suggest a generic framework for digital twins for manufacturing. These standards describe general principles and requirements [7], reference architecture with functional views [8], basic information attributes for the observable manufacturing elements (OMEs) [9], and technical requirements for information exchange between entities within the reference architecture [10]. Shao et al. [11] provide a meticulous review of the ISO 23247 digital twin framework for manufacturing, outlining its structured composition of entities and sub-entities. The framework consists of the user entity for hosting software systems and interfaces, the digital twin entity for the digital representation and synchronization of OMEs, and the device communication entity for data interaction and device control. Each entity is further divided into sub-entities and functional entities, such as the data collection sub-entity for data acquisition and pre-processing and the device control sub-entity for actuation and operational control.

The accuracy and reliability of digital twin systems are crucial, as any discrepancies could lead to increased operational costs. The frequency of data acquisition and processing should be carefully selected based on the specific application of the digital twin. For example, a digital twin of a biomass boiler may effectively operate with minute-by-minute data, as more frequent data might not provide substantial additional insights but will increase resource utilization. Despite collecting data at minute intervals, digital twins can accumulate extensive datasets, necessitating advanced methodologies for efficient processing in real or near-real-time [12].

This article introduces EA-SAS, a newly developed digital twin platform, and presents a systematic exploration of its unique ETL scheduling framework, underscoring our approach to optimizing data management within this context. It begins with a literature analysis, situating our research within the broader academic context. Subsequent sections discuss data processing modalities in digital twin systems, introduce a Docker-integrated architectural framework, and evaluate task scheduling system performance. The architecture we present addresses the complex demands of data processing within digital twins, a cornerstone for achieving operational efficiency and adaptability in contemporary industrial settings. By optimizing these processes, our study contributes to the advancement of automated control mechanisms, laying the groundwork for more intelligent, responsive, and efficient industrial operations.

2. Related Work

2.1. Big Data in Digital Twin

In general literature and industry practice, digital twins are frequently recognized as a significant instance of big data applications due to their characteristics:

1. Data Volume: Digital twins generate large volumes of data as they continuously collect information from various sensors and devices to create a real-time, dynamic representation of a physical object or system.
2. Data Variety: The data associated with digital twins comes from a diverse range of sources, including IoT sensors, operational systems, and environmental data, encompassing a wide variety of formats.
3. Real-Time Processing: Digital twins often require real-time or near-real-time data processing to accurately reflect the current state of the physical entity they represent. This demands efficient and robust big data processing capabilities.
4. Complex Analytics: The use of digital twins involves complex analytics, including predictive modeling and simulation, to gain insights and make decisions based on the data collected. This requires sophisticated data processing and analysis techniques, which are hallmarks of big data applications.
5. Integration Challenges: Like other big data applications, digital twins face challenges in integrating and harmonizing data from disparate sources, ensuring data quality, and managing the scale of data.

Building on the work of Tao [13,14], the concept of big data and its integration with digital twin technology in the manufacturing industry is comprehensively summarized. This integration is evident in various applications, including product design, production planning, manufacturing, and predictive maintenance. Tao's analysis not only outlines the similarities and differences between big data and digital twin technologies from both a holistic and data-centric perspective but also delves into the structure and operational mechanisms of digital twin systems (DTS). This thorough examination underscores the classification of digital twins as a significant instance of big data application, particularly in the context of advanced manufacturing processes ([15] citing Tao [13,14]).

Despite the established nature of traditional ETL approaches in big data, there is an emerging consensus on the necessity to adopt more innovative technologies. These technologies are essential to navigate the growing complexities and requirements of big data environments, exemplified by digital twins.

2.2. Current Approaches to ETL

The [16] study identified nine popular approaches to implement ETL solutions:

- Service-oriented architecture (SOA);
- Web-based technologies (e.g., semantic web);
- Fault-tolerant algorithms;
- Structured Query Languages (SQL);
- Parallelization (e.g., MapReduce);
- Domain ontology;
- Multi-agent systems (MAS);
- Conceptual modeling (e.g., Unified Modeling Language (UML) and Business Process Modeling Notation (BPMN));
- Metadata repository [16].

UML-based models standardize the design process but also have limitations in handling unstructured data and user-defined functions (UDFs). BPMN-based models offer semi-autonomous behavior and are geared toward translating business requirements into conceptual models. However, they require specific knowledge of BPMN and Business Process Execution Language (BPEL) for implementation [17].

In manufacturing, data are categorized into real-time perception data, production process data, and production activity plan data. This data must be collected, fused, and transmitted effectively for constructing a digital twin (DT) [18]. For small- and medium-sized enterprises, sensor-based tracking and machine vision are crucial for data acquisition. The collected data are then fused, integrated, and uploaded to a database, often via networks like 5G, for a higher bandwidth and lower latency. Technologies like the Hierarchical Data Format Version 5 (HDF5) are used for flexible data storage. Data from various sources, including manufacturing execution systems and networked machine tools, are merged into the database.

Optimizing algorithms for speed and accuracy and integrating different communication protocols and interfaces for a unified DT platform is highlighted as a necessity [16].

2.3. Challenges and Limitations in Existing Systems

For proactive decision-making, digital twins must acquire context information in real-time or near-real-time, tailored to their specific use cases [12]. These applications necessitate a range of data analysis techniques, from machine learning to statistical models, involving sophisticated data mining methods like clustering, classification, association rules, regression, prediction, and deviation analysis [19,20].

A digital twin's function transcends mere real-time mirroring of a physical system; it also sends optimized commands to the control system. Systems that only reflect the state of a physical asset in real-time are labeled 'digital shadows', whereas digital twins, by definition, interact with and impact the physical system [2,21,22].

To facilitate this bidirectional communication, an effective architectural framework is vital. This involves extracting data from the client's infrastructure and processing it through dedicated algorithms, typically under the extract–transform–load (ETL) processes.

Traditional ETL processes, often executed as background tasks, can be resource-intensive and potentially challenging to support over time, especially when older tasks fail due to library updates. Digital twin applications utilize various libraries for tasks ranging from mathematical and statistical algorithms to data manipulation and communication facilitation. Updates to these libraries, if not managed properly, can result in lost system traceability and increased time expenditures. Isolating each task in a separate environment by the task scheduling system is a potential solution.

Digital twins in manufacturing require specific considerations for data processing and circulation, such as latency, bandwidth, data security, and quality [23]. Researchers assert that low latency and scalable capacity are crucial for the successful implementation of digital twins [24].

The majority of the current systems face challenges in real-time big data processing due to scalability issues, offline data cleansing needs, or complexities in integrating varied data types and sources in real time. Most analyses currently occur on offline datasets [12]. While some digital twin models focus on offline batch analysis, there is a significant demand for real-time analysis and prediction [25]. Wallner et al. emphasize the significance of considering life cycle changes in digital twins, advocating for a holistic view of digital twin applications to reduce the implementation effort and to ensure that applications remain valid through changes [26].

The time required for data exchange and processing in digital twin systems is typically minimal when considered against the intended purpose and application of the digital twin [3]. The application scenario dictates the necessary communication latency between a physical device and its digital twin. As the need for immediate data processing and communication increases, so do the complexities and costs of system development [27]. For certain entities and applications, immediate processing, communication, and storage capabilities might be essential to meticulously monitor events and status transitions. For others, infrequent updates, perhaps daily or even less frequent, might suffice [3].

Despite the popularity of conceptual modeling like UML and BPMN in ETL implementation, concerns about an overemphasis on this approach are raised, particularly regarding its effectiveness in addressing future challenges in data complexity, volume, and heterogeneity [16].

2.4. Overview of Existing Solutions: Apache Airflow and AWS Batch

The realm of distributed ETL task-processing features notable solutions, among which Apache Airflow and AWS Batch stand out. This section examines these two platforms.

Apache Airflow, a platform extensively used for orchestrating complex computational workflows, manages tasks using directed acyclic graphs (DAGs) [28]. However, its complexity poses certain operational challenges. For instance, the lack of inherent synchronization of DAG configurations across multiple servers requires manual updates for any task modification, often necessitating additional systems for distribution. Additionally, Airflow's scheduling model, despite being robust, can lead to increased consumption of scheduler resources.

AWS Batch, on the other hand, is part of the Amazon Web Services ecosystem, offering a managed environment with the ease of a software-as-a-service (SaaS) model [29]. It allows task execution in isolated environments, like containers, facilitating the separation of concerns. However, it is not optimized for tasks requiring rapid execution, with recommended task durations spanning several minutes to avoid resource wastage [30]. The use of AWS EC2 servers, though not directly priced for AWS Batch, can make the overall solution costlier than regional alternatives by a significant margin [31,32].

EA-SAS is the name of our comprehensive digital twin platform, within which this paper specifically delves into the EA-SAS Cloud Scheduler, the subsystem designed to

manage data processing and ETL processes efficiently, showcasing the innovative architecture that forms the backbone of the platform's data handling capabilities Differing from existing solutions, the EA-SAS Cloud Scheduler is tailored with a novel architecture to meet the specific requirements of digital twin functionalities, especially focusing on the ETL scheduling framework. A comparative analysis of these ETL scheduling tools is presented in Table 1, highlighting their distinctive features and limitations.

Table 1. Comparative analysis of ETL scheduling tools.

Criteria	Apache Airflow	AWS Batch	EA-SAS Cloud Scheduler
Open ID connect compatibility	Yes	No	Yes
Ability to launch tasks within 3 seconds (short task launch delay)	No	No	Yes
Task execution in isolated environments	With extension	Yes	Yes
Configuration of tasks via user interface	No	Yes	Yes
Real-time task status and reporting via user interface	Yes	Yes	Yes
Automatic retry of failed tasks	Yes	Yes	Conditional

The tabular analysis underscores the distinct features and limitations inherent to each tool, guiding users in selecting the most appropriate ETL scheduling solution for their specific needs.

2.5. Relevance to the Present Study

Ali and Wrembel [17] point out a key limitation in current ETL tools: the lack of efficient workflow development support due to missing automated optimization and fine-tuning capabilities. This gap often leads to the creation of bespoke ETL tools tailored to specific business needs [15,33,34]. The increasing data volume adds complexity to the ETL workflow design, elevating execution costs and risking operational delays or failures.

To address these challenges, the EA-SAS Cloud Scheduler has been developed, featuring horizontal scaling and directed acyclic graph (DAG) task structures within Docker containers for efficient task scheduling. This is crucial to minimize execution times and optimize resource utilization [33].

Data in digital twin applications can be processed either through batch processing or stream processing. These methods are further elaborated on in the following chapter.

3. Data Processing Modalities in Digital Twin Systems

3.1. Micro-Batch Data Processing

In batch processing, data are accumulated into groups or "batches" and processed collectively once a certain threshold or time limit is reached [34]. This approach, while an efficient and simplifying system design, can introduce latency due to the gap between data collection and processing. For applications requiring minimal latency, micro-batch processing is often utilized.

Micro-batch processing breaks down ETL operations into smaller chunks, usually encompassing data for about a minute. This aligns well with digital twin models of complex systems, facilitating near-real-time data access. Although this method does not allow for real-time adaptation, it supports the integration of new data during model updates, enhancing subsequent analysis accuracy [35]. Studies indicate that micro-batch processing often outperforms stream processing in accuracy when analyzing stored data, owing to its repetitive nature. The structure of micro-batch processing is depicted in Figure 1.

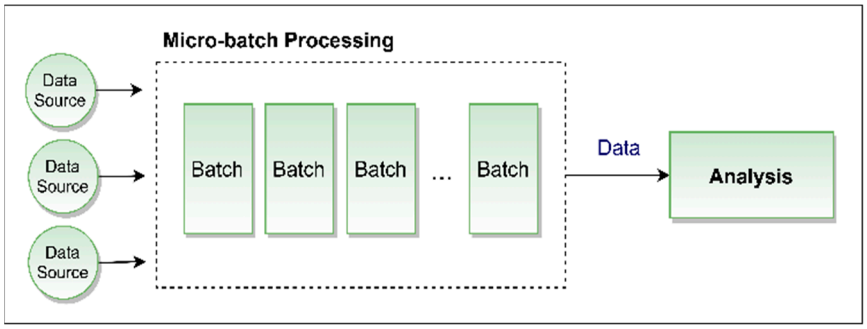

Figure 1. Micro-batch processing logic.

For digital twin applications, the latency in micro-batch processing can range from a few minutes to seconds, making it a suitable option for real-time scenarios.

3.2. Stream Data Processing

Stream processing is a real-time computational method that processes data as it arrives [35]. It enables rapid computations with typical latencies ranging from milliseconds to seconds and is incrementally adaptive to new data streams. This approach is especially effective when data relevance is more immediate and less dependent on historical records.

In stream processing, systems continuously operate on incoming data with minimal delay, maintaining an event-driven architecture. The fluctuating rate of incoming data can sometimes lead to increased computational demands, particularly during high data inflow periods [34]. While stream processing is adept at handling real-time data, integrating historical data analysis, crucial in digital twin models, may necessitate an additional system design. The structure of stream processing is depicted in Figure 2.

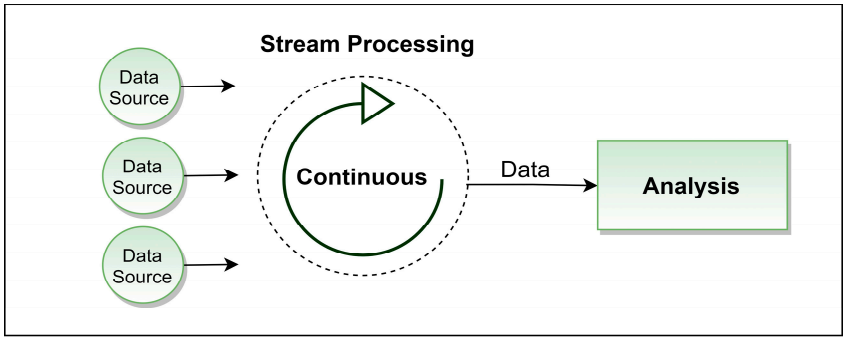

Figure 2. Stream processing logic.

In the context of industrial digital twin architectures, the immediacy provided by stream processing may not always be crucial. Real-time data fidelity is important, but slight latencies, ranging from seconds to minutes, are typically acceptable. For instance, in industrial operations like biomass boiler functioning or thermal regulation, decisions do not usually require millisecond precision. Understanding the stream processing's role is enhanced when compared with micro-batch processing, which offers a middle ground between real-time and traditional batch methods, suitable for scenarios where infrastructure may not fully support continuous stream processing but requires near-real-time data analysis.

3.3. Rationale behind Adopting Batch Processing

The choice between processing paradigms, such as stream and batch models, is driven by their unique advantages and suitability for specific applications [35]. In anomaly detection, for instance, batch processing often enhances the accuracy of identifying new anomalies offline at regular intervals, such as daily, weekly, or monthly. Conversely, stream

processing is favored for immediate online anomaly detection in certain industrial scenarios [35]. However, a holistic assessment indicates that a purely stream-based approach may introduce complexities that do not necessarily yield proportional benefits.

In many industrial applications, the necessity for data processing at second-level intervals is rare. Within this context, micro-batch processing emerges as a more fitting approach for real-time digital twin applications. It offers a detailed data analysis while meeting the latency requirements of real-time digital twins, effectively merging the strengths of both stream and batch processing. This makes micro-batch processing a balanced solution, especially considering the challenges associated with stream processing in certain digital twin scenarios.

Table 2 provides a comparative overview of micro-batch and stream processing from the perspective of digital twin technology.

Table 2. Comparative overview of data processing methodologies in the context of digital twin technology.

Criteria	Micro-Batch Processing	Stream Processing
Data Collection	Data aggregated over defined short intervals.	Continuous data streaming.
Data Processing	Processing occurs subsequent to collection.	Data are processed incrementally.
Advantages	Enables comprehensive data analysis, simpler implementation, and increased applicability.	Offers swift processing and real-time analytics.
Disadvantages	It may introduce variable latency.	Presents implementation complexity and specific applicability challenges.
Suitability	Ideal for large datasets needing in-depth analysis.	Less preferred for projects requiring extensive data analysis or large data volumes.

4. Architectural Framework of Docker-Integrated Task Management in the EA-SAS Cloud Scheduler

4.1. System Components and Topology

The EA-SAS Cloud Scheduler, part of our distinctively developed digital twin platform, EA-SAS, encompasses a set of interconnected structural units, each playing a pivotal role in the platform's innovative ETL scheduling framework. The EA-SAS Cloud Scheduler system comprises five interconnected structural units (as shown in Figure 3) as follows:

- Reverse Proxy Server: Functions as an intermediary for all external system access. It enhances traffic monitoring and keeps task executor servers secure from external access, forming a crucial part of the company's infrastructure.
- Keycloak Server: Hosts the Keycloak authentication service, centralizing all system authentication processes. This server is a critical component of the infrastructure.
- Scheduler Server: Contains the scheduler, user interface components, and a PostgreSQL database. It is primarily responsible for task scheduling and storing execution histories.
- RabbitMQ Server: Hosts the RabbitMQ message-queuing service, facilitating a significant portion of the communication between the scheduler and task executor servers.
- Worker Server: Represents the task executor subsystem. The number of these servers is theoretically unlimited, though the centralized architecture might impose some constraints. Each server performs tasks within separate containers and maintains task logs.

The architectural schematic (Figure 3) illustrates the EA-SAS Cloud Scheduler's design. At the core of this structure is the scheduler server, which manages task allocation and interfaces with the RabbitMQ message-queuing system. RabbitMQ acts as an intermediary, channeling tasks to a series of workers (Worker 1 through Worker N). The diagram's arrowed lines show the unidirectional flow of tasks and data from the scheduler to RabbitMQ and then to the workers. This setup ensures systematic task distribution and execution within the cloud-based framework.

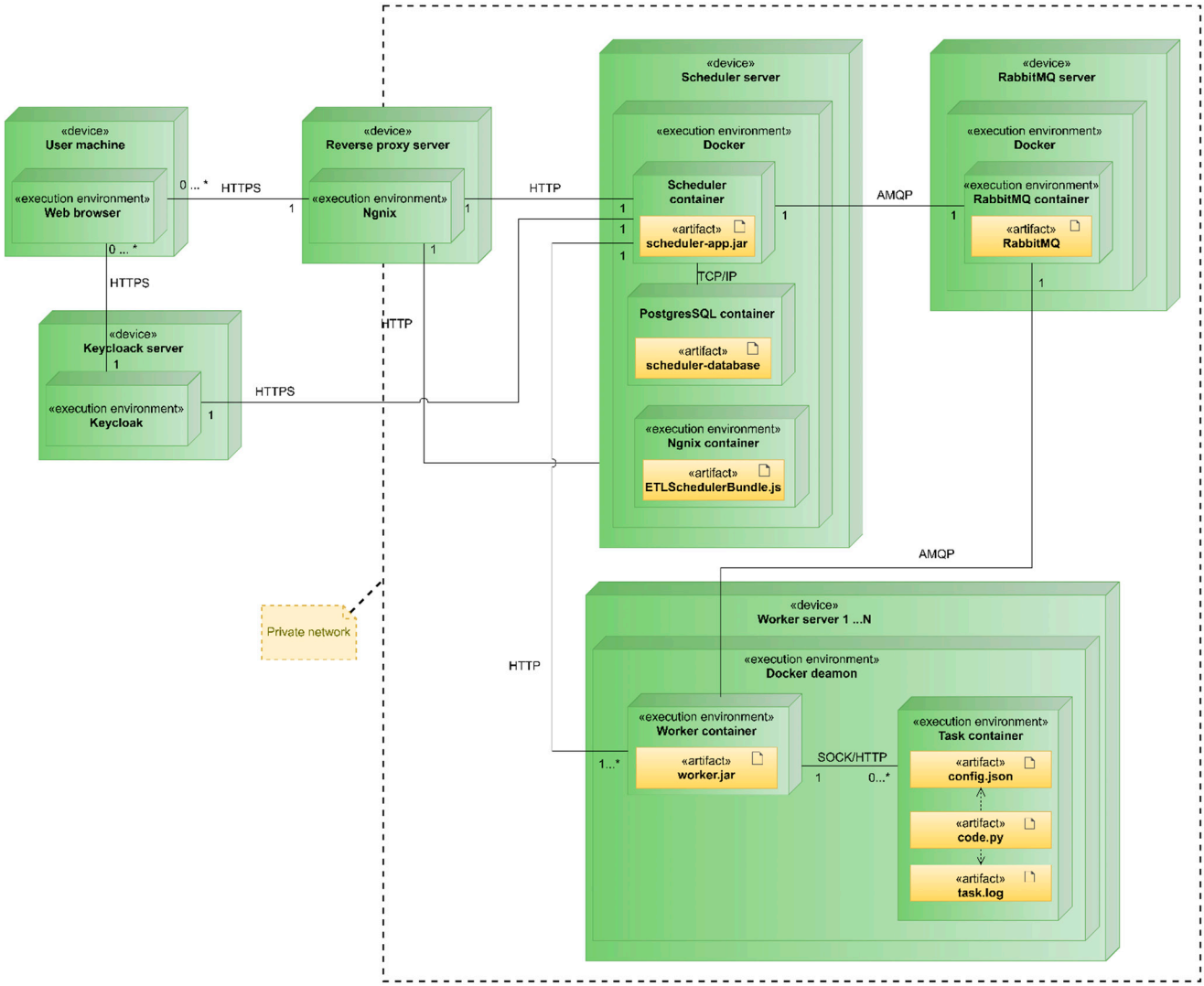

Figure 3. Deployment diagram of the EA-SAS Cloud Scheduler.

Communication between the executor and the scheduler is maintained through a 'heartbeat signal', providing real-time updates on the executor's status. The system uses Advanced Message Queuing Protocol (AMQP) messages for this communication, with RabbitMQ overseeing the interactions. A notable feature of RabbitMQ is its failover mechanism, which redirects tasks to an alternate executor in case of disruptions, ensuring consistent and reliable task execution.

The EA-SAS Cloud Scheduler's architecture aligns with contemporary standards in digital twin technology, echoing the guidelines of ISO/IEC AWI Standard 30172 [36] and ISO/IEC AWI Standard 30173 [37]. These standards, as discussed in [38], underscore the necessity for robust and flexible digital twin frameworks. They advocate for a design that can adapt to diverse-use cases while maintaining clear terminology and operational transparency, principles that are fundamentally embedded within the EA-SAS Cloud Scheduler's design and operational ethos.

4.2. Integration and Communication

The core of the EA-SAS Cloud Scheduler is its 'task flows', organized in a directed acyclic graph (DAG) format. Each task is linked to a specific execution code and a Docker environment, ensuring streamlined execution.

4.2.1. Docker Execution Environment

This environment exhibits a hierarchical structure with Docker images acting as metadata repositories, mainly for library versions. Tasks access their respective container's metadata upon queueing, with the system autonomously sourcing images from a Docker registry if absent. This design isolates each task, mitigating disruptions from library update incompatibilities.

4.2.2. Temporal Composition of a Task

Task execution within the digital twin platform involves distinct time intervals, representing various stages of the task's life cycle, shown in Figure 4:

1. Scheduler Delay (t_1): This interval commences with the scheduled time of task execution and culminates when the task is triggered. It encapsulates the delay between when a task is scheduled and its initiation.
2. Queuing/Task Distribution Delay (t_2): Post triggering, the task enters a queuing system. The duration represented by t_2 captures the time taken from the task entry into this queue until the system identifies and designates a suitable worker for its execution.
3. Config Fetching (t_3): During this phase, the system retrieves the task metadata essential for determining the conditions under which the task will be executed.
4. Data Fetching (Extract, t_4): Here, specific datasets, as outlined in the previously fetched configuration, are acquired to facilitate task execution.
5. Calculations (Transform, t_5): This interval is central to the task's purpose, wherein the actual computational operations are executed.
6. Uploading/Saving Data (Load, t_6): Upon computation completion, the results are transmitted and stored within the digital twin platform.
7. Confirm Delay (t_7): This final interval signifies the time lapse between task execution completion and its acknowledgment on the user interface.

Overhead		Task Execution				Overhead
Scheduler delay	Queuing / task distribution delay	Config fetching	Data fetching (**E**)	Calculations (**T**)	Uploading / saving data (**L**)	Confirm delay
t_1	t_2	t_3	t_4	t_5	t_6	t_7

Figure 4. Temporal composition of a task in the digital twin platform EA-SAS Cloud Scheduler.

The execution phases (config fetching and ETL processes) are represented as:

$$t_{\text{execution}} = t_3 + t_4 + t_5 + t_6 \quad (1)$$

This equation represents the sum of the time intervals for config fetching (t_3), data fetching (t_4), calculations (t_5), and uploading/saving data (t_6).

The overhead time is represented as:

$$t_{\text{overhead}} = t_1 + t_2 + t_7 \quad (2)$$

This equation aggregates the time intervals for scheduler delay (t_1), queuing/task distribution delay (t_2), and confirm delay (t_7).

In sum, the EA-SAS Cloud Scheduler's integration with Docker ensures enhanced reliability and efficiency in task management. By utilizing Docker for individual task environments and integrating RabbitMQ for efficient task distribution, the system achieves both streamlined execution and robust failover capabilities. Figure 4 elucidates the task's life cycle and associated intervals, highlighting the system's operational strengths and potential areas for optimization. As we transition into the next chapter, "Evaluation of Task

Scheduling System Performance," we will further examine the practical implications and performance metrics of this architectural framework.

5. Results

5.1. Objective Derivation and Hypothesis Formation

The primary objective of our study was to measure and compare the duration of the task execution overhead ($t_{overhead}$), as described by Equation (2). Our methodology provided a uniform platform for appraising the efficiency of task scheduling and execution, deliberately excluding the variable of task complexity. Our hypothesis was that the overhead time would increase with a rising number of tasks per minute, and differences between the schedulers would become evident.

5.2. Experimental Setup and Methodology

1. Task Design: We utilized a basic Python task to maintain consistency in our measurements, thereby removing any discrepancies that could result from intricate task executions or data retrieval processes.
2. Test Configuration: Our testing procedure involved establishing a directed acyclic graph (DAG)/task flow with the aforementioned task. We meticulously recorded the interval from the task's scheduling point to the confirmation of the DAG/task flow.
3. Test Scope: The experiment spanned a wide range of task counts, from 1 to 1000 per minute, to thoroughly assess the performance of the schedulers under varying operational loads.
4. Infrastructure: Both scheduling tools were assessed using the identical virtual private server (VPS) setup, ensuring a controlled environment. Executors were isolated on a separate server to preclude any potential disturbances to the scheduling assessment. The VPS's specifications are detailed in Table 3.

Table 3. Characteristics of the testing Virtual Private Server.

Parameter	Characteristic
Processor	Intel Xeon (Skylake), 4 cores @ 2.6 GHz
RAM	16 GB
Storage Media	SSD

5.3. Quantitative Metrics and Analytical Outcomes

The synthesis of our findings is encapsulated in Table 4 and Figure 5, which collectively demonstrate a direct relationship between task count increments and execution time. Apache Airflow consistently manifested more substantial lags in comparison to the EA-SAS Cloud Scheduler. For example, when handling 1000 tasks per minute, the EA-SAS Cloud Scheduler maintained a lean overhead of just 1.5 s, whereas Apache Airflow lagged with an overhead of 23.4 s.

Table 4. Task Execution overhead comparison.

Task Count per Minute	Task Execution Overhead, Seconds	
	Apache Airflow	EA-SAS Cloud Scheduler
1	7.1	0.6
5	7.8	0.6
10	8.8	0.7
20	10.6	0.7
50	12.1	0.7
100	13.2	0.7
200	16.2	0.8
500	18.9	1.1
1000	23.4	1.5

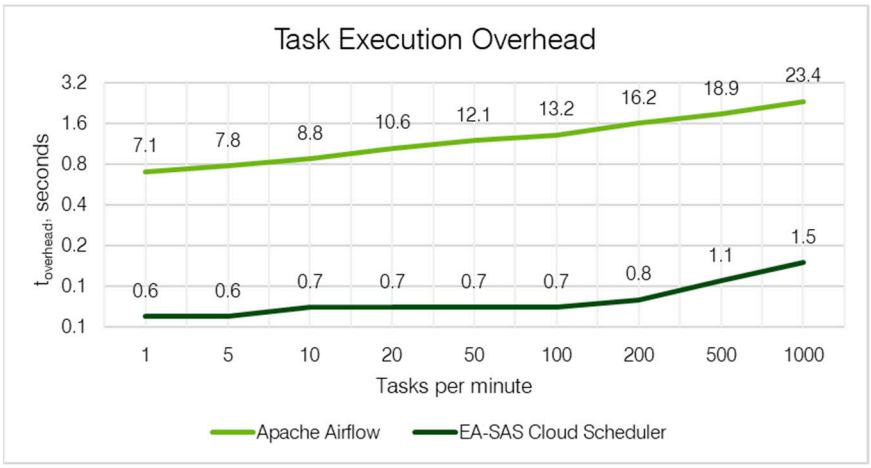

Figure 5. Task execution's overhead comparison results.

Apache Airflow's extended overhead time can be attributed to its intricate architecture, optimized for diverse data processing needs. This system is particularly adept at managing workflows that evolve over longer periods, such as days or weeks. However, in scenarios like digital twin applications, where real-time analysis is crucial, Airflow's comprehensive feature set may inadvertently increase the task execution overhead.

EA-SAS Cloud currently operates through Apache Airflow and has seven workers. Metering the data of task distribution between workers is depicted in Figure 6. The total executed task count between various digital twins is depicted in Figure 7. Each of the colors represents a different digital twin. Names of the companies are not shown due to confidentiality reasons.

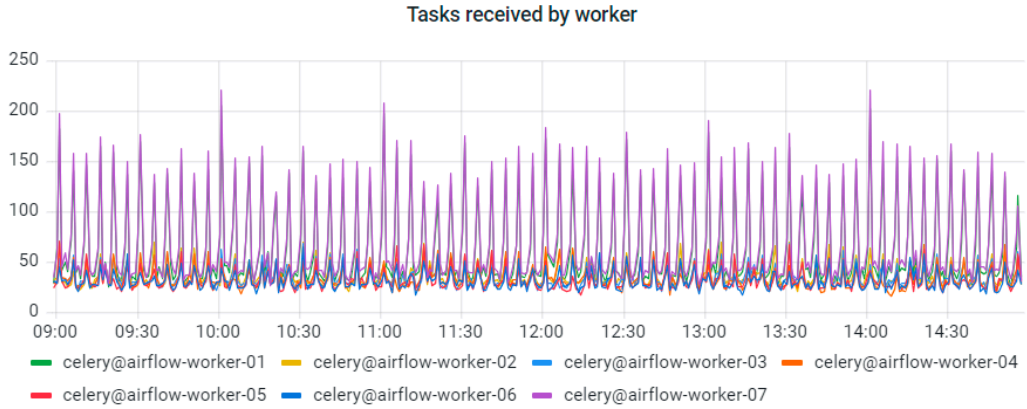

Figure 6. Metering data of task distribution between workers at EA-SAS.

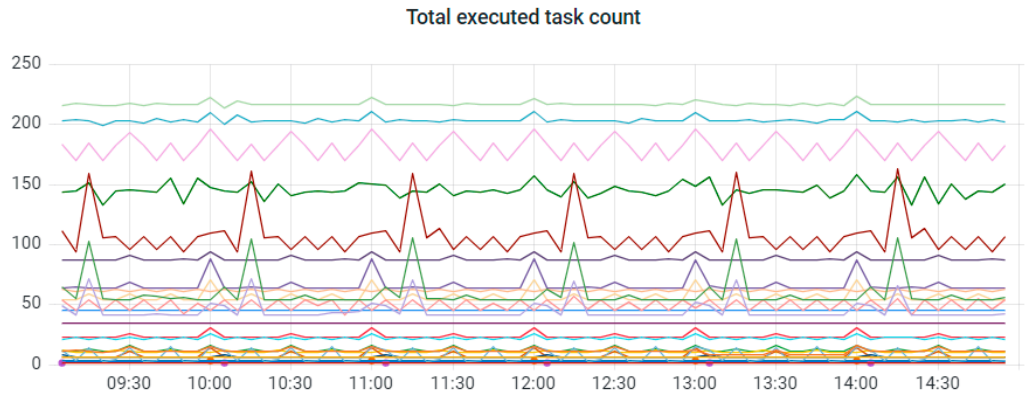

Figure 7. Total executed task count by different digital twins.

6. Conclusions

This study has meticulously explored the EA-SAS Cloud Scheduler, a core component of the EA-SAS digital twin platform, emphasizing its pivotal role in optimizing ETL scheduling to enhance automation and control in industrial systems. The experimental results demonstrate the scheduler's exceptional efficiency in managing a high volume of tasks, indicating a profound improvement in operational performance. Specifically, the EA-SAS Cloud Scheduler excels in scenarios demanding frequent and rapid task processing, significantly reducing overhead times when compared to other systems. This underlines the scheduler's capability to align with the temporal requirements of diverse applications, ensuring optimal performance, particularly in environments that necessitate swift task execution and meticulous monitoring.

However, it is crucial to acknowledge certain limitations. While the EA-SAS Cloud Scheduler is engineered for high efficiency, its performance in varying industrial scenarios needs extensive real-world validation. The experimental setup, though comprehensive, represents a controlled environment that may not capture the full complexity of real-world operations. Moreover, the system's reliance on specific technologies like Docker and RabbitMQ, while beneficial for the tasks demonstrated, might pose challenges in terms of the broader applicability and integration into diverse IT ecosystems.

In conclusion, the EA-SAS Cloud Scheduler emerges as a significant advancement in digital twin technology, offering substantial improvements in ETL scheduling and task management. Future studies should aim to validate these findings in a broader array of industrial settings and explore the integration of the scheduler with different technologies, further enhancing its versatility and applicability in the evolving landscape of industrial digitalization.

Author Contributions: Conceptualization, V.S. and M.S.; methodology, V.S., M.K. and M.J.; software, M.J.; validation, V.S., M.J. and A.D.; formal analysis, V.S., M.K. and M.J.; investigation, A.D. and M.J.; resources, M.J.; data curation, M.J.; writing—original draft preparation, A.D.; writing—review and editing, A.D.; visualization, A.D.; supervision, V.S.; project administration, A.D.; funding acquisition, V.S. All authors have read and agreed to the published version of the manuscript.

Funding: The authors disclosed receipt of the following financial support for the research, authorship, and/or publication of this article: This work was supported by the Norwegian financial mechanism and the state budget of the Republic of Lithuania funds (grant number LT-07-1-EIM-K01-006).

Data Availability Statement: The data that support the findings of this study are not publicly available due to confidentiality agreements with our clients. These agreements prohibit the sharing of the data outside of the specific permissions granted for the research and publication of this manuscript.

Conflicts of Interest: All authors are employed by the Company "Energy Advice". The authors declare that the research was conducted in the absence of any commercial or financial relationships that could be construed as a potential conflict of interest.

References

1. EA-SAS Digital Twin. Available online: https://www.energyadvice.lt/en/products/ (accessed on 8 August 2022).
2. Fuller, A.; Fan, Z.; Day, C.; Barlow, C. Digital Twin: Enabling Technologies, Challenges and Open Research. *IEEE Access* **2020**, *8*, 108952–108971. [CrossRef]
3. Minerva, R.; Lee, G.M.; Crespi, N. Digital Twin in the IoT Context: A Survey on Technical Features, Scenarios, and Architectural Models. *Proc. IEEE* **2020**, *108*, 1785–1824. [CrossRef]
4. Loaiza, J.H.; Cloutier, R.J. Analyzing the Implementation of a Digital Twin Manufacturing System: Using a Systems Thinking Approach. *Systems* **2022**, *10*, 22. [CrossRef]
5. Wang, Y.; Chen, Q.; Kang, C.; Xia, Q. Clustering of Electricity Consumption Behavior Dynamics Toward Big Data Applications. *IEEE Trans. Smart Grid* **2016**, *7*, 2437–2447. [CrossRef]
6. Wang, Y.; Kang, X.; Chen, Z. A survey of Digital Twin techniques in smart manufacturing and management of energy applications. *Green Energy Intell. Transp.* **2022**, *1*, 100014. [CrossRef]
7. *ISO 23247-1:2021*; Automation Systems and Integration—Digital Twin Framework for Manufacturing—Part 1: Overview and General Principles. International Organization for Standardization: Geneva, Switzerland, 2021. Available online: https://www.iso.org/standard/75066.html (accessed on 18 January 2024).

8. *ISO 23247-2:2021*; Automation Systems and Integration—Digital Twin Framework for Manufacturing—Part 2: Reference Architecture. International Organization for Standardization: Geneva, Switzerland, 2021. Available online: https://www.iso.org/standard/78743.html (accessed on 18 January 2024).
9. *ISO 23247-3:2021*; Automation Systems and Integration—Digital Twin Framework for Manufacturing—Part 3: Digital Representation of Manufacturing Elements. International Organization for Standardization: Geneva, Switzerland, 2021. Available online: https://www.iso.org/standard/78744.html (accessed on 18 January 2024).
10. *ISO 23247-4:2021*; Automation Systems and Integration—Digital Twin Framework for Manufacturing—Part 4: Information Exchange. International Organization for Standardization: Geneva, Switzerland, 2021. Available online: https://www.iso.org/standard/78745.html (accessed on 18 January 2024).
11. Shao, G.; Frechette, S.; Srinivasan, V. An analysis of the new ISO 23247 series of standards on digital twin framework for manufacturing. In Proceedings of the ASME 2023 18th International Manufacturing Science and Engineering Conference, New Brunswick, NJ, USA, 12–16 June 2023.
12. Hribernik, K.; Cabri, G.; Mandreoli, F.; Mentzas, G. Autonomous, context-aware, adaptive Digital Twins—State of the art and roadmap. *Comput. Ind.* **2021**, *133*, 103508. [CrossRef]
13. Tao, F.; Cheng, Y.; Cheng, J.; Zhang, M.; Xu, W.; Qi, Q. Theories and technologies for cyber-physical fusion in digital twin shop-floor. *Jisuanji Jicheng Zhizao Xitong/Comput. Integr. Manuf. Syst. CIMS* **2017**, *23*, 1603–1611. [CrossRef]
14. Zhou, G.; Zhang, C.; Li, Z.; Ding, K.; Wang, C. Knowledge-driven digital twin manufacturing cell towards intelligent manufacturing. *Int. J. Prod. Res.* **2020**, *58*, 1034–1051. [CrossRef]
15. Zhang, R.; Wang, F.; Cai, J.; Wang, Y.; Guo, H.; Zheng, J. Digital twin and its applications: A survey. *Int. J. Adv. Manuf. Technol.* **2022**, *123*, 4123–4136. [CrossRef]
16. Nwokeji, J.; Aqlan, F.; Anugu, A.; Olagunju, A. Big data etl implementation approaches: A systematic literature review. In Proceedings of the International Conference on Software Engineering and Knowledge Engineering, SEKE, San Francisco, CA, USA, 1–3 July 2018; pp. 714–715. [CrossRef]
17. Ali, S.M.F.; Wrembel, R. From conceptual design to performance optimization of ETL workflows: Current state of research and open problems. *VLDB J.* **2017**, *26*, 777–801. [CrossRef]
18. Hu, W.; Zhang, T.; Deng, X.; Liu, Z.; Tan, J. Digital twin: A state-of-the-art review of its enabling technologies, applications and challenges. *J. Intell. Manuf. Spec. Equip.* **2021**, *2*, 1–34. [CrossRef]
19. Siddiqa, A.; Hashem, I.A.T.; Yaqoob, I.; Marjani, M.; Shamshirband, S.; Gani, A.; Nasaruddin, F. A survey of big data management: Taxonomy and state-of-the-art. *J. Netw. Comput. Appl.* **2016**, *71*, 151–166. [CrossRef]
20. Tao, F.; Zhang, M.; Nee, A.Y.C. Digital Twin and Big Data. In *Digital Twin Driven Smart Manufacturing*; Academic Press: Cambridge, MA, USA, 2019; pp. 183–202. [CrossRef]
21. Sepasgozar, S.M.E. Differentiating Digital Twin from Digital Shadow: Elucidating a Paradigm Shift to Expedite a Smart, Sustainable Built Environment. *Buildings* **2021**, *11*, 151. [CrossRef]
22. El Mokhtari, K.; Panushev, I.; McArthur, J.J. Development of a Cognitive Digital Twin for Building Management and Operations. *Front. Built Environ.* **2022**, *8*, 856873. [CrossRef]
23. Tao, F.; Zhang, M.; Nee, A.Y.C. *Digital Twin and Cloud, Fog, Edge Computing. Digital Twin Driven Smart Manufacturing*; Academic Press: Cambridge, MA, USA, 2019; pp. 171–181. [CrossRef]
24. Al-Ali, A.R.; Gupta, R.; Batool, T.Z.; Landolsi, T.; Aloul, F.; Al Nabulsi, A. Digital Twin Conceptual Model within the Context of Internet of Things. *Futur. Internet* **2020**, *12*, 163. [CrossRef]
25. Li, X.; Liu, H.; Wang, W.; Zheng, Y.; Lv, H.; Lv, Z. Big data analysis of the Internet of Things in the digital twins of smart city based on deep learning. *Futur. Gener. Comput. Syst.* **2022**, *128*, 167–177. [CrossRef]
26. Wallner, B.; Zwölfer, B.; Trautner, T.; Bleicher, F. Digital Twin Development and Operation of a Flexible Manufacturing Cell using ISO 23247. *Procedia CIRP* **2023**, *120*, 1149–1154. [CrossRef]
27. Lu, Y.; Liu, C.; Kevin, I.; Wang, K.; Huang, H.; Xu, X. Digital Twin-driven smart manufacturing: Connotation, reference model, applications and research issues. *Robot. Comput. Integr. Manuf.* **2020**, *61*, 101837. [CrossRef]
28. Best Practices—Airflow Documentation. Available online: https://airflow.apache.org/docs/apache-airflow/stable/best-practices.html (accessed on 8 August 2022).
29. Aquilanti, P.-Y.; Kendrex, S.; Koop, M. AWS Batch Dos and Don'ts: Best Practices in a Nutshell | AWS HPC Blog. Available online: https://aws.amazon.com/blogs/hpc/aws-batch-best-practices/ (accessed on 8 August 2022).
30. Liston, B. Creating a Simple 'Fetch & Run' AWS Batch Job | AWS Compute Blog. Available online: https://aws.amazon.com/blogs/compute/creating-a-simple-fetch-and-run-aws-batch-job/ (accessed on 8 August 2022).
31. VPS Serveriai—Interneto Vizija. Available online: https://www.iv.lt/vps-serveriai/#konteineriai (accessed on 10 August 2022).
32. Amazon EC2 Pricing—Amazon Web Services. Available online: https://aws.amazon.com/ec2/pricing/ (accessed on 10 August 2022).
33. Khalid, M.; Yousaf, M.M. A Comparative Analysis of Big Data Frameworks: An Adoption Perspective. *Appl. Sci.* **2021**, *11*, 11033. [CrossRef]
34. Rovnyagin, M.M.; Shipugin, V.A.; Ovchinnikov, K.A.; Durachenko, S.V. Intelligent container orchestration techniques for batch and micro-batch processing and data transfer. *Procedia Comput. Sci.* **2021**, *190*, 684–689. [CrossRef]
35. Pishgoo, B.; Azirani, A.A.; Raahemi, B. A hybrid distributed batch-stream processing approach for anomaly detection. *Inf. Sci.* **2020**, *543*, 309–327. [CrossRef]

36. *ISO/IEC TR 30172:2023*; Internet of THINGS (loT)—Digital twin—Use Cases. International Organization for Standardization: Geneva, Switzerland, 2023. Available online: https://www.iso.org/standard/81578.html (accessed on 24 January 2024).
37. *ISO/IEC 30173:2023*; Digital Twin—Concepts and Terminology. International Organization for Standardization: Geneva, Switzerland, 2023. Available online: https://www.iso.org/standard/81442.html (accessed on 24 January 2024).
38. Wang, Z.; Gupta, R.; Han, K.; Wang, H.; Ganlath, A.; Ammar, N.; Tiwari, P. Mobility Digital Twin: Concept, Architecture, Case Study, and Future Challenges. *IEEE Internet Things J.* **2022**, *9*, 17452–17467. [CrossRef]