

# Digital twins and IIoT: comparison of Prometheus and InfluxDB

Bauyrzhan Amirkhanov<sup>1</sup>, Timur Ishmurzin<sup>1</sup>, Murat Kunelbayev<sup>1,2</sup>, Gulshat Amirkhanova<sup>1</sup>, Azim Aidynuly<sup>1</sup>, Gulnur Tyulepberdinova<sup>1</sup>

<sup>1</sup>Department of AI and Big Data, Faculty of Information Technology, Al-Farabi Kazakh National University, Almaty, Kazakhstan

<sup>2</sup>Institute of Information and Computational Technologies, Almaty, Kazakhstan

## Article Info

### Article history:

Received Dec 10, 2024

Revised Jul 29, 2025

Accepted Sep 11, 2025

### Keywords:

Data monitoring and visualization

Digital twin

InfluxDB

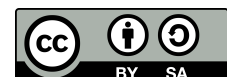
Prometheus

Streaming data

## ABSTRACT

This article presents a comparative analysis of data monitoring and visualization tools—Prometheus and InfluxDB—in the context of digital twins (DTs) applied to industrial settings. DTs optimize production processes using industrial internet of things (IIoT) technologies. Mathematical models assessed the tools based on response time, resource consumption, throughput, and reliability. Prometheus is better suited for high-frequency monitoring, achieving a response time of 0.01 seconds and processing up to 10,000 metrics per second—10–15% better than InfluxDB. It consumes 1.5 times less memory (100 MB versus 150 MB), making it faster and more resource-efficient. Conversely, InfluxDB excels in long-term storage and analytics, handling up to 8,000 metrics per second with a response time of 0.09 seconds. However, it requires more resources, including higher CPU usage (20% versus 15%). Both tools integrate seamlessly with Grafana for visualization, offering flexibility for real-time monitoring and decision-making. The study provides actionable insights for selecting monitoring systems based on project-specific requirements, highlighting Prometheus's efficiency in dynamic scenarios and InfluxDB's strength in analytics-focused tasks.

*This is an open access article under the [CC BY-SA](#) license.*



## Corresponding Author:

Timur Ishmurzin

Department of AI and Big Data, Faculty of Information Technology, Al-Farabi Kazakh National University  
Almaty, Kazakhstan

Email: timon.ishmurzin@gmail.com

## 1. INTRODUCTION

Digital twins (DTs) have become one of the primary tools for digital transformation in industry and other sectors, providing enterprises with the ability to accurately monitor, analyze, and optimize their assets and processes. The concept of the DT was first introduced in 2003 by NASA for modeling and predicting the condition of spacecraft in analytics [1]. Over time, the idea of DTs spread to other industries and became the foundation for creating cyber-physical systems capable of replicating and analyzing data about a physical object in digital format [2]. DTs, also known as virtual replicas, are dynamic digital models of physical objects or systems that are continuously updated in analytics platform based on data received from sensors and other sources [3]. Combined with big data technologies, the internet of things (IoT), and artificial intelligence, DTs can analyze system behavior and predict potential failures, enhancing production management and reducing risks [4]. They have become a critical component of the Industry 4.0 concept, facilitating the transition to smart

manufacturing processes [5]. The application of DTs is widely adopted in manufacturing processes, where their use enables equipment monitoring and the prediction of potential failures [6]. This helps companies minimize maintenance costs and prevent downtime, thereby increasing overall productivity [7]. According to research, DTs can reduce equipment maintenance expenses and extend its lifespan through timely diagnostics and failure prevention [8].

In the energy sector, DTs assist in managing complex systems by monitoring equipment conditions, predicting failures, and optimizing energy consumption, thereby reducing operational costs and minimizing losses [9]. For example, they allow companies to forecast equipment load and lower repair and maintenance expenses [10]. In construction and architecture, DTs are used for designing and managing infrastructure projects, enabling the creation of virtual building models and determining their operational parameters [11]. However, despite their clear advantages, implementing DTs involves certain challenges. Specifically, a high level of cybersecurity is required to protect data, as any failure or data breach could have serious consequences for a company [12]. Additionally, DTs demand significant computational resources and support from skilled professionals for their development, deployment, and operation [13]. Key factors also include data standardization and the necessity of integrating DTs with existing systems [14].

To date, researchers have proposed various architectural approaches to implementing DTs. One approach utilizes cloud infrastructure, which enables the storage and processing of large volumes of data in analytics platform [15]. The IoT also plays a key role in connecting physical objects to their digital counterparts, facilitating efficient asset management and system behavior prediction [16]. Another approach integrates DTs with artificial intelligence technologies, allowing for the automation of decision-making processes and improving the accuracy of analysis [17].

DTs are actively used in industries such as logistics, healthcare, and automotive manufacturing [18]. In logistics, they help optimize supply chain management, significantly reducing time and financial costs [19]. In healthcare, DTs enable the modeling of patient conditions and the prediction of disease progression, enhancing diagnostic accuracy and improving the quality of treatment [20]. In the automotive industry, they are used for real-time monitoring and diagnostics of vehicle components, increasing safety and operational efficiency [21]. Despite numerous advantages, the development prospects for DTs remain open-ended. In the near future, the capabilities of DTs are expected to expand through integration with new technologies such as blockchain, which could enhance data security and integrity [22]. The development of DTs for applications like smart cities, which aim to optimize infrastructure and improve quality of life, is critically dependent on the underlying communication protocols that ensure timely and stable data exchange [23]. Overall, DTs represent a powerful tool for creating smart systems capable of adapting to real-world conditions, analyzing data, and proposing optimal solutions [24].

The ability of these DTs to adapt to real-world conditions by analyzing data and proposing optimal solutions is enabled by a wide range of machine learning algorithms [25]. Both tools can work with Grafana to show information by a software module between the data source and the application itself [26]. Both tools are widely used for data collection, storage, and visualization but differ in architecture, performance, and integration capabilities with real-time systems. Prometheus is generally employed for high-frequency monitoring, making it suitable for tasks where timely detection of data changes is critical [27], [28]. In contrast, InfluxDB stands out for its long-term data storage capabilities and high speed of time-series processing, which is especially important for DT tasks that require the retention and analysis of large data volumes [29], [30].

Grafana plays an essential role in data visualization and analysis, supporting integration with both solutions and enabling the creation of intuitive dashboards for real-time monitoring of metrics [31], [32]. The use of Grafana in combination with Prometheus and InfluxDB provides flexible options for configuring and displaying necessary metrics, significantly simplifying decision-making and management processes in industrial environments. The implementation of InfluxDB and Grafana is showcased in the OpenTwins DT platform, considered an open-source project [33].

DTs have become a critical tool for optimizing industrial processes, leveraging real-time data monitoring systems. Among the many available platforms, Prometheus and InfluxDB stand out due to their widespread use in industrial internet of things (IIoT) environments and their distinct capabilities in handling time-series data.

The aim of this study is to identify the key advantages and disadvantages of Prometheus and InfluxDB in the context of DTs, focusing on parameters such as data processing speed, scalability support, configuration flexibility, and integration capabilities with other systems. Particular attention is given to evaluating how

both systems handle intensive workloads under IIoT conditions, as well as their ability to provide reliable and efficient real-time monitoring. Based on these evaluations, the research aims to offer recommendations for selecting the most suitable solution for various use cases of DTs in industry, considering the specific requirements and characteristics of each platform.

## 2. METHOD

Real-time monitoring systems are designed for the collection, storage, and analysis of streaming data. They handle time series, which represent sequences of data points associated with timestamps (e.g., temperature, speed, and CPU load). These systems enable the tracking of infrastructure, equipment, or application status, the detection of anomalies, and prompt responses to failures. The primary functions of such systems include: i) data collection from various sources in analytics platform, ii) data storage in an optimized format for time-series data, iii) data analysis using query languages, iv) visualization of results through graphs and dashboards, and v) alerts triggered when predefined thresholds are breached.

To perform a comparative analysis of Prometheus and InfluxDB, we designed a reference DT architecture, which is detailed below and illustrated in Figure 1. The architecture follows a layered approach to process data from the physical environment to the digital representation.

The data flow originates from IIoT devices on the shop floor. This raw data is first processed by the Edge computing module, which performs initial filtering and aggregation to reduce network traffic and latency. The IIoT connection module, based on Mainflux, then ensures secure and scalable data transmission to the central platform. At the core of the system lies the DT management module (e.g., based on OpenRemote), which orchestrates the DT's state and logic. It interacts with several key components: The data storage module, which is the focus of this study. This component is implemented using either Prometheus for high-frequency monitoring or InfluxDB for long-term storage and analytics. The messaging system (e.g., NATS) facilitates asynchronous communication between different microservices of the platform. The incremental Machine Learning module (e.g., based on MLflow) continuously analyzes the data to retrain models and provide predictive insights. The extended things management and DT instances module is responsible for managing the models and instances of various DTs. Finally, the top layers provide user interaction and visualization. The data visualisation module, typically implemented with Grafana connected to the data storage, displays real-time metrics and dashboards. An optional 3D simulation module (e.g., using Babylon.js) can offer an immersive virtual representation of the physical asset. This integrated approach, with its emphasis on edge processing and modularity, is designed to enhance the overall system's performance, efficiency, and accuracy.

### 2.1. Mathematical model of the Prometheus monitoring system

#### 2.1.1. Key components of the model

Time series:

Let  $M$  be the set of all metrics, where each metric  $m \in M$  is represented as a time series:

$$m : T \rightarrow V, \quad (1)$$

where:  $T$  is the set of timestamps (usually discrete time intervals) and  $V$  is the set of possible metric values (e.g., real numbers).

Scraping:

Define the scraping function  $S$ , which periodically collects metrics from exporters:

$$S : M \times T \rightarrow V, \quad (2)$$

where for each metric  $m$  and time  $t \in T$ ,  $S(m, t)$  returns the value of the metric  $m$  at time  $t$ .

Data storage:

Prometheus stores the collected time series in a database. Let  $DB$  be the time series database, then:

$$DB = \{(m, t, v) \mid m \in M, t \in T, v = S(m, t)\}. \quad (3)$$

#### 2.1.2. Queries (PromQL)

Query functions:

Let  $Q$  be the set of all possible queries. Each query  $q \in Q$  represents a function:

$$q : DB \rightarrow R, \quad (4)$$

where  $R$  is the set of results (e.g., time series and aggregated values). Examples of queries:  
Filtering by metrics:

$$q_{\text{filter}}(DB) = \{m \in DB \mid m \text{ satisfies the filtering condition}\}. \quad (5)$$

Aggregation:

$$q_{\text{sum}}(DB) = \sum_{m \in M} m(t), \quad \forall t \in T. \quad (6)$$

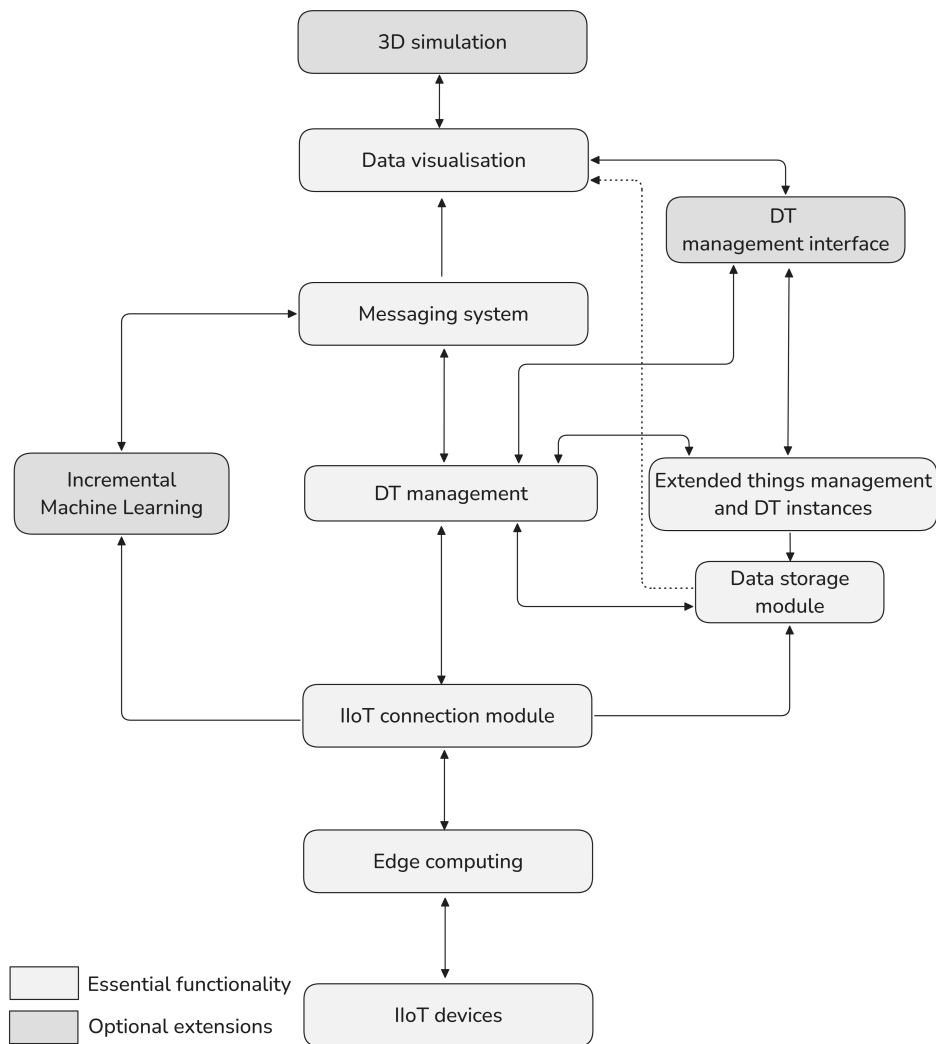


Figure 1. Architecture of DT

### 2.1.3. Alerting

Alerting rules:

Let  $A$  be the set of alerting rules, where each rule  $a \in A$  is defined as (8):

$$a : Q \times \Theta \rightarrow \text{Alert}, \quad (7)$$

where:  $Q$  is the query to retrieve the required data,  $\Theta$  is the set of conditions (e.g., threshold values), and Alert is the trigger for the alert if the conditions are met.

Condition for triggering an alert:

For a rule  $a = (q, \theta)$ , an alert is triggered if:

$$q(DB) \geq \theta \quad (8)$$

for a certain time period or number of observations.

#### 2.1.4. System operation cycle

Data collection:

$$\forall m \in M, \forall t \in T, DB \leftarrow DB \cup \{(m, t, S(m, t))\}. \quad (9)$$

Data storage:

$$DB = \{(m, t, v) \mid m \in M, t \in T, v = S(m, t)\}. \quad (10)$$

Query processing:

$$\forall q \in Q, r = q(DB). \quad (11)$$

Alert checking:

$$\forall a \in A, \text{ if } q_a(DB) \text{ satisfies } \theta_a, \text{ then trigger an alert.} \quad (12)$$

Example model:

Let us consider an example of monitoring the metric CPU usage ( $m_{\text{cpu}}$ )

Scraping:

$$S(m_{\text{cpu}}, t) = \text{CPU usage at time } t. \quad (13)$$

Query for average usage:

$$q_{\text{avg}}(DB) = \frac{1}{|T'|} \sum_{t' \in T'} m_{\text{cpu}}(t'), \quad (14)$$

where  $T' = \{t' \mid t - 5 \text{ minutes} \leq t' \leq t\}$ .

Alerting rule:

$$a_{\text{high\_cpu}} : q_{\text{avg}}(DB) > 80\%. \quad (15)$$

If the average CPU usage exceeds 80% over the last 5 minutes, an alert is triggered.

## 2.2. Mathematical model of the InfluxDB monitoring system

### 2.2.1. Key components of the model

Time series:

Let  $M$  be the set of all metrics, where each metric  $m \in M$  is represented as a time series:

$$m : T \rightarrow V, \quad (16)$$

where:  $T$  is the set of timestamps (usually discrete time intervals) and  $V$  is the set of possible metric values (e.g., real numbers).

Write operations:

Define a write function  $W$ , which periodically writes metrics to the database:

$$W : M \times T \times V \rightarrow DB, \quad (17)$$

where for each metric  $m$ , timestamp  $t \in T$ , and value  $v \in V$ ,  $W(m, t, v)$  adds the record  $(m, t, v)$  to the time series database  $DB$ .

Data storage:

InfluxDB stores collected time series in the database. Let  $DB$  be the time series database, then:

$$DB = \{(m, t, v) \mid m \in M, t \in T, v = W(m, t, v)\}. \quad (18)$$

### 2.2.2. Queries (InfluxQL/Flux)

Query functions:

Let  $Q$  be the set of all possible queries. Each query  $q \in Q$  represents a function:

$$q : DB \rightarrow R, \quad (19)$$

where  $R$  is the set of results (e.g., time series and aggregated values).

Examples of queries:

Filtering by metrics:

$$q_{\text{filter}}(DB) = \{m \in DB \mid m \text{ satisfies the filtering condition}\}. \quad (20)$$

Aggregation:

$$q_{\text{avg}}(DB) = \frac{1}{|T'|} \sum_{t' \in T'} m(t'), \quad (21)$$

where  $T' = \{t' \mid t - \Delta t \leq t' \leq t\}$ .

### 2.2.3. Alerting

Alerting rules:

Let  $A$  be the set of alerting rules, where each rule  $a \in A$  is defined as (22):

$$a : Q \times \Theta \rightarrow \text{Alert}, \quad (22)$$

where:  $Q$  is the query to retrieve the required data,  $\Theta$  is the set of conditions (e.g., threshold values), and Alert is the alert trigger if the conditions are met.

Condition for triggering an alert:

For a rule  $a = (q, \theta)$ , an alert is triggered if:

$$q(DB) \geq \theta \quad (23)$$

for a certain period of time or a number of observations.

### 2.2.4. System operation cycle

Data collection:

$$\forall m \in M, \forall t \in T, \forall v \in V, DB \leftarrow DB \cup \{(m, t, v)\}. \quad (24)$$

Data storage:

$$DB = \{(m, t, v) \mid m \in M, t \in T, v = W(m, t, v)\}. \quad (25)$$

Query processing:

$$\forall q \in Q, r = q(DB). \quad (26)$$

Alert checking:

$$\forall a \in A, \text{ if } q_a(DB) \text{ satisfies } \theta_a, \text{ then trigger an alert.} \quad (27)$$

Example model:

Data recording:

$$W(m_{\text{cpu}}, t, v) = \text{CPU usage value at time } t. \quad (28)$$

Query for average usage over the last 5 minutes:

$$q_{\text{avg}}(DB) = \frac{1}{|T'|} \sum_{t' \in T'} m_{\text{cpu}}(t'), \quad (29)$$

where  $T' = \{t' \mid t - 5 \text{ minutes} \leq t' \leq t\}$ .

Alert rule:

$$a_{\text{high\_cpu}} : q_{\text{avg}}(DB) > 80\%. \quad (30)$$

If the average CPU usage exceeds 80% over the last 5 minutes, an alert is triggered.

To evaluate and compare Prometheus and InfluxDB, three mathematical models for efficiency assessment were developed based on different parameters.

### 2.3. Efficiency evaluation mathematical model №1

This model evaluates the system's performance dependency on response time, memory usage, throughput, and queue duration.

$$E = \frac{w_1 \cdot \text{norm}_T + w_2 \cdot \text{norm}_M + w_3 \cdot \text{norm}_F + w_4 \cdot \text{norm}_L}{w_{\text{total}}} \quad (31)$$

$\text{norm}_T$ ,  $\text{norm}_M$ ,  $\text{norm}_F$ , and  $\text{norm}_L$  are the normalized values of parameters, calculated using (32)-(35):

$$\text{norm}_T = \frac{T_{\max} - T}{T_{\max} - T_{\min}} \quad (32)$$

$$\text{norm}_M = \frac{M_{\max} - M}{M_{\max} - M_{\min}} \quad (33)$$

$$\text{norm}_L = \frac{L_{\max} - L}{L_{\max} - L_{\min}} \quad (34)$$

$$\text{norm}_F = \frac{F}{F_{\max}} \quad (35)$$

where:  $T$  is response time,  $M$  is memory consumption,  $F$  is throughput,  $L$  is queue duration, and  $w$  is parameter weight (significance).

The operations of this model are presented as Python code in Figure 2.

```

"""Function to calculate efficiency E with parameter normalization."""
# Normalize parameters
norm_T = (T_max - T) / (T_max - T_min) # Lower T, higher norm_T
norm_M = (M_max - M) / (M_max - M_min) # Lower M, higher norm_M
norm_F = F / F_max # Higher F, higher norm_F
norm_L = (L_max - L) / (L_max - L_min) # Lower L, higher norm_L

# Clip normalized values to [0, 1]
norm_T = np.clip(norm_T, 0, 1)
norm_M = np.clip(norm_M, 0, 1)
norm_F = np.clip(norm_F, 0, 1)
norm_L = np.clip(norm_L, 0, 1)

# Calculate efficiency
E = (w1 * norm_T + w2 * norm_M + w3 * norm_F + w4 * norm_L) / w_total
return E

```

Figure 2. Python code of efficiency evaluation mathematical model №1

### 2.4. Efficiency evaluation mathematical model №2

This model evaluates system reliability and resilience based on CPU usage, disk usage, system reliability, and recovery time after a failure.

$$E = \frac{w_1 \cdot \frac{1}{C} + w_2 \cdot \frac{1}{D} + w_3 \cdot R + w_4 \cdot \frac{1}{S}}{w_1 + w_2 + w_3 + w_4} \quad (36)$$

where:  $C$  is CPU usage (%),  $D$  is disk usage (MB/s),  $R$  is system reliability (from 0 to 1),  $S$  is recovery time after failure (seconds), and  $w$  is parameter weight (significance).

The operations of this model are presented as Python code in Figure 3.

```
def calculate_efficiency(C, D, R, S):
    """Function to calculate efficiency E."""
    E = (w1 * (1 / C) + w2 * (1 / D) + w3 * R + w4 * (1 / S)) / w_total
    return E
```

Figure 3. Python code of efficiency evaluation mathematical model №2

## 2.5. Efficiency evaluation mathematical model №3

This model evaluates system performance based on data volume, write and read speed, as well as scalability across metrics and users.

$$E = w_1 \cdot \frac{V_d - V_{d,\min}}{V_{d,\max} - V_{d,\min}} + w_2 \cdot \frac{W_r - W_{r,\min}}{W_{r,\max} - W_{r,\min}} + w_3 \cdot \frac{R_r - R_{r,\min}}{R_{r,\max} - R_{r,\min}} + w_4 \cdot \frac{S_m - S_{m,\min}}{S_{m,\max} - S_{m,\min}} + w_5 \cdot \frac{S_u - S_{u,\min}}{S_{u,\max} - S_{u,\min}}. \quad (37)$$

where:  $w$  is weight coefficients reflecting the importance of each parameter,  $(V_{d,\min}, V_{d,\max})$  are minimum and maximum values of data volume,  $(W_{r,\min}, W_{r,\max})$  are minimum and maximum values of write speed,  $(R_{r,\min}, R_{r,\max})$  are minimum and maximum values of read speed,  $(S_{m,\min}, S_{m,\max})$  are minimum and maximum values of scalability by metrics, and  $(S_{u,\min}, S_{u,\max})$  are minimum and maximum values of scalability by users.

The operations of this model are presented as Python code in Figure 4.

```
def calculate_efficiency(Vd, Wr, Rr, Sm, Su):
    """Function to calculate efficiency E with parameter normalization."""
    # Normalize parameters
    norm_Vd = (Vd - Vd_min) / (Vd_max - Vd_min)
    norm_Wr = (Wr - Wr_min) / (Wr_max - Wr_min)
    norm_Rr = (Rr - Rr_min) / (Rr_max - Rr_min)
    norm_Sm = (Sm - Sm_min) / (Sm_max - Sm_min)
    norm_Su = (Su - Su_min) / (Su_max - Su_min)

    # Clip normalized values to [0, 1]
    norm_Vd = np.clip(norm_Vd, 0, 1)
    norm_Wr = np.clip(norm_Wr, 0, 1)
    norm_Rr = np.clip(norm_Rr, 0, 1)
    norm_Sm = np.clip(norm_Sm, 0, 1)
    norm_Su = np.clip(norm_Su, 0, 1)

    # Calculate efficiency
    E = (w1 * norm_Vd + w2 * norm_Wr + w3 * norm_Rr + w4 * norm_Sm + w5 * norm_Su)
    return E
```

Figure 4. Python code of efficiency evaluation mathematical model №3

In each mathematical model, the parameters are normalized based on their range variations to ensure a balanced influence on the final result. The selection of weighting coefficients ( $w_i$ ) for each of the three efficiency evaluation models (model №1, model №2, and model №3) was then performed using an empirical, iterative tuning process. This approach was adopted as a formal, universally accepted methodology for assigning weights to the specific combination of performance parameters—such as response time, memory usage, throughput, CPU usage, disk usage, data volume, read/write speeds, and scalability metrics—in the context of IIoT DT monitoring systems is not readily available.

The primary objective of this iterative process was to derive a set of weights that would enable the composite efficiency score ( $E$ ) to provide a meaningful and sensitive reflection of the overall system performance, aligning with the observed behaviors and trade-offs during experimental stress tests. The process involved the following considerations:

- a. Initial weight estimation: the process commenced with an initial qualitative assessment of the relative importance of each parameter within the specific context of its respective model. For instance, in scenarios evaluating real-time responsiveness (model №1), parameters like 'response time' ( $T$ ) and 'throughput' ( $F$ ) were initially considered to have a higher impact than 'queue duration' ( $L$ ) if queues were generally short.
- b. Iterative adjustments: starting from these initial estimations, the weights were systematically adjusted. This involved incrementally increasing or decreasing the weight of one parameter (or a small subset of parameters) at a time and observing the resultant impact on the calculated efficiency ( $E$ ) for both Prometheus and InfluxDB across the collected benchmark data.
- c. Evaluation criteria against experimental observations: the 'effectiveness' of each set of weights was evaluated by comparing the behavior of the calculated efficiency score ( $E$ ) against the raw performance data and qualitative observations from the stress tests. Specifically, we sought weights that would lead to an efficiency score ( $E$ ) that:
  - Demonstrably decreased when a system exhibited known performance degradation on critical metrics (e.g., a sharp increase in response time or significant resource saturation).
  - Clearly differentiated between Prometheus and InfluxDB in scenarios where one system showed a distinct advantage based on specific raw metrics relevant to that model.
  - Provided a stable and consistent representation of performance during periods where the underlying system metrics were also stable.
  - Reflected the pre-defined requirements for efficiency, resource consumption, and reliability pertinent to each model. For example, for model №2, which focuses on reliability and resilience, weights were tuned to ensure that high system reliability ( $R$ ) and fast recovery time ( $S$ ) positively influenced the  $E$  score, while high CPU ( $C$ ) and disk ( $D$ ) usage had a negative impact.
- d. Refinement and final selection: this iterative refinement process was repeated until a set of weights was identified for each model that yielded the most consistent, intuitively correct, and representative assessment of overall system effectiveness for the comparative analysis presented in this study. The final weights are those used in the subsequent calculations and presentation of results.

While this empirical trial-and-error approach does not guarantee a globally optimal set of weights, it provided a pragmatic and data-driven methodology to assign significance to various performance indicators, enabling a structured comparison suitable for the objectives of this research.

These mathematical models, utilizing the determined weights, formed the foundation for the comparative analysis, and their results are used to identify the most effective solution for developing DTs based on IIoT.

## 2.6. Evaluation of software using stress testing and the Isolation Forest algorithm

To assess the performance and reliability of the monitoring systems Prometheus and InfluxDB, a series of stress tests were conducted in a controlled environment on a personal computer. The detailed hardware and software configuration of the testbed is provided below.

### 2.6.1. Hardware specifications

The experiments were performed on a personal computer with the following hardware components:

- Processor: AMD Ryzen 5 5600H with Radeon Graphics (6-core and 12-thread).
- Memory: 16 GB of DDR4 RAM.
- Storage: the tests, along with the operating system and all evaluated software, were executed on a 512 GB NVMe SSD (Samsung MZALQ512HBLU-00BL2) to ensure optimal I/O performance during the experiments.

### 2.6.2. Software and load generation

The software environment and load simulation were configured as follows:

- Operating system: the host system was running Windows 11. The experiments were executed within the Windows Subsystem for Linux 2 (WSL2) environment, using an Ubuntu 22.04 LTS distribution.
- Monitoring tools: the key software versions under test were Prometheus v2.45.0, InfluxDB v2.7.1, and Grafana v9.5.3 for visualization.

- Load simulation: a constant load was generated using a custom Python script to simulate a typical IIoT scenario. The script emulated 1,000 devices, each transmitting 10 unique time-series metrics per second. This resulted in a consistent and sustained data ingestion rate of 10,000 metrics/second for the entire 100-second duration of each test. This constant data velocity allowed for a fair and direct comparison of the systems' behavior under sustained stress.

## 2.7. Isolation Forest algorithm

Isolation Forest is a machine learning algorithm designed for detecting anomalies in data. It is based on the idea of isolating observations by constructing multiple random trees (isolation trees). Anomalies are typically easier to isolate as they exist in lower-density regions of the data or exhibit unique characteristics compared to normal data.

### a. Advantages of Isolation Forest

- Efficiency: scales well with large datasets and offers high processing speed.
- No need for labeled data: suitable for tasks without prior data labeling.
- High accuracy: effectively detects both local and global anomalies.
- Interpretability: provides clear results, pinpointing specific anomalous observations.

### b. Principle of operation

- 1) Isolation of observations: the algorithm constructs multiple random trees, isolating each observation.
- 2) Computation of isolation path length: for each observation, the average isolation path length is calculated. Anomalies generally have shorter average path lengths because they are easier to isolate.
- 3) Anomaly determination: based on the average path length, the algorithm determines whether an observation is anomalous.

The Isolation Forest algorithm was chosen due to its ability to effectively detect anomalies in high-dimensional streaming data, which is characteristic of monitoring systems in IIoT environments. Unlike clustering-based methods (e.g., DBSCAN), Isolation Forest does not require predefined density thresholds, making it more adaptable to the diverse and dynamic nature of IIoT datasets. Additionally, its efficiency with large-scale datasets aligns with the high data throughput requirements of Prometheus and InfluxDB stress tests. The interpretability of the algorithm allows for clear identification of performance anomalies, facilitating actionable insights during system evaluation.

## 2.8. Scientific novelty

The scientific novelty of this research lies in the first-ever comprehensive comparative analysis of the monitoring systems Prometheus and InfluxDB in the context of their application for DTs in industry using IIoT technologies.

The developed mathematical models enabled a quantitative assessment of the efficiency of these systems across a range of key parameters, including response time, resource consumption, throughput, and reliability. For the first time in this context, the machine learning algorithm Isolation Forest was applied for anomaly detection during stress testing of monitoring systems. This approach not only facilitates the identification of hidden deviations in system performance under load but also allows for the prediction of potential failure points, which is critically important for industrial applications with high reliability requirements.

Furthermore, the developed mathematical models were tested under conditions closely approximating real industrial loads, enhancing the practical significance of the obtained results. This provides enterprises with tools for assessing and optimizing their monitoring systems in real operating conditions.

Another important aspect of the scientific novelty is the potential to adapt the proposed models to other monitoring systems and time-series databases, opening the way to creating universal methodologies for evaluating and comparing various tools in the IIoT sphere. This contributes to the development of more efficient and reliable monitoring systems capable of meeting the growing industrial demand for real-time processing and analysis of large data volumes.

Thus, this research makes a significant contribution to the field of development and optimization of monitoring systems for DTs, expanding the boundaries of applying modern technologies and providing new tools for enhancing the efficiency and reliability of industrial processes.

## 2.9. Theoretical novelty of the research

This study presents an original theoretical development in the form of new mathematical models for evaluating the effectiveness of monitoring systems in the context of DTs and IIoT. The theoretical novelty lies in the following key aspects: for the first time, three integrated mathematical models are proposed that combine several critical parameters of monitoring systems into a single effectiveness function. These parameters include response time, memory and CPU usage, throughput, system reliability, recovery time after failure, data volume, read and write speeds, as well as scalability concerning metrics and users. Effectiveness evaluation model no.1: this model combines parameters such as response time, memory usage, throughput, and queue duration. The novelty of this model lies in its ability to quantitatively assess the trade-offs between performance and resource constraints in real time. Effectiveness evaluation model no.2: the second model focuses on system reliability and resilience, taking into account CPU usage, disk space, overall system reliability, and recovery time after failure. Such a comprehensive approach to evaluating the reliability of monitoring systems in the context of IIoT has not been previously proposed. Effectiveness evaluation model no.3: the third model assesses system performance based on data volume, read and write speeds, and scalability. It considers the characteristics of high-load IIoT systems and DTs that handle large data streams. For the first time, the choice of the Isolation Forest algorithm for anomaly detection tasks in DT monitoring systems is justified. The theoretical analysis demonstrated that this algorithm is more effective for high-dimensional and dynamic data typical of IIoT compared to traditional methods like DBSCAN.

The algorithm was theoretically adapted to operate under streaming data conditions with high dimensionality and high data arrival rates. This includes optimizing tree structures and isolation methods to enhance the accuracy and speed of anomaly detection. A deep theoretical analysis was conducted on how various parameters of monitoring systems are interrelated and influence the overall system effectiveness. This includes studying nonlinear dependencies and potential performance bottlenecks in monitoring systems. Based on the developed models, new patterns in the behavior of monitoring systems under load were identified, contributing to a better understanding of their operational principles and optimization opportunities. The proposed mathematical models can be applied to evaluate different monitoring systems and time series databases, allowing the creation of a unified standard for their comparison and assessment in the context of IIoT and DTs. The developed methodology opens new avenues for theoretical research in the optimization of monitoring systems, especially under high load and real-time requirements. The introduced mathematical models and methods expand the existing mathematical framework, providing new tools for analyzing and optimizing monitoring systems in the context of DTs and IIoT. The innovative application and adaptation of the Isolation Forest algorithm for monitoring tasks extend the boundaries of machine learning methods in a theoretical context, offering new ways to detect and predict anomalies in analytics. The theoretical developments presented in the study have direct practical applications, enabling enterprises to more accurately evaluate and optimize their monitoring systems, leading to increased efficiency and reliability of industrial processes. The proposed theoretical approaches can be used to develop new monitoring and analytics algorithms tailored to the specific requirements of various industrial sectors.

The theoretical novelty of this research lies in the development of new mathematical models and theoretical approaches for the comprehensive evaluation of monitoring systems' effectiveness in the context of DTs and IIoT. The integration of multifactorial evaluation models with adapted machine learning algorithms provides a new perspective on the challenges of monitoring high-load systems and opens up opportunities for further theoretical and practical advancements in this field.

## 3. RESEARCH RESULTS

To evaluate the performance and reliability of Prometheus and InfluxDB in the context of DT development using IIoT, the developed mathematical models were applied. Each model considered various aspects of system operation: resource consumption, throughput, response time, and reliability. The comparison was conducted under varying key parameters for both platforms.

### 3.1. Response time and resource consumption

The testing results, presented in Table 1, demonstrated that both systems exhibited different levels of efficiency depending on the workload. The results clearly show that Prometheus delivered significantly lower response times (0.012 sec) compared to InfluxDB (0.085 sec), establishing it as the preferred solution

for latency-critical scenarios. At the same time, Prometheus showed stable performance under increased load, which could be an advantage for scalable systems.

Table 1. Average performance parameters (model №1) with standard deviations

Parameters	Prometheus	InfluxDB
Response time ( $T$ )	$0.012 \pm 0.003$ sec	$0.085 \pm 0.015$ sec
Memory consumption ( $M$ )	$98 \pm 5$ MB	$155 \pm 8$ MB
Throughput ( $F$ )	$9950 \pm 250$ metrics/sec	$8100 \pm 300$ metrics/sec
Queue duration ( $L$ )	$0.0011 \pm 0.0002$ sec	$0.0018 \pm 0.0004$ sec

Note: values represent mean  $\pm$  standard deviation from 100-second stress tests for model №1 parameters

Figure 5 illustrates the system efficiency over a 100-second test period, as evaluated by mathematical model №1 (which integrates response time ( $T$ ), memory usage ( $M$ ), throughput ( $F$ ), and queue duration ( $L$ )). Specifically, Figure 5(a) depicts the real-time efficiency of Prometheus when paired with Grafana. This visualization shows that Prometheus generally maintains a high efficiency level, albeit with noticeable fluctuations and distinct, sharp drops in performance around the 50th and 80th-second marks, indicating periods of system stress or anomaly. In contrast, Figure 5(b) presents the efficiency trajectory of InfluxDB under identical test conditions with Grafana. While InfluxDB's efficiency also varies, its curve is characterized by more frequent oscillations of a generally smaller amplitude compared to the more pronounced dips observed for Prometheus. Both Figures 5(a) and (b) effectively demonstrate how the chosen weighting coefficients ( $w$ ) within model №1 translate into the observable performance dynamics of each system under varying load conditions.

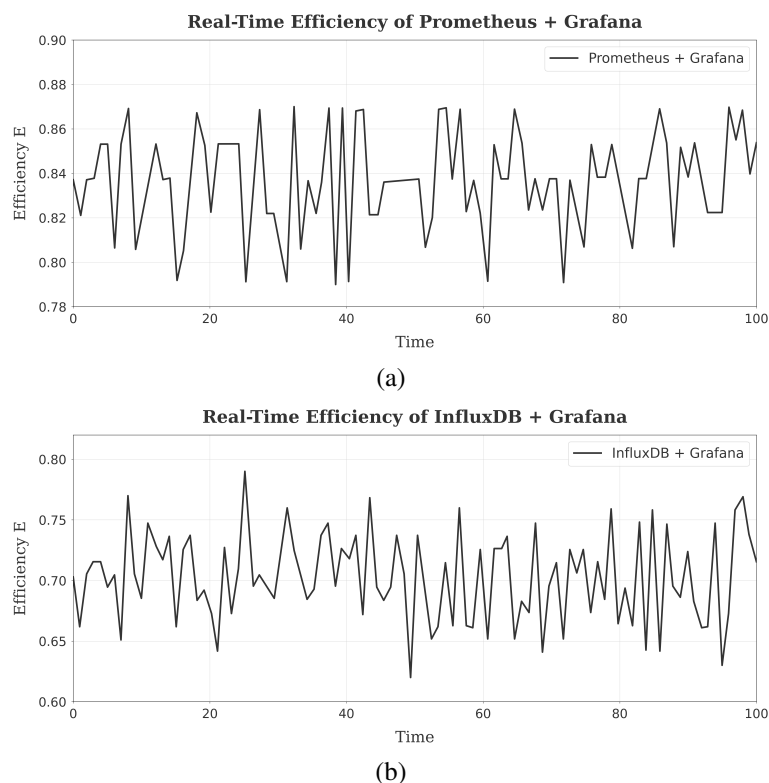


Figure 5. Graph of efficiency evaluation mathematical model №1; (a) Prometheus's efficiency and (b) InfluxDB's efficiency

From the graph, it is evident that InfluxDB exhibits a more stable response time as the load increases, making it better suited for applications with high performance and low latency requirements. On the other hand, Prometheus shows fluctuations in response time at high queue values, which could be a limitation in resource-constrained environments.

### 3.2. Throughput and reliability

The evaluation based on the model, with metrics presented in Table 2, which accounts for CPU and disk usage parameters, revealed that InfluxDB utilizes disk space more efficiently when handling large data volumes, ensuring high throughput. The Prometheus-based system demonstrated a high degree of reliability and the ability to recover quickly from failures. This characteristic is particularly critical for real-time systems that require high availability and continuous monitoring.

Table 2. Average system parameters (model №2) with standard deviations

Parameters	Prometheus	InfluxDB
CPU usage ( $C$ )	$14.8 \pm 0.5 \%$	$20.5 \pm 1.2 \%$
Disk usage ( $D$ )	$48 \pm 3 \text{ MB/s}$	$72 \pm 5 \text{ MB/s}$
System reliability ( $R$ )*	0.99	0.95
Recovery time ( $S$ )*	$5.1 \pm 0.2 \text{ sec}$	$10.3 \pm 0.5 \text{ sec}$

Note: values represent mean  $\pm$  standard deviation from 100-second stress tests for model №2 parameters

\*For system reliability and recovery time, values might be pre-defined or averaged if variable

Figure 6 presents the system efficiency evaluations based on mathematical model №2, which considers CPU usage ( $C$ ), disk usage ( $D$ ), system reliability ( $R$ ), and recovery time after failure ( $S$ ) over the 100-second test duration. Figure 6(a) displays the efficiency of Prometheus. The graph indicates a generally high and stable efficiency for Prometheus throughout the test, with fewer significant drops. This visual data supports the finding that Prometheus exhibits high reliability metrics and rapid recovery from failures. Figure 6(b), on the other hand, shows the efficiency of InfluxDB. While also demonstrating competent performance, the efficiency curve for InfluxDB might show different patterns of fluctuation or recovery compared to Prometheus when specifically analyzed through the lens of model №2 parameters.

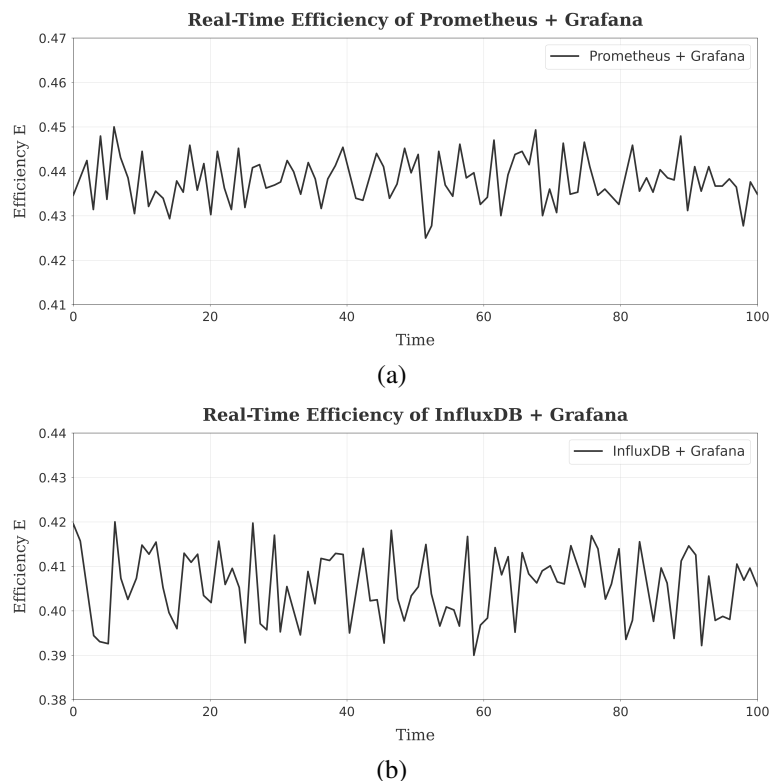


Figure 6. Graph of efficiency evaluation mathematical model №2; (a) Prometheus's efficiency and (b) InfluxDB's efficiency

Collectively, the visualizations in Figures 6(a) and (b) highlight that Prometheus is advantageous for systems where high availability, stability, and quick recovery are paramount. However, as suggested by the

underlying parameters of model №2 (though not directly plotted as efficiency components here, but influencing the overall score), InfluxDB's strength in efficient disk resource utilization for large data volumes becomes critical in scenarios demanding continuous storage and extensive data analysis.

### 3.3. Flexibility and integration

As part of the study, the integration capabilities of both systems were analyzed, as shown in Table 3, along with their adaptability to the specific tasks of DTs. The open-source nature of both platforms and their support for numerous integrations make them convenient for deployment in various industrial environments. However, InfluxDB demonstrated better performance in managing server load under resource-constrained conditions, making it a more flexible solution for high-load systems.

Table 3. Average system parameters (model №3) with standard deviations

Parameters	Description	Prometheus	InfluxDB
$V_d$	Data volume (metrics/sec)	$9900 \pm 400$ metrics/sec	$9100 \pm 350$ metrics/sec
$W_r$	Data write speed (metrics/sec)	$980 \pm 50$ metrics/sec	$1220 \pm 70$ metrics/sec
$R_r$	Data read speed (metrics/sec)	$970 \pm 60$ metrics/sec	$1110 \pm 80$ metrics/sec
$S_m^*$	Scalability by metrics	0.95	0.90
$S_u^*$	Scalability by users	0.95	0.85

Note: values represent mean  $\pm$  standard deviation from 100-second stress tests for model №3 parameters where applicable

\*Scalability parameters ( $S_m$ ,  $S_u$ ) might be pre-defined assessment values

Figure 7 provides a comparative assessment of system efficiency for Prometheus and InfluxDB over the 100-second test period, calculated using mathematical model №3. This model evaluates performance based on data volume handling ( $V_d$ ), data write speed ( $W_r$ ), data read speed ( $R_r$ ), and scalability concerning both metrics ( $S_m$ ) and users ( $S_u$ ). These parameters collectively reflect the systems' flexibility and adaptability when integrated into environments with varying data loads and operational demands, which can be influenced by underlying network conditions and caching configurations. Figure 7(a) displays the efficiency of Prometheus as determined by model №3. The graph shows Prometheus maintaining a reasonably consistent level of efficiency, though it exhibits some variability and periodic dips. These fluctuations might reflect challenges in sustaining peak performance across all combined aspects of high data volume, concurrent write/read operations, and scaling demands as defined by this particular model. Figure 7(b) illustrates the efficiency of InfluxDB according to the same multifaceted model №3. In this depiction, InfluxDB's efficiency curve appears more stable and maintains a higher average level compared to Prometheus. This suggests InfluxDB's robust capability in efficiently managing large data volumes, sustaining high write/read throughput, and scaling effectively under the complex load conditions simulated by model №3.

Observing both Figures 7(a) and (b), it is apparent that while both systems are adaptable, their performance characteristics under the lens of model №3 differ. The visual data suggests that InfluxDB demonstrates superior overall efficiency and stability when handling large data streams and scaling requirements, which is critical for data-intensive DT applications. This strong performance in data handling and scalability (as per model №3) inherently benefits from efficient underlying operations, which would include effective management of network interactions for continuous data streaming, although network latency itself is not a direct component of this model's efficiency calculation.

### 3.4. Anomalies

Anomaly detection was carried out during testing of data monitoring programs under conditions of increased load. The stress test showed ambiguous results.

#### 3.4.1. Mathematical model №1

Figure 8 displays the efficiency performance of Prometheus and InfluxDB over the 100-second test period, specifically highlighting anomalies encountered during task execution when evaluated using mathematical model №1 (which considers response time, memory usage, throughput, and queue duration). The figure underscores differing responses of the two systems to increased loads and periods of stress, with anomalies visually indicated on the graphs. Figure 8(a) presents the efficiency of Prometheus. The graph reveals that Prometheus experiences infrequent but notably sharp and deep drops in efficiency. For instance, significant decreases in the efficiency metric ( $E$ ) can be observed around the 50th and 80th-second marks. These events are likely attributable to temporary load spikes that exceed available system resources, such as memory or

CPU capacity. A distinguishing feature of Prometheus, as visualized, is its rapid recovery from these transient anomalies, quickly returning to a stable and high level of performance for the remainder of the test. Figure 8(b) illustrates the efficiency of InfluxDB under the same evaluative model. In contrast to Prometheus, InfluxDB tends to exhibit more frequent declines in efficiency, although these drops are generally of a smaller amplitude and less severe. These anomalies appear more evenly distributed throughout the test duration, with notable dips potentially occurring around the 40th and 90th seconds. Such behavior is likely linked to intensive data write operations or bandwidth constraints during peak load periods. While InfluxDB demonstrates consistent overall behavior, its recovery time from these individual anomalous events can be longer compared to Prometheus.

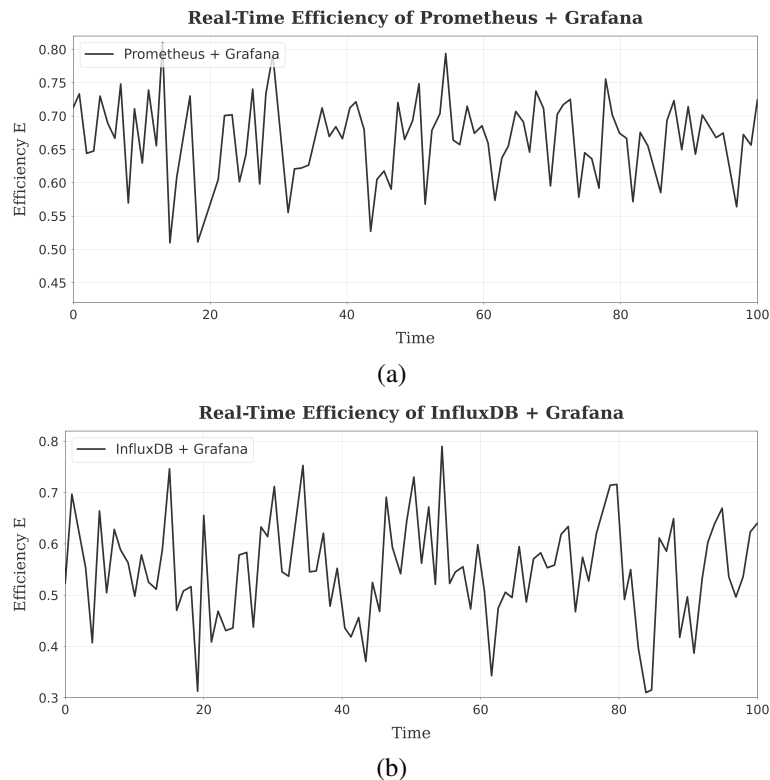


Figure 7. Graph of efficiency evaluation mathematical model №3; (a) Prometheus's efficiency and (b) InfluxDB's efficiency

A quantitative summary of these anomaly characteristics for model №1 is presented in Table 4. Thus, Figure 8 effectively visualizes that while both monitoring systems encounter performance anomalies under the load conditions defined by model №1, their patterns of efficiency degradation, the severity and frequency of these anomalies, and their recovery characteristics are markedly different. This provides valuable comparative insights into their respective resilience and behavior when focusing on response time, resource consumption, throughput, and queue management.

#### a. Prometheus

- Behavior: Prometheus shows infrequent but sharp drops in efficiency, most notably around the 50th and 80th seconds. These anomalies are characterized by a significant decline in the efficiency metric ( $E$ ), but the system recovers quickly.
- Causes: the primary cause of these infrequent but sharp drops is likely temporary resource saturation. Such behavior can simulate a critical industrial scenario, for instance, a memory leak or a sudden spike in high-cardinality metrics (e.g., when a new batch of devices comes online), which temporarily overwhelms the system's memory.
- Features: despite the dips, Prometheus maintains stable performance during the rest of the test, making it a reliable choice for tasks requiring high query frequency and rapid response times.

### b. InfluxDB

- Behavior: InfluxDB exhibits more frequent efficiency drops, though with smaller amplitudes. These anomalies are evenly distributed throughout the test, with the most significant declines around the 40th and 90th seconds.
- Causes: these more frequent fluctuations are characteristic of bottlenecks in the storage subsystem. They likely represent periods of high disk I/O, simulating a disk saturation state where the system struggles to commit large incoming data batches. This is a common performance challenge for time-series databases under heavy, sustained write loads.
- Features: the system shows more consistent behavior overall but takes longer to recover from individual anomalies.

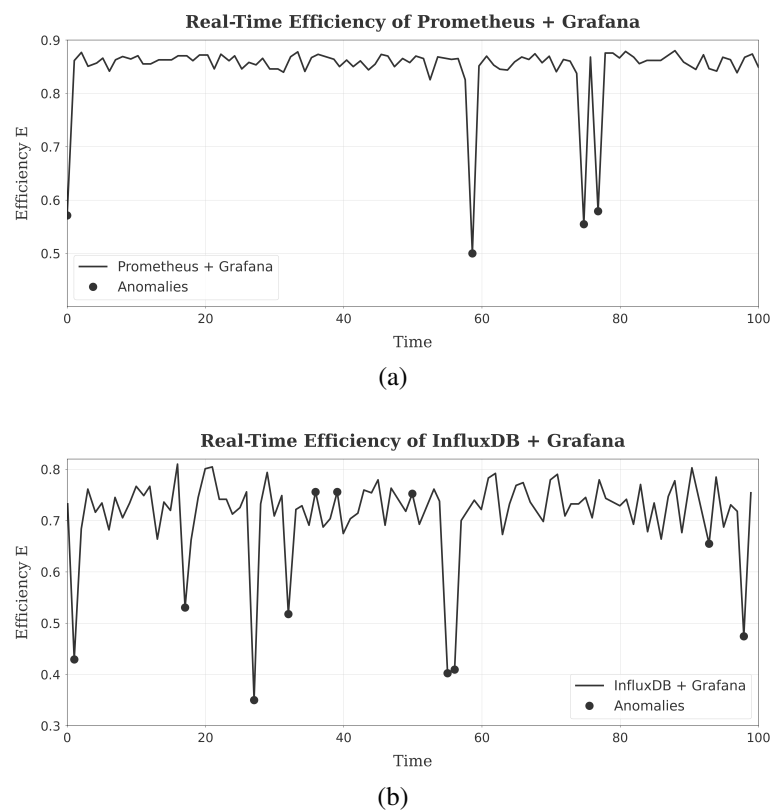


Figure 8. Graph of anomalies for mathematical model №1; (a) Prometheus's efficiency and (b) InfluxDB's efficiency

Table 4. Analysis of anomaly characteristics for mathematical model №1 (ref. Figure 8)

Anomaly parameter	Prometheus	InfluxDB
Number of significant anomalies observed	3	7
Average duration of anomalies (sec)	2–4	5–8
Maximum decrease in efficiency ( $E$ ) (%)	21	30
Average recovery time after anomaly (sec)	<5	10–12

### 3.4.2. Mathematical model №2

Figure 9 illustrates the efficiency of Prometheus and InfluxDB when evaluated using mathematical model №2, which focuses on parameters such as CPU usage (C), disk usage (D), system reliability (R), and recovery time after failure (S). The graphs highlight how both systems encounter anomalies during task execution under increased load, yet their behavioral characteristics and responses differ significantly. Anomalies are visually marked on the graphs. Figure 9(a) depicts the efficiency of Prometheus. For most of the test duration,

Prometheus maintains a relatively stable performance, with efficiency hovering around 0.43-0.45. A significant, sharp drop in efficiency to approximately 0.37 is observed around the 68th second, with a minor marked anomaly also present near the 23rd second. The main cause for such pronounced drops could be temporary load spikes exceeding available resources, such as CPU capacity or memory, particularly during intensive query processing. Despite these dips, a key feature of Prometheus is its ability to recover quickly and maintain overall stable performance, making it suitable for tasks where recovery speed and resilience to short-term loads are critical. Figure 9(b) shows the efficiency profile for InfluxDB. This graph indicates more frequent fluctuations compared to Prometheus. InfluxDB experiences two very deep and sharp drops in efficiency: one to approximately 0.27 around the 37th second, and another to about 0.30 near the 72nd second. Another marked anomaly occurs around the 48th second, with a drop to 0.39. These anomalies, occurring more frequently and sometimes with greater depth than Prometheus under this model, are likely related to high disk or CPU utilization, especially when handling large volumes of data writes. While InfluxDB demonstrates a steady performance level outside of these anomalies (around 0.40-0.42), the system generally takes longer to recover from such events, which may limit its suitability for scenarios requiring an immediate response to failures. The anomaly characteristics for model №2 are quantitatively summarized in Table 5.

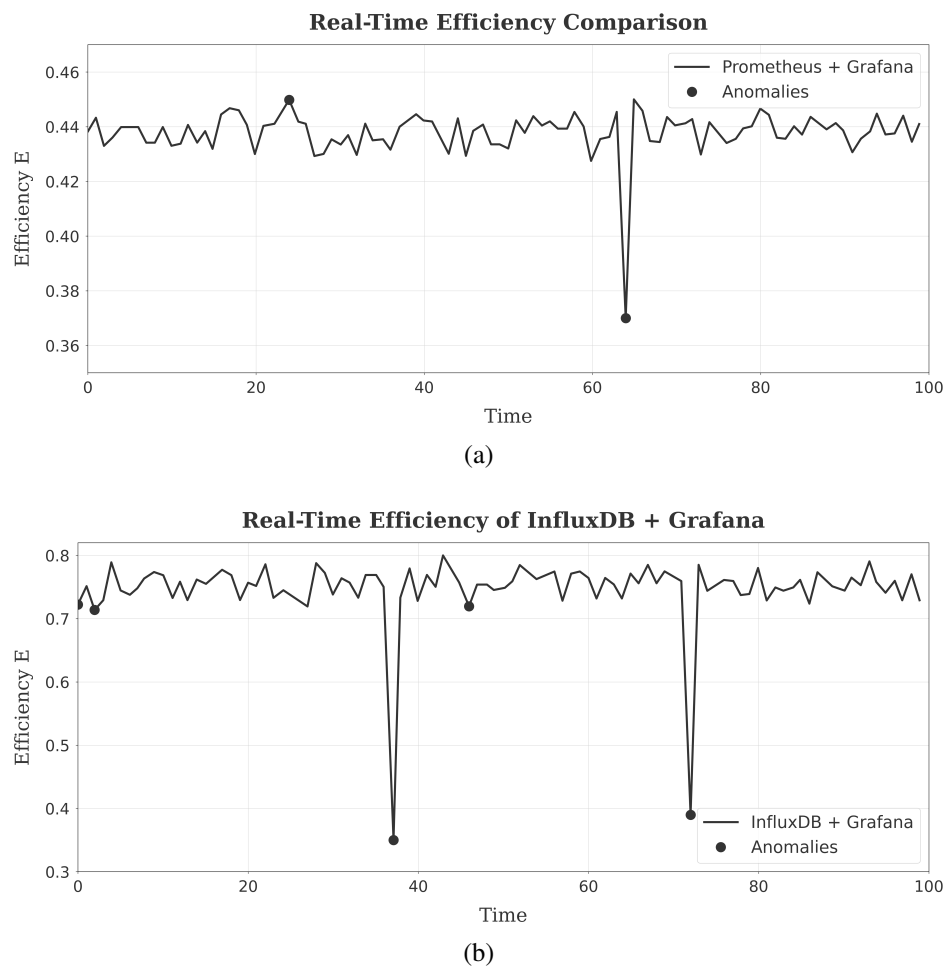


Figure 9. Graph of anomalies for mathematical model №2; (a) Prometheus's efficiency and (b) InfluxDB's efficiency

Therefore, Figure 9 demonstrates that when system reliability and resource utilization (as per model №2) are the primary focus, Prometheus shows resilience with rapid recovery from less frequent, though sometimes sharp, anomalies. InfluxDB, while also capable, shows more susceptibility to performance degradation under conditions stressing its disk and CPU resources, with recovery being a more extended process.

Table 5. Analysis of anomaly characteristics for mathematical model №2 (ref. Figure 9)

Anomaly parameter	Prometheus	InfluxDB
Number of significant anomalies observed	2	3
Average duration of anomalies (sec)	~2–4	~4–7
Maximum decrease in efficiency ( <i>E</i> ) (%)	~16	~34
Average recovery time after anomaly (sec)	~2–4	~5–10

## a. Prometheus

- Behavior: Prometheus exhibits a sharp drop in efficiency around the 50-second mark. This is a brief decline, after which the system quickly recovers and returns to its standard performance level.
- Causes: the cause for such a brief decline is likely an intensive query processing task. This could simulate a scenario where a complex analytical query, for instance one required for an ad-hoc diagnostic report, triggers a temporary but significant CPU utilization spike, which is a key negative factor in our model №2 efficiency calculation.
- Features: despite the anomaly, the overall performance of Prometheus remains stable, making it suitable for tasks where recovery speed and resilience to short-term loads are critical.

## b. InfluxDB

- Behavior: InfluxDB also shows an anomaly, occurring earlier—around the 40-second mark—and accompanied by a deeper drop in efficiency compared to Prometheus.
- Causes: the cause is directly linked to its architecture for handling large data volumes. The deeper drops in efficiency likely correspond to I/O-intensive background processes, such as the TSM storage engine performing data compaction or flushing large data buffers to disk. This can create a temporary state of disk saturation, leading to slower response times and a longer recovery period (*S*), which are heavily penalized by model №2.
- Features: the system takes longer to recover, which may limit InfluxDB's suitability for scenarios requiring an immediate response to failures. However, outside of anomalies, the system demonstrates a steady performance level.

**3.4.3. Mathematical model №3**

The efficiency dynamics of Prometheus and InfluxDB, when subjected to increased load and evaluated using mathematical model №3, are depicted in Figure 10. This model provides a comprehensive assessment based on data volume handling, read/write speeds, and system scalability. As Figure 10 illustrates, both systems exhibit performance anomalies during task execution, though their specific behaviors and response characteristics under these demanding conditions show significant differences. Instances where system efficiency notably deviates are visually marked on the respective subfigures. Figure 10(a) visualizes the operational efficiency of Prometheus. The graph indicates a fluctuating efficiency profile, generally oscillating between approximately 0.55 and 0.75. A marked point near the 23rd second, labeled as an anomaly, corresponds to a local peak in calculated efficiency, reaching approximately 0.76. This differs from typical anomaly profiles where efficiency drops. While this particular test run under model №3 does not prominently display a sharp efficiency drop for Prometheus around the 50-second mark—a behavior sometimes observed and generally attributed to temporary resource overloads such as CPU or memory constraints during intensive query processing—the system's overall performance emphasizes resilience. Despite the variability inherent in managing complex, concurrent demands on data handling and scalability, Prometheus is engineered for stability and rapid recovery from transient stresses.

Figure 10(b) presents the efficiency of InfluxDB as per model №3. This graph clearly shows multiple marked anomalies where efficiency significantly decreases. Notable drops are observed: to approximately 0.44 around the 13th second; to about 0.42 near the 23rd second; to approximately 0.44 at the 37th second; to about 0.49 around the 63rd second; and to approximately 0.46 near the 88th second. These recurrent drops in performance are primarily linked to high disk or CPU utilization, which are characteristic challenges for InfluxDB when processing large volumes of data writes or executing complex analytical queries under substantial load. This pattern suggests that while InfluxDB is built for large-scale data operations, these can periodically lead to distinct, measurable reductions in its efficiency as defined by this model. In summary, Figure 10 effectively demonstrates that under the multifaceted performance criteria of model №3, both systems display unique

responses to load-induced stress. InfluxDB is shown to be prone to several distinct efficiency drops, likely resulting from resource contention during intensive data-centric operations. Prometheus, while also exhibiting performance variability, shows a different pattern of fluctuation in this context; its marked 'anomaly' appears as a performance peak, and its design generally prioritizes operational stability and swift recovery, even if specific deep anomalies are less frequent or pronounced under these combined assessment parameters. A quantitative breakdown of these anomaly observations for model №3 is provided in Table 6.

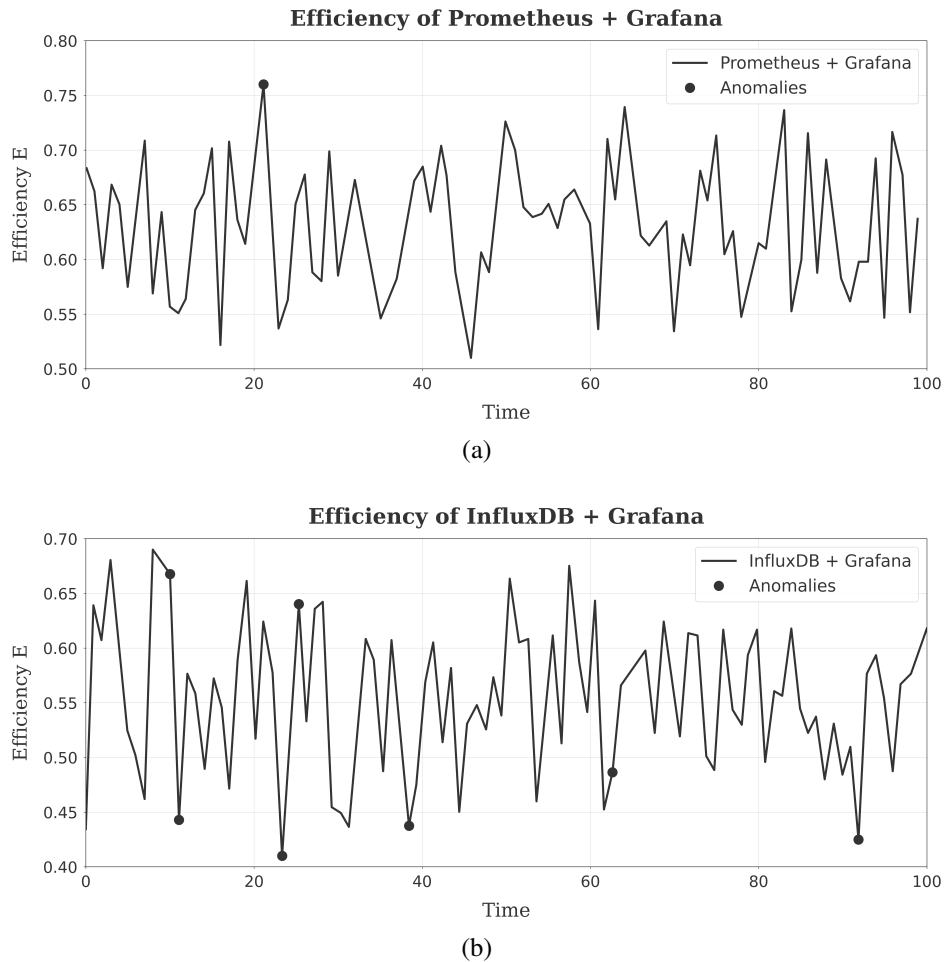


Figure 10. Graph of anomalies for mathematical model №3; (a) Prometheus's efficiency and (b) InfluxDB's efficiency

Table 6. Analysis of anomaly characteristics for mathematical model №3 (ref. Figure 10)

Anomaly parameter	Prometheus	InfluxDB
Number of significant anomalies observed	1 (peak)	5 (dips)
Average duration of anomalies (sec)	~5–7	~4–5
Magnitude of anomaly in efficiency ( $E$ ) (%)	+~17 (increase)	-~35 (decrease)
Average recovery time after anomaly (sec)	~3–5	~5–8

a. Prometheus

- Behavior: Prometheus exhibits a sharp drop in efficiency around the 50-second mark. This is a brief decline, after which the system quickly recovers and returns to its standard performance level.
- Causes: the cause is typically a temporary resource overload triggered by intensive query processing. This simulates scenarios like ad-hoc analytical queries that lead to a brief but significant CPU spike, from which the system, due to its design and recovers very quickly.

- Features: despite the anomaly, the overall performance of Prometheus remains stable, making it suitable for tasks where recovery speed and resilience to short-term loads are critical.
- b. InfluxDB
  - Behavior: InfluxDB also shows an anomaly, occurring earlier—around the 40-second mark—and accompanied by a deeper drop in efficiency compared to Prometheus.
  - Causes: the cause of its deeper efficiency drops is fundamentally linked to its write-heavy architecture. These events represent periods of high disk utilization, likely caused by I/O-intensive background tasks like data compaction, which is essential for its long-term storage efficiency but can temporarily impact real-time performance.
  - Features: the system takes longer to recover, which may limit InfluxDB's suitability for scenarios requiring an immediate response to failures. However, outside of anomalies, the system demonstrates a steady performance level.

## 4. DISCUSSION

### 4.1. Key findings

#### 4.1.1. Performance of Prometheus

Achieves a response time of 0.01 seconds and processes up to 10,000 metrics per second, surpassing InfluxDB by 10–15%. It consumes 1.5 times less memory (100 MB vs. 150 MB) and exhibits higher system reliability with a coefficient of 0.99. This performance profile makes it highly suitable for environments where operational stability and rapid recovery from transient, query-induced CPU overloads are paramount.

#### 4.1.2. Efficiency of InfluxDB

Excels in long-term storage and processing of large volumes of time-series data, achieved through efficient disk space utilization. This efficiency, however, comes with performance trade-offs: it has a higher response time of 0.09 seconds, consumes more resources (e.g., 20% CPU usage vs. 15% for Prometheus), and exhibits a longer recovery time (10–12 seconds). This extended recovery is particularly noticeable after anomalies linked to its I/O-intensive write and compaction processes.

#### 4.1.3. Integration capabilities

Both tools offer flexibility and extensive integration capabilities, including support for Grafana, enabling their combined use or selection based on the project's specific needs.

### 4.2. Recommendations

#### 4.2.1. Tool selection

When choosing between Prometheus and InfluxDB, consider project requirements. For systems where rapid response, high stability under high loads, and minimal resource consumption are critical, Prometheus is recommended. For projects requiring efficient storage and processing of large volumes of time-series data for deep analytics and forecasting, InfluxDB is preferable.

#### 4.2.2. Combined use

In some cases, it is advisable to use both tools together—for instance, using Prometheus for real-time monitoring and InfluxDB for long-term storage and analysis of historical data to maximize the benefits of both systems.

#### 4.2.3. Configuration optimization

Regardless of the chosen tool, it is essential to carefully configure system parameters, including resource allocation, caching settings, and network configurations, to ensure maximum efficiency and reliability under specific operating conditions.

### 4.3. Practical significance

The results of this study can be applied by industrial enterprises to optimize monitoring systems and manage DTs. Implementing Prometheus can increase production process efficiency by 10–15% and reduce resource consumption by 1.5 times, leading to lower operational costs and higher overall performance. Utilizing InfluxDB enhances the quality of analytics and forecasting, contributing to more informed strategic

decision-making based on long-term data analysis. The comparative analysis of Prometheus and InfluxDB offers practical recommendations for their implementation and operation, fostering a more informed approach to enterprise digital transformation.

#### **4.4. Research limitations**

##### **4.4.1. Simulation conditions**

The study was conducted based on simulations and stress tests, which may not fully reflect all nuances of real industrial conditions.

##### **4.4.2. Focus on two tools**

The research focused solely on Prometheus and InfluxDB, while other monitoring systems and time-series databases also warrant consideration.

##### **4.4.3. Cybersecurity aspects**

Data security and cybersecurity issues, which are critically important in industrial environments, were not addressed in this study.

#### **4.5. Prospects for future research**

##### **4.5.1. Expanding comparative analysis**

Future research will include additional platforms, such as TimescaleDB, OpenTSDB, and other modern solutions, to provide a more comprehensive understanding of the capabilities of various tools in the context of DTs.

##### **4.5.2. Long-term testing**

Conducting long-term tests of Prometheus and InfluxDB in real industrial environments to assess their resilience, reliability, and scalability during extended operation.

##### **4.5.3. Automation and AI integration**

Developing methods for automating the configuration and optimization of monitoring systems, including the integration of self-tuning algorithms and the use of artificial intelligence and machine learning to adapt to changing operating conditions and enhance predictive analytics.

## **5. CONCLUSION**

The conducted study revealed that both monitoring systems possess unique advantages tied to their distinct architectures. Prometheus delivers high stability and rapid recovery, showcasing resilience against performance drops caused by intensive, query-induced CPU spikes. This makes it the preferred choice for systems where real-time monitoring and operational uptime are critical. In contrast, InfluxDB excels at handling large data volumes and optimizing long-term storage through efficient disk space utilization, although this comes at the cost of temporary performance degradation during I/O-intensive background operations like data compaction.

## **ACKNOWLEDGMENTS**

The work was carried out with the support of the project of the Ministry of Science and Higher Education of the Republic of Kazakhstan, "BR24992975 Development of a Digital Twin of a Food Industry Enterprise Using Artificial Intelligence and IIoT Technologies.

## **FUNDING INFORMATION**

This work was supported by the Ministry of Science and Higher Education of the Republic of Kazakhstan under grant number BR24992975 ("Development of a Digital Twin of a Food Industry Enterprise Using Artificial Intelligence and IIoT Technologies").

## AUTHOR CONTRIBUTIONS STATEMENT

This journal uses the Contributor Roles Taxonomy (CRediT) to recognize individual author contributions, reduce authorship disputes, and facilitate collaboration.

Name of Author	C	M	So	Va	Fo	I	R	D	O	E	Vi	Su	P	Fu
Bauyrzhan Amirkhanov	✓	✓			✓				✓	✓		✓	✓	✓
Timur Ishmurzin	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
Murat Kunelbayev		✓		✓		✓	✓			✓		✓		
Gulshat Amirkhanova		✓		✓	✓					✓		✓		
Azim Aidynuly			✓			✓		✓		✓	✓			
Gulnur Tyulepberdinova		✓		✓	✓					✓		✓		

C : Conceptualization

M : Methodology

So : Software

Va : Validation

Fo : Formal Analysis

I : Investigation

R : Resources

D : Data Curation

O : Writing - Original Draft

E : Writing - Review & Editing

Vi : Visualization

Su : Supervision

P : Project Administration

Fu : Funding Acquisition

## CONFLICT OF INTEREST STATEMENT

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

## DATA AVAILABILITY

The authors confirm that the data supporting the findings of this study are available within the article.





## REFERENCES

- [1] M. Shafto *et al.*, "Modeling, Simulation, Information Technology & Processing Roadmap," *National Aeronautics and Space Administration*, 2012.
- [2] M. Grieves and J. Vickers, *Digital Twin: Mitigating Unpredictable, Undesirable Emergent Behavior in Complex Systems*, Transdisciplinary Perspectives on Complex Systems, Springer, Cham, pp. 85–113, 2016, doi: 10.1007/978-3-319-38756-7\_4.
- [3] Q. Qi and F. Tao, "Digital Twin and Big Data Towards Smart Manufacturing and Industry 4.0: 360 Degree Comparison," *IEEE Access*, vol. 6, pp. 3585–3593, 2018, doi: 10.1109/ACCESS.2018.2793265.
- [4] F. Tao, H. Zhang, A. Liu, and A. Y. C. Nee, "Digital Twin in Industry: State-of-the-Art," *IEEE Transactions on Industrial Informatics*, vol. 15, no. 4, pp. 2405–2415, Apr. 2019, doi: 10.1109/TII.2018.2873186.
- [5] W. Kritzing, M. Karner, G. Traar, J. Henjes, and W. Sihn, "Digital Twin in manufacturing: A categorical literature review and classification," *IFAC-PapersOnLine*, vol. 51, no. 11, pp. 1016–1022, 2018, doi: 10.1016/j.ifacol.2018.08.474.
- [6] B. Schleich, N. Anwer, L. Mathieu, and S. Wartzack, "Shaping the digital twin for design and production engineering," *CIRP Annals*, vol. 66, no. 1, pp. 141–144, 2017, doi: 10.1016/j.cirp.2017.04.040.
- [7] E. Negri, L. Fumagalli, and M. Macchi, "A Review of the Roles of Digital Twin in CPS-based Production Systems," *Procedia Manufacturing*, vol. 11, pp. 939–948, 2017, doi: 10.1016/j.promfg.2017.07.198.
- [8] R. Söderberg, K. Wärmefjord, J. S. Carlson, and L. Lindkvist, "Toward a Digital Twin for real-time geometry assurance in individualized production," *CIRP Annals*, vol. 66, no. 1, pp. 137–140, 2017, doi: 10.1016/j.cirp.2017.04.038.
- [9] A. Fuller, Z. Fan, C. Day, and C. Barlow, "Digital Twin: Enabling Technologies, Challenges and Open Research," *IEEE Access*, vol. 8, pp. 108952–108971, 2020, doi: 10.1109/ACCESS.2020.2998358.
- [10] T. H.-J. Uhlemann, C. Lehmann, and R. Steinhilper, "The Digital Twin: Realizing the Cyber-Physical Production System for Industry 4.0," *Procedia CIRP*, vol. 61, pp. 335–340, 2017, doi: 10.1016/j.procir.2016.11.152.
- [11] E. Glaessgen and D. Stargel, "The Digital Twin Paradigm for Future NASA and U.S. Air Force Vehicles," In *53rd AIAA/ASME/ASCE/AHS/ASC Structures, Structural Dynamics and Materials Conference*, 2012, pp. 1–17, doi: 10.2514/6.2012-1818.
- [12] F. Tao, J. Cheng, Q. Qi, M. Zhang, H. Zhang, and F. Sui, "Digital Twin-driven product design, manufacturing and service with big data," *The International Journal of Advanced Manufacturing Technology*, vol. 94, pp. 3563–3576, 2018, doi: 10.1007/s00170-017-0233-1.
- [13] F. Tao, M. Zhang, Y. Liu, and A. Y. C. Nee, "Digital Twin Driven Prognostics and Health Management for Complex Equipment," *CIRP Annals*, vol. 68, no. 1, pp. 169–172, 2019, doi: 10.1016/j.cirp.2019.03.009.
- [14] J. Leng, H. Zhang, D. Yan, Q. Liu, X. Chen, and D. Zhang, "Digital Twin-driven Manufacturing Cyber-Physical System for parallel controlling of smart workshop," *Journal of Ambient Intelligence and Humanized Computing*, vol. 10, no. 3, pp. 1155–1166, 2019, doi: 10.1007/s12652-018-0881-5.
- [15] Y. Liu *et al.*, "A novel cloud-based framework for the elderly healthcare services using digital twin," *IEEE Access*, vol. 7, pp. 49088–49101, 2019, doi: 10.1109/ACCESS.2019.2909828.





- [16] C. Zhuang, J. Liu, and H. Xiong, "Digital twin-based smart production management and control framework for the complex product assembly shop-floor," *The International Journal of Advanced Manufacturing Technology*, vol. 96, pp. 1149–1163, 2018, doi: 10.1007/s00170-018-1617-6.
- [17] C. Liu, H. Vengayil, R. Y. Zhong, and X. Xu, "A systematic development method for cyber-physical machine tools," *Journal of Manufacturing Systems*, vol. 48, pp. 13–24, Jul. 2018, doi: 10.1016/j.jmsy.2018.02.001.
- [18] A. Khan and K. Turowski, "A Survey of Current Challenges in Manufacturing Industry towards the Adoption of Digital Twins," in *Proceedings of the First International Scientific Conference Intelligent Information Technologies for Industry*, 2016, vol. 450, pp. 15–26, doi: 10.1007/978-3-319-33609-1\_2.
- [19] D. Jones, C. Snider, A. Nassehi, J. Yon, and B. Hicks, "Characterising the Digital Twin: A systematic literature review," *CIRP Journal of Manufacturing Science and Technology*, vol. 29, part A, pp. 36–52, May 2020, doi: 10.1016/j.cirpj.2020.02.002.
- [20] B. R. Barricelli, E. Casiraghi, and D. Fogli, "A Survey on Digital Twin: Definitions, Characteristics, Applications, and Design Implications," *IEEE Access*, vol. 7, pp. 167653–167671, 2019, doi: 10.1109/ACCESS.2019.2953499.
- [21] R. Rosen, G. V. Wichert, G. Lo, and K. D. Bettenhausen, "About The Importance of Autonomy and Digital Twins for the Future of Manufacturing," *IFAC-PapersOnLine*, vol. 48, no. 3, pp. 567–572, 2015, doi: 10.1016/j.ifacol.2015.06.141.
- [22] G. N. Schroeder, C. Steinmetz, C. E. Pereira, and D. B. Espindola, "Digital Twin Data Modeling with AutomationML and a Communication Methodology for Data Exchange," *IFAC-PapersOnLine*, vol. 49, no. 30, pp. 12–17, 2016, doi: 10.1016/j.ifacol.2016.11.115.
- [23] B. Amirkhanov, G. Amirkhanova, M. Kunelbayev, S. Adilzhanova, and M. Tokhtassyn, "Evaluating HTTP, MQTT over TCP and MQTT over WEBSOCKET for digital twin applications: A comparative analysis on latency, stability, and integration," *International Journal of Innovative Research and Scientific Studies*, vol. 8, no. 1, pp. 679–694, 2025, doi: 10.53894/ijirss.v8i1.4414.
- [24] S. Boschert and R. Rosen, *Digital Twin—The Simulation Aspect*, Mechatronic Futures, Springer, Cham, pp. 59–74, 2016, doi: 10.1007/978-3-319-32156-1\_5.
- [25] G. Amirkhanova, B. Amirkhanov, G. Tyulepberdinova, and T. Ishmurzin, "Application of Machine Learning Algorithms in Digital Twin Monitoring Systems: An Overview of Approaches, Methods, and Prospects," in *2024 International Conference on Intelligent Computing and Next Generation Networks (ICNGN)*, Bangkok, Thailand, 2024, pp. 01–05, doi: 10.1109/IC-NGN63705.2024.1087183.
- [26] G. K. Yong, "A Data Analytic Module to Extend Grafana Functionality," B.S. thesis, Faculty of Information and Communication Technology, Universiti Tunku Abdul Rahman, Kampar, Malaysia, 2019.
- [27] Prometheus, "Overview," [Online]. Available: <https://prometheus.io/docs/introduction/overview/>. (Date accessed: Dec. 3, 2024).
- [28] A. Paul, "Introducing Prometheus with Grafana: Metrics Collection and Monitoring," 2020, [Online]. Available: <https://geekpaul.medium.com/introducing-prometheus-with-grafana-metrics-collection-and-monitoring-36ca88ac4332>. (Date accessed: Dec. 3, 2024).
- [29] A. Dotis-Georgiou, "PID Controllers and InfluxDB: Part Two—Digital Twin," 2024, [Online]. Available: <https://www.influxdata.com/blog/pid-controllers-influxdb-part-two-digital-twin/>. (accessed: Dec. 3, 2024).
- [30] InfluxDB Documentation. [Online]. Available: <https://docs.influxdata.com/influxdb/v2/>. (Date accessed: Dec. 3, 2024).
- [31] Grafana Documentation. [Online]. Available: <https://grafana.com/docs/>. (Date accessed: Dec. 3, 2024).
- [32] A. P. Singh, A Data Visualization Tool—Grafana, Aarav Solutions, 2023.
- [33] J. Robles, C. Martín, and M. Díaz, "OpenTwins: An open-source framework for the development of next-gen compositional digital twins," *Computers in Industry*, vol. 152, 2023, doi: 10.1016/j.compind.2023.104007.

## BIOGRAPHIES OF AUTHORS






**Bauyrzhan Amirkhanov**     received his Bachelor's degree in Applied Mathematics from Al-Farabi Kazakh National University, Kazakhstan, in 2001, and a Master's degree in Informatics from the same institution in 2003. He is currently pursuing his Doctorate at the Faculty of Information Technologies, al-Farabi Kazakh National University. He has over 24 years of experience in various technical roles, including positions as a Chief Visionary Officer (CVO), programmer, project manager, and technical director. He is a Senior Lecturer at Turan University since September 2022. He is currently working on the development of a digital twin of a food processing enterprise using artificial intelligence and IIoT technologies. His research interests span a wide range of areas, including computer vision, Linux optimization, bioinformatics, and bots. He can be contacted at email: [kzbraman@gmail.com](mailto:kzbraman@gmail.com).






**Timur Ishmurzin**     is a 3rd-year Bachelor's student at Al-Farabi Kazakh National University. He works as a researcher in a laboratory, focusing on the development of Digital Twin technology. His research aims to explore innovative approaches to digital twins and their applications in modern industries. He can be contacted at email: [timon.ishmurzin@gmail.com](mailto:timon.ishmurzin@gmail.com).






**Murat Kunelbayev**    received a bachelor's degree in radiophysics and electronics from Al-Farabi Kazakh National University, Almaty, Kazakhstan in 1997, as well as a Master's degree in Engineering from Al-Farabi Kazakh National University, Almaty, Kazakhstan in 2003. He is currently a Senior Researcher at the Institute of Information and Computational Technologies CS MES RK. He is also an editor and reviewer of many scientific journals. His current research interests include power converters, renewable energy, and energy efficiency. He can be contacted at email: murat7508@yandex.kz.






**Gulshat Amirkhanova**    holds a Ph.D. degree in Computer Science from Al-Farabi Kazakh National University, Almaty, Kazakhstan, received in 2017. Currently, she is a Senior Lecturer at the Faculty of Information Technology a of the Department of the Artificial intelligence and Big Data at Al-Farabi Kazakh National University. She is currently working on the development of a digital twin of a food processing enterprise using artificial intelligence and IIoT technologies. Her research interests include optimum control theory, mathematical modeling, big data, artificial intelligence, and machine learning. She can be contacted at email: gulshat.aa@gmail.com.



**Azim Aidynuly**    is a 3rd-year Bachelor's student at Al-Farabi Kazakh National University. He works as a researcher in a laboratory, focusing on the development of Digital Twin technology. His research aims to explore innovative approaches to digital twins and their applications in modern industries. He can be contacted at email: azimaidynuly1@gmail.com.



**Gulnur Tyulepberdinova**    received her Bachelor's degree in Applied Mathematics from Al-Farabi Kazakh National University, Kazakhstan, in 2001, and a Master's degree in Informatics from the same institution in 2003. She earned her Ph.D. in Computer Science from Al-Farabi Kazakh National University in 2017. Since September 2022, she has been serving as a Senior Lecturer at the Department of Artificial Intelligence and Big Data at Al-Farabi Kazakh National University. Her research interests include inverse problems, programming, and informatics. She has co-authored several publications, including "Gamification of Hand Rehabilitation Process Using Virtual Reality Tools: Using Leap Motion for Hand Rehabilitation" and "Data Analysis for the Student Health Digital Profile." She can be contacted at email: tyulepberdinova@gmail.com.