Review of modern tools for edge computing systems quality assurance

Viktor O. Maliarskyi[1], Vasyl P. Oleksiuk[1,2]

[1]*Ternopil Volodymyr Hnatiuk National Pedagogical University, 2 Maxyma Kryvonosa Str., Ternopil, 46027, Ukraine*
[2]*Institute for Digitalisation of Education of the NAES of Ukraine, 9 M. Berlynskoho Str., Kyiv, 04060, Ukraine*

Abstract

Reviewing relevant QA tools is essential for improving efficiency, collaboration, and software quality while reducing costs. It enables QA teams to meet the fast-changing demands of software development and deliver reliable software on time and within budget. Strategic tool selection, emphasizing standardization, integration, and alignment with business goals, streamlines processes and reduces complexity. Edge computing environments require a fundamentally different approach to testing compared to traditional cloud or on-premise applications. While traditional testing tools are valuable, edge computing requires specialized solutions that can handle distributed architectures and variable network conditions. This paper aims to analyze challenges and key approaches of Edge Computing systems' testing, evaluate the most suitable QA tools, organize them, and highlight their benefits to help QA teams identify tools that meet their requirements. Bibliometric analysis of researched field discovered that the topic of quality assurance of different types of edge computing systems is highly discussed and important nowadays. The main features of providing efficient quality assurance process for edge computing systems is to set up appropriate testing of their performance, reliability, security, integrity and usability. The authors have defined the criteria for selecting edge computing system testing tools, such as scalability, compatibility, integrity, cost savings, licensing, and ease of use. For each of them, several indicators are proposed that allow measuring the value of these indicators. The most relevant tools that match defined criteria are suggested.

Keywords

Edge computing, quality assurance, tools, challenges, approaches, criteria and indicators

1. Introduction

Modern software applications are more complex than ever, involving multiple platforms, devices, APIs, and integrations [1]. Modern businesses demand a greater transparency and accountability of their products. Standardized QA tools provide consistent metrics and insights into testing progress and quality. Also they enable traceability of bugs and test cases, which is crucial for audits and compliance.

Edge computing can be defined as an emerging technology that uses cloud computing to leverage edge data centers to process, store, and analyze data close to the source. Traditional cloud computing architectures are not designed for latency-critical applications such as AI (artificial intelligence) and IoT (Internet of things) because they rely on low data volumes generated by applications running near highly-populated areas [2].

Performing functional and integration testing, along with scalability and robustness assessments on the codebases is generally limited to configurations involving a single or a small number of physical servers. In contrast, edge architectures necessitate rigorous evaluation of functionalities designed to address the challenges posed by geographically distributed infrastructures, particularly when architectural models exhibit fundamentally distinct configurations. To ensure stable and reliable results, it is advisable to adopt best practices from the scientific community.

Testing, should not only be rooted in precision engineering, but also involve creative problem-solving. Low-level code testing, such as unit tests or API response validation, is relatively straightforward and facilitates the creation of frameworks capable of executing automated test suites that meet criteria such

doors-2025: 5th Edge Computing Workshop, April 4, 2025, Zhytomyr, Ukraine
maliarskyi_vo@fizmat.tnpu.edu.ua (V. O. Maliarskyi); oleksyuk@fizmat.tnpu.edu.ua (V. P. Oleksiuk)
https://tnpu.edu.ua/faculty/fizmat/oleksyuk-vasil-petrovich.php (V. P. Oleksiuk)
0009-0006-7289-7672 (V. O. Maliarskyi); 0000-0003-2206-8447 (V. P. Oleksiuk)

as repeatability, replicability, and reproducibility. However, testing integrated systems to replicate the configurations and operational conditions of production environments presents considerably greater complexity [3].

Reviewing and continuous monitoring of modern QA tools is crucial for efficiency, collaboration, cost savings, and quality improvement. It helps QA teams adapt to the fast-paced, dynamic demands of software development while maintaining high standards of software reliability and user satisfaction. With proper systematization, teams are better equipped to deliver exceptional software on time and within budget [4]. To address this, QA teams must take a strategic approach to tool selection, focusing on standardization, integration, and alignment with business goals. By managing this variety effectively, organizations can streamline their QA processes and maintain high-quality software delivery without unnecessary complexity. Modern QA increasingly relies on advanced features like AI-driven test generation, defect prediction, and analytics. Systematized tools help QA teams stay up-to-date and integrate innovative technologies without disrupting workflows [5].

The purpose of this article is to analyze and estimate the most suitable QA tools for edge computing systems in 2024, review them and highlight their benefits to help QA teams to find tools that meet their requirements.

2. Bibliometric analysis of research field

To explore the development of research in edge computing system's quality assurance field and to define the key concepts of the research by studying the keywords of related scientific papers and articles, the bibliometric analyses was performed using data from two sources: Scopus and Web of Science databases. As an example of bibliometric analysis workflow, the methodologies of Mintii and Semerikov [6], Vinueza-Naranjo et al. [7] were used. In accordance with these methodologies, some search queries were performed.

Scopus search query from January 6, 2025:

```
TITLE-ABS-KEY ( software ) AND TITLE-ABS-KEY ( "Edge Computing" )
AND TITLE-ABS-KEY ( testing OR "quality assurance" ) AND ( tool* )
AND PUBYEAR > 2014 AND PUBYEAR < 2026 AND ( LIMIT-TO ( SUBJAREA , "COMP" )
OR LIMIT-TO ( SUBJAREA , "ENGI" ) OR LIMIT-TO ( SUBJAREA , "MATH" )
OR LIMIT-TO ( SUBJAREA , "DECI" ) OR LIMIT-TO ( SUBJAREA , "ECON" )
OR LIMIT-TO ( SUBJAREA , "MULT" ) ) AND ( LIMIT-TO ( DOCTYPE , "ar" )
OR LIMIT-TO ( DOCTYPE , "cp" ) ) AND ( LIMIT-TO ( LANGUAGE , "English" ) )
```

Web of Science search query from January 6, 2025:

```
AK="Edge Computing" AND AK=(test* OR "quality assurance") AND AK=software
```

The Web of Science query line is much shorter than it is for Scopus, because the functionality of Web of Science platform's user interface is not automatically adding used checkbox filters to its searching query. But all the searching conditions was the same for both sources. After uniting duplicate results, the final documents list contained 70 files. VOSviewer [8] was used as a tool for bibliometric analysis. Analysis was performed on keywords using default algorithm of clustering. The result of bibliometric analysis is a map of 21 keywords divided into 4 clusters that is displayed on figure 1.

Analyzing the researched field's keywords grouped by VOSviewer and identifying associated thematic areas, we suggest to name formed clusters next way:

- Cluster 1 (red) – Edge computing system types and their performance.
- Cluster 2 (green) – Usage of machine learning in edge computing systems testing.
- Cluster 3 (Blue) – Open source systems.
- Cluster 4 (Yellow) – Network architecture.

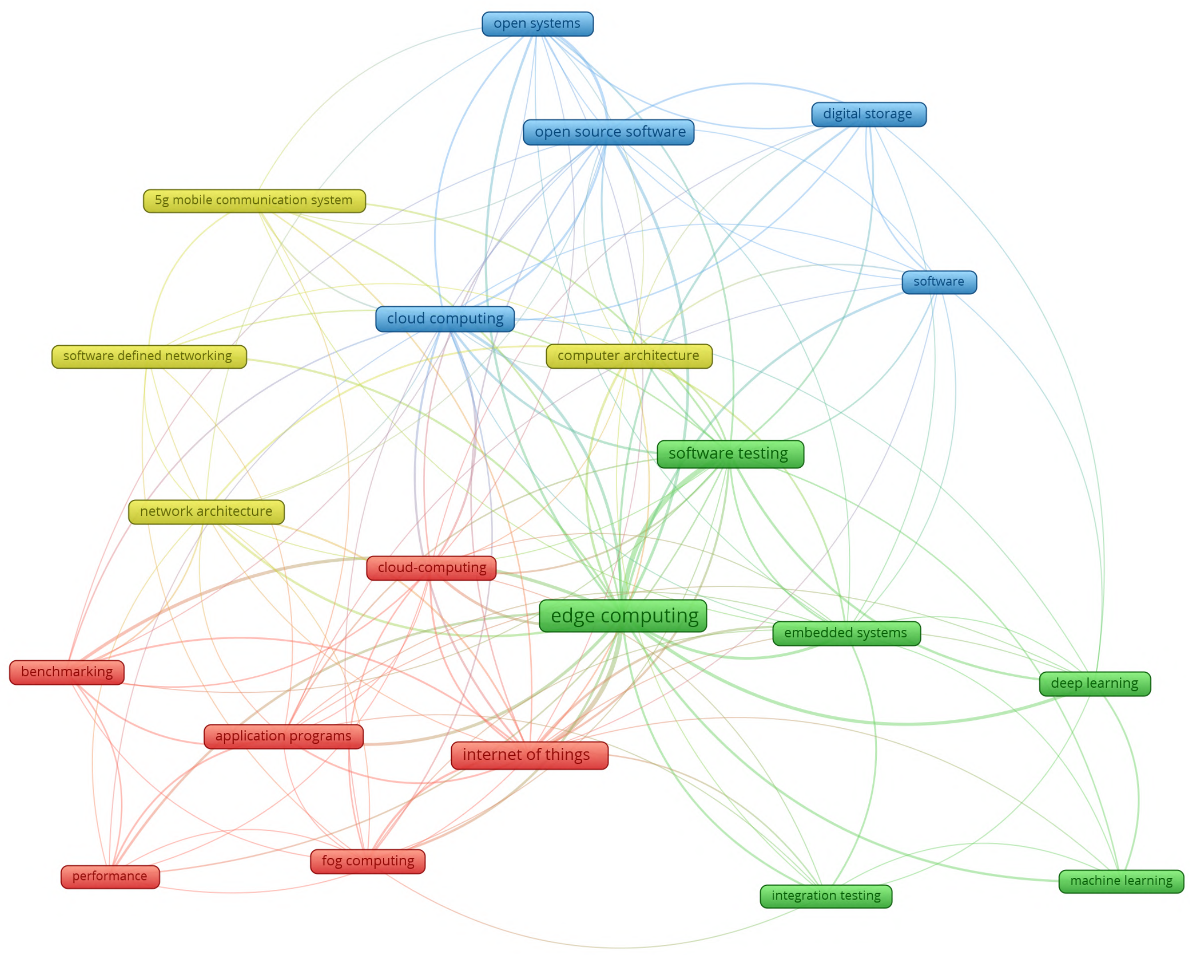

Figure 1: Network visualization of keywords relations in edge computing QA field.

To analyze generated map, we can use 4 main attributes:

- weight<Links> (w_l) – the number of links of an element with other elements
- weight<Total link strength> (w_{tls}) – general strength of links between target element and other elements
- weight<Occurrences> (w_o) – related to keywords it shows the number of their occurrences in analyzed documents
- score<Avg. pub. year> (s_{apy}) – average year of documents' publication where target keyword is used.

All the keywords and their analysis attributes values are visualized in table 1.

Analysis data discovers that keywords *Edge Computing* and *Software Testing* from cluster 2 have the highest numbers of all attributes between all the other keywords (edge computing: w_l – 20, w_{tls} – 143, w_o – 55; software testing: w_l – 19, w_{tls} – 70, w_o – 27). It means that the topic of quality assurance of different types of edge computing systems is highly discussed and important. Moreover it keeps changing, developing and always requires new investigations due to average year of documents' publication attribute values (edge computing: s_{apy} – 2021,95; software testing: s_{apy} – 2021.89).

The Internet of Things (IoT) is one of the largest and most influential network architectures. It describes access and processing of the data given by devices with sensors via wired and wireless networks. IoT requires scalable and flexible management of many interconnected devices and sensors today [9]. Due to the analysis result (w_l – 19, w_{tls} – 61), it can be caused by the fact that its errors can have direct and significant consequences on human lives. Addressing these errors requires rigorous

Table 1
Attributes of bibliometric analysis of keywords.

Keyword	Cluster	w_l	w_{tls}	w_o	s_{apy}
application programs	1	14	28	8	2022,50
benchmarking	1	11	27	9	2022,89
cloud-computing	1	14	29	7	2023,43
fog computing	1	13	22	5	2022,00
internet of things	1	19	61	18	2021,89
performance	1	9	17	5	2023,40
deep learning	2	10	24	10	2021,50
edge computing	2	20	143	55	2021,95
embedded systems	2	16	32	9	2022,56
integration testing	2	9	15	5	2022,20
machine learning	2	6	13	6	2021,50
software testing	2	19	70	27	2021,89
cloud computing	3	16	41	10	2021,10
digital storage	3	10	20	7	2022,14
open source software	3	14	35	9	2021,78
open systems	3	13	28	7	2021,29
software	3	11	18	6	2021,17
5g mobile comunication	4	9	16	7	2020,43
computer architecture	4	13	25	7	2021,71
network architecture	4	12	21	5	2022,20
software defined networking	4	8	13	5	2020,80

testing and validation processes; however, testing and validating a heterogeneous, dynamically evolving, and planetary-scale ecosystem presents unique challenges [10]. Considering some challenges in testing IoT architectures, such as the variety and large numbers of devices, dynamic environments, security challenges, and real-time performance, the authors [11] designed the framework for modelling the behaviour of the Internet of Things and edge computing environments. Its efficacy was validated using a case study for an electricity management and billing application within a smart city [12].

Also due to Occurrences and Avg. Pub. Year attributes' values we can note that the most relevant and popular keywords of investigated field are cloud-computing systems (w_o – 10, s_{apy} – 2023,43) and general edge computing system's performance testing (s_{apy} – 2023,40). It shows that the discussion of cloud computing and edge computing systems' synergy is developing and becoming more attractive for developers and scientists in many fields such as distributed computing [13], software engineering [14], computer networks [15], education [16, 17], etc. At the same time the problem of these systems' performance is rising and its solution will allow to develop more efficient and useful projects.

The bibliometric analysis shows the rise of attention to such studies as the use of AI for deep learning of edge devices [18], solving classification and segmentation tasks [19], reliability assessment of AI-systems, vehicle control [20], video monitoring [21], etc. Another direction of current research is the integration of Machine learning into the testing process of edge computing systems. This approach extends and speeds up the analyzes of edge computing capabilities, enabling real-time data processing and more intelligent decision-making.

3. Review of modern edge computing system's testing challenges and features

Edge computing environments become more popular and require a fundamentally different approach to testing compared to traditional cloud or on-premise applications. As emphasized in the ISTQB Advanced Level Test Automation Engineer syllabus [22], software designed for edge computing must be capable of effectively managing challenges such as intermittent connectivity, resource-constrained devices, and highly variable network conditions. These unique characteristics necessitate tailored testing strategies to ensure reliability, performance, and resilience in edge deployments. Basic test strategies must be expanded and account the requirements of edge computing environment (figure 2).

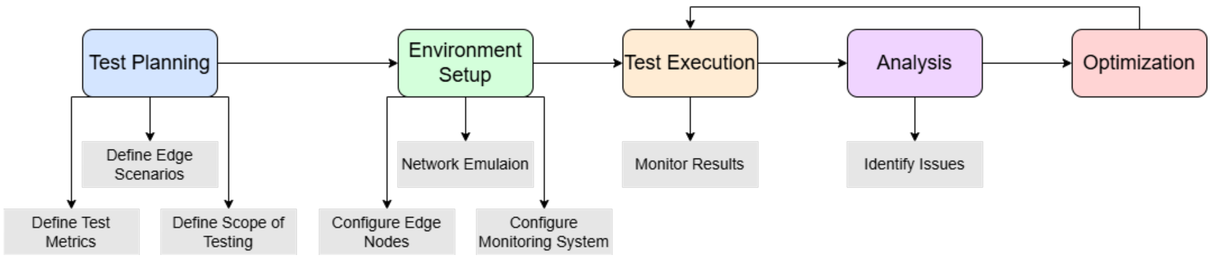

Figure 2: Edge computing's testing strategy [23].

Edge computing applications, such as autonomous vehicles and smart manufacturing systems, demand instantaneous data processing and response, leaving minimal tolerance for delay or error. Latency in data handling can lead to critical safety risks or significant operational inefficiencies. Data testing should ensure that systems meet the stringent time-sensitive requirements of edge environments by validating that data is processed and acted upon within milliseconds. For instance, in an autonomous vehicle, real-time testing verifies that the vehicle's sensors and AI algorithms accurately detect obstacles and respond promptly, thereby mitigating the risk of accidents and ensuring safe operation.

System reliability is a critical consideration in edge computing, where edge devices function autonomously, often without direct dependence on centralized data centers. Testing should identify potential system failures before they escalate into critical issues. By continuously monitoring the performance of edge devices, testing must detect early indicators of hardware malfunctions, software anomalies, or network disruptions.

Security is a paramount concern in edge computing environments, where data is processed outside the controlled and secure boundaries of traditional data centers. Data testing must enhance security by continuous monitoring of data flows for vulnerabilities or breaches. For instance, real-time testing can detect anomalous data patterns indicative of cyberattacks or unauthorized access to edge devices [24]. By integrating real-time testing with robust security protocols, organizations can rapidly identify and respond to potential threats, thereby protecting sensitive data and ensuring adherence to industry compliance standards.

In edge computing, the seamless coordination and performance of multiple devices operating across a distributed network are essential for ensuring smooth and efficient operations. Data testing plays a crucial role in verifying that each component within the edge network functions optimally. This involves validating the efficiency of inter-device communication, evaluating the speed and accuracy of data processing, and identifying potential bottlenecks that could impair overall performance. By implementing continuous testing, organizations can enhance the performance of their edge systems, achieving greater speed, reliability, and operational effectiveness [25].

Edge computing applications often operate in dynamic and challenging environments, such as offshore oil rigs, remote industrial sites, or aerospace systems. These settings pose unique difficulties, including variable network connectivity, extreme environmental conditions, and restricted access to centralized resources. Testing must address these challenges by validating the adaptability and resilience of edge systems. We should ensure that edge devices maintain functionality and reliability under unstable network conditions or harsh environmental factors, thereby supporting continuous and dependable operations in critical applications.

Edge computing relies on uninterrupted data flow and processing to deliver the low-latency and high-performance advantages it promises. Continuous monitoring and verifying that data flows seamlessly from edge devices to local processing units is essential. This ongoing evaluation ensures that any interruptions, delays, or inconsistencies in data transmission are promptly detected and resolved, enabling edge systems to sustain a consistent and reliable flow of real-time data for optimal performance.

In edge computing, where data processing occurs closer to its source, real-time data testing plays a pivotal role in ensuring systems operate with precision, efficiency, and dependability. Edge environments frequently manage time-critical data, where delays, inaccuracies, or failures can lead to severe consequences, particularly in domains such as autonomous vehicles, industrial IoT, and healthcare monitoring. By continuously validating the integrity, performance, and functionality of edge devices and systems under live conditions, real-time data testing mitigates these risks, enabling reliable and responsive operations in dynamic and high-stakes scenarios [26].

The primary objective of real-time data testing is to guarantee the accuracy and reliability of data processed at the edge. In decentralized systems, edge devices—such as sensors, cameras, and smart gateways—generate and collect substantial volumes of data. Real-time testing ensures that this data is correctly formatted, transmitted, and synchronized across distributed networks. By detecting inconsistencies or corruptions in the data flow, real-time testing prevents potential inaccuracies that could otherwise result in erroneous decisions or malfunctioning operations in edge-based applications.

Burn-in testing is a rigorous process aimed at identifying early failures in components and minimizing the likelihood of defects and malfunctions during field operation. This testing subjects computing components to extreme operating conditions, such as high and low temperature extremes, intensive usage cycles, and elevated voltages. The goal is to identify and eliminate defective components or those with inherently short lifespans before their deployment in operational systems. The primary purpose of burn-in testing is to ensure that components can endure severe environmental and operational stresses, demonstrating their durability and dependability under diverse and challenging conditions. By doing so, burn-in testing helps prevent costly or potentially catastrophic system failures, thereby enhancing the reliability and safety of mission-critical applications [27].

4. Tools for edge computing systems‘ testing review

Manual and automation testing leverages a wide array of tools designed to help QA engineers create, organize, and track test cases, manage bugs, and ensure overall software quality. These tools can focus on specific testing areas, such as functional, API, or web service testing, or be used for broader approaches like exploratory testing. Comprehensive QA management systems support the entire process, from planning and execution to analysis and reporting. They often integrate seamlessly with collaboration platforms, making it easier to assign tasks and coordinate efforts within teams [28]. This combination of tools and services allows specialists to streamline Edge Computing systems testing workflows, improve efficiency, and optimize the software testing process.

In general we can categorize testing tools by their usage in different test types (functional, performance, API, load, etc), by test management area (test case management, bug tracking, logging, monitoring, etc.) and by general testing purposes (test data generation, device simulation, screen recording). Suggested categorization is displayed on figure 3

Considering all the testing challenges and features of edge computing systems' testing described at previous chapter, we can formulate list of criteria to provide appropriate QA tools selection (table 2).

Most of the above indicators can be assessed based on official documentation from software vendors or cloud service providers. However, such indicators as load handling, horizontal scaling, vertical scaling, stress testing, latency maintenance and tool customizability should be determined based on expert assessment with subsequent verification of the respondents' degree of agreement. So, developing a methodology for integrated assessment according to the criteria and indicators we have defined using both approaches is advisable.

In general, QA team must aggregate full set of tools to be able to provide:

- functional testing;
- latency and performance testing;
- security testing;
- fault tolerance and resilience testing;

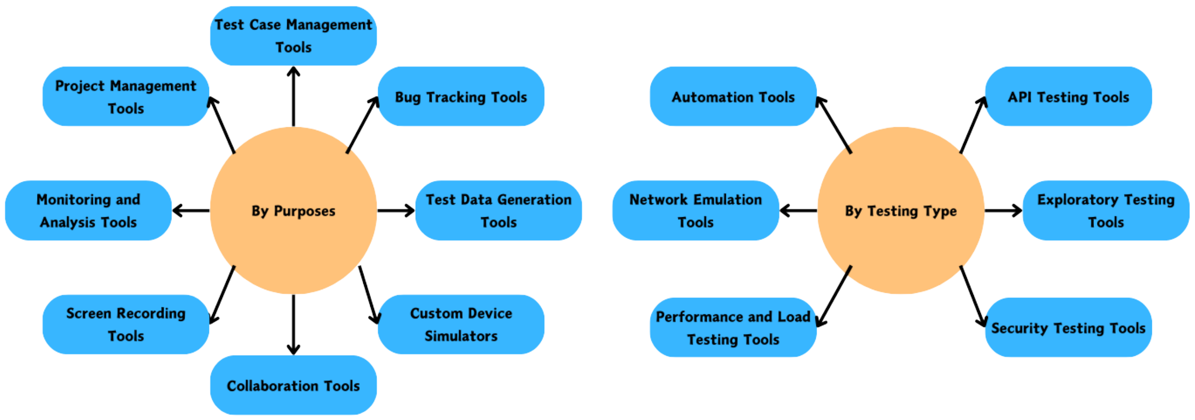

Figure 3: Testing tool categories.

- monitoring and measurement.

To achieve a comprehensive coverage of special requirements to edge computing systems' testing, there is a need for detailed selection of appropriate tools. As a result of investigation, tools indicated in table 3 can be suggested as the most suitable to perform needed workflows and assertions.

While traditional testing tools are valuable, edge computing requires specialized solutions that can handle distributed architectures and variable network conditions. Based on the papers analyzed above and our own experience, we offer the set of essential tools for edge computing QA:

The Linux-based network emulation tool **NetEm (Network Emulator)** [29] simulates network conditions such as latency, packet loss, duplication, and jitter. It is commonly employed to test application performance under adverse network scenarios. By configuring kernel parameters, it allows developers to mimic real-world network behaviors, enabling testing of edge systems in highly dynamic and unreliable network environments, such as those seen in remote or mobile edge nodes.

NetEm's flexibility and granular control make it a valuable tool for testing the resilience of edge applications under challenging conditions. It helps developers assess system behavior during network disruptions, which is critical for applications requiring high reliability and low latency.

WANem (Wide Area Network Emulator) [30] is an open-source tool designed to replicate WAN characteristics such as high latency, limited bandwidth, and network congestion. It is widely used to test distributed applications that operate over wide geographic distances. WANem offers a user-friendly interface for creating scenarios that emulate the communication between edge nodes and cloud data centers, allowing organizations to assess system behavior in geographically distributed edge computing setups.

WANem provides a straightforward approach to emulating WAN scenarios, ensuring that edge systems can handle bandwidth constraints and varying latency. While it is effective for general testing, it may lack some advanced features required for highly specific edge environments.

Simple yet effective tool for injecting network faults into local network communications is **Clumsy** [31]. It allows users to introduce packet drops, latency, corruption, and duplication into a network to test the resilience of software or devices. While primarily designed for smaller-scale or local testing environments, Clumsy can be highly useful for edge system developers to evaluate edge device behavior under adverse network conditions during early-stage development.

Clumsy provides basic but effective network emulation, making it ideal for testing small edge deployments. Its limited feature set may not suffice for more complex distributed environments, but it's a good starting point for basic network testing.

For performance testing **Apache JMeter** [32] is a powerful open-source tool that supports a wide range of protocols, including HTTP, FTP, JDBC, and SOAP. It is used to measure application performance, load, and scalability under various conditions. In the context of edge computing, JMeter is particularly

Table 2
Edge computing systems' testing tools selection criteria.

Criteria	Indicators
1. Scalability	1.1 Load Handling: Maximum number of concurrent users, devices, or data streams the tool can manage. 1.2 Horizontal Scaling: Support for distributed environments by adding more edge nodes or servers. 1.3 Vertical Scaling: Ability to optimize performance by increasing resource utilization (CPU, memory, storage) on existing nodes. 1.4 Stress Testing: Performance under high data loads, network traffic, or concurrent operations. 1.5 Latency Maintenance: Ability to maintain low latency as the system scales.
2. Compatibility	2.1 Multi-Platform Support: Compatibility with operating systems and hardware architectures 2.2 Protocol Support: Support for edge-specific protocols 2.3 Containerization: Support for container technologies (e.g., Docker, Kubernetes) often used in edge deployments. 2.4 Interoperability: Integration with different IoT platforms, cloud backends, and middleware. 2.5 Real-Time Processing: Support for real-time data processing and decision-making capabilities.
3. Integrity	3.1 CI/CD Support: Compatibility with CI/CD tools like Jenkins, GitLab CI/CD, GitHub Actions, Azure DevOps, or CircleCI. 3.2 Automated Test Execution: Capability to trigger tests automatically during build, deployment, or code changes. 3.3 Version Control Integration: Compatibility with version control systems like Git for managing code and configurations.
4. Cost saving	4.1 Upfront Cost: Competitive price of purchasing the tool or system. 4.2 Subscription Model: Availability of flexible pricing models, such as pay-as-you-go or annual subscriptions. 4.3 Free Trial or Open Source: Availability of free versions or open-source alternatives.
5. Licensing	5.1 Licensing Transparency: Support for commercial, academic, or non-profit licensing.
6. Ease of Use	6.1 User Interface (UI): Presence of an intuitive and visually accessible graphical interface. 6.2 Documentation: Availability of comprehensive user guides, tutorials, and knowledge bases. 6.3 Support Channels: Availability of help via chat, forums, email, or phone support. 6.4 Customizability: Flexibility to tailor the tool's settings, workflows, or reports to specific needs.

valuable for testing APIs, microservices, and edge-based web applications, helping organizations ensure these components can handle real-world traffic loads and response times.

JMeter's ability to simulate heavy user loads and test APIs ensures thorough performance validation in edge environments. While it excels in scalability testing, it may struggle to handle complex distributed setups without additional configuration.

As for the load testing **Gatling**[33] is a popular open-source tool specifically designed for high-throughput HTTP applications. Known for its performance and efficiency, Gatling is widely used to simulate concurrent user behavior and analyze response times. For edge systems, Gatling can be employed to test the responsiveness of services that require low latency, such as real-time IoT data processing or edge-based decision-making systems.

Gatling's intuitive interface and efficient resource usage make it a strong choice for edge performance testing. While it requires some initial learning, it provides comprehensive insights into system behavior under load.

Python-based load testing tool that enables developers to simulate user behavior across distributed systems is **Locust** [34]. It supports thousands of concurrent simulated users, making it an excellent choice for edge environments with multiple devices or nodes. Locust's flexibility allows for custom test

Table 3
Systematized tools suitable for edge computing systems' testing.

Type	**Name**	**Benefits for edge computing systems' testing**
Network emulators	Linux NetEM	Simulates real-world network conditions like latency, packet loss, and jitter, helping to test edge systems' performance under diverse and challenging network scenarios.
	WANem	Provides an easy-to-use interface for emulating wide-area network conditions, enabling the testing of edge system performance under unreliable and constrained connectivity.
	Clumsy	Allows network fault injection (latency, packet drops, etc.), ensuring robustness and reliability of edge systems operating in fluctuating network environments.
Performance and load testing	JMeter	Simulates high user loads to evaluate edge systems' performance, scalability, and ability to process simultaneous requests effectively.
	Gatling	Delivers detailed performance metrics for edge systems by testing data processing speeds, throughput, and response times under varying loads.
	Locust	Facilitates distributed load testing, ideal for edge environments with numerous connected devices, to ensure the scalability of the overall system.
Security testing	NMap	Scans edge devices for open ports, which is key strategy to find network vulnerabilities or system miss-configurations.
	Shodan	Search engine that can be used to scan IoT devices connected to the same network and find potential threats or vulnerabilities.
	Metasploit	Framework for penetration testing to simulate attacks on edge devices and applications by supporting exploit matching, execution, and development.
Monitoring & metrics	Prometheus	Enables real-time monitoring of edge devices and applications, identifying performance bottlenecks and ensuring reliable data collection and processing.
	Grafana	Provides visual analytics and dashboards for edge system metrics, allowing quick identification of anomalies or performance trends.
	ELK Stack	Offers comprehensive logging, monitoring, and data analysis for edge systems, ensuring efficient troubleshooting and root cause analysis in distributed environments.
	NewRelic	Delivers full-stack monitoring for edge devices, providing actionable insights on system health, performance, and end-user experiences.
	Nagios	Monitors the health, uptime, and performance of edge devices and infrastructure, ensuring early detection of issues in critical environments.
AI/ML frameworks	TensorFlow Extended (TFX)	Supports the end-to-end testing of AI/ML pipelines in edge environments, ensuring accurate model deployment, data integrity, and consistent predictions in real-time.
	DataRobot	Provides automated AI/ML model testing and monitoring, ideal for validating the performance of intelligent systems deployed at the edge.
Test automation	Katalon Studio	Simplifies functional, API, and UI testing for edge applications, ensuring robust performance across diverse device configurations.

scripts, making it highly adaptable for edge applications that demand specific interaction patterns, such as smart manufacturing or autonomous vehicle networks.

Locust's flexibility and scalability allow for detailed performance evaluations of edge systems. Its reliance on scripting requires programming knowledge but enables highly customizable testing scenarios.

Moving to network security testing tools, we can suggest **Nmap** [35]. Open-source network scanning tool that provides detailed insights into hosts, open ports, and potential vulnerabilities. It is widely used to assess the security posture of edge devices and detect misconfigurations that could expose systems to cyber threats.

Nmap's scripting engine enables automated scans and custom security assessments, making it highly effective for distributed edge environments. Its command-line interface may be intimidating for beginners, but its versatility and accuracy in network reconnaissance make it a crucial tool for security professionals.

Another useful tool is **Shodan** [36]. A search engine for internet-connected devices that provides visibility into exposed edge systems, open ports, and security risks. It is widely used to identify vulnerable IoT devices, industrial control systems, and cloud-connected edge nodes.

Shodan's ability to scan and categorize devices based on their fingerprints makes it invaluable for cybersecurity teams. Its premium features can be costly, but its powerful search capabilities make it an essential tool for identifying and securing publicly exposed edge infrastructure.

Penetration testing is a crucial stage of system's security assurance flow. **Metasploit** [37] is an open-source security testing framework that allows teams to simulate real-world cyberattacks on edge computing systems. It is widely used to exploit vulnerabilities in networks, applications, and connected devices to assess their resilience against threats.

Metasploit's extensive exploit database and automation features make it a powerful tool for proactive security testing. Its complexity may require a learning curve for new users, but its ability to simulate advanced attack scenarios makes it indispensable for ethical hackers and security teams.

Really useful would be **Prometheus** [38]. It's an open-source monitoring and alerting toolkit designed for collecting and querying time-series data. It integrates well with edge systems to monitor the health and performance of edge devices and microservices. Prometheus is particularly suited for edge testing environments that require real-time insights into resource usage, network conditions, or system availability.

Prometheus's detailed metrics and real-time alerts make it indispensable for maintaining edge system reliability. Its complexity can pose a challenge for new users, but its integration with other tools like Grafana enhances its utility.

As an alternative tool for Prometheus, we can use **Grafana** [39] – open-source data visualization and monitoring tool that integrates with Prometheus and other data sources. It allows users to create dynamic dashboards for real-time analysis. In edge computing, Grafana is invaluable for visualizing metrics such as latency, throughput, and resource utilization across distributed edge systems.

Grafana's customizable dashboards provide clear insights into edge system performance. It requires time to set up effectively, but its visualization capabilities make it a valuable monitoring tool.

Also, the powerful kit is **ELK Stack** [40], a combination of three powerful tools Elasticsearch for log storage and search, Logstash for log collection and processing, and Kibana for visualization. This suite is widely used for centralizing and analyzing logs from distributed systems. In edge environments, ELK helps manage the extensive logging generated by edge devices and microservices, ensuring insights into system performance and error tracking.

It handles large volumes of log data efficiently, aiding in debugging and performance optimization. Its resource intensity requires robust infrastructure, but its analytical power is unmatched for log-based insights.

One more monitoring tool is **New Relic** [41], a cloud-based observability platform offering performance monitoring, distributed tracing, and analytics. It is widely used to analyze application and system behavior in real-time, making it highly applicable for edge computing systems. New Relic provides end-to-end visibility across the entire system, including edge devices, microservices, and cloud backends, helping to ensure seamless and reliable operations.

New Relic simplifies observability with intuitive dashboards and detailed traces, ensuring edge systems operate smoothly. Its licensing costs and dependency on the internet may limit its applicability for some teams.

For infrastructure monitoring we would use **Nagios** [42]. It is an open-source system monitoring tool that provides real-time insights into network, server, and application performance. It is widely used to monitor the health and availability of edge devices and nodes. Nagios can alert teams to hardware failures, software issues, or network disruptions in edge environments, ensuring proactive problem resolution.

Nagios's customization options ensure effective monitoring of distributed edge systems. It's dated interface may deter some users, but its reliability and plugin ecosystem are significant advantages.

When working with artificial intelligence and machine learning systems, it's important to use specific tools, that are able to process models assessment and validation.

End-to-end platform for deploying and managing machine learning models in production is **TensorFlow Extended (TFX)** [43]. It is often used in edge environments where AI models must run locally on edge devices, such as image recognition in smart cameras or predictive analytics in industrial IoT. TFX enables model testing, validation, and monitoring, ensuring robust and accurate AI deployment at the edge.

TFX ensures robust model validation and deployment in edge AI setups. Its steep learning curve requires expertise but offers unmatched reliability and performance for AI-driven edge applications.

Also, it's worth mentioning **DataRobot** [44], an automated machine learning (AutoML) platform designed to simplify and accelerate the development and deployment of AI models. In edge computing, it helps create models that can operate locally on edge devices, enabling predictive analytics, anomaly detection, or AI-based decision-making directly at the edge. DataRobot ensures that models are efficient, accurate, and optimized for deployment in resource-constrained edge environments.

DataRobot accelerates AI development for edge applications, providing robust model validation. Its cost and limited customization may pose challenges for small teams, but its capabilities significantly enhance edge AI operations.

As an optimal test automation tool we would suggest **Katalon Studio** [45]. It's a versatile platform that supports web, mobile, API, and desktop applications. For edge computing, Katalon can automate functional testing of applications running on edge devices or microservices. Its simplicity and versatility make it ideal for testing interfaces, workflows, and APIs in edge environments.

Katalon simplifies functional and API testing for edge applications. While limited for system-level testing, its ease of use and automation capabilities are ideal for validating workflows.

To perform real-time testing on edge computing systems like IoT sensors that require continuous data streaming it is necessary to use systems that allow to create custom data pipelines and analyze them. For that porpose we would offer to use Apache products: **Kafka** [46] and **Flink** [47]. Though these are not a testing tools but their usage can bring great benefits to QA processes.

Apache Kafka is a distributed messaging system designed for real-time data streaming. It is commonly used in edge computing to facilitate the movement of data between edge devices, processing nodes, and cloud systems. Kafka is particularly useful for building scalable and fault-tolerant edge architectures that handle high volumes of real-time data from IoT devices or edge sensors.

Kafka's high throughput and scalability make it essential for edge data streaming. Its configuration complexity requires expertise, but its ability to handle real-time data flows ensures reliable edge operations.

Apache Flink is a distributed stream-processing framework optimized for real-time analytics. It supports low-latency processing of data streams, making it an excellent choice for edge applications like predictive maintenance, real-time monitoring, and analytics. Flink's ability to process massive data streams in distributed environments ensures reliable edge data handling and analysis.

Flink's low-latency processing capabilities make it a top choice for edge applications requiring real-time insights. Its complexity makes it suitable for advanced use cases and experienced teams.

5. Conclusions

Today, edge computing introduces challenges, including decentralized operations, time-sensitive data processing, and security vulnerabilities, all of which demand robust testing strategies.

Due to performed bibliometric analysis we can note that current state of edge computing system's quality assurance field is mostly oriented on IoT, cloud-computing, embedded and open-source systems testing. But the problems of justification and definition criteria for selections edge computing systems' testing tools are still releavant. At the same time, it is necessary to consider the specifics of modern peripheral devices.

Continuous validation of system reliability, adaptability to harsh environments, and seamless data flow are essential to maintaining operational efficiency and safeguarding against failures or breaches. These testing approaches collectively ensure that edge systems meet stringent performance, safety, and security requirements, supporting their deployment in mission-critical and dynamic environments.

Burn-in testing and real-time data testing are fundamental methodologies in ensuring the reliability and performance of edge computing systems, particularly in critical applications. Burn-in testing identifies and eliminates defective components through rigorous stress testing, which enhances the durability and dependability of hardware in challenging operational environments. Real-time data testing, on the other hand, plays a pivotal role in validating the integrity, accuracy, and responsiveness of edge devices, ensuring smooth operations in dynamic and high-stakes scenarios.

To provide appropriate QA tools selection, next list of criteria is suggested: scalability, compatibility with edge architectures, integration with CI/CD pipelines, cost suitability, licensing and ease of use. Most of these criteria can be estimated by checking tool's documentation or by their exploratory use. But the assessment of more complicated and data-driven indicators require expert analysis which can be discovered in future investigations.

So the main task for QA teams is to aggregate full set of tools to be able to provide functional latency, performance, security, fault tolerance, and resilience testing. Also, the tools for monitoring and measurement are required.

Author Contributions: Conceptualization, Viktor O. Maliarskyi and Vasyl P. Oleksiuk; writing – original draft, Viktor O. Maliarskyi; writing—review and editing, Vasyl P. Oleksiuk. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Data Availability Statement: No new data were created or analysed during this study. Data sharing is not applicable.

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

Declaration on Generative AI: While preparing this work, the authors used ChatGPT and Grammarly to improve sentence structure and word choice.

References

[1] F. Okezie, I. Odun-Ayo, S. Bogle, A Critical Analysis of Software Testing Tools, Journal of Physics: Conference Series 1378 (2019) 042030. doi:10.1088/1742-6596/1378/4/042030.

[2] V. B. Ramu, Edge Computing Performance Amplification, arXiv:2305.16175 (2023). doi:10.48550/arXiv.2305.16175.

[3] B. Cohen, G. Csatári, S. Huang, B. Jones, A. Lebre, D. Paterson, I. Váncsa, Edge Computing: Next Steps in Architecture, Design and Testing, 2023. URL: https://www.openstack.org/use-cases/edge-computing/edge-computing-next-steps-in-architecture-design-and-testing.

[4] R. Timbó, Best QA Testing Tools in 2023, 2023. URL: https://www.revelo.com/blog/qa-testing-tools.

[5] R. Sujatha, 10 AI Testing Tools to Streamline Your QA Process in 2024, 2024. URL: https://www.digitalocean.com/resources/articles/ai-testing-tools.

[6] I. Mintii, S. Semerikov, Optimizing Teacher Training and Retraining for the Age of AI-Powered Personalized Learning: A Bibliometric Analysis, in: E. Faure, Y. Tryus, T. Vartiainen, O. Danchenko, M. Bondarenko, C. Bazilo, G. Zaspa (Eds.), Information Technology for Education, Science, and Technics, volume 222 of *Lecture Notes on Data Engineering and Communications Technologies*, Springer Nature Switzerland, Cham, 2024, pp. 339–357. doi:10.1007/978-3-031-71804-5_23.

[7] P. G. Vinueza-Naranjo, J. Chicaiza, R. Rumipamba-Zambrano, Fog Computing Technology Research: A Retrospective Overview and Bibliometric Analysis, ACM Comput. Surv. 57 (2024). doi:10.1145/3702313.

[8] N. J. van Eck, L. Waltman, VOSviewer Manual, 2023. URL: https://www.vosviewer.com/documentation/Manual_VOSviewer_1.6.20.pdf.

[9] T. A. Vakaliuk, O. V. Andreiev, O. F. Dubyna, O. L. Korenivska, Y. O. Andreieva, Wireless technologies in IoT projects with distributed computing, in: T. A. Vakaliuk, S. O. Semerikov (Eds.), Proceedings of the 4th Edge Computing Workshop (doors 2024), Zhytomyr, Ukraine, April 5, 2024, volume 3666 of *CEUR Workshop Proceedings*, CEUR-WS.org, 2024, pp. 4–13. URL: https://ceur-ws.org/Vol-3666/paper01.pdf.

[10] F. Nikolaidis, A. Chazapis, M. Marazakis, A. Bilas, Event-Driven Testing For Edge Applications, 2022. URL: http://arxiv.org/abs/2212.12370v1.

[11] D. N. Jha, K. Alwasel, A. Alshoshan, X. Huang, R. K. Naha, S. K. Battula, S. Garg, D. Puthal, P. James, A. Zomaya, S. Dustdar, R. Ranjan, IoTSim-Edge: A simulation framework for modeling the behavior of Internet of Things and edge computing environments, Software - Practice and Experience 50 (2020) 844 – 867. doi:10.1002/spe.2787.

[12] K. Alwasel, D. N. Jha, F. Habeeb, U. Demirbaga, O. Rana, T. Baker, S. Dustdar, M. Villari, P. James, E. Solaiman, R. Ranjan, IoTSim-Osmosis: A framework for modeling and simulating IoT applications over an edge-cloud continuum, Journal of Systems Architecture 116 (2021). doi:10.1016/j.sysarc.2020.101956.

[13] V. Daneshmand, K. C. Subratie, R. J. Figueiredo, PolyNet: Cost- and Performance-Aware Multi-Criteria Link Selection in Software-Defined Edge-to-Cloud Overlay Networks, in: 2024 IEEE 10th International Conference on Network Softwarization (NetSoft), 2024, pp. 127–135.

[14] B. Varghese, P. Leitner, S. Ray, K. Chard, A. Barker, Y. Elkhatib, H. Herry, C.-H. Hong, J. Singer, F. P. Tso, E. Yoneki, M.-F. Zhani, Cloud Futurology, Computer 52 (2019) 68–77. doi:10.1109/MC.2019.2895307.

[15] O. S. Holovnia, V. P. Oleksiuk, Selecting cloud computing software for a virtual online laboratory supporting the Operating Systems course, in: A. E. Kiv, S. O. Semerikov, M. P. Shyshkina (Eds.), Proceedings of the 9th Workshop on Cloud Technologies in Education, CTE 2021, Kryvyi Rih, Ukraine, December 17, 2021, volume 3085 of *CEUR Workshop Proceedings*, CEUR-WS.org, 2021, pp. 216–227.

[16] S. Papadakis, S. O. Semerikov, A. M. Striuk, H. M. Kravtsov, M. P. Shyshkina, M. V. Marienko, H. B. Danylchuk, Embracing digital innovation and cloud technologies for transformative learning experiences, CEUR Workshop Proceedings 3679 (2024) 1–21. URL: https://ceur-ws.org/Vol-3679/paper00.pdf.

[17] S. O. Semerikov, T. A. Vakaliuk, I. S. Mintii, V. A. Hamaniuk, O. V. Bondarenko, P. P. Nechypurenko, S. V. Shokaliuk, N. V. Moiseienko, Designing an immersive cloud-based educational environment for universities: a comprehensive approach, CEUR Workshop Proceedings 3844 (2024) 107 – 116. URL: https://ceur-ws.org/Vol-3844/paper09.pdf.

[18] S. Busaeed, I. Katib, A. Albeshri, J. M. Corchado, T. Yigitcanlar, R. Mehmood, LidSonic V2.0: A LiDAR and Deep-Learning-Based Green Assistive Edge Device to Enhance Mobility for the Visually Impaired, Sensors 22 (2022) 7435. doi:10.3390/s22197435.

[19] K. Memon, N. Yahya, M. Z. Yusoff, R. Remli, A.-W. M. M. Mustapha, H. Hashim, S. S. A. Ali, S. Siddiqui, Edge Computing for AI-Based Brain MRI Applications: A Critical Evaluation of Real-Time Classification and Segmentation, Sensors 24 (2024) 7091. doi:10.3390/s24217091.

[20] S. Grigorescu, T. Cocias, B. Trasnea, A. Margheri, F. Lombardi, L. Aniello, Cloud2Edge Elastic AI Framework for Prototyping and Deployment of AI Inference Engines in Autonomous Vehicles, Sensors 20 (2020) 5450. doi:10.3390/s20195450.

[21] F. P. Scalcon, R. Tahal, M. Ahrabi, Y. Huangfu, R. Ahmed, B. Nahid-Mobarakeh, S. Shirani, C. Vidal, A. Emadi, AI-Powered Video Monitoring: Assessing the NVIDIA Jetson Orin Devices for Edge Computing Applications, in: 2024 IEEE Transportation Electrification Conference and Expo, ITEC 2024, 2024. doi:10.1109/ITEC60657.2024.10598994.

[22] ISTQB® Certified Tester Advanced Level – Test Automation Engineering certification, 2023. URL: https://www.istqb.org/certifications/certified-tester-advanced-level-test-automation-engineering-ctal-tae-v2-0/.

[23] B. Fellows, How to Test Software for Edge Computing Environments, 2024. URL: https://www.workwithloop.com/blog/how-to-test-software-for-edge-computing-environments.

[24] J. Malik, F. Pastore, An empirical study of vulnerabilities in edge frameworks to support security testing improvement, Empirical Software Engineering 28 (2023). doi:10.1007/s10664-023-10330-x.
[25] J. Beilharz, P. Wiesner, A. Boockmeyer, L. Pirl, D. Friedenberger, F. Brokhausen, I. Behnke, A. Polze, L. Thamsen, Continuously Testing Distributed IoT Systems: An Overview of the State of the Art, in: Service-Oriented Computing – ICSOC 2021 Workshops, Springer International Publishing, 2022, pp. 336–350. doi:10.1007/978-3-031-14135-5_30.
[26] P. Gupta, Edge Computing and Real-Time Data Testing: Enhancing System Reliability, 2023. URL: https://fpgainsights.com/test-measurement/edge-computing-and-real-time-data-testing.
[27] S. Durrett, Ensure Reliability in Edge Computing with Burn-In Testing, 2023. URL: https://www.smithsinterconnect.com/smiths-interconnect-blog/ensure-reliability-in-edge-computing-with-burn-in-testing.
[28] T. Khomenko, Top 15 Manual Testing Tools Checklist To Know In 2024, 2023. URL: https://testomat.io/blog/top-15-manual-testing-tools-to-know/#what-tools-are-used-in-manual-testing.
[29] Linux NetEM manual, 2025. URL: https://man7.org/linux/man-pages/man8/tc-netem.8.html.
[30] The Wide Area Network emulator, 2014. URL: https://wanem.sourceforge.net/.
[31] Clumsy 0.3, 2021. URL: https://jagt.github.io/clumsy/.
[32] Apache JMeter™, 2024. URL: https://jmeter.apache.org/index.html.
[33] Gatling Official Page, 2025. URL: https://gatling.io/.
[34] Locust Official Page, 2025. URL: https://locust.io/.
[35] Nmap Official Page, 2021. URL: https://nmap.org/.
[36] Shodan Official Page, 2023. URL: https://www.shodan.io/.
[37] Metasploit Official Page, 2025. URL: https://www.metasploit.com/.
[38] Prometheus Official Page, 2025. URL: https://prometheus.io/.
[39] Grafana Official Page, 2025. URL: https://grafana.com/.
[40] Elastic Stack Official Page, 2025. URL: https://www.elastic.co/elastic-stack.
[41] NewRelic Official Page, 2025. URL: https://newrelic.com/.
[42] Nagios Official Page, 2025. URL: https://www.nagios.org/.
[43] TensorFlow Extended (TFX) Official Page, 2025. URL: https://www.tensorflow.org/tfx/.
[44] DataRobot Official Page, 2025. URL: https://www.datarobot.com/.
[45] Katalon Studio Official Page, 2025. URL: https://katalon.com/.
[46] Apache Kafka Official Page, 2025. URL: https://kafka.apache.org/.
[47] Apache Flink Official Page, 2025. URL: https://flink.apache.org/.