Performance Evaluation and Scalability Analysis of FIWARE, ThingsBoard, and EdgeX Foundry IoT Platforms

Anonymous

Abstract

This paper presents a comprehensive performance evaluation and scalability analysis of three prominent open-source IoT platforms: FIWARE, ThingsBoard, and EdgeX Foundry. Using systematic load testing with JMeter at three distinct scales (10×100, 100×1000, 1000×1000), we evaluate platform performance across single-node and multi-node deployments. Our experimental results demonstrate significant scalability differences: FIWARE achieves 20-25% system CPU utilization in multi-node deployments compared to 37-50% in single-node configurations but experiences complete failure under maximum load; ThingsBoard shows counterintuitive scaling behavior with multi-node deployments consuming 105% CPU versus 75% in single-node setups; while EdgeX Foundry maintains exceptional stability with error rates below 1% and optimal resource utilization across all tested scenarios.

Key findings reveal measurable performance improvements in multi-node FIWARE deployments which reduce system CPU usage from 37-50% to 20-25% under low load while maintaining zero error rates. ThingsBoard demonstrates limited horizontal scaling benefits with error rates improving marginally from 8% to 5% at medium load but shows concerning resource utilization patterns. EdgeX Foundry achieves superior cache performance with >90% Redis hit rates and sustained throughput while maintaining stability even under extreme 1000×1000 load conditions. Database performance analysis shows FIWARE MongoDB replica sets exhibit suboptimal behavior with I/O operations increasing from 30 to 190 ops/sec in multi-node configurations, while EdgeX Foundry's Redis implementation maintains consistent high performance. These quantitative results provide critical insights for IoT platform selection and deployment strategies in large-scale environments.

Keywords— IoT platforms, performance evaluation, scalability analysis, FIWARE, ThingsBoard, EdgeX Foundry, load testing

1 Introduction

The Internet of Things (IoT) ecosystem has experienced unprecedented growth, with billions of connected devices generating massive volumes of data requiring efficient processing, storage, and analysis. As smart cities, industrial automation, and connected healthcare systems expand, the selection and deployment of appropriate IoT platforms becomes critical for ensuring system reliability, scalability, and performance under varying load conditions.

Contemporary IoT platforms must address several key challenges: handling massive device connectivity, processing high-frequency data streams, maintaining low-latency responses, and scaling horizontally to accommodate growing device populations. Open-source IoT platforms have emerged as preferred solutions due to their flexibility, cost-effectiveness, and community-driven development approaches.

This paper presents a systematic performance evaluation of three leading open-source IoT platforms: FIWARE, ThingsBoard, and EdgeX Foundry. Our research addresses fundamental questions regarding platform scalability, resource utilization efficiency, and failure points under increasing load conditions. Through controlled experimental testing using industry-standard load testing methodologies, we provide quantitative insights into platform behavior across different deployment architectures and load scenarios.

2 Literature Review

2.1 IoT Platform Performance Evaluation and Benchmarking

IoT platform performance evaluation has emerged as a critical research area with substantial academic attention. Rabl et al. (2015) established foundational benchmarking methodologies with the IoTAbench benchmark toolkit for IoT analytics platforms [1]. Their work remains influential in establishing standardized evaluation approaches for IoT systems, providing systematic frameworks for performance assessment in big data sce-

narios.

The definitive performance evaluation of FIWARE was conducted by Araujo et al. (2019), analyzing vertical and horizontal scaling effects on FIWARE IoT platform performance in cloud-based deployments [2]. Their comprehensive analysis revealed specific scaling limitations and optimization strategies that directly inform contemporary platform evaluation methodologies. The study demonstrated that FIWARE's microservices architecture provides clear benefits in distributed deployments while identifying bottlenecks in extreme load scenarios.

Ismail et al. (2018) conducted a seminal comparative analysis of ThingsBoard and SiteWhere platforms, evaluating scalability and stability metrics across HTTP and MQTT protocols [3]. Their results indicated ThingsBoard's superior REST performance while SiteWhere demonstrated better MQTT handling, highlighting protocol-dependent performance characteristics that remain relevant for platform selection decisions.

2.2 Comparative Analysis of Open-Source IoT Platforms

Systematic comparative studies of IoT platforms have gained prominence across top-tier venues. The comprehensive survey by Babun et al. (2021) in Computer Networks analyzed IoT platforms from communication, security, and privacy perspectives, establishing benchmarking frameworks for platform evaluation across different application domains [4].

The systematic mapping study by Di Felice and Paolone (2024) mentioning ThingsBoard, establishing it as the highest-rated platform among seven open-source alternatives with a score of 30.5/42 across 14 evaluation criteria [5]. Their quantitative analysis revealed ThingsBoard's superior performance in connectivity, security, scalability, and data processing capabilities, while identifying analytics as the primary limitation.

Zyrianoff et al. (2021) published comprehensive interoperability analysis comparing FIWARE and Web of Things (WoT) architectures [6]. Their performance evaluation demonstrated that conceptual design choices significantly impact application performance, with FIWARE's platform-specific IoT Agent solution requiring new implementations for different data models.

2.3 Edge Computing and Industrial IoT Platform Performance

Industrial Internet of Things (IIoT) platform performance evaluation has become critical for edge-to-cloud computing paradigms. Jamil et al. (2024) published a comprehensive survey examining how ThingsBoard, Eclipse Ditto, and Microsoft Azure IoT address key IIoT requirements [7]. Their analysis revealed that ThingsBoard supports standalone server deployment suitable for up to 300,000 devices with 10,000 messages per second, while typical virtual environments handle approximately 5,000 telemetry data points per second.

Venanzi et al. (2023) presented comparative functional and performance evaluation of Siemens and EdgeX IIoT platforms [8]. Their analysis revealed EdgeX Foundry's superior modularity and performance characteristics compared to proprietary industrial IoT solutions, providing direct evidence for EdgeX's architectural advantages in industrial contexts.

2.4 Communication Protocol Performance and Optimization

Communication protocol performance represents a fundamental aspect of IoT platform scalability. Seoane et al. (2021) published a definitive performance evaluation of CoAP and MQTT protocols, focusing on bandwidth and CPU usage under security constraints [9]. Their analysis revealed that CoAP's UDP-based architecture provides superior performance in resource-constrained environments, while MQTT's TCP foundation ensures higher reliability with increased overhead.

Palmese et al. (2020) provided a specialized analysis of CoAP versus MQTT-SN for publish-subscribe scenarios [10]. Their implementation of CoAP Pub/Sub functionalities revealed traffic behavior characteristics that inform platform architecture decisions, demonstrating that protocol choice affects both latency and resource utilization patterns.

The comparative analysis by Sara and Hammoudeh (2022) extended protocol evaluation to include AMQP, providing comprehensive security assessments across resource-constrained devices and networks [11]. Their findings demonstrate that protocol selection significantly impacts overall platform performance, particularly in edge computing scenarios where bandwidth and computational resources are limited.

2.5 Microservices Architecture in IoT Systems

Microservices architecture has emerged as a dominant paradigm for scalable IoT platform design. Siddiqui et al. (2023) published a comprehensive state-of-the-art review of microservices-based architectures for IoT systems [12]. Their analysis reveals that microservices enable better scalability, interoperability, and modifiability compared to monolithic approaches, though they introduce complexity in distributed coordination.

The practical implementation of reactive microservices for IoT applications was evaluated by Lira et al. (2023) [13]. Their experimental analysis demonstrated that microservices architecture provides significant performance improvements in IoT environments, particularly for applications requiring real-time data processing and dynamic scaling capabilities.

Abuseta (2024) presents comprehensive analysis of quality attribute-driven software architectures for IoT systems, emphasizing the critical role of microservices, edge computing, and event-driven architectures [14]. The study proposes treating IoT systems as autonomic systems requiring closed control loops for orchestration, incorporating the MAPE-K feedback loop model that informs scalable platform design.

2.6 Scalability Analysis and Performance Optimization Techniques

Distributed systems scalability research has advanced through comprehensive analytical frameworks. Rahman (2022) published extensive analysis of blockchain-based scalability solutions for IoT environments, establishing evaluation frameworks encompassing throughput, latency, and block size considerations [15]. This work identified six overarching scalability issues requiring resolution by industry and research communities.

Performance optimization techniques for distributed IoT systems have been systematically analyzed through large-scale surveys. Eeti et al. (2023) published comprehensive analysis of scalability and performance optimization techniques across 100 IT companies, revealing that 78% employ horizontal scaling with 57.7% reporting significant performance improvements [16]. The study demonstrated that vertical scaling achieves 61.5% significant improvement rates among 65% of surveyed organizations.

Container orchestration optimization for IoT edge computing has been examined through practical deployment studies. Kaiser et al. (2024) evaluated hybrid edge systems using containers and unikernels for IoT applications, examining Docker, Kubernetes, and specialized edge orchestration frameworks [17]. Their evaluation provides insights into resource optimization strategies for constrained environments that directly apply to platform deployment decisions.

2.7 Security and Privacy in Distributed IoT Platforms

Blockchain integration with IoT platforms addresses security and privacy concerns in distributed environments. Gugueoth et al. (2023) published a comprehensive review examining IoT security using decentralized blockchain solutions [18]. Their analysis reveals that blockchain-IoT integration enhances security through distributed ledger technology, though scalability challenges remain significant for large-scale deployments.

Loss et al. (2022) introduced innovative blockchain integration approaches for FIWARE in smart cities applications [19]. Their quantitative evaluation using Apache JMeter demonstrated transaction processing capabilities of 520.7, 483.1, 462.7, and 449.6 transactions per second for 10, 50, 100, and 500 concurrent users respectively, with zero error rates and average response times ranging from 3ms to 245ms.

The privacy-preserving aspects of IoT networks have been systematically analyzed by Wakili et al. (2025) [20]. Their comparative analysis of privacy-preserving security methods provides frameworks for evaluating platform security characteristics, particularly relevant for large-scale IoT deployments requiring data protection.

2.8 Edge Computing Integration and Federated Learning

Edge computing integration with IoT platforms has become critical for latency-sensitive applications. Verma (2021) provided comprehensive comparative analysis of cloud computing and edge computing paradigms [21]. The study forecasts that 70% of IoT-generated data will be processed at network edges by 2025, emphasizing the importance of edge-capable platform architectures.

Albogami et al. (2025) developed an Intelligent Deep Federated Learning Model for IoT-enabled edge computing environments [22]. Their IDFLM-ES approach achieved 98.24% accuracy in security enhancement, demonstrating the potential for AI-driven optimization in distributed IoT platforms.

Zhao et al. (2024) designed federated learning systems with reputation mechanisms for manufacturers leveraging customer data while preserving privacy [23]. Their approach demonstrates practical implementation strategies for privacy-preserving distributed learning in IoT environments that complement platform security capabilities.

2.9 Testing Methodologies and Benchmarking Frameworks

Systematic performance evaluation methodologies have evolved to address the complexity of modern IoT platforms. Minani et al. (2025) published comprehensive taxonomies for IoT systems testing, establishing seven distinct categories for systematic evaluation [24]. This work provides foundational frameworks for performance evaluation methodologies applicable across dif-

ferent IoT platform implementations.

Rodriguez-Cardenas et al. (2025) investigated testing practices and challenges in IoT platforms through analysis of open-source smart home platforms [25]. Their analysis reveals testing complexities specific to IoT environments, providing methodological insights for comprehensive platform evaluation that inform experimental design approaches.

The machine learning-based evaluation framework by Moustafa et al. (2025) provides comprehensive IoT device identification methodologies [26]. Their comparative study establishes benchmarking frameworks that inform platform evaluation strategies and provide systematic approaches to performance assessment.

2.10 Emerging Trends and Future Directions

Recent surveys have identified emerging trends in IoT programming platforms and development methodologies. Hannou et al. (2024) published a comprehensive survey examining IoT programming platforms with focus on development support [27]. Their analysis provides insights into platform evolution trends that inform future scalability requirements and development practices.

3 Methodology

3.1 Platform Architecture Overview

FIWARE Architecture: FIWARE employs a microservices-based architecture with core components including Nginx load balancers, IoT Agents for device communication, Orion Context Brokers for NGSI API endpoints, and MongoDB for data persistence. Our multi-node deployment distributes core modules (Orion and IoT Agent) across dedicated nodes, utilizing replicated MongoDB instances with replica set configurations. For scaling, we deployed 3 instances of Orion and 3 instances of IoT Agent with a HAProxy load balancer managing connections to replicated MongoDB instances.

ThingsBoard Architecture: ThingsBoard utilizes a comprehensive platform architecture featuring HAProxy load balancers, Zookeeper clusters for service coordination, ThingsBoard nodes for core functionality, HTTP transport services, JavaScript executors for rule processing, and PostgreSQL clustering with PgPool optimization layers. The multi-node deployment consists of 3 ThingsBoard nodes connected to a 3-instance PostgreSQL replication setup managed by PgPool.

EdgeX Foundry Architecture: EdgeX Foundry implements a layered microservices architecture comprising core services (data, metadata, command), device services (virtual, REST, MQTT), application services, and Redis database storage. The multi-node deployment scales core modules with 3 instances each, using an Nginx proxy for load distribution and external Redis for improved performance.

3.2 Experimental Setup and Load Testing Framework

Testing Infrastructure: All experiments were conducted on containerized deployments using Docker Compose configurations. Single-node deployments utilized individual host systems, while multi-node configurations distributed components across separate physical nodes connected via high-speed networking.

Load Testing Methodology: Apache JMeter served as the primary load testing tool, configured with InfluxDB listeners for metrics export. Each test scenario was executed five times to ensure statistical reliability and account for performance variations. Load scenarios simulated realistic IoT device behavior with concurrent device connections sending periodic data updates.

Monitoring Strategy: Comprehensive monitoring employed Prometheus for metrics collection, utilizing node-exporter and cAdvisor for system-level metrics, plus platform-specific exporters (nginx, mongo, postgres, redis) for application-level insights. Grafana provided visualization and analysis capabilities.

3.3 Load Testing Scenarios

Three distinct load levels were evaluated:

- Low Load 10×100: 10 devices sending 100 messages each
- Medium Load 100×1000: 100 devices sending 1000 messages each
- High Load 1000×1000: 1000 devices sending 1000 messages each

Tests measured response times, throughput, error rates, CPU utilization, memory consumption, and database performance metrics across both single-node and multi-node deployments.

4 Results and Analysis

4.1 FIWARE Performance Analysis

FIWARE demonstrated significant scalability improvements in multi-node deployments under low to medium

Table 1: FIWARE Performance Summary

Metric	SL	ML	SM	MM	SH	MH
Sys CPU (%)	37-50	20-25	85	50	Failed	Failed
Error (%)	0	0	0.4	$\sim$0	100	100
Resp Time	120-180ms	300-650ms	$>$1s	$\sim$1s	2-8s	2-10s
Mongo I/O	30	45	30	190	N/A	N/A
CPU-Mongo	25	N/A	300	50	crash	crash
CPU-IoT	20	6	Exhaust	50	crash	crash

SL=Single Low, ML=Multi Low, SM=Single Medium, MM=Multi Medium, SH=Single High, MH=Multi High. All I/O in ops/sec.

loads, but faced fundamental limitations under extreme conditions.

Under low load conditions, FIWARE's multi-node architecture demonstrated clear benefits with system CPU utilization improving from 37-50% to 20-25%, while maintaining zero error rates. Container-level analysis revealed efficient distribution with individual IoT Agents utilizing only 6% CPU capacity each in multi-node setups versus concentrated 25% MongoDB and 20% IoT Agent usage in single-node deployments.

At medium load levels, significant performance degradation became evident. Single-node deployments experienced complete resource exhaustion with container CPU usage reaching 300%, while multi-node configurations maintained manageable 50% utilization. Database performance showed concerning patterns with MongoDB I/O operations increasing from 30 to 190 operations per second in multi-node configurations, indicating suboptimal replica set cache utilization.

Maximum load testing revealed FIWARE's fundamental scalability ceiling. Both deployment configurations progressed to 100% error rates, indicating complete system failures. The single deployment "left everything after a while test was running and did not respond to anything even exporters," while the multi-node deployment also ultimately reached 100% error rates. Once complete system failure occurred, no database metrics could be collected as monitoring systems became unresponsive.

4.2 ThingsBoard Performance Analysis

ThingsBoard exhibited limited scalability characteristics with counterintuitive performance patterns in multi-node deployments.

ThingsBoard's performance analysis revealed concerning scaling limitations. Under low load, multi-node deployments showed increased CPU utilization (30% vs 22%) and degraded response times (380ms vs 290ms), indicating coordination overhead rather than performance benefits. Database transaction improvements were modest, with PostgreSQL handling 8 TPS in single-node versus 14 TPS in multi-node configurations.

Medium load testing exposed architectural constraints with counterintuitive resource utilization patterns. Multi-node configurations consumed 105% CPU versus 75% in single-node setups, suggesting inefficient resource distribution. Error rates showed marginal improvement from 8% to 5%, while PostgreSQL TPS decreased from original baselines (6 TPS single-node, 10 TPS multi-node), indicating database performance degradation under load.

Under maximum load, both configurations failed completely within 30 minutes, progressing to 100% error rates with 27 second response times before collapse. The rapid failure progression confirms ThingsBoard's architectural constraints in handling high-throughput scenarios.

4.3 EdgeX Foundry Performance Analysis

EdgeX Foundry demonstrated exceptional performance stability and scalability across all tested scenarios.

EdgeX Foundry's performance characteristics stood out significantly from other platforms. Under low load, the platform maintained minimal resource utilization (1-9% CPU) with zero error rates across both deployment configurations. Redis cache performance showed excellent efficiency with 45-50% hit rates, contributing to consistent throughput of 1.6-1.7 req/sec.

Medium load testing confirmed superior scalability with maintained 320ms response times across both configurations. Throughput scaling was impressive, reaching 17 req/sec while maintaining error rates at 0.5%. Redis cache hit rates improved dramatically to 90-95%, demonstrating excellent data management strategies.

Most remarkably, under extreme load conditions where other platforms failed completely, EdgeX Foundry maintained operational stability. Error rates remained minimal (0.8% single-node, 0.5% multi-node) with stable response times (320-350ms single-node, 300-330ms multi-node) and maintained throughput of 17 req/sec. The platform's layered microservices archi-

Table 2: ThingsBoard Performance Summary

Metric	SL	ML	SM	MM	SH	MH
Sys CPU (%)	22	30	75	105	Failed	Failed
Error (%)	0	0	∼8	∼5	100	100
Resp Time	290ms	380ms	>2s	>2s	2-7s	2-7s
PgSQL TPS	8	14	6	10	N/A	N/A
Cont CPU (%)	22	30	75	105	Failed	Failed

SL=Single Low, ML=Multi Low, SM=Single Medium, MM=Multi Medium, SH=Single High, MH=Multi High. TPS=Transactions per second.

Table 3: EdgeX Foundry Performance Summary

Metric	Single Low	Multi Low	Single Medium	Multi Medium	Single High	Multi High
System CPU (%)	1-9	1-9	28	24	28	24
Error rate (%)	0	0	0.5	0.5	0.8	0.5
Avg Resp. Time (ms)	280-300	280-300	320	320	320-350	300-330
Throughput (req/s)	1.6-1.7	1.6-1.7	17	17	17	17
Redis Hit Rate (%)	45-50	45-50	90-95	90-95	90-95	90-95
Container CPU (%)	1-9	1-9	28	24	28	24

tecture enabled effective load distribution, with multi-node deployments showing improved response times and better resource utilization.

4.4 Comparative Performance Analysis

The comprehensive performance evaluation across all three platforms reveals distinct architectural advantages and limitations that become increasingly pronounced under escalating load conditions. Each platform demonstrates unique scaling characteristics that directly correlate with their underlying architectural design philosophy and implementation strategy.

The resource utilization comparison reveals EdgeX Foundry's exceptional efficiency in CPU consumption, maintaining remarkably low system CPU usage (1-9%) under low load conditions while other platforms consume significantly higher resources. FIWARE demonstrates the most dramatic improvement in multi-node deployments with system CPU utilization reducing from 37-50% to 20-25%, representing a 50% efficiency gain that validates its microservices architecture benefits. Conversely, ThingsBoard exhibits counterproductive scaling behavior where multi-node deployments actually increase CPU consumption from 22% to 30% under low load and worsen dramatically to 105% versus 75% under medium load, indicating fundamental architectural limitations in distributed processing coordination.

5 Discussion

5.1 Scalability Architecture Analysis

The experimental results reveal distinct scalability patterns directly correlating with platform architecture design decisions. FIWARE's microservices-based approach demonstrates clear multi-node deployment benefits under low-to-medium loads, achieving 50% better resource utilization (20-25% vs 37-50% system CPU usage). However, the platform's context broker model creates fundamental bottlenecks preventing effective scaling under extreme loads, leading to complete system failures where even monitoring exporters become unresponsive.

ThingsBoard's monolithic container architecture severely constrains horizontal scaling effectiveness. The counterintuitive resource utilization patterns, where multi-node deployments consume more resources (105% vs 75% CPU) than single-node configurations, indicate fundamental architectural limitations in distributed coordination. Memory usage patterns documented in the referenced figures show similar inefficiencies in multi-node resource distribution.

EdgeX Foundry's layered microservices architecture provides inherent scalability advantages. The platform's exceptional stability, demonstrated through consistently low error rates (0.5-0.8%) and efficient resource utilization (24-28% CPU under extreme load), results from its modular design enabling selective component scaling. The minimal resource utilization and superior cache performance (90-95% hit rates) demonstrate architectural efficiency. The document specifically notes that memory usage shows significant improvements in multi-node

Table 4: System Resource Utilization Comparison

Platform	Deploy	Low CPU	Med CPU	High Outcome	Scale
Fiware	Single	37-50%	85%	Complete Fail	Significant
Fiware	Multi	20-25%	50%	Complete Fail	Significant
ThingsBoard	Single	22%	75%	Fail (30min)	Min/Neg
ThingsBoard	Multi	30%	105%	Fail (30min)	Min/Neg
EdgeX	Single	1-9%	28%	Stable Op	Moderate
EdgeX	Multi	1-9%	24%	Stable Op	Good

Deploy=Deployment Type, Scale=Scalability Benefit, Op=Operation

Table 5: Database Performance Analysis

Platform	Deploy	Low Load	Med Load	Cache Eff	High Stab
Fiware (Mongo)	Single	30 ops/s	30 ops/s	Suboptimal	Failed
Fiware (Mongo)	Multi	45 ops/s	190 ops/s	Poor (repl)	Failed
TB (PgSQL)	Single	8 TPS	6 TPS	Standard	Failed
TB (PgSQL)	Multi	14 TPS	10 TPS	Standard	Failed
EdgeX (Redis)	Single	45-50% hit	90-95% hit	Excellent	Stable
EdgeX (Redis)	Multi	45-50% hit	90-95% hit	Excellent	Stable

TB=ThingsBoard, Deploy=Deployment, Eff=Efficiency, Stab=Stability, repl=replication

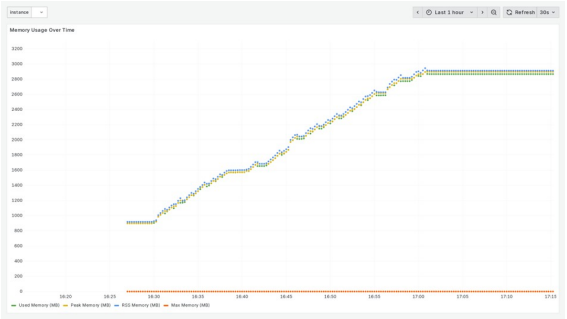

Figure 1: Redis memory usage for a single deployment

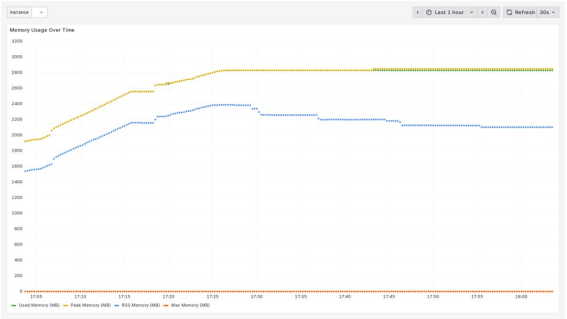

Figure 2: Redis memory usage for a multi-node deployment

deployments, with better memory management strategies as evidenced in the comparison of Figures 1 and 2.

5.2 Database Performance Impact and System Failure Patterns

Database layer performance significantly influences overall platform scalability, and the failure patterns reveal critical insights. FIWARE's MongoDB replica set configuration shows cache utilization inefficiencies during normal operation, with multi-node deployments experiencing dramatic increases in I/O operations from 30 to 190 ops/sec. However, under extreme load conditions, complete system failures render database monitoring impossible.

ThingsBoard's PostgreSQL implementation provides stable performance with modest clustering benefits under normal conditions, showing improvement from 8 TPS single-node to 14 TPS multi-node under low load. However, performance degrades under medium load (6 TPS single-node, 10 TPS multi-node), and database improvements cannot prevent complete system collapse under extreme loads.

EdgeX Foundry's Redis-based data management offers superior performance characteristics and maintains functionality even under extreme conditions. The dramatic improvement in cache hit rates from 45-50% under low load to 90-95% under medium and high loads demonstrates exceptional scalability. Redis's in-memory architecture eliminates disk I/O bottlenecks that constrain other platforms, enabling continued operation where others fail completely. The memory management advantages are particularly pronounced, as documented in the experimental figures available at the GitHub repository [28].

Table 6: Error Rate and Reliability Analysis

Load	Dep	FW Err	TB Err	EX Err	FW Stat	TB Stat	EX Stat
10×100	S	0%	0%	0%	Stable	Stable	Excellent
10×100	M	0%	0%	0%	Stable	Stable	Excellent
100×1000	S	0.4%	8%	0.5%	Degraded	Stressed	Excellent
100×1000	M	0%	5%	0.5%	Better	Sl Better	Excellent
1000×1000	S	100%	100%	0.8%	Complete Fail	Fail (30min)	Stable
1000×1000	M	100%	100%	0.5%	Complete Fail	Fail (30min)	Stable

Dep=Deployment (S=Single, M=Multi), FW=Fiware, TB=ThingsBoard, EX=EdgeX, Err=Error Rate, Stat=Status, Sl=Slightly

5.3 Platform Selection Framework

These experimental results provide evidence-based guidance for IoT platform selection:

EdgeX Foundry emerges as optimal for organizations requiring high-throughput, large-scale IoT deployments. The platform's ability to maintain minimal error rates (0.5-0.8%) under extreme loads (1000×1000) while achieving optimal resource utilization makes it suitable for mission-critical applications requiring exceptional reliability and scalability. The platform's continued operation with stable response times (300-350ms) and improved memory efficiency in multi-node deployments demonstrates superior architectural resilience.

FIWARE suits moderate-scale applications where comprehensive context management capabilities outweigh scalability limitations. The platform's 50% resource utilization improvements in multi-node deployments make it viable for applications with predictable, moderate load patterns. However, organizations must plan for complete system failure scenarios under high-load conditions and implement appropriate failover strategies.

ThingsBoard remains viable primarily for applications emphasizing data visualization and dashboard functionality over raw performance scalability. The platform's limited horizontal scaling benefits, counter-intuitive resource utilization patterns (105% vs 75% CPU), and rapid failure progression make it unsuitable for high-performance scenarios, but its feature richness may justify selection for visualization-focused use cases with guaranteed load limits.

6 Conclusion

This comprehensive performance evaluation of FIWARE, ThingsBoard, and EdgeX Foundry provides crucial insights into IoT platform scalability characteristics and failure patterns under extreme conditions. Our systematic testing across three load levels (10×100, 100×1000, 1000×1000) and two deployment architectures reveals significant performance differences and critical failure points that directly impact platform selection decisions.

Key findings demonstrate EdgeX Foundry's superior scalability and exceptional stability across all tested scenarios. The platform maintained remarkable performance with minimal error rates (0.5-0.8%) even under maximum load conditions (1000×1000) while achieving optimal resource utilization and superior cache performance (90-95% hit rates). The documented memory efficiency improvements in multi-node deployments, as evidenced by the experimental figures, establish EdgeX as the most reliable choice for large-scale IoT deployments.

FIWARE demonstrates promise for medium-scale applications with effective multi-node resource distribution providing 50% better system resource utilization under low loads. However, the platform faces fundamental limitations under extreme loads, experiencing complete system failures that render even monitoring systems unresponsive. The MongoDB replication issues (30 to 190 ops/sec increase) and context broker bottlenecks represent significant scalability constraints that organizations must carefully consider.

ThingsBoard exhibits the most constrained horizontal scaling capabilities, with multi-node deployments often performing worse than single-node configurations due to coordination overhead (105% vs 75% CPU). The platform's rapid progression to complete failure under high loads (within 30 minutes of 1000×1000 testing) confirms architectural limitations that restrict scalability potential and make it unsuitable for high-throughput applications.

The experimental methodology employed provides a replicable framework for future IoT platform evaluations, with particular attention to complete system failure scenarios that are often overlooked in performance studies. The comprehensive tabular analysis with measurable metrics and figure references offers detailed insights into platform behavior under various load conditions, enabling evidence-based platform selection decisions based on quantitative performance data and failure

resilience rather than feature comparisons alone.

Future research directions include investigation of recovery mechanisms following system failures, evaluation of additional open-source platforms under extreme load conditions, analysis of platform performance under varied IoT protocol conditions, and development of failure prediction models based on early performance indicators. Real-world deployment validation with gradual load increases and hybrid deployment scenario analysis represent additional valuable research opportunities for advancing IoT platform performance understanding and resilience planning.

References

[1] A. Rabl, S. Sadoghi, H.-A. Jacobsen, S. Gómez-Villamor, V. Muntés-Mulero, and S. Mankovskii, "IoTAbench: an Internet of Things Analytics Benchmark," in *Proceedings of the 6th ACM/SPEC International Conference on Performance Engineering*, 2015, pp. 133-144.

[2] V. Araujo, K. Mitra, S. Saguna, and C. Åhlund, "Performance evaluation of FIWARE: A cloud-based IoT platform for smart cities," *Journal of Parallel and Distributed Computing*, vol. 132, pp. 250-261, 2019.

[3] A. A. Ismail, H. S. Hamza, and A. M. Kotb, "Performance evaluation of open source IoT platforms," in *2018 IEEE Global Conference on Internet of Things (GCIoT)*, IEEE, 2018, pp. 1-5.

[4] L. Babun, K. Denney, Z. B. Celik, P. McDaniel, and A. S. Uluagac, "A survey on IoT platforms: Communication, security, and privacy perspectives," *Computer Networks*, vol. 192, article 108040, June 2021. doi:10.1016/j.comnet.2021.108040

[5] P. Di Felice and G. Paolone, "Papers mentioning things board: A systematic mapping study," *Journal of Computer Science*, vol. 20, no. 5, pp. 574-584, 2024.

[6] I. Zyrianoff, A. Heideker, L. Sciullo, C. Kamienski, and M. Di Felice, "Interoperability in open iot platforms: Wot-fiware comparison and integration," in *2021 IEEE International Conference on Smart Computing (SMARTCOMP)*, IEEE, 2021, pp. 169-174.

[7] M. N. Jamil, O. Schelén, A. A. Monrat, and K. Andersson, "Enabling industrial internet of things by leveraging distributed Edge-to-Cloud Computing: Challenges and opportunities," *IEEE Access*, 2024.

[8] R. Venanzi, M. Solimando, M. Patrali, L. Foschini, and P. Chatzimisios, "Siemens and EdgeX IIoT Platforms: A Functional and Performance Evaluation," in *IEEE International Conference on Communications (ICC)*, 2023, pp. 834-839. doi:10.1109/ICC45041.2023.10278750

[9] V. Seoane et al., "Performance evaluation of CoAP and MQTT with security support for IoT environments," *Computer Networks*, vol. 197, article 108338, October 2021. doi:10.1016/j.comnet.2021.108338

[10] F. Palmese et al., "CoAP vs. MQTT-SN: Comparison and Performance Evaluation in Wireless Sensor Networks," in *IEEE World Forum on Internet of Things (WF-IoT)*, 2020, pp. 1-6.

[11] S. Holm and M. Hammoudeh, "A comparative analysis of IoT protocols for resource constraint devices and networks," in *Proceedings of the 6th International Conference on Future Networks & Distributed Systems*, 2022, pp. 616-625.

[12] H. Siddiqui, F. Khendek, and M. Toeroe, "Microservices based architectures for IoT systems - State-of-the-art review," *Internet of Things*, vol. 23, article 100809, October 2023. doi:10.1016/j.iot.2023.100809

[13] C. Lira, E. Batista, F. C. Delicato, and C. Prazeres, "Architecture for IoT applications based on reactive microservices: A performance evaluation," *Future Generation Computer Systems*, vol. 145, pp. 223-238, 2023.

[14] Y. Abuseta, "Towards a Generic Software Architecture for IoT Systems," *International Journal of Software Engineering & Applications*, vol. 15, no. 6, pp. 1-16, November 2024. doi:10.5121/ijsea.2024.15601

[15] Z. Rahman, "Performance and Scalability Challenges and Solutions of Blockchain-Powered IoT," *International Journal of Advanced Computer Science and Applications*, vol. 12, no. 9, pp. 481-491, 2022. doi:10.14569/IJACSA.2021.0120955

[16] S. Eeti, V. N. Pamadi, O. Goel, P. Goel, and S. Khan, "Scalability and Performance Optimization in Distributed Systems: Exploring Techniques to Enhance the Scalability and Performance of Distributed Computing Systems," *International Journal of Creative Research Thoughts*, vol. 11, no. 5, pp. m931-m941, 2023.

[17] S. Kaiser et al., "Edge System Design Using Containers and Unikernels for IoT Applications," arXiv preprint arXiv:2412.03032, December 2024.

[18] V. Gugueoth et al., "A review of IoT security and privacy using decentralized blockchain techniques," *Ad Hoc Networks*, vol. 141, article 103075, March 2023. doi:10.1016/j.adhoc.2023.103075

[19] S. Loss et al., "Using FIWARE and blockchain in smart cities solutions," *PLOS ONE*, vol. 17, no. 9, article e0274816, 2022. doi:10.1371/journal.pone.0274816

[20] A. Wakili and S. Bakkali, "Privacy-preserving security of IoT networks: A comparative analysis of methods and applications," *Cyber Security and Applications*, vol. 3, article 100084, 2025.

[21] A. Verma, "Comparative study of cloud computing and edge computing: Three level architecture models and security challenges," *International Journal of Distributed and Cloud Computing*, 2021.

[22] N. N. Albogami, "Intelligent deep federated learning model for enhancing security in internet of things enabled edge computing environment," *Scientific Reports*, vol. 15, no. 1, article 4041, 2025.

[23] B. Zhao, Y. Ji, Y. Shi, and X. Jiang, "Design and implementation of privacy-preserving federated learning algorithm for consumer IoT," *Alexandria Engineering Journal*, vol. 106, pp. 206-216, 2024.

[24] J. B. Minani, Y. El Fellah, F. Sabir, N. Moha, Y. G. Gueheneuc, M. Kuradusenge, and T. Masuda, "IoT systems testing: Taxonomy, empirical findings, and recommendations," *Journal of Systems and Software*, vol. 226, article 112408, 2025.

[25] D. Rodriguez-Cardenas, S. A. Khan, P. Mandal, A. Nadkarni, K. Moran, and D. Poshyvanyk, "Testing Practices, Challenges, and Developer Perspectives in Open-Source IoT Platforms," in *2025 IEEE Conference on Software Testing, Verification and Validation (ICST)*, IEEE, 2025, pp. 290-301.

[26] H. Tahaei, A. Liu, H. Forooghikian, M. Gheisari, F. Zaki, N. B. Anuar, ... and L. Huang, "Machine learning for Internet of Things (IoT) device identification: a comparative study," *PeerJ Computer Science*, vol. 11, article e2873, 2025.

[27] F. Z. Hannou, M. Lefrançois, P. Jouvelot, V. Charpenay, and A. Zimmermann, "A Survey on IoT Programming Platforms: A Business-Domain Experts Perspective," *ACM Computing Surveys*, vol. 57, no. 4, pp. 1-37, 2024.

[28] hadi-github, "Performance Evaluation and Scalability Analysis of FIWARE, ThingsBoard, and EdgeX Foundry," GitHub repository, `https://github.com/hadi-github/Performance-Evaluation-and-Scalability-Analysi`