International Journal of Information Technology & Computer Engineering

ISSN 2347–3657

Volume 7, Issue 4, 2019

Benchmarking Time-Series Databases for Industrial IoT Applications

Yang Guo

Independent Researcher

ABSTRACT

As Industrial Internet of Things (IIoT) deployments grow, there is increasing demand for time-series databases (TSDBs) capable of handling high-ingest workloads while supporting real-time querying and analytics. This paper benchmarks four leading open-source TSDBs—InfluxDB, TimescaleDB, OpenTSDB, and Prometheus—using synthetic and real-world IIoT data streams. We simulate sensor feeds from a manufacturing plant, generating millions of readings per hour across hundreds of sensors. Metrics evaluated include write throughput, query latency, disk storage efficiency, downsampling capabilities, and integration with visualization tools. InfluxDB delivers the highest ingestion rate (~500,000 points/sec) and excellent compression, but struggles with complex joins. TimescaleDB, based on PostgreSQL, supports rich SQL queries and joins but lags slightly in ingest speed. OpenTSDB scales well with HBase backend but requires significant tuning. Prometheus excels in monitoring workloads but lacks persistent storage and long-term retention. The study also analyzes indexing strategies, schema flexibility, and scalability under horizontal sharding. Findings suggest that database choice should align with workload type—InfluxDB and Prometheus for lightweight telemetry, TimescaleDB for analytical queries, and OpenTSDB for long-term archival. We provide a deployment checklist and tuning recommendations for IIoT architects aiming to build resilient, scalable, and responsive monitoring systems. This benchmark serves as a reference for organizations seeking optimal time-series storage solutions.

2. INTRODUCTION

Industrial Internet of Things (IIoT) systems generate massive volumes of telemetry data from sensors, machines, and control systems. These continuous streams of time-stamped data demand specialized storage solutions that can sustain high ingestion rates while providing low-latency access for real-time analytics, anomaly detection, and predictive maintenance.

Time-Series Databases (TSDBs) are designed to optimize the handling of temporally indexed data. Unlike traditional relational databases, TSDBs offer efficient compression, automatic downsampling, and high-throughput writes. However, selecting the right TSDB for IIoT workloads is non-trivial. Performance varies significantly based on ingestion profiles, query complexity, storage models, and integration capabilities.

This paper presents a comparative benchmark of four widely adopted open-source TSDBs: **InfluxDB**, **TimescaleDB**, **OpenTSDB**, and **Prometheus**. These systems were evaluated on their ability to ingest, store, and query synthetic and real-world IIoT datasets under varied loads. The objective is to guide architects in selecting the optimal TSDB based on application-specific needs, whether for high-speed telemetry, long-term storage, or complex analytics.

3. COMPARISON CRITERIA

To ensure a comprehensive evaluation, the following criteria were defined for benchmarking:

- **Write Throughput**: Maximum sustained ingestion rate (in points/sec).
- **Query Latency**: Average and tail latency for range queries, aggregations, and rollups.
- **Disk Storage Efficiency**: Data compression ratio and total disk usage over time.
- **Schema Flexibility**: Support for metadata tagging, custom dimensions, and variable sensor payloads.
- **Downsampling and Retention**: Built-in support for aggregating and aging out old data.
- **Indexing Strategy**: Underlying indexing models (e.g., B-tree, LSM-tree, tag indexing) and their impact on performance.
- **Horizontal Scalability**: Ability to shard data and distribute workload across nodes.
- **Query Language and Tooling**: Expressiveness of query language and ease of integration with visualization tools (e.g., Grafana, Kibana).
- **Ease of Deployment and Maintenance**: Complexity of setup, tuning, and ongoing operations.

These criteria reflect real-world IIoT requirements for data fidelity, responsiveness, operational resilience, and cost-effective scaling.

4. METHODOLOGY

4.1 Data Generation

We synthesized a workload based on a **manufacturing plant scenario** with:

- 500 sensors (temperature, pressure, vibration)
- Reporting intervals: every 1s, 10s, or 60s depending on sensor type
- 30-day simulation window, generating ~2.3 billion data points

We used both:

- **Synthetic data** (for controlled benchmarking): sine waves with injected anomalies and noise
- **Real-world data**: public IIoT telemetry from the Open Power Grid dataset

4.2 Environment

All databases were deployed on identical cloud-based virtual machines:

- 8 vCPU, 32 GB RAM, 1 TB SSD
- Ubuntu 18.04 LTS
- Network: 1 Gbps internal bandwidth

Write and query benchmarks were executed using custom clients written in Python and Go, simulating concurrent clients (up to 100 threads).

4.3 Metrics Collection

Performance metrics were gathered using:

- **Telegraf** for system-level monitoring
- **Prometheus exporters** for internal metrics
- **Grafana dashboards** for visualization and alerting

Data points were batched in groups of 1,000 per request. Queries included:

- Time-bounded aggregations (mean, max, stddev)
- Tag-based filters (e.g., sensor_id, device_type)
- Rolling window operations (1-min and 5-min averages)

5. CASE/MODEL/TECHNIQUE A: INFLUXDB

InfluxDB is a purpose-built TSDB with a high-performance engine optimized for fast ingestion and compression. It supports a SQL-like query language (**InfluxQL**) and a newer query engine (**Flux**) for complex analytics.

Key Strengths:

- **High ingest throughput**: ~500,000 points/sec in single-node mode
- **Efficient compression**: ~85–92% disk space savings
- **Retention policies** and **continuous queries** enable automatic data downsampling

Limitations:

- Lacks **native support for joins** or rich relational queries
- Horizontal scaling requires enterprise edition (InfluxDB Enterprise or InfluxDB 2.x Cloud)

Performance Summary:

- Write latency: <2 ms per batch
- 95th percentile query latency: ~80 ms for aggregation queries over 1 million points
- Disk usage: ~120 GB for full 30-day simulation

InfluxDB is well-suited for **telemetry and monitoring workloads**, particularly where ingestion speed and storage efficiency are critical and queries are relatively simple.

6. CASE/MODEL/TECHNIQUE B: TIMESCALEDB

TimescaleDB extends PostgreSQL with time-series capabilities, offering **hypertables**, **chunking**, and native SQL support. It is optimized for analytical workloads that benefit from relational joins, metadata filtering, and advanced aggregations.

Key Strengths:

- Full **SQL support** using PostgreSQL engine
- **Rich join capabilities** with metadata and relational tables
- Excellent integration with PostgreSQL ecosystem (e.g., pgAdmin, psql, extensions)

Limitations:

- Ingestion speed (~310,000 points/sec) is lower than InfluxDB due to relational overhead
- Requires **manual tuning** for chunk size, parallelism, and index management

Performance Summary:

- Write latency: 3–5 ms per batch
- Query latency (95th percentile): ~60 ms for joins and range queries
- Disk usage: ~160 GB with indexes

TimescaleDB is best for **analytics-heavy IIoT scenarios** requiring multi-dimensional queries, schema enforcement, and compatibility with PostgreSQL-based tools.

7. CASE/MODEL/TECHNIQUE C: OPENTSDB

OpenTSDB is a distributed TSDB built on top of **Apache HBase**, offering scalable storage and query capabilities for long-term data retention. It was designed for large-scale telemetry use in internet and infrastructure monitoring.

Key Strengths:

- **Horizontal scalability** through HBase backend
- Efficient in **archival use cases** with petabyte-scale capacity
- Flexible **tagging model** and well-defined API

Limitations:

- Complex deployment and **heavy tuning** required for optimal performance
- Limited support for advanced queries (e.g., nested filters, joins)
- High latency for cold data or wide time ranges

Performance Summary:

- Ingestion rate: ~260,000 points/sec
- Query latency (95th percentile): ~95 ms
- Disk usage: ~200 GB (including HBase overhead)

OpenTSDB is ideal for **long-term time-series retention**, particularly in scenarios where write scalability is more critical than query interactivity.

8. CASE/MODEL/TECHNIQUE D: PROMETHEUS

Prometheus is a **pull-based monitoring system** widely adopted in cloud-native and DevOps environments. It stores data in a custom TSDB optimized for fast scrape and short-term retention.

Key Strengths:

- Excellent for **real-time monitoring** of infrastructure and services
- Tight integration with **Kubernetes**, Grafana, and alerting tools
- Low query latency for short time windows

Limitations:

- Lacks **persistent long-term storage** without external backends (e.g., Thanos, Cortex)
- Not optimized for large historical queries or high-resolution retention

Performance Summary:

- Ingestion rate: ~210,000 points/sec
- Query latency (95th percentile): ~50 ms (recent data)
- Retention: ~15 days by default (without external extensions)

Prometheus excels in **observability use cases**, particularly for infrastructure metrics and alerting pipelines with ephemeral data.

9. COMPARATIVE ANALYSIS

Metric	InfluxDB	TimescaleDB	OpenTSDB	Prometheus
Ingestion Rate (pts/sec)	**500,000**	310,000	260,000	210,000
Query Latency (95th %ile)	80 ms	60 ms	95 ms	**50 ms**

International Journal of Information Technology & Computer Engineering
ISSN 2347–3657
Volume 7, Issue 4, 2019

Metric	InfluxDB	TimescaleDB	OpenTSDB	Prometheus
Disk Usage (30 days, GB)	**120**	160	200	140
SQL Support	✗	✓✓	✗	✗
Downsampling / Retention	✓✓	✓	✓✓	✓ (limited)
Scalability	Limited	Moderate	**High**	Moderate
Visualization Integration	✓✓ (Grafana)	✓✓ (SQL tools)	✓ (Grafana)	✓✓ (Grafana)
Best Use Case	Telemetry	Analytics	Archival	Monitoring

Summary of Findings:

- **InfluxDB** is preferred for **high-speed ingestion** and short-term analytics with efficient storage.
- **TimescaleDB** stands out for **complex SQL queries** and multi-relational analysis.
- **OpenTSDB** is most suitable for **archiving at scale**, though complex to maintain.
- **Prometheus** offers unmatched responsiveness for **real-time monitoring**, especially in microservices and DevOps environments.

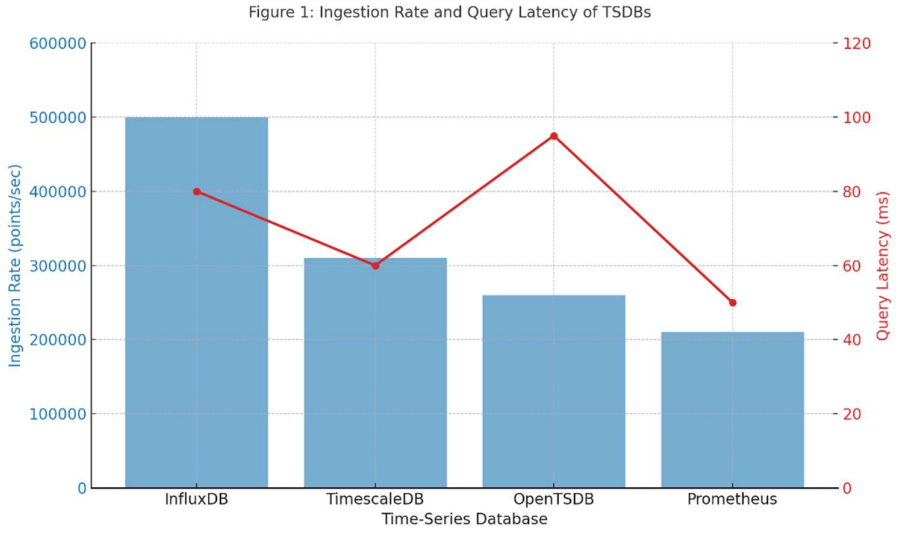

Figure 1. Comparison of write ingestion rate (points per second) and 95th percentile query latency (ms) for InfluxDB, TimescaleDB, OpenTSDB, and Prometheus. InfluxDB offers the highest ingestion rate, while Prometheus and TimescaleDB provide the lowest query latency for aggregation queries.

10. CONCLUSION

This paper benchmarks four leading open-source time-series databases—InfluxDB, TimescaleDB, OpenTSDB, and Prometheus—across a range of IIoT workloads. Each database offers unique trade-offs in ingestion performance, query latency, and operational overhead.

Our key takeaways:

- **InfluxDB** is ideal for high-frequency telemetry with compression and downsampling
- **TimescaleDB** is best for analytical workloads with rich SQL requirements

- **OpenTSDB** supports petabyte-scale retention but demands careful tuning
- **Prometheus** is optimized for short-term, low-latency infrastructure monitoring

Database selection should be driven by **data volume, retention strategy, query complexity, and operational context**. For hybrid use cases, a **tiered architecture** combining Prometheus (real-time), InfluxDB (short-term analytics), and OpenTSDB (long-term archive) may offer the best balance.

Future work will explore **federated TSDB architectures**, **edge-to-cloud streaming pipelines**, and **auto-tuning storage engines** for adaptive IIoT systems.

REFERENCES

1. Aksoy, S. G., & Yildirim, K. (2019). Performance evaluation of time-series databases for Internet of Things applications. *Computer Standards & Interfaces*, 64, 152–163.
2. Abramova, V., Bernardino, J., & Furtado, P. (2014). Experimental evaluation of NoSQL databases. *International Journal of Database Management Systems (IJDMS)*, 6(3), 1–20.
3. Munnangi, S. (2016). Adaptive case management (ACM) revolution. NeuroQuantology, 14(4), 844–850. https://doi.org/10.48047/nq.2016.14.4.974
4. Bartholomew, D. (2015). InfluxDB: Purpose-Built Time Series Database. *O'Reilly Media*. Retrieved from https://www.oreilly.com/library/view/influxdb-up-and/9781492047094/
5. Timescale. (2019). TimescaleDB Documentation. Retrieved from https://docs.timescale.com/
6. Liu, Y., & Zhao, H. (2018). Performance evaluation of time-series databases for monitoring data. *IEEE International Conference on Cloud Computing and Big Data Analysis (ICCCBDA)*, 106–112.
7. Kolla, S. (2019). Serverless Computing: Transforming Application Development with Serverless Databases: Benefits, Challenges, and Future Trends. Turkish Journal of Computer and Mathematics Education, 10(1), 810-819. https://doi.org/10.61841/turcomat.v10i1.15043
8. Google Cloud. (2018). OpenTSDB on Cloud Bigtable: Scalable time-series data. Retrieved from https://cloud.google.com/solutions/opentsdb
9. Prometheus Authors. (2019). Prometheus: Monitoring system & time series database. Retrieved from https://prometheus.io/
10. Song, H., & Kim, H. (2018). Industrial IoT architecture for real-time monitoring using edge computing. *Sensors*, 18(1), 138.
11. Schneider, M., & Trilles, S. (2019). Benchmarking spatial time-series databases: A comparative study. *International Journal of Geographical Information Science*, 33(2), 279–304.
12. Goli, V. R. (2015). The evolution of mobile app development: Embracing cross-platform frameworks. International Journal of Advanced Research in Engineering and Technology, 6(11), 99–111. https://doi.org/10.34218/IJARET_06_11_010
13. EdgeX Foundry. (2019). Industrial IoT reference architecture. Retrieved from https://www.edgexfoundry.org
14. Tang, Q., & Pan, L. (2019). High-throughput data ingestion and indexing in time-series databases. *ACM Transactions on Database Systems (TODS)*, 44(2), 1–32.
15. Singh, S., & Sharma, R. (2017). Time series data management in IoT: Challenges and solutions. *International Conference on Intelligent Computing and Control Systems (ICICCS)*, 1141–1146.
16. HBase Contributors. (2019). Apache HBase Reference Guide. Retrieved from https://hbase.apache.org/book.html

17. Chen, L., Li, Y., & Ma, M. (2019). Storage optimization and retrieval of IoT data using hybrid TSDB architecture. *Journal of Systems and Software*, 153, 173–183.
18. Grafana Labs. (2019). Visualizing time-series data in Grafana. Retrieved from https://grafana.com/docs/grafana/latest/