# Benchmarking and Evaluating Time Series Databases for Appliance-Level Energy Consumption Data

Simin Shehbaz
*Faculty of Computer Science*
*University of New Brunswick*
Fredericton, Canada
simin.shehbaz@unb.ca

Mohammad Mehabadi
*Faculty of Computer Science*
*University of New Brunswick*
Fredericton, Canada
m.mehabadi@unb.ca

Kenneth B. Kent
*Faculty of Computer Science*
*University of New Brunswick*
Fredericton, Canada
ken@unb.ca

***Abstract*—Time series databases (TSDBs) are widely used to store high-frequency energy consumption data, but their performance varies depending on workload characteristics. This paper benchmarks leading TSDBs to identify their suitability to handle appliance-level, per-minute energy data. While prior work has evaluated TSDBs, limited research has been done on TSDBs for wide-format, fine-grained residential energy data at the appliance level. We introduce a custom-generated dataset simulating the usage of twenty-six appliances across five household types in a wide-format schema. Building on the TSM-Bench framework, we adapt it to support our appliance-level dataset, domain-specific workloads, and evaluation metrics. We analyze ingestion and query performance across three TSDBs that support this wide-format, while highlighting the trade-offs in latency, resource usage, throughput and storage.**

**Looking ahead, we plan to evaluate schema transformations (wide to narrow), explore additional TSDBs optimized for narrow-format ingestion and compare their performance against wide-format results. These measures aim to provide a comprehensive and format-aware comparison of TSDBs.**

***Index Terms*—time series databases, benchmarking, IoT, energy consumption, benchmark workloads, ingestion & query performance**

## I. Introduction

The rise of smart homes, Internet of Things (IoT)-enabled devices, and renewable energy systems have made **appliance-level energy consumption data** a critical resource for building intelligent infrastructure. Capturing this data at **minute-level frequency** from dozens of appliances produces large-scale, multi-dimensional time series data—demanding efficient storage, querying, and analysis. Time series databases (TSDBs) are essential for managing large volumes of time-stamped data with **low-latency read and write performance**. They are preferred over general-purpose databases in real-time monitoring applications, due to their **scalability, compression efficiency**, and native support for streaming analytics [1]. TSDBs use time-based indexing, columnar storage, down-sampling strategies, and offer built-in support for temporal queries such as sliding windows, aggregations, and time filters [1]. Unlike traditional relational databases, TSDBs are optimized for **high ingestion throughput** and storage efficiency [1], [2].

A crucial design choice when storing time series data is the schema format—wide vs. narrow. In the **wide format**, each row contains multiple sensor readings for a single timestamp. In contrast, the **narrow format** stores one measurement per row, requiring additional columns for the sensor identifier and its value. Wide-format schemas are typically better suited for high-dimensional sensor data where all sensors report at the same interval, enabling faster cross-sensor operations and improved compression due to columnar locality [3]. However, they may be less flexible for variable-frequency streams or semi-structured data, where narrow formats are more common [2].

In the context of the appliance-level energy data that we are addressing in this paper, typical analyses on this data would include: **trends over time** (e.g., how the heater consumption varies seasonally), **anomaly detection** that identifies surges in energy readings (e.g., devices left on), **device-level usage comparison**, and checking for **peak demands**, showing the critical role of TSDBs in analytics.

Existing benchmarks have not evaluated systems under appliance-level energy workloads. They have focused on specific domains like financial transactions, smart buildings, and industrial instruments [4]–[6]. TSM-Bench comes close in scope, and caters to wide-formatted data [1]. Hence we extend this framework, tailoring it to our fine-grained, appliance-level, energy dataset that exhibits: high dimensionality, wide-format rows, frequent data points (minute-level), and realistic workloads.

Every testing system is optimized around trade-offs made in storage formatting, indexing, compression and query execution [1]–[3]. Therefore, this work presents a comprehensive benchmarking framework for appliance-level energy data that extends the TSM-Bench, while making the following contributions:

- Developing a benchmarking framework tailored to appliance-level energy consumption, evaluating both ingestion and query performance.
- Generating data workloads using **'DGSim'**, a custom simulator that simulates realistic energy consumption

patterns for 26 appliances across five household types, producing wide-format, minute-level time series data [7].
- Defining a varying set of workload strains including write-heavy streams, range queries, time aggregations (daily/monthly), and appliance-level comparisons.
- Benchmarking three leading TSDBs: TimescaleDB, QuestDB and ClickHouse.

The remainder of the paper is structured as follows: Section II reviews related work on TSDB benchmarking and appliance-level datasets. Section III presents the study setup, Section **??** details the benchmarking framework. Section **??** reports and analyzes the experimental results. Section **??** concludes the paper, and Section **??** outlines future research directions.

## II. Related Work

Numerous benchmarks have been proposed to evaluate TSDBs, each tailored to specific application domains and workload characteristics.

**TSM-Bench** is among the most comprehensive benchmarks for TSDBs used in monitoring environments [1]. It introduces offline and online workload tiers, a suite of fundamental time-based queries, and a novel Generative Adversarial Network (GAN)-based data generator (TS-LSH) tailored specifically for hydrological data. While powerful, its design is domain-specific and does not generalize well to energy datasets. **TSBS** (Time Series Benchmark Suite) is widely used in industry to compare storage size, loading time, and aggregation performance across systems like InfluxDB and TimescaleDB [4]. However, TSBS is heavily geared toward finance and DevOps metrics, and lacks native support for wide-format ingestion and complex device-level analytics [4]. **SciTS** benchmarks TSDBs for scientific instrumentation use cases. It uses synthetic data to evaluate ingestion and query latency [6]. SmartBench evaluates data systems for smart buildings [5]. However, it limits its scope to correlation queries and basic filtering on noisy datasets. **IoTDB-Benchmark** focuses on heterogeneous IoT data with different sensor types and frequencies [8]. It evaluates ingestion throughput and query performance but assumes narrow-format, homogeneous time series and does not support appliance-level complexity. **IoTAbench** was proposed as a benchmark for IoT analytics platforms and uses real-world sensor data to assess ingestion and analytics performance [9]. While highlighting power usage scenarios, it uses simplified event models and focuses on system-level throughput over schema variations. **TS-Benchmark** is tailored to benchmark electricity data from wind turbines, while incorporating realistic workloads including anomaly detection and multi-sensor fetches. However, it is limited to offline evaluation and assumes a relatively simple data layout patterns [10]. An important system-level study is the encoding analysis for **Apache IoTDB**, which compares time series compression formats under various characteristics including data repetition, sparsity, and delta distribution [11]. This study indirectly evaluates performance on wide-format data by examining columnar compression effects, making it relevant for storage strategy comparisons.

Collectively, these benchmarks have advanced TSDB evaluation. However, none offer an end-to-end assessment for appliance-level energy data, which requires support for wide and narrow schema comparisons, multi-sensor queries, and concurrent streaming and analytics—core demands in smart energy systems.

## III. Experimental Setup

This section outlines the data generation process, data characteristics, and the process of selecting TSDBs that are used in this study.

### A. Data Generation

The dataset used in this study is generated using **"DGSim"**, developed in prior work under this study [7]. This models appliance-level energy consumption based on the usage patterns of various household appliances. It has been designed to model realistic energy consumption patterns by appliances including those of heating, ventilation, and air conditioning (HVAC) systems, refrigerators, and washing machines. DGSim adopts a **Markov, state-based modeling technique**, where each appliance is modeled as a probabilistic state machine [12]. This simulator provides a scalable, reproducible environment for generating data as required by the experiment. DGSim originally generated aggregated energy consumption profiles for a large number of households. For the purpose of this benchmarking study, the simulator was modified to output disaggregated, appliance-level energy consumption data for individual households.

### B. Data Characteristics

The generated dataset follows a wide format, where each row represents a timestamp and each column corresponds to a specific appliance. It captures minute-level energy consumption for twenty-six appliances over a user-defined time range. DGSim supports five distinct household types: single-family attached, single-family detached, mobile home, two-to-four room apartment and apartments with more than five rooms. Each of these have different consumption patterns. For this study, datasets were exported as CSV files prior to ingestion. Table I summarizes the structure and attributes of the dataset.

TABLE I
Characteristics of the Appliance-Level Energy Consumption Dataset

| Characteristic | Description |
|---|---|
| Time Granularity | 1-minute intervals |
| Time Range | User-defined |
| Appliances Modeled | 26 appliances |
| Number of Columns | 30 columns |
| Dataset Format | Wide format |
| House Types | 5 distinct types |
| Operations | Simulate $\rightarrow$ Insert $\rightarrow$ Query |
| Storage Format | CSV (pre-ingestion), TSDB (post-ingestion) |

### C. TSDBs' Inclusion and Exclusion Criteria

We have defined a set of inclusion and exclusion criteria to select the appropriate TSDBs for this benchmarking study. The criteria are as follows:

*1) Inclusion Criteria:*

- Native support for time-series data, indexing, and storage
- Open-source or free for academic use
- Ability to ingest wide and narrow format data
- Support for time-based queries, filtering, and aggregation
- Native support for Docker-based deployment

*2) Exclusion Criteria:*

- Unmaintained or inactive projects, or projects that have had no updates in over a year

### D. Selected TSDBs for Benchmark Study

Starting with a list of thirty-one TSDBs identified through literature and community sources, we applied the above criteria to narrow the selection to four candidates. ClickHouse is a high-performance columnar database with strong support for wide schemas. Its **MergeTree** engine enables high ingestion rates through **delta encoding** and **Single Instruction Multiple Data (SIMD)-optimized scans** [3]. TimescaleDB, built on PostgreSQL, stores time series data in row-oriented **hypertables**, partitioned by time. It handles wide rows effectively by using array-based storage and **chunk-level compression** [2]. QuestDB is a lightweight TSDB using column-store layout with time-partitioned tables and **memory-mapped I/O**. It supports wide schemas but lacks advanced compression, leading to a larger storage footprint [13]. Although flexible and JSON-friendly, CrateDB has an inverted index structure and is inefficient for wide-schema ingestion with frequent updates [14]. During preliminary testing, it showed significant bottlenecks, and was excluded from comparative analysis. The final TSDBs selected were: **TimescaleDB**, **QuestDB**, and **ClickHouse**.

### E. Units

- Use either SI (MKS) or CGS as primary units. (SI units are encouraged.) English units may be used as secondary units (in parentheses). An exception would be the use of English units as identifiers in trade, such as "3.5-inch disk drive".
- Avoid combining SI and CGS units, such as current in amperes and magnetic field in oersteds. This often leads to confusion because equations do not balance dimensionally. If you must use mixed units, clearly state the units for each quantity that you use in an equation.
- Do not mix complete spellings and abbreviations of units: "Wb/m$^2$" or "webers per square meter", not "webers/m$^2$". Spell out units when they appear in text: ". . . a few henries", not ". . . a few H".
- Use a zero before decimal points: "0.25", not ".25". Use "cm$^3$", not "cc".)

### F. Equations

Number equations consecutively. To make your equations more compact, you may use the solidus ( / ), the exp function, or appropriate exponents. Italicize Roman symbols for quantities and variables, but not Greek symbols. Use a long dash rather than a hyphen for a minus sign. Punctuate equations with commas or periods when they are part of a sentence, as in:

$$a + b = \gamma \tag{1}$$

Be sure that the symbols in your equation have been defined before or immediately following the equation. Use "(1)", not "Eq. (1)" or "equation (1)", except at the beginning of a sentence: "Equation (1) is . . ."

### G. LaTeX-Specific Advice

Please use "soft" (e.g., `\eqref{Eq}`) cross references instead of "hard" references (e.g., `(1)`). That will make it possible to combine sections, add equations, or change the order of figures or citations without having to go through the file line by line.

Please don't use the `{eqnarray}` equation environment. Use `{align}` or `{IEEEeqnarray}` instead. The `{eqnarray}` environment leaves unsightly spaces around relation symbols.

Please note that the `{subequations}` environment in LaTeX will increment the main equation counter even when there are no equation numbers displayed. If you forget that, you might write an article in which the equation numbers skip from (17) to (20), causing the copy editors to wonder if you've discovered a new method of counting.

BibTeX does not work by magic. It doesn't get the bibliographic data from thin air but from .bib files. If you use BibTeX to produce a bibliography you must send the .bib files.

LaTeX can't read your mind. If you assign the same label to a subsubsection and a table, you might find that Table I has been cross referenced as Table IV-B3.

LaTeX does not have precognitive abilities. If you put a `\label` command before the command that updates the counter it's supposed to be using, the label will pick up the last counter to be cross referenced instead. In particular, a `\label` command should not go before the caption of a figure or a table.

Do not use `\nonumber` inside the `{array}` environment. It will not stop equation numbers inside `{array}` (there won't be any anyway) and it might stop a wanted equation number in the surrounding equation.

### H. Some Common Mistakes

- The word "data" is plural, not singular.
- The subscript for the permeability of vacuum $\mu_0$, and other common scientific constants, is zero with subscript formatting, not a lowercase letter "o".
- In American English, commas, semicolons, periods, question and exclamation marks are located within quotation marks only when a complete thought or name is cited,

such as a title or full quotation. When quotation marks are used, instead of a bold or italic typeface, to highlight a word or phrase, punctuation should appear outside of the quotation marks. A parenthetical phrase or statement at the end of a sentence is punctuated outside of the closing parenthesis (like this). (A parenthetical sentence is punctuated within the parentheses.)

- A graph within a graph is an "inset", not an "insert". The word alternatively is preferred to the word "alternately" (unless you really mean something that alternates).
- Do not use the word "essentially" to mean "approximately" or "effectively".
- In your paper title, if the words "that uses" can accurately replace the word "using", capitalize the "u"; if not, keep using lower-cased.
- Be aware of the different meanings of the homophones "affect" and "effect", "complement" and "compliment", "discreet" and "discrete", "principal" and "principle".
- Do not confuse "imply" and "infer".
- The prefix "non" is not a word; it should be joined to the word it modifies, usually without a hyphen.
- There is no period after the "et" in the Latin abbreviation "et al.".
- The abbreviation "i.e." means "that is", and the abbreviation "e.g." means "for example".

An excellent style manual for science writers is [**?**].

### *I. Authors and Affiliations*

**The class file is designed for, but not limited to, six authors.** A minimum of one author is required for all conference articles. Author names should be listed starting from left to right and then moving down to the next line. This is the author sequence that will be used in future citations and by indexing services. Names should not be listed in columns nor group by affiliation. Please keep your affiliations as succinct as possible (for example, do not differentiate among departments of the same organization).

### *J. Identify the Headings*

Headings, or heads, are organizational devices that guide the reader through your paper. There are two types: component heads and text heads.

Component heads identify the different components of your paper and are not topically subordinate to each other. Examples include Acknowledgments and References and, for these, the correct style to use is "Heading 5". Use "figure caption" for your Figure captions, and "table head" for your table title. Run-in heads, such as "Abstract", will require you to apply a style (in this case, italic) in addition to the style provided by the drop down menu to differentiate the head from the text.

Text heads organize the topics on a relational, hierarchical basis. For example, the paper title is the primary text head because all subsequent material relates and elaborates on this one topic. If there are two or more sub-topics, the next level head (uppercase Roman numerals) should be used and, conversely, if there are not at least two sub-topics, then no subheads should be introduced.

### *K. Figures and Tables*

*a) Positioning Figures and Tables:* Place figures and tables at the top and bottom of columns. Avoid placing them in the middle of columns. Large figures and tables may span across both columns. Figure captions should be below the figures; table heads should appear above the tables. Insert figures and tables after they are cited in the text. Use the abbreviation "Fig. 1", even at the beginning of a sentence.

TABLE II
TABLE TYPE STYLES

| **Table Head** | **Table Column Head** | | |
|---|---|---|---|
| | ***Table column subhead*** | ***Subhead*** | ***Subhead*** |
| copy | More table copy[a] | | |

[a]Sample of a Table footnote.

Fig. 1. Example of a figure caption.

Figure Labels: Use 8 point Times New Roman for Figure labels. Use words rather than symbols or abbreviations when writing Figure axis labels to avoid confusing the reader. As an example, write the quantity "Magnetization", or "Magnetization, M", not just "M". If including units in the label, present them within parentheses. Do not label axes only with units. In the example, write "Magnetization (A/m)" or "Magnetization {A[m(1)]}", not just "A/m". Do not label axes with a ratio of quantities and units. For example, write "Temperature (K)", not "Temperature/K".

## ACKNOWLEDGMENT

The preferred spelling of the word "acknowledgment" in America is without an "e" after the "g". Avoid the stilted expression "one of us (R. B. G.) thanks ...". Instead, try "R. B. G. thanks...". Put sponsor acknowledgments in the unnumbered footnote on the first page.

## REFERENCES

Number footnotes separately in superscripts. Place the actual footnote at the bottom of the column in which it was cited. Do not put footnotes in the abstract or reference list. Use letters for table footnotes.

Unless there are six authors or more give all authors' names; do not use "et al.". Papers that have not been published, even if they have been submitted for publication, should be cited as "unpublished" [**?**]. Papers that have been accepted for publication should be cited as "in press" [**?**]. Capitalize only

the first word in a paper title, except for proper nouns and element symbols.

For papers published in translation journals, please give the English citation first, followed by the original foreign-language citation [**?**].

## REFERENCES


[1] A. Khelifati, M. Khayati, A. Dignös, D. Difallah, and P. Cudre-Mauroux, "TSM-Bench: Benchmarking time series database systems for monitoring applications," *Proc. VLDB Endow.*, vol. 16, no. 11, pp. 3363–3376, 2023.

[2] S. K. Jensen, T. B. Pedersen, and C. Thomsen, "Time Series Management Systems: A Survey," *IEEE Transactions on Knowledge and Data Engineering*, vol. 29, no. 11, pp. 2581–2600, 2017.

[3] ClickHouse Team, "ClickBench: Benchmarking Analytical Databases," https://github.com/ClickHouse/ClickBench, 2021, accessed: 2025-06-19.

[4] Timescale Team, "Time Series Benchmark Suite (TSBS)," https://github.com/timescale/tsbs, 2020, accessed: 2025-06-19.

[5] P. Gupta, M. J. Carey, S. Mehrotra, and R. Yus, "SmartBench: A Benchmark for Data Management in Smart Spaces," *Proc. VLDB Endow.*, vol. 13, no. 12, pp. 1807–1820, 2020. [Online]. Available: http://www.vldb.org/pvldb/vol13/p1807-gupta.pdf

[6] J. Mostafa, S. Wehbi, S. Chilingaryan, and A. Kopmann, "SciTS: A Benchmark for Time-Series Databases in Scientific Experiments and Industrial IoT," in *Proceedings of the ACM International Conference on Management of Data (SIGMOD)*, 2022.

[7] B. S. P. Addala, M. M. Mohammadi, and K. B. Kent, "DGSim: A Scalable Framework for Simulating Energy Consumption of Household Appliances," in *Proceedings of the 39th ECMS International Conference on Modelling and Simulation*, Catania, Italy, Jun. 2025, pp. 562–569.

[8] R. Liu, J. Yuan, and X. Huang, "Benchmarking Time Series Databases with IoTDB-Benchmark for IoT Scenarios," https://arxiv.org/abs/1901.08304, 2019, arXiv:1901.08304.

[9] M. Arlitt, M. Marwah, G. Bellala, A. Shah, J. Healey, and B. Vandiver, "IoTAbench: an Internet of Things Analytics Benchmark," in *Proc. of the 6th ACM/SPEC International Conference on Performance Engineering (ICPE)*. ACM, 2015, pp. 133–144.

[10] Y. Hao, X. Qin, Y. Chen, Y. Li, X. Sun, and Y. Tao, "TS-Benchmark: A Benchmark for Time Series Databases," in *Proc. of the 2021 IEEE 37th International Conference on Data Engineering (ICDE)*. IEEE, 2021.

[11] J. Xiao, Y. Huang, C. Hu, S. Song, X. Huang, and J. Wang, "Time Series Data Encoding for Efficient Storage: A Comparative Analysis in Apache IoTDB," *Proc. VLDB Endow.*, vol. 15, no. 10, pp. 2148–2160, 2022.

[12] I. L. MacDonald, W. Zucchini, and R. Langrock, *Hidden Markov Models for Time Series: An Introduction Using R*, 2nd ed. CRC Press, 2016.

[13] QuestDB Team, "Questdb documentation: Architecture overview," 2024, accessed: 2025-06-19. [Online]. Available: https://questdb.io/docs/reference/architecture/

[14] S. Rinaldi, F. Bonafini, P. Ferrari, A. Flammini, E. Sisinni, and D. Bianchini, "Impact of Data Model on Performance of Time Series Database for Internet of Things Applications," in *IEEE Int. Instrumentation and Measurement Technology Conf. (I2MTC)*, 2019, pp. 1–6.

[15] W. Felter, A. Ferreira, R. Rajamony, and J. Rubio, "An Updated Performance Comparison of Virtual Machines and Linux Containers," in *2015 IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS)*. IEEE, 2015, pp. 171–172.