SEER: An End-to-End Toolkit for Benchmarking Time Series Database Systems in Monitoring Applications

Luca Althaus
University of Fribourg
Switzerland
luca.althaus@unifr.ch

Mourad Khayati
University of Fribourg
Switzerland
mourad.khayati@unifr.ch

Abdelouahab Khelifati
University of Fribourg
Switzerland
abdelouahab.khelifati@unifr.ch

Anton Dignös
Free University of Bozen-Bolzano
Italy
anton.dignoes@unibz.it

Djellel Difallah
NYU Abu Dhabi
United Arab Emirates
djellel@nyu.edu

Philippe Cudré-Mauroux
University of Fribourg
Switzerland
pcm@unifr.ch

ABSTRACT

Time series database systems (TSDBs) are prevalent in many applications ranging from monitoring and IoT devices to scientific research. Those systems are specifically designed to efficiently manage data indexed by time. Because of the variety of workloads, the diversity of time series features, and the sophistication of existing TSDBs, there is no clear way to pick the most suitable system.

In this demo, we introduce SEER, an automated, configurable, and interactive toolkit to evaluate TSDBs. SEER is based on TSM-Bench, a benchmark tailored for time series database systems used in monitoring applications. It implements an end-to-end pipeline for database benchmarking from data generation and feature contamination to workload evaluation. Users can define their portfolios by configuring and parameterizing custom queries, specifying their frequencies, controlling the type and level of data features, and indicating the type of workloads. Moreover, they can deploy new systems and/or reconfigure the pre-installed ones. SEER would process users' requests and gracefully recommend the best system on a use-case basis.

PVLDB Reference Format:
Luca Althaus, Mourad Khayati, Abdelouahab Khelifati, Anton Dignös, Djellel Difallah, and Philippe Cudré-Mauroux. SEER: An End-to-End Toolkit for Benchmarking Time Series Database Systems in Monitoring Applications. PVLDB, 17(12): 4361 - 4364, 2024.
doi:10.14778/3685800.3685875

PVLDB Artifact Availability:
The source code, data, and/or other artifacts have been made available at https://github.com/eXascaleInfolab/seer.

1 INTRODUCTION

Time series database systems (TSDBs) have become a foundational component within a data management landscape increasingly driven by real-time analytics and data-intensive applications. The growing demand for stream processing and monitoring capabilities has spurred significant research into developing TSDBs that can process, compress, and store time series data effectively. However, the diversity and complexity of these systems pose a challenge for end users in selecting the most suitable TSDB for their specific requirements, especially in scenarios involving real-time monitoring applications. With the current state of affairs, a new comprehensive benchmarking solution has become necessary to understand and compare the capabilities of various TSDBs under modern use cases.

To address this need, we recently introduced TSM-Bench [5], a comprehensive time series benchmark for time series monitoring applications. TSM-Bench builds upon previous TSDB benchmarking efforts [2, 4, 10] to assess novel facets such as query latency, throughput, data ingestion rate, compression, and scalability. It implements a suite of performance assessment tools, which include online execution, diverse query variations, and the generation of realistic data. The data generation relies on a novel method that efficiently augments seed real-world time series datasets. Furthermore, TSM-Bench provides insights through a detailed examination of seven TSDB systems, alongside practical recommendations for navigating their architectural designs.

In this paper, we demonstrate our TSM-Bench benchmark with SEER, a graphical tool allowing users to compare and evaluate various TSDBs through an interactive dashboard. SEER serves two main purposes. **(1) TSM-Bench Results Navigator:** we publish the results of hundreds of experiments we obtained while studying seven TSDB systems, using all TSM-Bench queries across a wide range of query parameters. **(2) Custom Benchmarking with User Configuration:** we enable the users to benchmark their own TSDB systems with custom configurations and to compare and store their results. Moreover, SEER introduces new functionalities, such as synthetic data augmentation to mimic real-world data scenarios and parameterizable and composable TSM-Bench queries for mixed workload benchmarking. We make SEER available as an open-source toolkit, as well as a pre-deployed website preloaded with extensive results from several TSDBs.

The rest of this paper is organized as follows. Section 2 provides an overview and necessary background on TSM-Bench. Then, in Section 3, we present SEER, detailing its architecture and functionalities. Finally, in Section 4, we discuss four demonstration scenarios enabled by SEER, highlighting how it can be used in different use cases to evaluate TSDBs.

Proceedings of the VLDB Endowment, Vol. 17, No. 12 ISSN 2150-8097.
doi:10.14778/3685800.3685875

2 TSM-BENCH OVERVIEW

TSM-Bench consists of three key components: a time series generator, an evaluation framework, and a time series feature injector. We provide a summary of those components and emphasize the benchmark pipeline. More details can be found in our paper [5].

Benchmarking Datasets. TSM-Bench's datasets have been generated using real-world seed water quality time series. The data was recorded in Swiss rivers using ten sensors from one station over a period of five days, resulting into 430K datapoints. There are two categories of time series. The first category includes water temperature series containing duplicates and similar consecutive values. The second category includes water level series, which are erratic and contain abrupt changes.

As mentioned earlier, TSM-Bench aims to evaluate the scalability of TSDBs. To do so, the seed dataset was augmented by generating a larger number of long real-like time series. As a result, we obtain two benchmarking datasets of 518M and 17.2B data points, respectively. We devised and implemented a new generation technique that augments the size of datasets in linear time. The generation builds on recent advances in generative models [8] and Locality Sensitive Hashing (LSH) [7]. Our generation is modular and extensible, making it possible to augment multiple series as well as to control the generation quality/efficiency trade-off.

Performance Evaluation Framework. TSM-Bench compares seven popular TSDBs: ClickHouse, Druid, eXtremeDB, InfluxDB, MonetDB, QuestDB, and TimescaleDB. It evaluates their computational and storage performance using standard metrics such as query latency, throughput, data ingestion rate, and compression.

To compare the computational performance, TSM-Bench implements two workload tiers that capture how much the query complexity impacts the query latency of the systems. The offline workload evaluates various temporal queries, assuming that query execution and insertion are separate. Seven queries of increasing level of complexity are implemented: Q1: Data Fetching, Q2: Data Fetching with Filter, Q3: Data Aggregation, Q4: Downsampling, Q5: Upsampling, Q6: Cross-sensor Average, and Q7: Correlation. The online workload simulates concurrent operations and evaluates those querying while increasing the data influx rate.

Data Encoding. Several recent works have studied the impact of time series key properties—features—on encoding schemes [1, 9]. TSM-Bench builds on those works and implements a systematic way to evaluate the impact of those features on the storage capabilities of TSDBs. Three types of features are considered: (i) repeated values, (ii) missing values, and (iii) consecutive values within a range. The scale of the features is varied, and the systems are ranked based on the storage size of the time series.

TSM-Bench packs all those components into a comprehensive benchmark. However, manually parsing the benchmark's results remains tedious as the user needs to analyze a large number of plots, i.e., around 68'482 results. To facilitate this effort, we propose an easy-to-use exploration toolkit that implements all the components of TSM-Bench in a few interactive plots. Moreover, we added additional utilities, such as a customized frontend, to configure the workloads, visualize their results on demand, deploy new systems, and reconfigure the pre-installed ones.

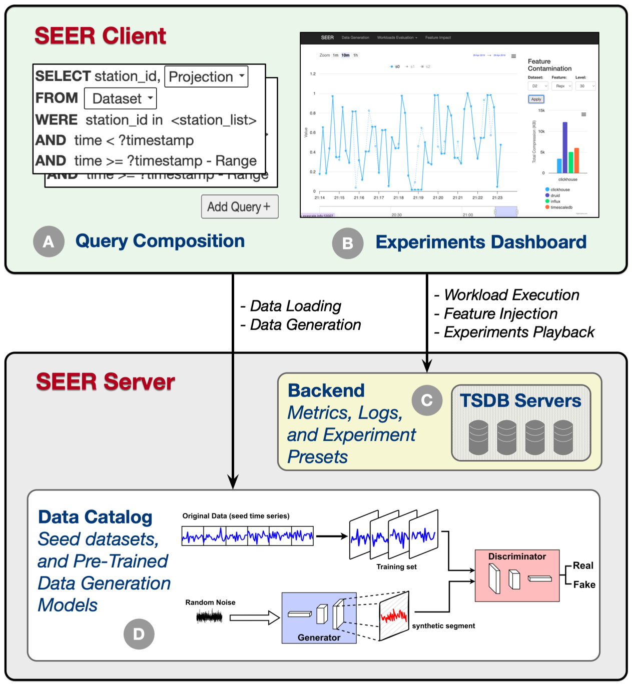

```
SELECT station_id, Projection
FROM  Dataset
WERE  station_id in  <station_list>
AND  time < ?timestamp
AND  time >= ?timestamp - Range
```

Figure 1: SEER Architecture

3 SEER TOOLKIT

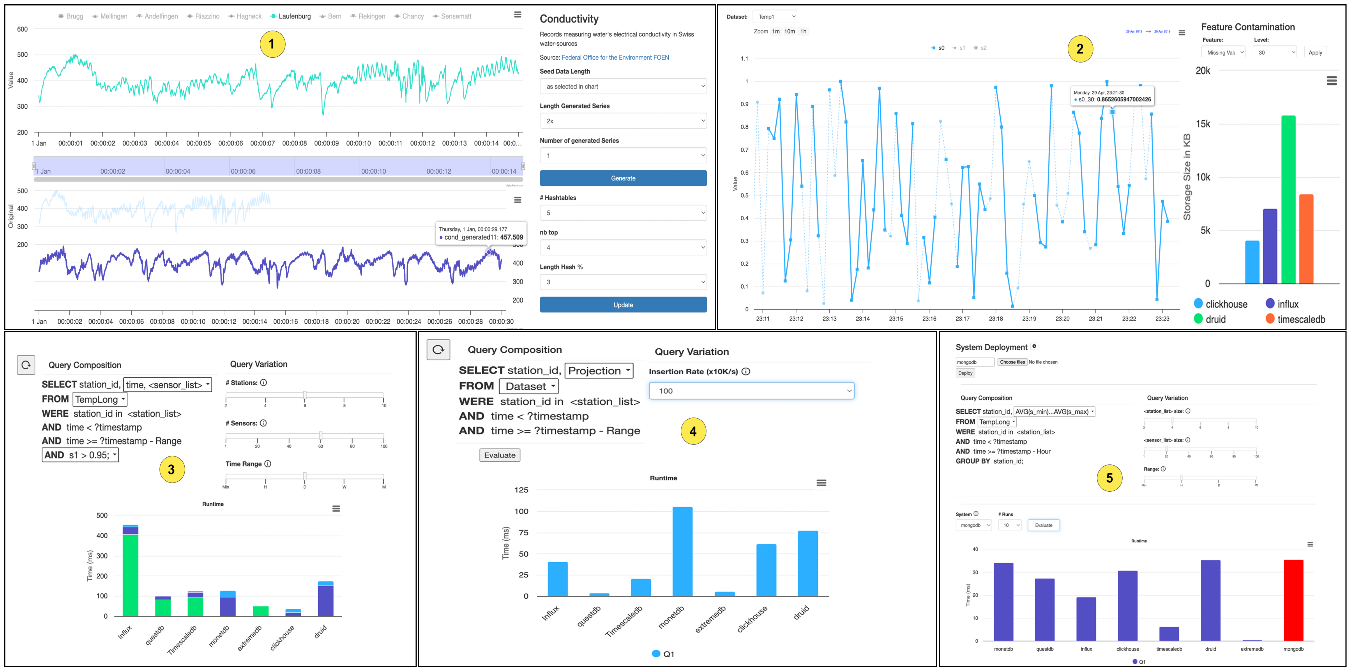
In this section, we outline the architecture of the SEER framework and the features it offers. SEER is written in Python and JavaScript for the backend and front-end functionalities. A demo version is deployed on our web server[1], offering TSDB connections. The source code of SEER and the used datasets are publicly accessible at: https://github.com/eXascaleInfolab/seer. Any other application can easily embed our data generation and query composition, thus acquiring the ability to perform automatic benchmarking.

3.1 SEER Architecture

SEER is implemented as a web application based on a client-server architecture (cf. Figure 1). On the client side A, users can select a target dataset, configure and execute queries from TSM-Bench, or generate mixtures of such queries. B The results, either generated or retrieved from the backend, are streamed to the dashboard for inspection and comparison across systems. C The backend server processes client requests to execute actions such as running real-time queries against a preconfigured TSDB connection, testing the impact of different systems through feature injection, or saving/loading the results of past experiments. Lastly, D the data catalog enables the selection of built-in seed datasets and allows for their augmentation with our data generation algorithm.

The system is optimized for interactivity, ensuring quick response times for a better user experience. The bulk of the workload results and feature impact are derived and interpolated from a repository of prior experimental results. Additionally, new executions on TSDBs are recorded in the experiments log.

[1] http://srv.exascale.info:12007/


Conductivity

Records measuring water's electrical conductivity in Swiss water-sources

Source: Federal Office for the Environment FOEN

Seed Data Length: as selected in chart

Length Generated Series: 2x

Number of generated Series: 1

# Hashtables: 5

nb top: 4

Length Hash %: 3

Thursday, 1 Jan, 00:00:29.177 ● cond_generated11: 457.509

Monday, 29 Apr, 23:21:30 ● s0_30: 0.8652605947002426

Dataset: Temp1

Feature Contamination — Feature: Missing Val. Level: 30

```
SELECT station_id, time, <sensor_list>
FROM TempLong
WERE station_id in <station_list>
AND time < ?timestamp
AND time >= ?timestamp - Range
AND s1 > 0.95;
```

```
SELECT station_id, Projection
FROM Dataset
WERE station_id in <station_list>
AND time < ?timestamp
AND time >= ?timestamp - Range
```

Insertion Rate (x10K/s): 100

System Deployment: mongodb — Choose files No file chosen

```
SELECT station_id, AVG(s_min)...AVG(s_max)
FROM TempLong
WERE station_id in <station_list>
AND time < ?timestamp
AND time >= ?timestamp - Hour
GROUP BY station_id;
```

System: mongodb # Runs: 10

Figure 2: SEER provides a GUI that enables (1) visualizing seed time series at different granularity and generating new series by augmenting the seed series' length and/or number. Users can (2) inject various features in the generated time series while specifying their type and frequency and measuring their impact on the storage of TSDBs. They can also compose their own query workloads and compute the performance of TSDBs in two ways: (3) 'offline' where query execution is separate from insertion, and (4) 'online' that simulates concurrent operations. The tool also allows (5) to deploy and reconfigure new TSDBs while comparing their performance against the stored results of pre-installed systems.

3.2 Functionalities

SEER exposes three core functionalities to the user: data generation, feature injection, and system evaluation. The main functionalities are illustrated in Figure 2.

Time Series Generator. SEER allows the generation of synthetic data with real-like properties (1). It uses Generative Adversarial Network (GAN) [3, 6] to build a pre-trained model, which takes as input seed time series segments and learns to imitate them by playing an adversarial game between two networks: a generator and a discriminator. The former is trained to map random noise to synthetic segments, while the latter is trained to distinguish the real segments from the fake ones (cf. Figure 1 – (D)).

We devise a novel Locality-Sensitive Hashing (LSH)-based method to concatenate the GAN-generated segments. The hashing is applied to index each synthetic segment into the hashing tables while ensuring that similar segments are more likely to be stored in the same bucket. Once the hashing tables are constructed, the codes for original segments are extracted and looked up in the hash tables to obtain the most similar synthetic segments. From these, one segment is selected and appended to the generated time series after applying a fitting function.

The quality of the generation depends on multiple parameters, including the number of hash tables (# hashtables), the number of top candidates representing real segments (nb top), and the size of the hash code for a time series segment (length hash %). Tuning those parameters helps achieve a high similarity between the generated and original segments while making room for novelty in the generation. When generating the data, the visualization in SEER provides an interactive exploration of the parameter tuning impact by displaying both the generated and original seed series.

Feature Injector. SEER also allows inspecting how particular features in the data affect the storage requirements of the different TSDB systems that exploit vastly different strategies for data compression (2). It does so by contaminating time series data with missing values, repeated consecutive values, outliers, and a specific delta between consecutive data points. The severity level of the injected contamination can be selected for each contamination. By applying the contamination, SEER visualizes the time series, where the applied contamination is indicated using dotted lines, and shows how, in terms of storage costs, the compression of the different TSDBs reacts to the injected feature.

TSDBs Evaluator. SEER's evaluator compares the query performance of different systems from three perspectives. In all cases, queries are based on TSM-Bench queries, ranging from simple data fetching to the computation of correlations (cf. Section 2), and can be conveniently formulated using SQL templates.

SEER allows the construction of workloads from a sequence of queries (3). A workload consists of an arbitrary number of queries, and for each query, the relative frequency within the workload can be specified. For each query, data access parameters, such as the number of involved series (stations and sensors) and accessed time range (minute, hour, day, etc.), can be specified. SEER will then show, using a stacked bar chart, the runtime performance of the different systems for the specified workload and individual queries within the workload.

In addition to the offline setting, SEER allows the inspection of the performance of queries in an online setting, i.e., when querying and ingestion occur concurrently (4). In this setting, a query and an ingestion rate in insertions per second need to be selected, upon which SEER will then display the query performance in the presence of concurrent data ingestion.

Another salient feature of SEER is the deployment and/or reconfiguration of systems (5). A new system can be deployed by specifying its installation, data loading, and configuration scripts. Multiple instances of the same system can be launched by modifying the configuration file. Once the system is deployed, SEER will visualize the query result as provided by the system and a visual summary of the variability of the runtime time in the form of a box plot. This allows for the analysis of query execution times of a particular system over several runs. When changing query parameters or the TSDB system, the box plot of previous executions remains in the visualization, allowing runtimes and variability of different queries and/or systems to be compared.

4 DEMONSTRATION SCENARIOS

In this section, we discuss the different scenarios for interacting with the web interface. Users will be invited to assess the performance of TSDB systems under various workload modes, query configurations, and dataset sizes.

Scenario 1: Real-like Time Series Generation. In this scenario, users can use a pre-trained GAN model from TSM-Bench to augment the size of the seed dataset with new time series. The first step is to load a dataset from a list of three real-world water monitoring datasets: Temperature, Conductivity, and pH Accuracy. Each dataset exhibits different properties, such as similar consecutive values or abrupt changes. Should the user wish, other datasets can also be uploaded. Data loading is followed by setting generation hyperparameters, which include the hash function, number of hash tables, and length of the hash binary code. Interested participants will be invited to examine how those parameters affect the generation quality and time. Users can also set the size of the generated dataset by varying the number and/or the length of the synthetic time series. They can also augment the series' length and/or number of specific subsequences.

Scenario 2: Feature Encoding Evaluator. In this scenario, users can alter the generated time series with synthetic features that impact the compression of TSDBs. They can also control the occurrence rate of the feature. This step can be repeated multiple times for different features and time series types. By clicking the "Run" button, the compression result of each system is displayed. Users can invoke different encoding schemes of the TSDB by adjusting the feature level to a different value and clicking the "Update" button.

Scenario 3: Customized Query Analysis. In this scenario, users will be able to simulate a common real-life task in which they design a query portfolio tailored to their needs. Users will be invited to compose their own personalized workload by choosing the operators from the Query Builder. They can also specify the configuration of the temporal queries by controlling the number of stations, the number of sensors, and the range. Another important feature of this scenario is to allocate a frequency to the workload. The audience will be then invited either to add a new query to their workload or click on the "Run" button, which will rank the TSDBs according to their weighted average performance on the selected queries. Furthermore, we will provide users with two evaluation modes: offline and online. Users shall observe how the query configuration, query type, and dataset size impact the performance of TSDBs.

Scenario 4: Deployment of New Systems. SEER will provide a more engaged experience with TSDBs. Attendees will be invited to deploy and evaluate new systems not covered by the TSM-Bench benchmark through the "Deploy" function. Upon clicking on the "Evaluate" button, SEER will display the runtime of the deployed system alongside the other systems' pre-calculated results. Alternatively, users can re-deploy an existing system with a different configuration of parameters (e.g., index, partitioning size, JIT compilation, etc.). They shall observe the impact of those parameters on the performance of TSDBs. Additionally, they can modify the number of runs, and SEER will display the variance incurred by the selected system across different query instances. This will help TSDB practitioners scrutinize the robustness of their systems.

REFERENCES

[1] Davis W. Blalock, Samuel Madden, and John V. Guttag. 2018. Sprintz: Time Series Compression for the Internet of Things. *Proc. ACM Interact. Mob. Wearable Ubiquitous Technol.* 2, 3 (2018), 93:1–93:23. https://doi.org/10.1145/3264903

[2] Peeyush Gupta, Michael J. Carey, Sharad Mehrotra, and Roberto Yus. 2020. SmartBench: A Benchmark For Data Management In Smart Spaces. *Proc. VLDB Endow.* 13, 11 (2020), 1807–1820. http://www.vldb.org/pvldb/vol13/p1807-gupta.pdf

[3] Geon Heo, Yuji Roh, Seonghyeon Hwang, Dayun Lee, and Steven Whang. 2020. Inspector Gadget: A Data Programming-based Labeling System for Industrial Images. *Proc. VLDB Endow.* 14, 1 (2020), 28–36. https://doi.org/10.14778/3421424.3421429

[4] Mostafa Jalal, Sara Wehbi, Suren Chilingaryan, and Andreas Kopmann. 2022. SciTS: A Benchmark for Time-Series Databases in Scientific Experiments and Industrial Internet of Things. (2022), 12:1–12:11. https://doi.org/10.1145/3538712.3538723

[5] Abdelouahab Khelifati, Mourad Khayati, Anton Dignös, Djellel Eddine Difallah, and Philippe Cudré-Mauroux. 2023. TSM-Bench: Benchmarking Time Series Database Systems for Monitoring Applications. *Proc. VLDB Endow.* 16, 11 (2023), 3363–3376.

[6] Jinfeng Peng, Derong Shen, Nan Tang, Tieying Liu, Yue Kou, Tiezheng Nie, Hang Cui, and Ge Yu. 2022. Self-supervised and Interpretable Data Cleaning with Sequence Generative Adversarial Networks. *Proc. VLDB Endow.* 16, 3 (2022), 433–446. https://www.vldb.org/pvldb/vol16/p433-peng.pdf

[7] Kexin Rong, Clara E. Yoon, Karianne J. Bergen, Hashem Elezabi, Peter Bailis, Philip Alexander Levis, and Gregory C. Beroza. 2018. Locality-Sensitive Hashing for Earthquake Detection: A Case Study Scaling Data-Driven Science. *Proc. VLDB Endow.* 11, 11 (2018), 1674–1687. https://doi.org/10.14778/3236187.3236214

[8] Yong Wang, Guoliang Li, Kaiyu Li, and Haitao Yuan. 2022. A Deep Generative Model for Trajectory Modeling and Utilization. *Proc. VLDB Endow.* 16, 4 (2022), 973–985. https://doi.org/10.14778/3574245.3574277

[9] Jinzhao Xiao, Yuxiang Huang, Changyu Hu, Shaoxu Song, Xiangdong Huang, and Jianmin Wang. 2022. Time Series Data Encoding for Efficient Storage: A Comparative Analysis in Apache IoTDB. *Proc. VLDB Endow.* 15, 10 (2022), 2148–2160. https://doi.org/10.14778/3547305.3547319

[10] Hao Yuanzhe, Qin Xiongpai, Chen Yueguo, Li Yaru, Sun Xiaoguang, Tao Yu, Zhang Xiao, and Du Xiaoyong. 2021. TS-Benchmark: A Benchmark for Time Series Databases. *2021 IEEE 37th International Conference on Data Engineering (ICDE)* (2021).