

Benchmarking or Benchmarketing ?



*Benchmarking is the practice of comparing **business processes** and **performance metrics** to industry bests and best practices from other companies. Dimensions typically measured are **quality, time** and **cost**.*

Source: Wikipedia

Benchmarking



“

*Benchmarking is the practice of comparing **business processes** and **performance metrics** to industry bests and best practices from other companies **for marketing purpose**. Dimensions typically measured are **quality, time and cost**.*

Source: ChatGPT

Benchmarking



**The Return of the H2O.ai Database-like
Ops Benchmark**

**Comparing InfluxDB, TimescaleDB, and
QuestDB Time-Series Databases**

**4Bn rows/sec query benchmark:
Clickhouse vs QuestDB vs Timescale**

**Timescale vs. Amazon RDS
PostgreSQL: Up to 350x Faster
Queries, 44 % Faster Ingest, 95
% Storage Savings for Time-
Series Data**

**TimescaleDB vs. InfluxDB:
Purpose Built Differently for
Time-Series Data**

Table of contents

01

Shortcomings

02

Improvements

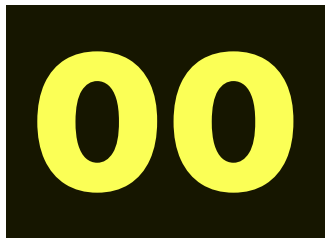
03

Frameworks

04

Takeaways





Misconception about Benchmarking

Benchmarking is always about

Performance



*Benchmarking is the practice of comparing business processes and performance metrics to industry bests and best practices from other companies. Dimensions typically measured are **quality, time and cost**.*

Source: Wikipedia

Benchmarking



Benchmarking is about three things

Reliability
Performance
Cost

Benchmarking is mainly about two things

Reliability
Performance

01

Shortcomings with Benchmarking Strategies

Biased Results

Knowledge disparity about all databases that are being benchmarked like

- Are best practices for these databases adopted ?
- Are these databases designed for this use-case ?
- Are configs for these databases set correctly ?



Non Reproducible Results

Missing mentions of parameters that may impact benchmarking results like

- Hardware specifications used.
- Cluster provisioning steps.
- Dataset used.



No Expert Review of Results

Most benchmarks are done in isolation.

- Before publishing the results, authors of these benchmarks should reach out to the subject-matter experts of each of these databases for reviewing the results.



02

Improvements with Benchmarking Strategies

Benchmark only your database

Run benchmarks on various dimensions on

- Setups in various cloud providers with different cluster topologies.
- Different type and size of datasets.
- Read and write performance.



Benchmark as part of CI

Run benchmarks as part of continuous integration

- Use results to measure performance of each component of your database.
- Track regressions using this benchmarking suite.



03

Frameworks for Benchmarking

TPC-C and TPC-H

Benchmark for OLTP and OLAP databases

- No centralized way of storing the results.



ClickBench

Benchmark for OLAP databases

Detailed Comparison						
	ClickHouse (tuned) (c6a.metal, 500gb gp2)	StarRocks (c6a.metal, 500gb gp2)	Databend (c6a.metal, 500gb gp2)	ClickHouse (c6a.metal, 500gb gp2)	ClickHouse (m6i.32xlarge)	ClickHouse (web) (c6a.metal, 500gb gp2)
Load time:	137s (*414.75)	433s (*1312.12)	70s (*212.12)	140s (*423.01)	458s (*1387.02)	0s (*1.00)
Data size:	13.57 GiB (*1.46)	16.49 GiB (*1.78)	19.49 GiB (*2.10)	13.58 GiB (*1.46)	13.58 GiB (*1.46)	13.56 GiB (*1.46)
Q0.	0.00s (*1.08)	0.03s (*3.94)	0.00s (*1.21)	0.00s (*1.08)	0.01s (*1.48)	0.00s (*1.18)
Q1.	0.04s (*2.25)	0.03s (*2.00)	0.01s (*1.20)	0.01s (*1.10)	0.01s (*1.15)	0.01s (*1.15)
Q2.	0.04s (*2.45)	0.06s (*3.50)	0.02s (*1.54)	0.02s (*1.50)	0.02s (*1.30)	0.02s (*1.65)
Q3.	0.03s (*1.75)	0.06s (*3.50)	0.02s (*1.46)	0.02s (*1.60)	0.01s (*1.25)	0.02s (*1.65)
Q4.	0.09s (*1.10)	0.08s (*1.00)	0.25s (*2.84)	0.19s (*2.21)	0.15s (*1.80)	0.20s (*2.34)
Q5.	0.12s (*1.00)	0.21s (*1.69)	0.27s (*2.12)	0.40s (*3.15)	0.20s (*1.59)	0.45s (*3.56)
Q6.	0.02s (*2.15)	0.02s (*2.31)	0.01s (*1.91)	0.02s (*2.15)	0.02s (*2.00)	0.02s (*2.00)
Q7.	0.02s (*1.97)	0.04s (*2.99)	0.02s (*1.69)	0.03s (*2.09)	0.03s (*2.09)	0.02s (*2.03)
Q8.	0.14s (*1.13)	0.12s (*1.00)	0.29s (*2.29)	0.25s (*1.98)	0.31s (*2.48)	0.27s (*2.12)
Q9.	0.27s (*1.00)	0.70s (*2.54)	0.30s (*1.13)	0.27s (*1.01)	0.33s (*1.22)	0.27s (*1.00)
Q10.	0.06s (*1.17)	0.05s (*1.00)	0.29s (*4.94)	0.09s (*1.67)	0.10s (*1.83)	0.10s (*1.82)
Q11.	0.06s (*1.00)	0.06s (*1.06)	0.18s (*2.83)	0.10s (*1.65)	0.11s (*1.82)	0.09s (*1.52)
Q12.	0.14s (*1.00)	0.17s (*1.22)	0.29s (*2.04)	0.17s (*1.20)	0.19s (*1.39)	0.18s (*1.32)
Q13.	0.18s (*1.00)	0.23s (*1.29)	0.43s (*2.36)	0.21s (*1.18)	0.23s (*1.27)	0.24s (*1.37)
Q14.	0.15s (*1.00)	0.18s (*1.17)	0.27s (*1.73)	0.17s (*1.12)	0.18s (*1.18)	0.21s (*1.38)
Q15.	0.12s (*1.47)	0.08s (*1.00)	0.22s (*2.54)	0.14s (*1.69)	0.15s (*1.74)	0.12s (*1.48)
Q16.	0.32s (*1.08)	0.30s (*1.00)	0.48s (*1.58)	0.35s (*1.17)	0.45s (*1.49)	0.35s (*1.15)

ClickBench

Handles the shortcomings of benchmarking strategies

- Easy to reproduce the results.
- Includes database provisioning steps.
- Includes hardware specifications.



ClickBench

Various databases run clickbench and contributes results

 **add datafusion** ✓

 **add SelectDB result** ✓

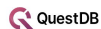
 **Add duckdb results on c5.4xlarge and c6a.metal** ✓

 **add Doris result** ✓

 **Benchmark for StarRocks** ✓

ClickBench

ClickBench helped other databases to improve

[Product](#) ▾[Learn](#) ▾[Docs](#)[Pricing](#)

How ClickBench helped us improve

Recently ClickHouse conducted a [benchmark](#) for their own database and many others, including QuestDB. The benchmark included data import as the first step. Since we were in the process of building a faster import, this benchmark provided us with nice test data and baseline results. So, what have we achieved? Let's find out. The benchmark was using QuestDB's HTTP [import endpoint](#) to ingest the data into an existing non-partitioned table. You may wonder why it doesn't use a [partitioned](#) table, which stores the data sorted by the timestamp values and provides many benefits for time series analysis. Most likely, the reason is terrible import execution time. Both HTTP-based import and pre-6.5 COPY SQL command are simply not capable of importing a big CSV file with unsorted data. Thus, the benchmark opts for a non-partitioned table with no designated timestamp column. The test CSV file may be downloaded and uncompressed following the commands:

[PRODUCTS](#) ▾[SOLUTIONS](#) ▾[STARROCKS](#)[RESOURCES](#) ▾[COMPANY](#) ▾

What ClickBench Can Tell You

ClickBench provides users with a fast and simple way to view and reproduce the ranking of databases by a performance (i.e., query speed). The results are based on a predefined dataset and a collection of queries.

[Contents](#)[Quick start](#)[> Concepts](#)[> Tutorials](#)[> Recommendations](#)[> Managing databases](#)[> Managing a cluster](#)[> YQL](#)

ClickBench load

The load is based on data and queries from the <https://github.com/ClickHouse/ClickBench> repository, and the queries and table layout are adapted to YDB.

The benchmark generates typical workload in the following areas: clickstream and traffic analysis, web analytics, machine-generated data, structured logs, and event data. It covers typical queries in analytics and real-time dashboards.

The dataset for this benchmark was obtained from an actual traffic recording of one of the world's largest web analytics platforms. It has been anonymized while keeping all the essential data distributions. The query set was improvised to reflect realistic workloads, while the queries are not directly from production.



ClickBench

Various databases integrated ClickBench in their daily continuous integration runs

Add ClickBench queries to DataFusion benchmark runner #7060

 Open alamb wants to merge 1 commit into `apache:main` from `alamb:alamb/clickbench_runner` 



04

Takeaways

Takeaways

- Every use-case is unique. Run your own benchmark.
- Not just performance, but reliability, scalability and security are equally important.



Thank you!