



FOSDEM'23

Brussels / 4 & 5 February 2023

Fast and Streaming Data Devroom

Ingesting over a million rows per second on a single instance

Time-series processing using



QuestDB

Javier Ramirez, Developer Advocate at QuestDB

@supercoco9

master 19 branches 67 tags Go to file Add file Code

tris0laris chore(ui): Update README.md (#2959)	35b748e 1 hour ago	3,925 commits
.github	test(build): validate PR title validation rules (#2730)	3 months ago
.idea	chore(core): make java code formatter to apply method sorting (#271...	3 months ago
artifacts	build: 6.6.2-SNAPSHOT	2 months ago
benchmarks	chore(wal): optimise WAL application (merge) to the table (#2922)	last week
ci	ci(build): fix snapshot pipeline (#2913)	3 weeks ago
core	chore(lip): reduce Out Of Order introduced by ILP writing for WAL ta...	7 hours ago
examples	build: 6.7.1-SNAPSHOT	2 weeks ago
i18n	docs(core): update readme queries (#2780)	2 months ago
pkg ami marketplace	chore(core): fix missing defaults for O3 min max commit lag (#2918)	2 weeks ago
utils	build: 6.7.1-SNAPSHOT	2 weeks ago
win64svc	feat(core): deterministically deposit hs_err_pid files into db dire...	6 months ago
.all-contributorsrc	docs: add suconghou as a contributor for bug (#2383)	6 months ago
.git-blame-ignore-revs	chore(build): git blame to ignore the reformatting commit (#2880)	last month
.gitignore	feat(core): make partitions attached via soft link read-only, protect...	2 weeks ago
CODEOWNERS	chore: switch to team-based codeowners (#1754)	last year
CODE_OF_CONDUCT.md	chore(docs): add Prettier formatting to project files (#1720)	last year
CONTRIBUTING.md	docs(core): add code formatting info to contributing guide (#2784)	2 months ago
LICENSE.txt	fix: license changed to Apache 2.0. Fixed #80	3 years ago
README.md	chore(ui): Update README.md (#2959)	1 hour ago
SECURITY.md	docs(core): add SECURITY policy (#2629)	3 months ago
examples.manifest.yaml	docs(lip): add an example with auth, but without TLS (#2455)	5 months ago
pom.xml	build: 6.7.1-SNAPSHOT	2 weeks ago

About

An open source time-series database for fast ingest and SQL queries

questdb.io

- java
- iot
- postgres
- sql
- database
- big-data
- time-series
- analytics
- cpp
- grafana
- postgresql
- simd
- low-latency
- financial-analysis
- tsdb
- hacktoberfest
- time-series-database
- questdb

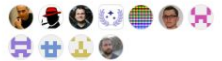
- Readme
- Apache-2.0 license
- Code of conduct
- Security policy
- 10.1k stars
- 116 watching
- 627 forks

Releases 60

6.7 Latest 2 weeks ago

+ 59 releases

Contributors 103



+ 92 contributors

Languages



README.md



slack 1859 all contributors 168 maven-central v6.7-jdk8

English | 简体中文 | 繁體中文 | العربية | Italiano | Українська | Español | Português | 日本語

QuestDB



We would like to be known for:

- Performance
 - Better performance with smaller machines
- Developer Experience
- Proudly Open Source (Apache 2.0)



Not all big (or fast)
data problems are the
same



Do you have a time-series problem? (1/2)

- Most of your queries are scoped to a time range
- You mostly insert data. You rarely update or delete individual rows
- It is likely you write data more frequently than you read data
- Since data keeps growing, you will very likely end up with much bigger data than your typical operational database would be happy with
- You often need to resample your data for aggregations/analytics
- You often need to align timestamps from multiple data series



Do you have a time-series problem? (2/2)

- You typically access recent/fresh data rather than older data
- But still want to keep older data around for occasional analytics
- Your data origin might experience bursts or lag, but keeping the correct order of events is critical for you
- But you typically request your reads to show data captured recently
- Both ingestion and querying speed are critical for your business

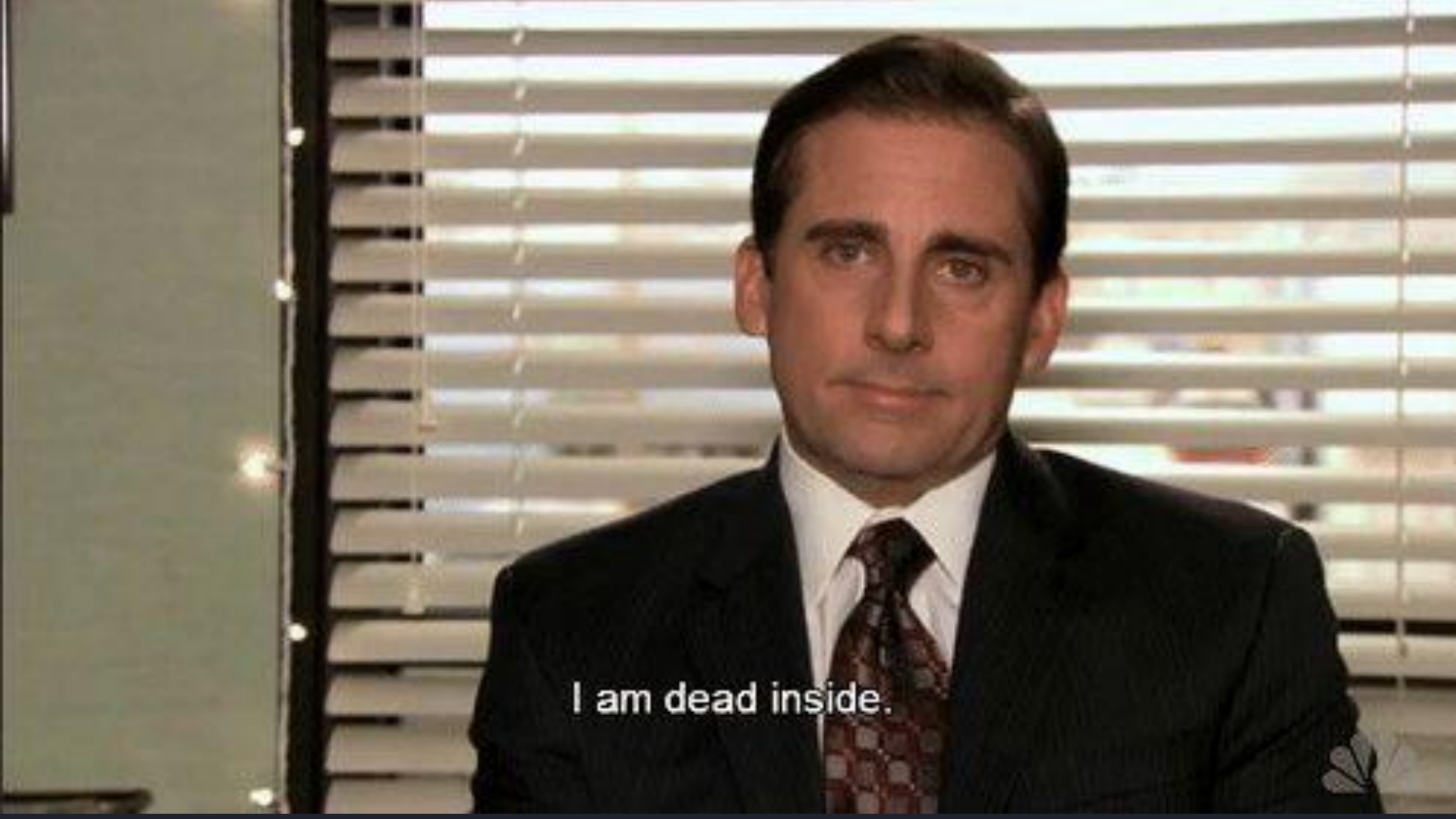


Some time-series demo queries

<https://demo.questdb.io/>



Ingesting over 1 million time series per second on a single instance

A medium shot of Steve Carell as Michael Scott from the TV show 'The Office'. He is wearing a dark suit, a white shirt, and a patterned tie. He is sitting in front of horizontal window blinds. The lighting is soft, coming from the blinds. The NBC peacock logo is visible in the bottom right corner.

I am dead inside.

All benchmarks are lies (but they give us a ballpark)



Ingesting over 1.4 million rows per second (using 5 CPU threads)

<https://questdb.io/blog/2021/05/10/questdb-release-6-0-tsbs-benchmark/>

While running queries scanning over 4 billion rows per second (16 CPU threads)

<https://questdb.io/blog/2022/05/26/query-benchmark-questdb-versus-clickhouse-timescale/>



Search or jump to...

Pull requests Issues Codespaces Marketplace Explore



timescale / tsbs Public

Watch 46

Fork 233

Starred 1k

<> Code Issues 52 Pull requests 21 Actions Projects Security Insights

master

15 branches

0 tags

Go to file

Add file

<> Code



puzpuzpuz Add QuestDB to makefile (#181)

bcc0013 on Jan 19, 2022 769 commits

folder	.github/workflows	Create go.yaml for GitHub workflow (#171)	last year
folder	cmd	Questdb benchmark support (#157)	last year
folder	docs	Questdb benchmark support (#157)	last year
folder	helm	TSBS Docker and helm chart	2 years ago
folder	internal	Questdb benchmark support (#157)	last year
folder	load	Enable persisting ingestion/query benchmark results in a common fo...	last year
folder	pkg	Questdb benchmark support (#157)	last year
folder	scripts	Questdb benchmark support (#157)	last year
file	.gitignore	Questdb benchmark support (#157)	last year
file	.travis.yml	Add multinode to master (#168)	last year
file	Dockerfile	TSBS Docker and helm chart	2 years ago
file	LICENSE	Update copyright year to 2021	2 years ago
file	Makefile	Add QuestDB to makefile (#181)	last year

About

Time Series Benchmark Suite, a tool for comparing and evaluating databases for time series data

- benchmarking
- cassandra
- mongodb
- influxdb
- time-series
- timescaledb

- Readme
- MIT license
- 1k stars
- 46 watching
- 233 forks

Releases

No releases published

Packages

No packages published



Technical decisions and trade offs we made
to get here



We can make many
assumptions about the shape
of the data and usage patterns

Written FROM SCRATCH for performant time-series



- Using JAVA unsafe mode, with zero GC and sharing memory with C++
- Writing our own IO functions, with native memory networking and zero GC
- Own implementation of String and other common classes, to avoid overhead
- Own implementation of Logger, for speed and to avoid interpolations



Down to the nanosecond

Benchmark	Mode	Cnt	Score	Error	Units
LogBenchmark.testLogOneIntBlocking	avgt	2	265.391		ns/op
LogBenchmark.testLogOneInt	avgt	2	82.985		ns/op
LogBenchmark.testLogOneIntDisabled	avgt	2	0.661		ns/op
Log4jBenchmark.testLogOneInt	avgt	2	877.266		ns/op
Log4jBenchmark.testLogOneIntDisabled	avgt	2	1.368		ns/op

QUESTDB'S APPROACH TO PERFORMANCE

A man with dark hair, wearing a black sweater, is shown from the chest up. He is holding a mobile phone to his ear with his right hand. His facial expression is one of intense focus or menace, with furrowed brows and a slight, unsettling smile. The background is a plain, light-colored wall.

**I WILL FIND YOU...
AND I WILL QUICKEN YOU**





<https://github.com/questdb/questdb>



<https://questdb.io/cloud/>



Quick recap

- Time-series problems can be hard
- QuestDB only does time-series
- Ingestion is done via official clients (or ILP over socket), queries are done via SQL
- QuestDB's storage model makes ingestion very fast, and indices unnecessary
- We measure-implement-repeat continuously to improve performance
- All benchmark are lies, but if you like them take a look at

<https://questdb.io/blog/tags/engineering/>



THANKS!

For more info, <https://questdb.io> and <https://demo.questdb.io>

We  contributions and  stars

github.com/questdb/questdb

Javier Ramirez, Developer Advocate at QuestDB
@supercoco9