CWI

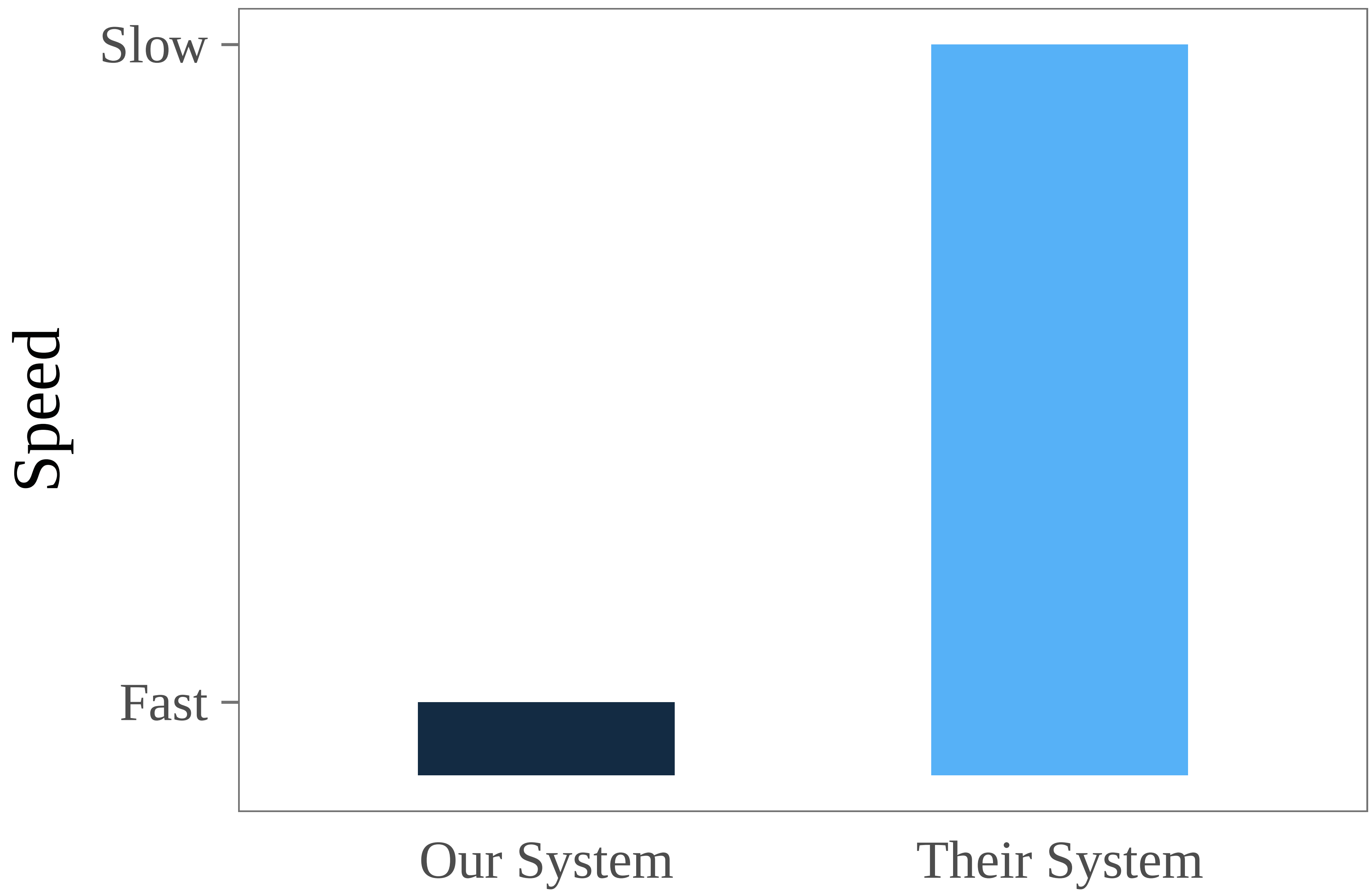
Database Architectures

Fair Benchmarking Considered Difficult: Common Pitfalls In Database Performance Testing

Mark Raasveldt, Pedro Holanda, Tim Gubner & Hannes Mühleisen

State of Things

- Many problems in Data Management Benchmarking
  - Industry 2018: White papers online, misleading ("Trust us, our product is perfect in every way")
  - Academia 2018: Unreproducible numbers in papers ("Trust me, my proposal is the best")
  - Paradox:
    Lots of results published, few are useful
  - Why?



Paper without this plot will not get accepted
Product without this plot will not get traction/sold

Motivating Example

- TPC-H Q1 benchmark in top conference paper
- Compared prototype against real DBMS (Hyper)
  - Hardcoded group counts + Hardcoded hash
  - Too small data types (float to hold aggregations)
  - Overflows not handled
- Surprise: They were faster
  - … but incorrect results (and crashes if the dataset changes)
- Doesn’t matter if you only look at the timings!

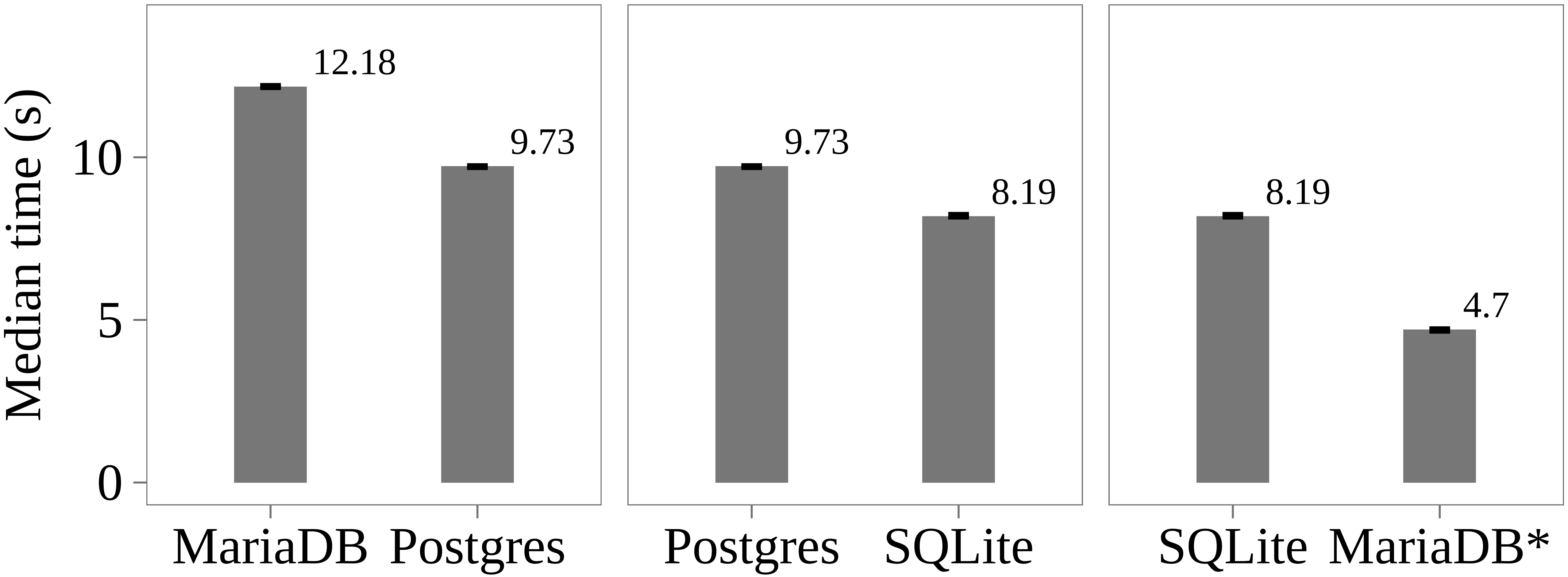
DB Benchmarks: Common Pitfalls

1. Non-Reproducibility
2. Failure to Optimise
3. Apples vs Oranges
4. Incorrect Results
5. Cold vs. Hot/Warm Runs
6. Data Preprocessing
7. Overly-Specific Tuning

Non-Reproducibility

- The example we gave was bad.
  - But we could at least spot the crimes!
- Could be worse:
  - Just nothing available. This is the normal case.
  - Very little consequences (paper acceptance)
- Noble Effort: SIGMOD Reproducibility
- Fix: Script that produces plots in paper *from scratch.* Source code etc. available.

Same query & data (TPCH SF1 Q1)



What’s the crime?

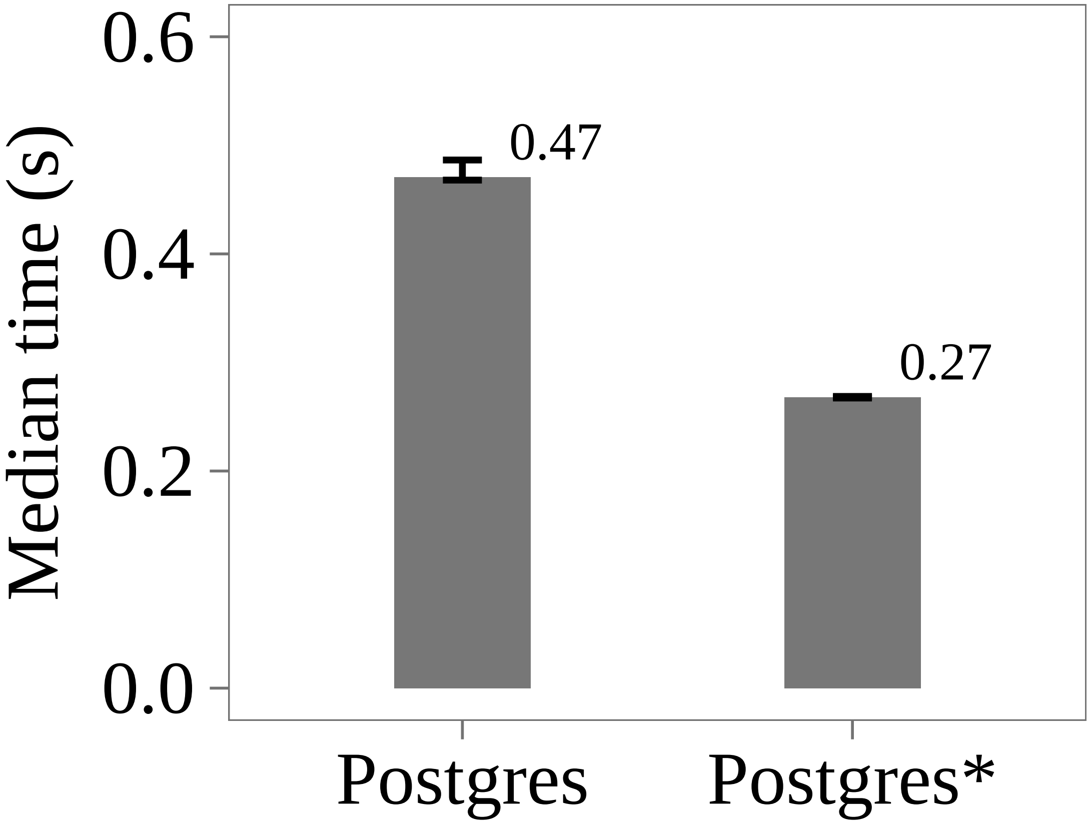
Same query & data (TPCH SF1 Q1)

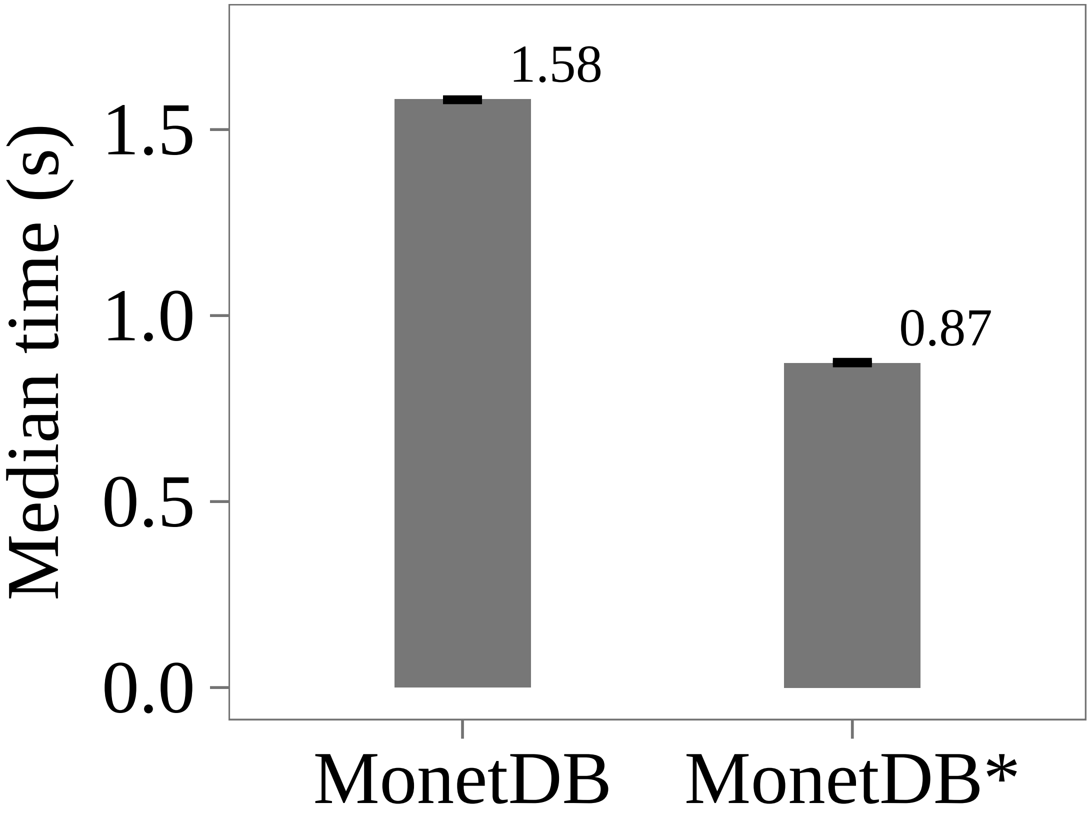
- …same configuration parameters
- …same compilation flags
- …same version number of the database
- …different schema!
    - DOUBLE instead of DECIMAL
- Still gives correct results according to TPC-H specification

Failure to Optimize

- Low incentive to optimise competition
- Compiler (-O1 vs -O3, version, …)
- Configuration
  - e.g. pg_shared_buffers=10GB, pg_effective_cache_size=6GB
- Fix: Involve competition!
  Have them configure their system.
  - Lots of work though, but more common.

Same query, data & schema (TPCH SF1 Q1)





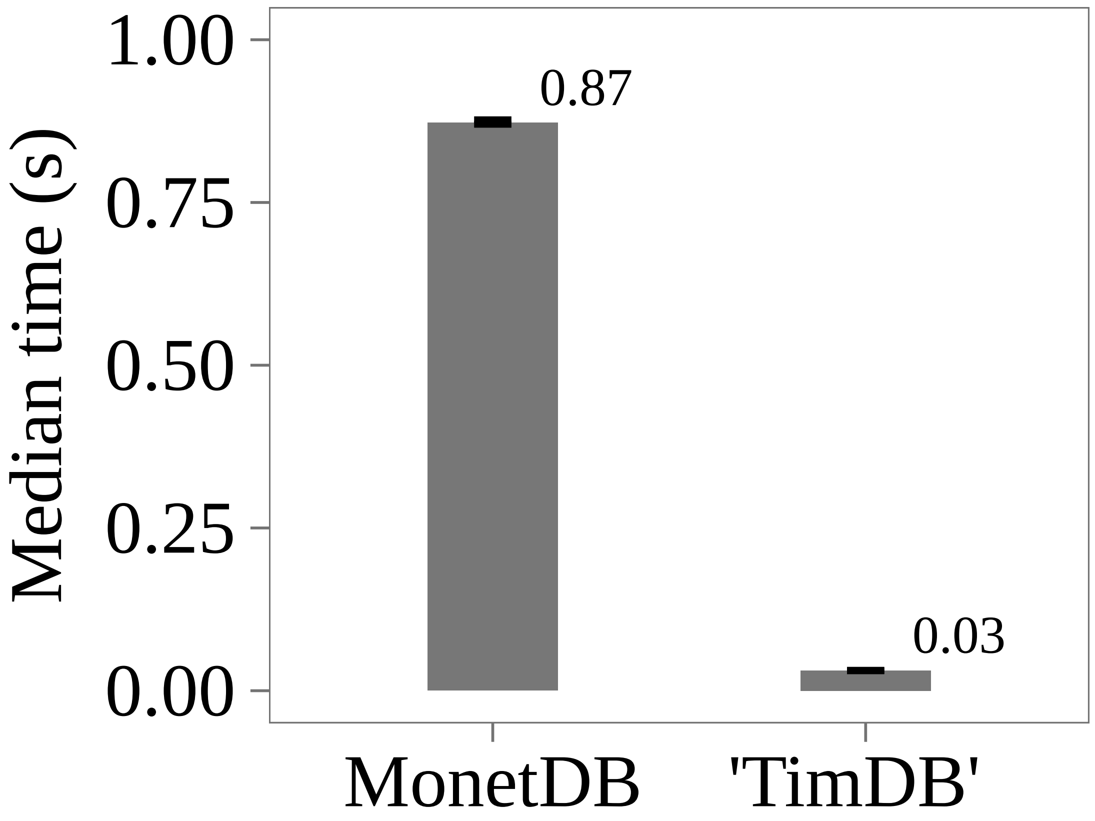
Config

Compilation Flags

Apples vs. Oranges

- Standalone vs. Full System
- Feature mismatch
  - Overflow checking on/off
  - Transactions on/off
- Fix: Hard. Integrate algorithms into full system.

Same query & data (TPCH SF1 Q1)



TimDB is hand-rolled standalone C program for Q1
TimDB is not a database. Common misrepresentation.

Incorrect Results

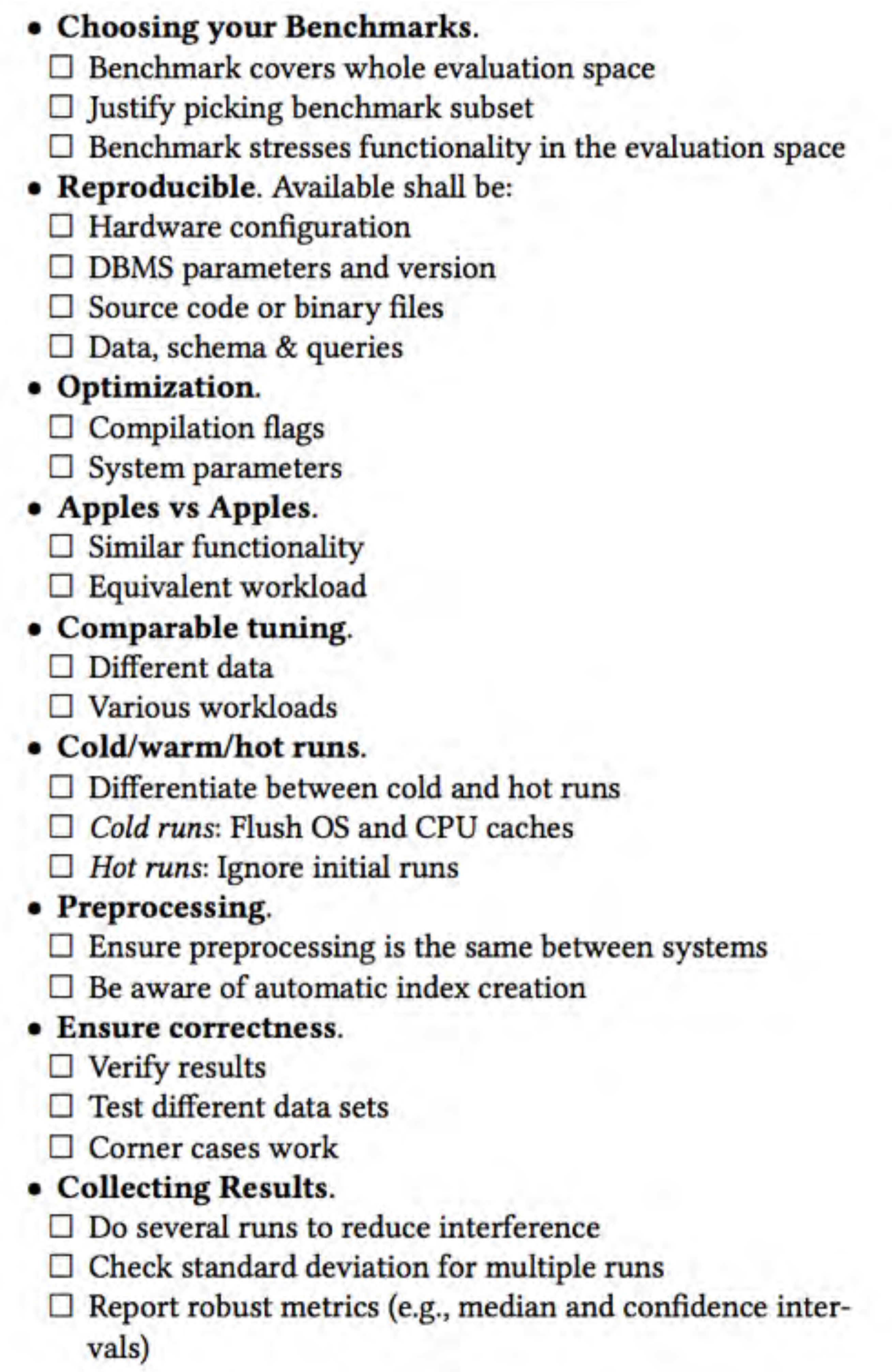
- Bugs sometimes make code very fast.
  - But incorrect, may be invisible in benchmark
- Always check results
- Run with different benchmark and dataset, too
- E.g. run with PostgreSQL and compare results

```
void tpchq1() {
  return;
}
```

Even TimDB can’t beat!

Summary

- Beware of these pitfalls when writing/reviewing
- We are by no means immune ourselves



- **Choosing your Benchmarks.**
  - ☐ Benchmark covers whole evaluation space
  - ☐ Justify picking benchmark subset
  - ☐ Benchmark stresses functionality in the evaluation space
- **Reproducible**. Available shall be:
  - ☐ Hardware configuration
  - ☐ DBMS parameters and version
  - ☐ Source code or binary files
  - ☐ Data, schema & queries
- **Optimization.**
  - ☐ Compilation flags
  - ☐ System parameters
- **Apples vs Apples.**
  - ☐ Similar functionality
  - ☐ Equivalent workload
- **Comparable tuning.**
  - ☐ Different data
  - ☐ Various workloads
- **Cold/warm/hot runs.**
  - ☐ Differentiate between cold and hot runs
  - ☐ *Cold runs*: Flush OS and CPU caches
  - ☐ *Hot runs*: Ignore initial runs
- **Preprocessing.**
  - ☐ Ensure preprocessing is the same between systems
  - ☐ Be aware of automatic index creation
- **Ensure correctness.**
  - ☐ Verify results
  - ☐ Test different data sets
  - ☐ Corner cases work
- **Collecting Results.**
  - ☐ Do several runs to reduce interference
  - ☐ Check standard deviation for multiple runs
  - ☐ Report robust metrics (e.g., median and confidence intervals)