TexBench: A Unified *Benchmarking Suite* for *Shifting Workloads*

Abhishek Chanda Shubham Kaushik Artem Lavrov Subhadeep Sarkar

Problem

Key-value (KV) benchmarks

KV-stores are everywhere

RocksDB, cassandra, levelDB, APACHE HBASE, SQLite, speedb, BigTable, CockroachDB, WT, SCYLLA, influxdb, FoundationDB

Many databases!

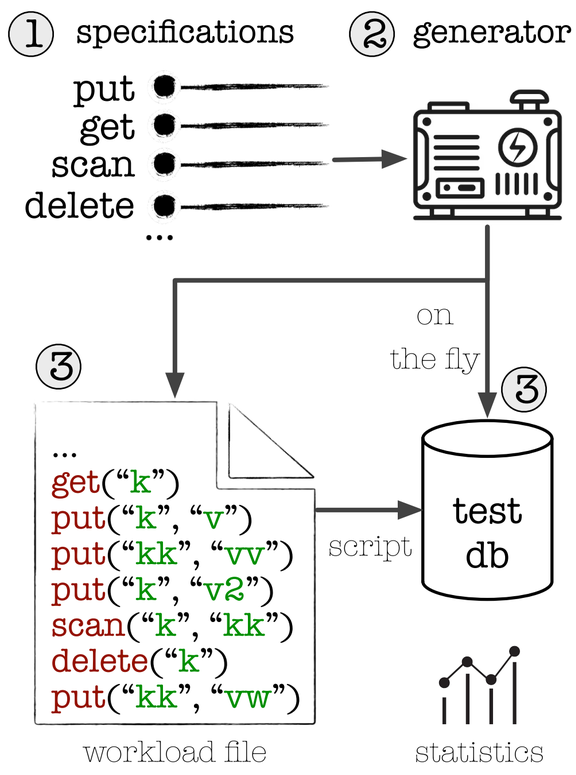
How to **compare** & **choose** the **best**?

Benchmarking databases

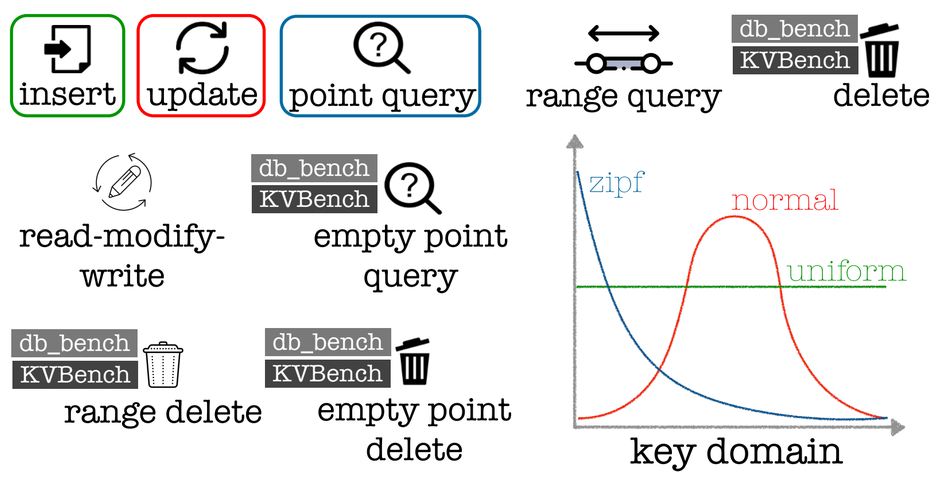
relational: TPCx-HS (TPCTC'14), TPC-DI (VLDB'14), OLTPBench (VLDB'13), PDGF (DBTest'11), Myriad (ASBD'11), TPC-E (TPCTC'09), SSB (TPCTC'09), TPC-DS (VLDB'06), MUDD (WOSP'04), TPC-C (SIGMOD'93)

non-relational — KV store benchmarks & generators: Tectonic (TPCTC'25), KVBench (DBTest'24) — workload generators; db_bench (RocksDB) — tightly coupled w/ RocksDB; YCSB (SoCC'10) — fixed legacy workloads

workload generators



Real-world workloads



exhibits diverse, operation-specific distributions

Challenges: KV benchmarks / generators

1 shifting workloads

2 complex key-value structure

<user_id>:<model_version>:<feature>:<timestamp>

Tectonic

```
{
 "sections":[{
  "groups":[
  {
   Phase 1: write-heavy
   "inserts":{95%},
   "point_queries":{5%}
  },{
   Phase 2: read-heavy
   "updates":{2%},
   "point_queries":{98%}
}]}]}
```

- support shifting workloads
- fine grained control over key-value
- generation of nearly sorted workloads
- emulate diverse access patterns

expects users to write complex JSON input

no side-by-side comparison

no support for results visualization

TexBench

Design overview

Presets: YCSB-A to F; db_bench; KVBench-I to V; **Tectonic-1 to 7**

OR

Custom: *"Generate a two-phase workload that starts insert-heavy and later shifts to read-heavy."*

choose one of the standard benchmarks

LLM core

input JSON

```
{
 "sections":[{
  "groups":[
  {
   Phase 1: write-heavy
   "inserts":{95%},
   "point_queries":{5%}
  },
  {
   Phase 2: read-heavy
   "updates":{2%},
   "point_queries":{98%}
  }]
}]}
```

Tectonic

generate multi-phased workload

benchmark each database

output

Workload specification

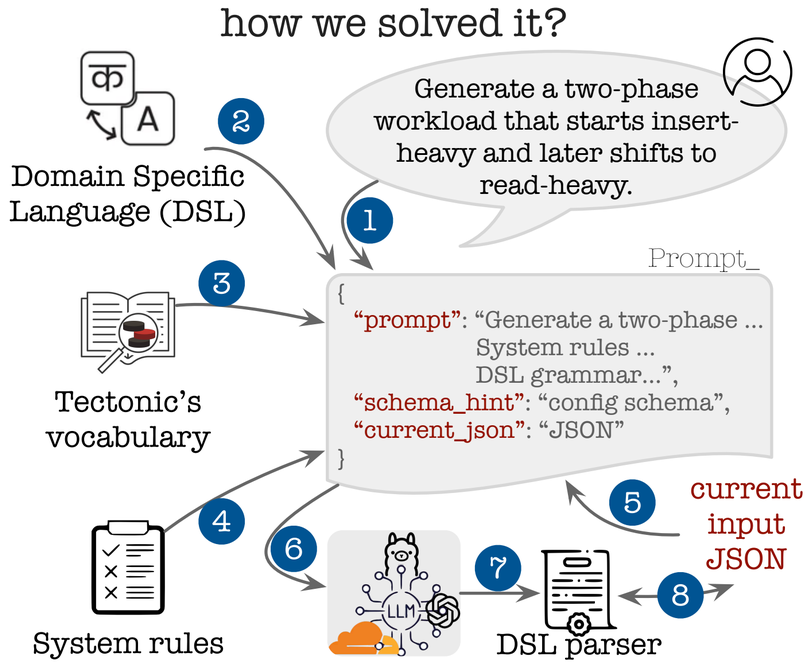
- support preset benchmarks out-of-the-box
- enable users to generate input JSON using LLMs
- compare databases side-by-side in the same interface

LLM core

challenges

- lost-in-the-middle effect
- hallucination
- invalid results
- parsing and extracting

how we solved it?

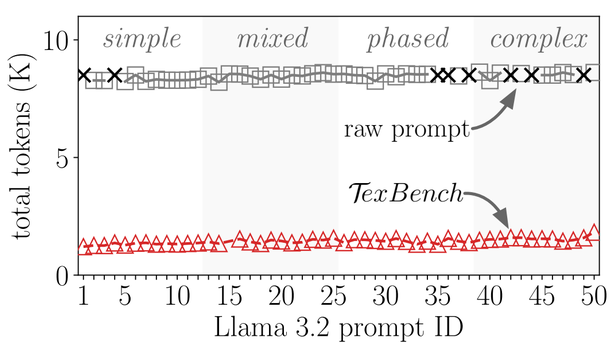


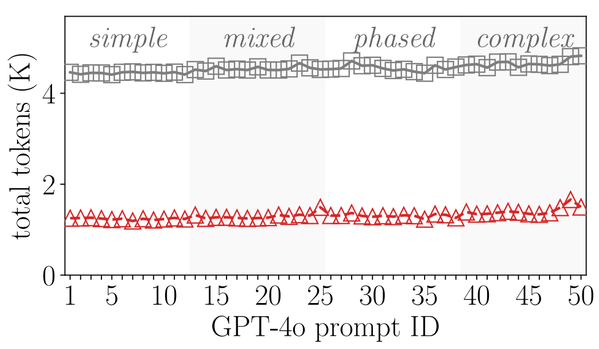
ensures the correctness of workload specification

enables users to describe workloads

Evaluation

Run a corpus of 50 prompts to generate simple, mixed operations, phased, and complex multi-phase workloads

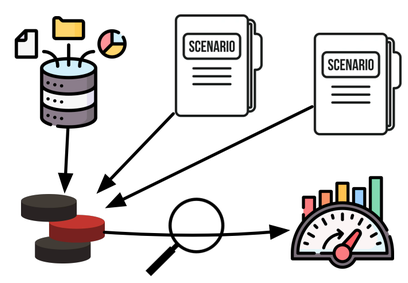




dimension/score	maximum	raw prompt	*TexBench*
syntactic validity	5	5	5
section cardinality	5	5	5
phase cardinality	10	5	10
key-domain	5	5	5
phase ordering	10	0	10
operation composition	20	8	20
operation properties	35	18.7	35
schema correctness	10	10	10
total	100	**56.7**	**100**

- reduces costs by using significantly fewer tokens
- independent of the LLM model, saving tokens
- more accurate JSON specifications compared to raw prompting

Future work



- add support for more databases
- evaluate workload modeling against real-world traces

Brandeis UNIVERSITY
Michtom School of Computer Science

VLDB 2026 BOSTON

lab