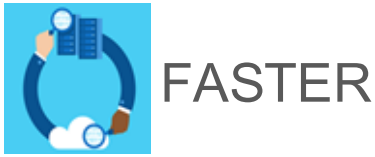
KVBench: A Key-Value Benchmarking Suite

Zichen Zhu, Arpita Saha, Manos Athanassoulis, Subhadeep Sarkar

BOSTON UNIVERSITY
Brandeis UNIVERSITY
The International Workshop on Testing Database Systems dbtest.io
ACM SIGMOD



Key-Value Stores

FASTER

new key-value stores

Key-Value

Stores

new application requirements

run experiments

apples to apples comparison

Key-Value Workload Generators & their Limitations

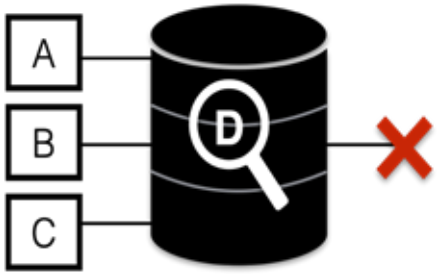
		operation								distribution			
		insert	update	non-empty point query	empty point query	range query	non-empty point delete	empty point delete	range delete	uniform	normal	beta	Zipfian
benchmark	YCSB	✓	✓	✓	✗	✓	✗	✗	✗	✓	✗	✗	✓
	db_bench	✓	✓	✓	✓	✓	✓	✓	✓	✗	✓	✗	✗
	KVBench	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓

BOSTON UNIVERSITY Brandeis UNIVERSITY

The International Workshop on Testing Database Systems dbtest.io

ACM SIGMOD

Challenges



empty point queries not supported

insert, update and point queries may have different distributions

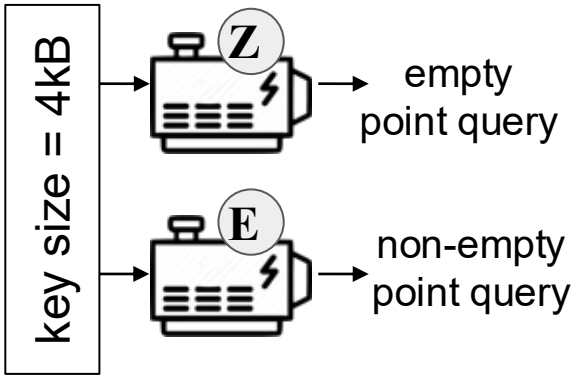
deletes maybe interleaved with updates and queries

BOSTON UNIVERSITY Brandeis UNIVERSITY

The International Workshop on Testing Database Systems dbtest.io

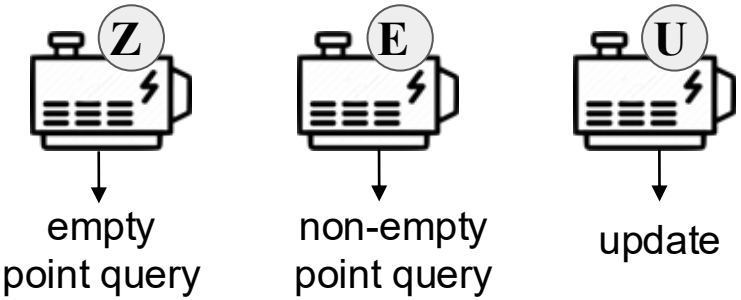
ACM SIGMOD

Contributions



two generators for empty & non-empty point query

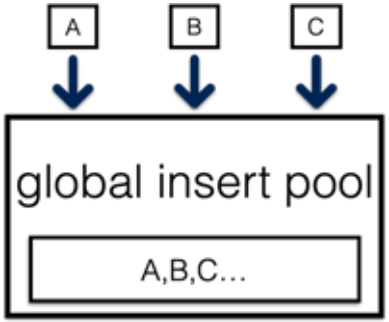
- mixed workload
- interleaved point queries



separate generator for update operation

- update and queries with different distributions

Contributions



global insert pool to keep track of existing entries

- deletes interleaved
- deletion on existing keys

```
KVBench-I
KVBench-II
KVBench-III
KVBench-IV
KVBench-V
```

new set of benchmark workloads

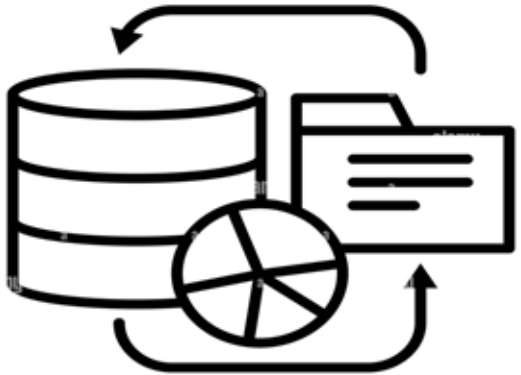
- inspired by real-life workloads

BOSTON UNIVERSITY Brandeis UNIVERSITY

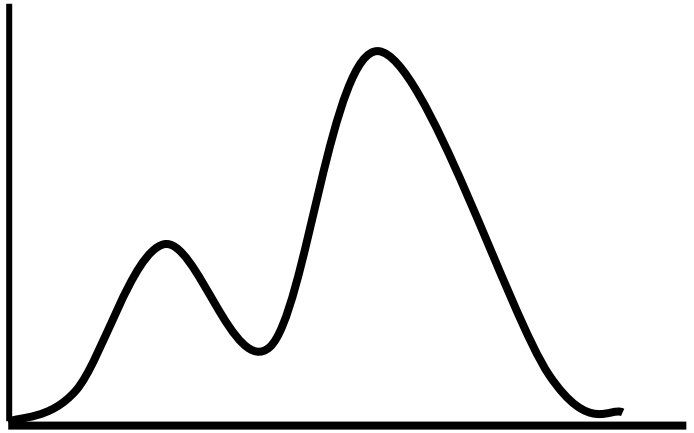
The International Workshop on Testing Database Systems dbtest.io

ACM SIGMOD

KVBench Suite



operations



distributions

BOSTON UNIVERSITY

Brandeis UNIVERSITY

The International Workshop on Testing Database Systems dbtest.io

ACM SIGMOD

Operations

insert

update

query

delete

BOSTON UNIVERSITY

Brandeis UNIVERSITY

The International Workshop on Testing Database Systems dbtest.io

ACM SIGMOD

Insert

- Create global insert pool
- Generate key until not matched in pool
- Insert new key into pool
- Output operation into workload file

```
HS = HashSet() //global insert pool
Key k = RandomKeyGenerator(dist)
while k ∈ HS : //check if key in pool
        k = RandomKeyGenerator(dist)
HS.insert(k) //insert new key in pool
Value v = RandomValueGenerator()
workload_file.output("Insert", k, v)
```

BOSTON UNIVERSITY Brandeis UNIVERSITY

The International Workshop on Testing Database Systems dbtest.io

ACM SIGMOD

Update

- Random key from global insert pool
- Random value generated
- Operation output to workload file

```
V = HS.toVector() //copy HS elements to vector
Index idx = RandomIndexGenerator(|V|, dist)
Key k = V.get(idx) //select key randomly from V
Value v = RandomValueGenerator()
workload_file.output("Update", k, v)
```

Point Query

- *Empty point query:*
  Key generated randomly until not in pool
- *Non-empty point query:*
  Key selected randomly from insert pool
- Operation output to workload file

Empty point-query:

```
Key k= RandomKeyGenerator()
while k ∈ HS : //generate key until not in pool
        k = RandomKeyGenerator()
workload_file.output("Query", k)
```

Non-empty point-query:

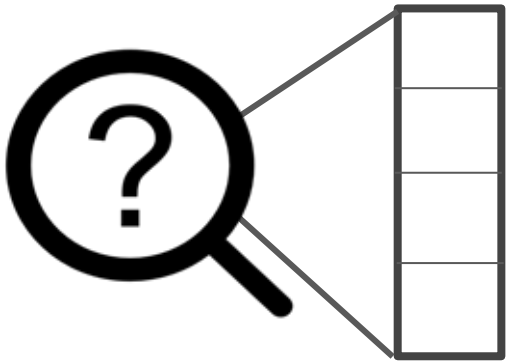
```
Index idx = RandomIndexGenerator(|V|, dist)
Key k = V.get(idx) //select key randomly from V
workload_file.output("Query", k)
```

BOSTON UNIVERSITY Brandeis UNIVERSITY

The International Workshop on Testing Database Systems dbtest.io

ACM SIGMOD

Range Query



- Random start key skipping Y*|V| entries
- Offset by selectivity for end key
- Operation output to workload file

```
V = V.sort() //sort vector
Index start_idx = RandomIndexGenerator(|V|*(1-Y), dist)
Key start_key = V.get(start_idx)
                    //random start key skipping Y*|V| entries

Index end_idx = start_idx+|V|*Y//offset by Y for end key
Key end_key = V.get(end_idx)
workload_file.output("RangeQuery", start_key, end_key)
```

BOSTON UNIVERSITY Brandeis UNIVERSITY

The International Workshop on Testing Database Systems dbtest.io

ACM SIGMOD

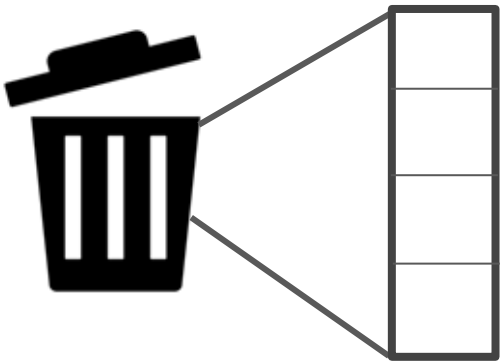
Point Delete

- Key selected randomly from insert pool
- Remove key from pool
- Delete operation output to workload file

```
Index idx = RandomIndexGenerator(|V|, dist)
Key k = V.get(idx) //random key from pool

V.remove(k) //remove key from vector
HS.remove(k) //remove key from hashset
workload_file.output("Delete", k)
```

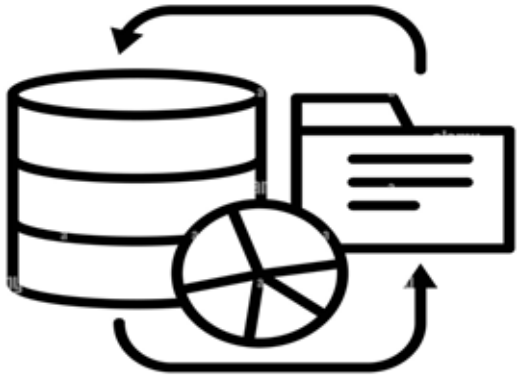
Range Delete



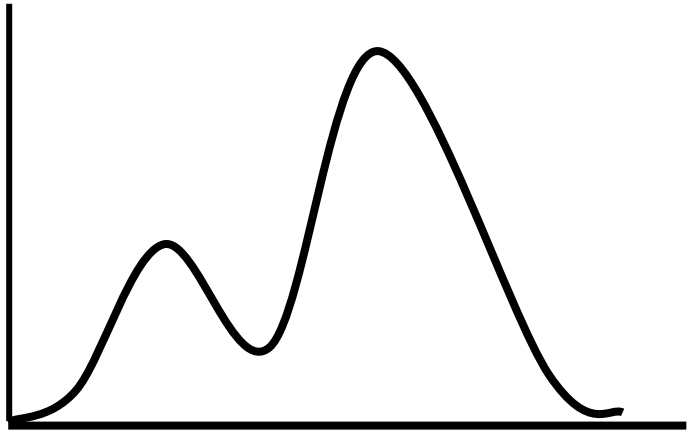
- Random start key generated
- Offset by selectivity for end key
- Remove keys in range from pool
- Operation output to workload file

```
V = V.sort() //sort vector
Index start_idx = RandomIndexGenerator(|V|*(1-Y), dist)
Key start_key = V.get(start_idx)//random start key
Index end_idx = start_idx+|V|*Y//offset by Y for end key
Key end_key = V.get(end_idx)
for idx in range(start_idx, end_idx):
        HS.remove(V.get(idx))
V.remove(start_idx,end_idx) //remove all keys in range from pool
workload_file.output("RangeQuery", start_key, end_key)
```

KVBench Suite



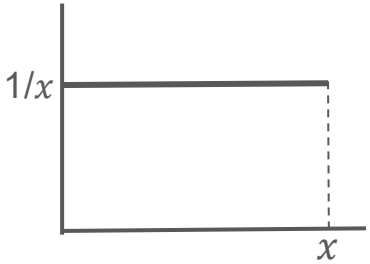
operations



distributions

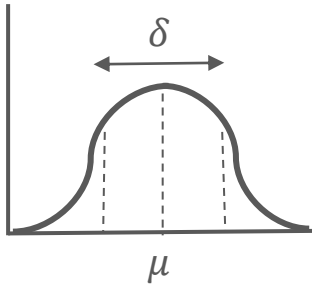
Distributions

Uniform



- uniform_int_distribution() in C++
- random keys for update and non-empty point queries

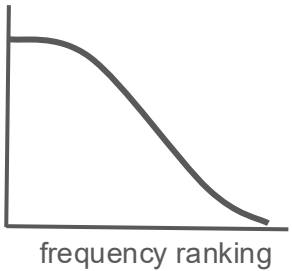
Normal



- normal_distribution in C++
- mean percentile for μ index and scaled δ
- keys chosen from vector accordingly

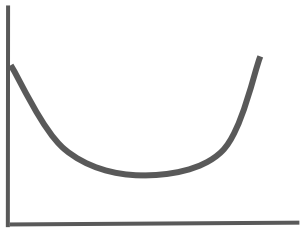
Distributions

Zipfian



- Shuffle keys randomly
- Index selected according to Zipfian distribution

Beta



- Gamma_distribution in C++
- 2 gamma to simulate a beta

Mixed Workloads

thresholding

- five thresholds
- # inserts before other operations

interleaving

- deletes/inserts
- vector updated, sorted
- probability recalculated

preloading

- existing insert-only workload
- read all inserts for HS and V

BOSTON UNIVERSITY Brandeis UNIVERSITY

The International Workshop on Testing Database Systems dbtest.io

ACM SIGMOD

Benchmark Workloads		KVB-I	KVB-II	KVB-III	KVB-IV	KVB-V
Preloading/Interleaving/Sequential		P	I	P	P	S
Operations	Insert	-	50%	-	-	95%
	Update	-	25%	50%	50%	-
	Non-empty point query	20%	-	25%	-	5%
	Empty point query	80%	15%	25%	-	-
	Range query	-	-	-	-	-
	Point delete	-	10%	-	-	-
	Range delete	-	-	-	50%	-
Distributions	Insert distribution	-	uniform	-	-	Zipfian
	Update distribution	-	uniform	Zipfian	Zipfian	-
	Non-empty pq distribution	beta	uniform	uniform	-	uniform
	Empty pq distribution	uniform	-	uniform	-	-

BOSTON UNIVERSITY

Brandeis UNIVERSITY

dbtest.io

ACM SIGMOD

Benchmark Setup

Metrics

- Query latency
- Throughput

Workload

- entry size: 1024B
- # I, Q, U = 10M

System setup

- 2 Intel Xeon CPU@2.1GHz
- 375 GB RAM, 350 GB SSD

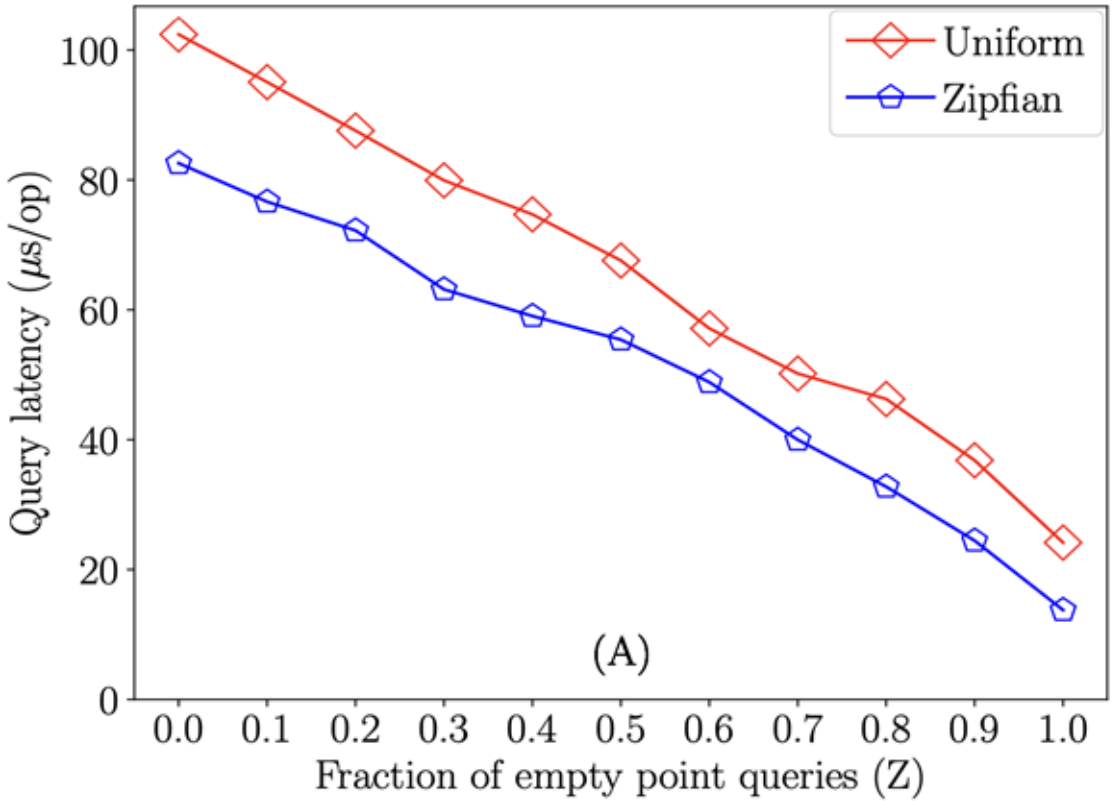
Index setup

- RocksDB
- LSM Tree (T =4 , P=4096, B=8)

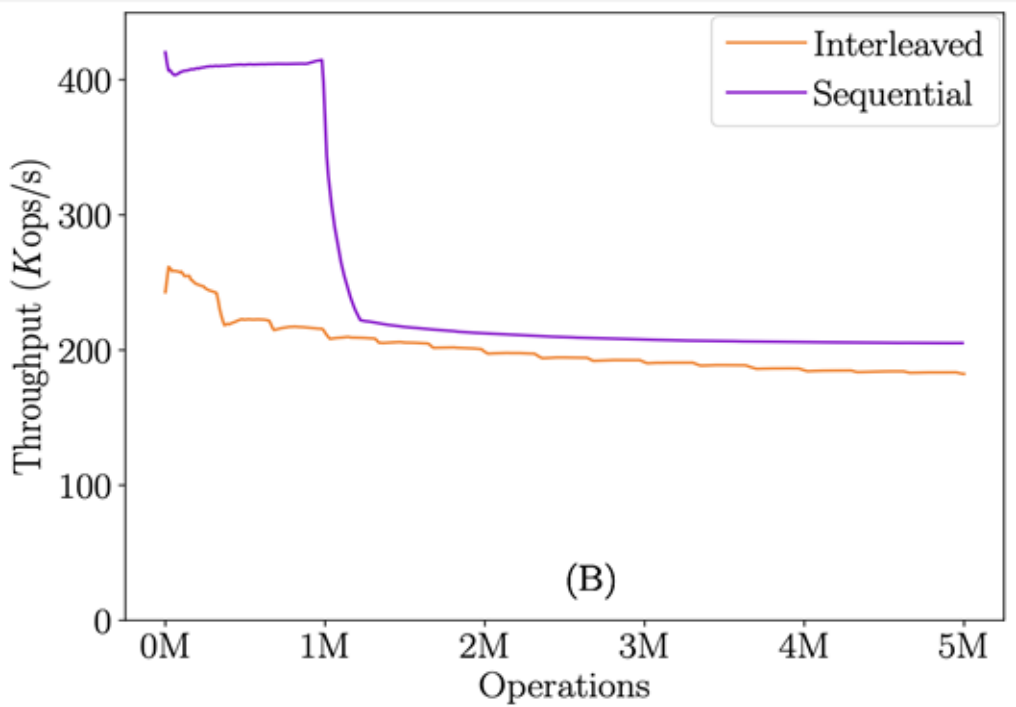
BOSTON UNIVERSITY Brandeis UNIVERSITY

The International Workshop on Testing Database Systems dbtest.io

ACM SIGMOD



The average latency per query decreases as there are more empty point queries for different distributions.



Workload

U = 0.9M
D = 0.1M
Q = 4M
I = 10M (P)

The throughput differs substantially between interleaved and sequential workloads even if they have the same workload composition.

Future Work

increase efficiency of generating workload

more operations (queries on deleted keys, repeatedly deleted keys)

different selectivity and distributions for different range queries

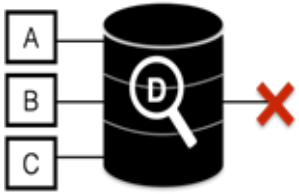
integrate the workload generator into a general framework/ existing benchmark framework to test more databases

BOSTON UNIVERSITY Brandeis UNIVERSITY

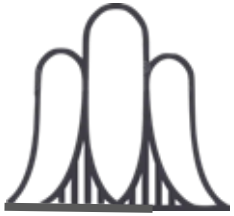
The International Workshop on Testing Database Systems dbtest.io

ACM SIGMOD

KVBench: A Key-Value Benchmarking Suite



Distinguish between empty and non-empty point lookups



Different operations can have different distributions

Deletes can be mixed with other operation types

BOSTON UNIVERSITY

Brandeis UNIVERSITY

The International Workshop on Testing Database Systems
dbtest.io

ACM SIGMOD

Thanks!


BOSTON
UNIVERSITY


Brandeis
UNIVERSITY


The International Workshop on Testing Database Systems
dbtest.io


ACM SIGMOD