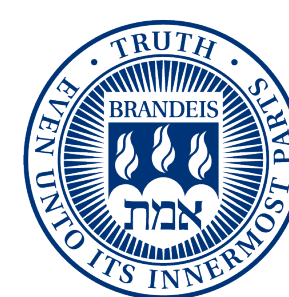


TexBench: Harnessing LLMs for Efficient Key-value Benchmarking

Shubham Kaushik

Abhishek Chanda

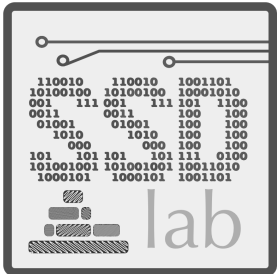
Subhadeep Sarkar



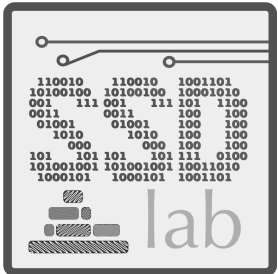
Brandeis
UNIVERSITY



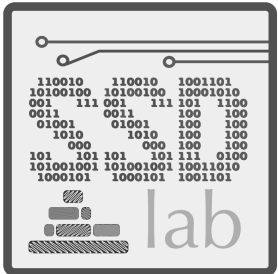
Key-Value Stores



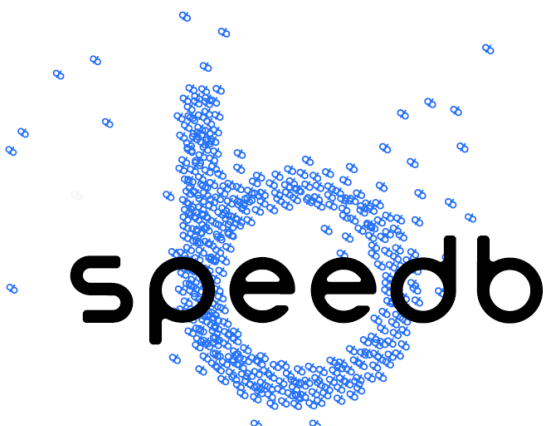
Key-Value Stores

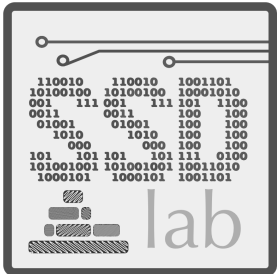


Key-Value Stores

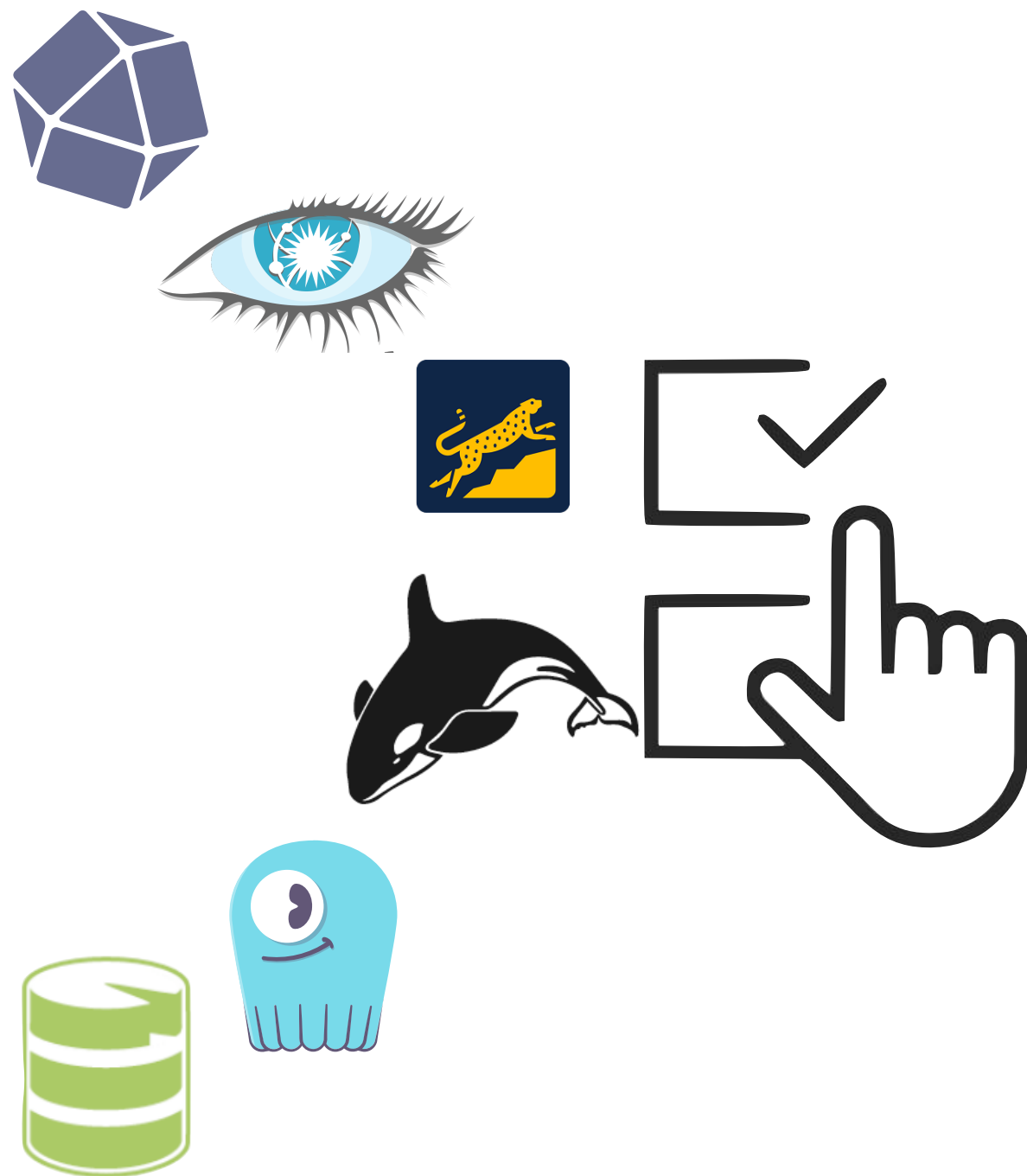


Key-Value Stores

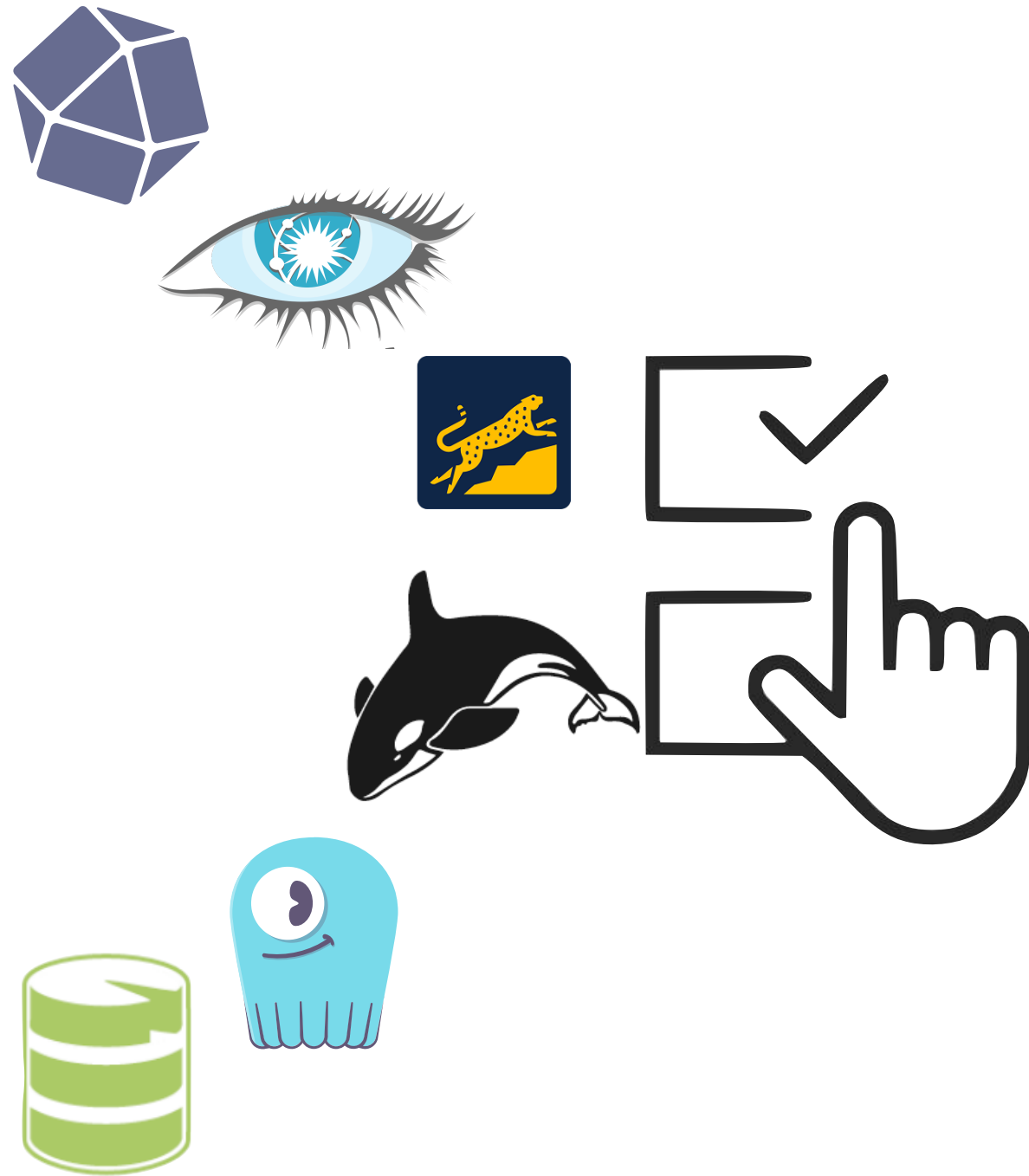




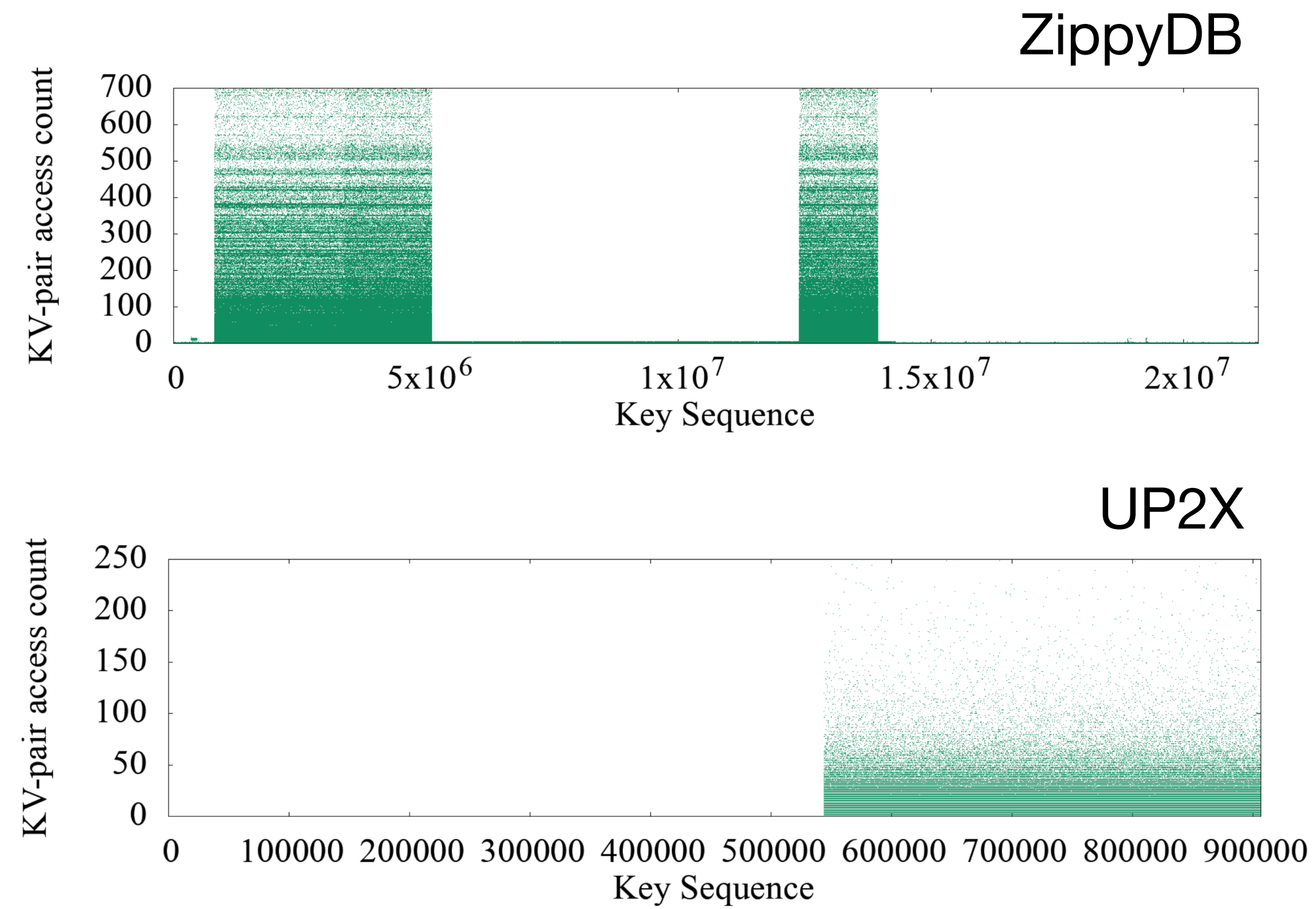
Brandeis
UNIVERSITY



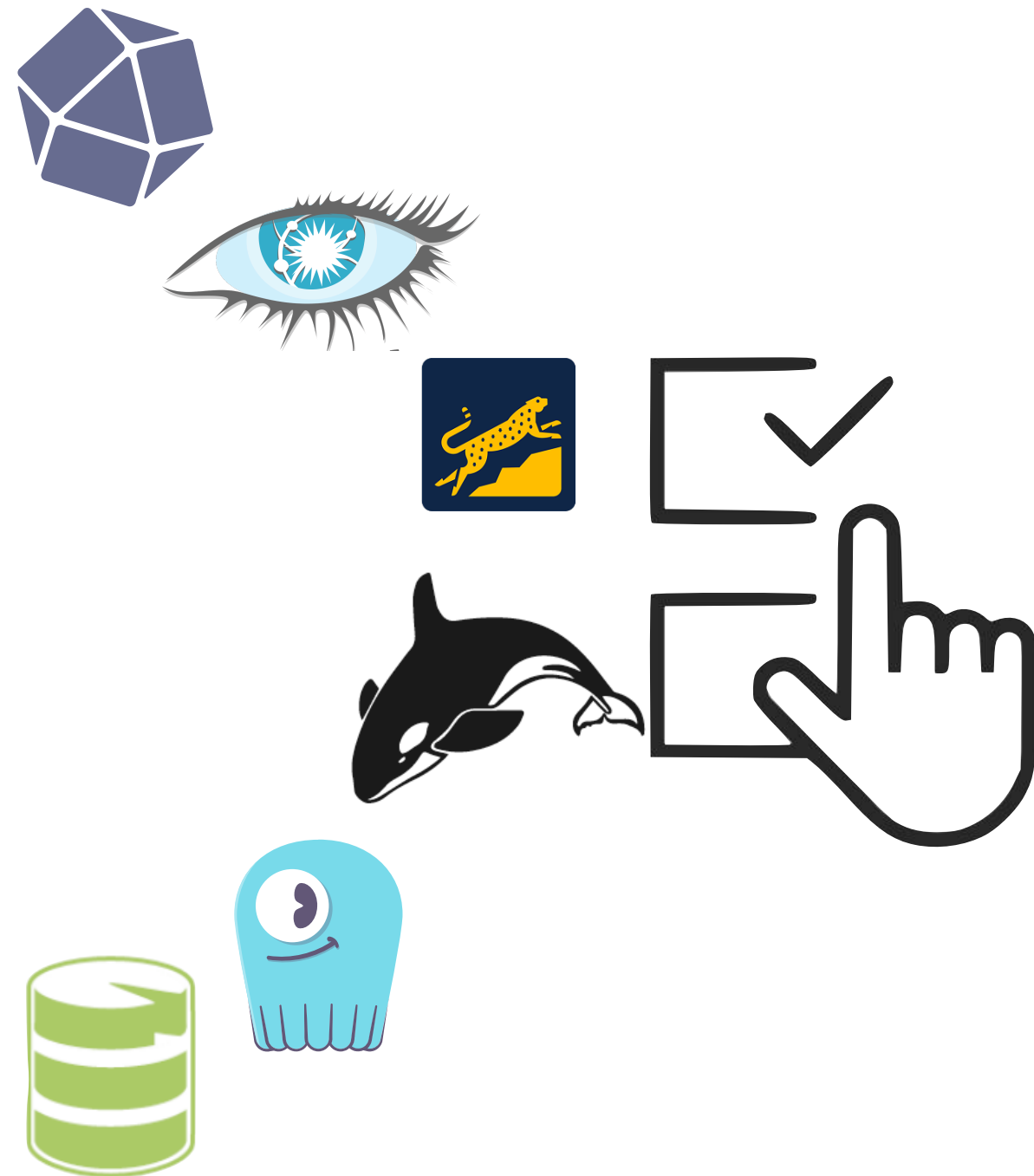
**which one fits for
the scenario at hand?**



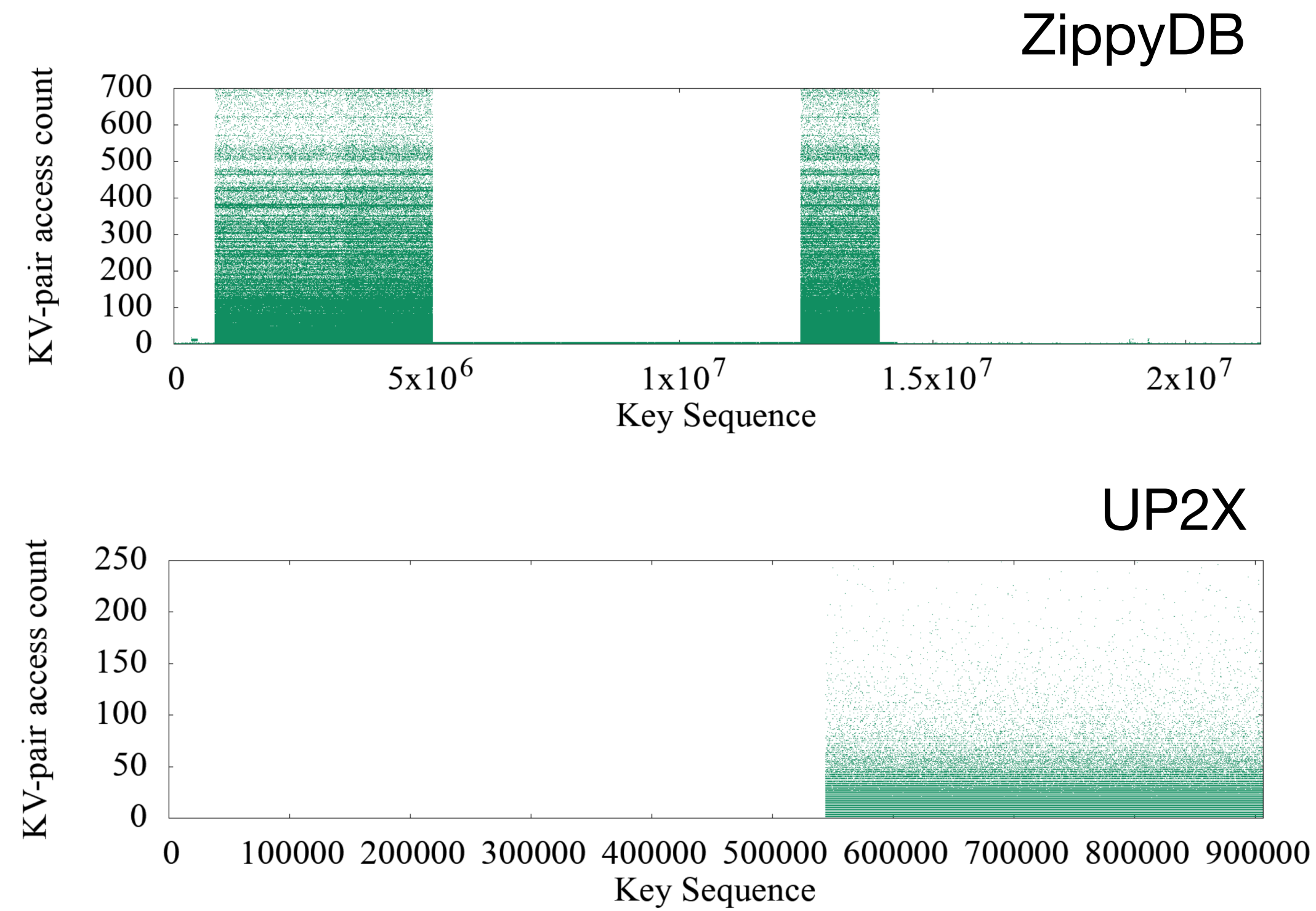
**which one fits for
the scenario at hand?**



**diverse workload scenarios
in real world**

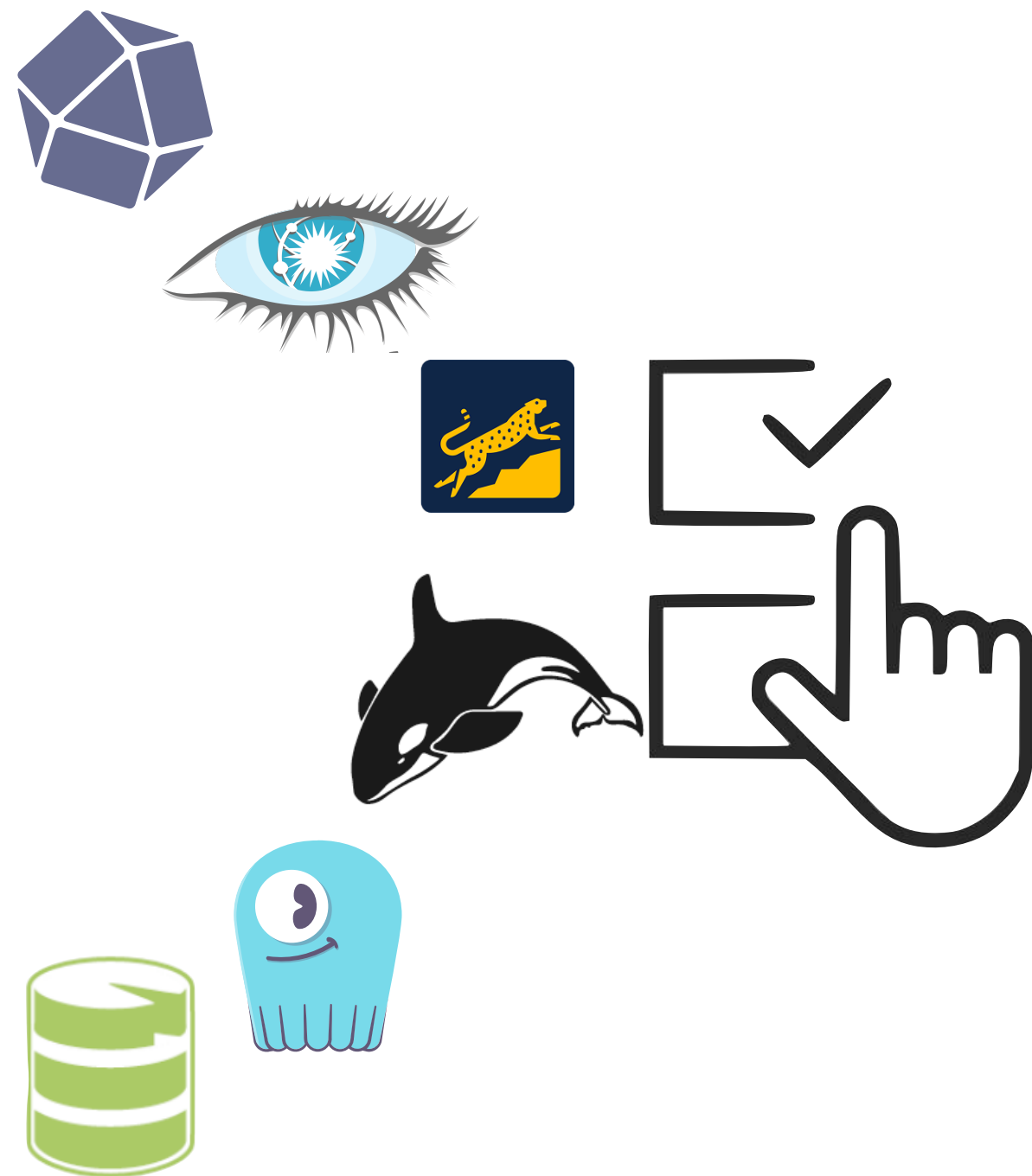


which one fits for
the scenario at hand?

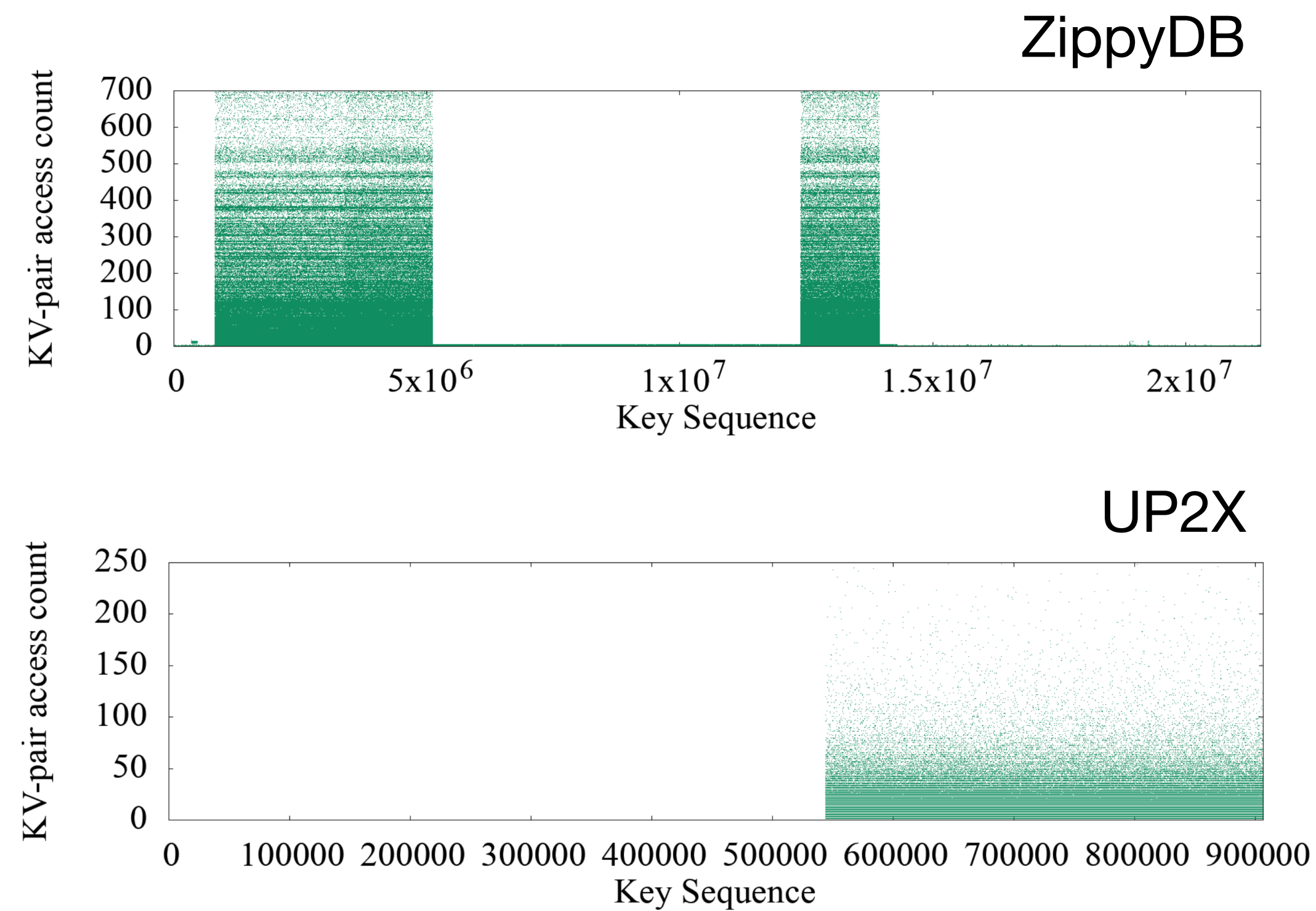


diverse workload scenarios
in real world

How to **choose** the
right database?



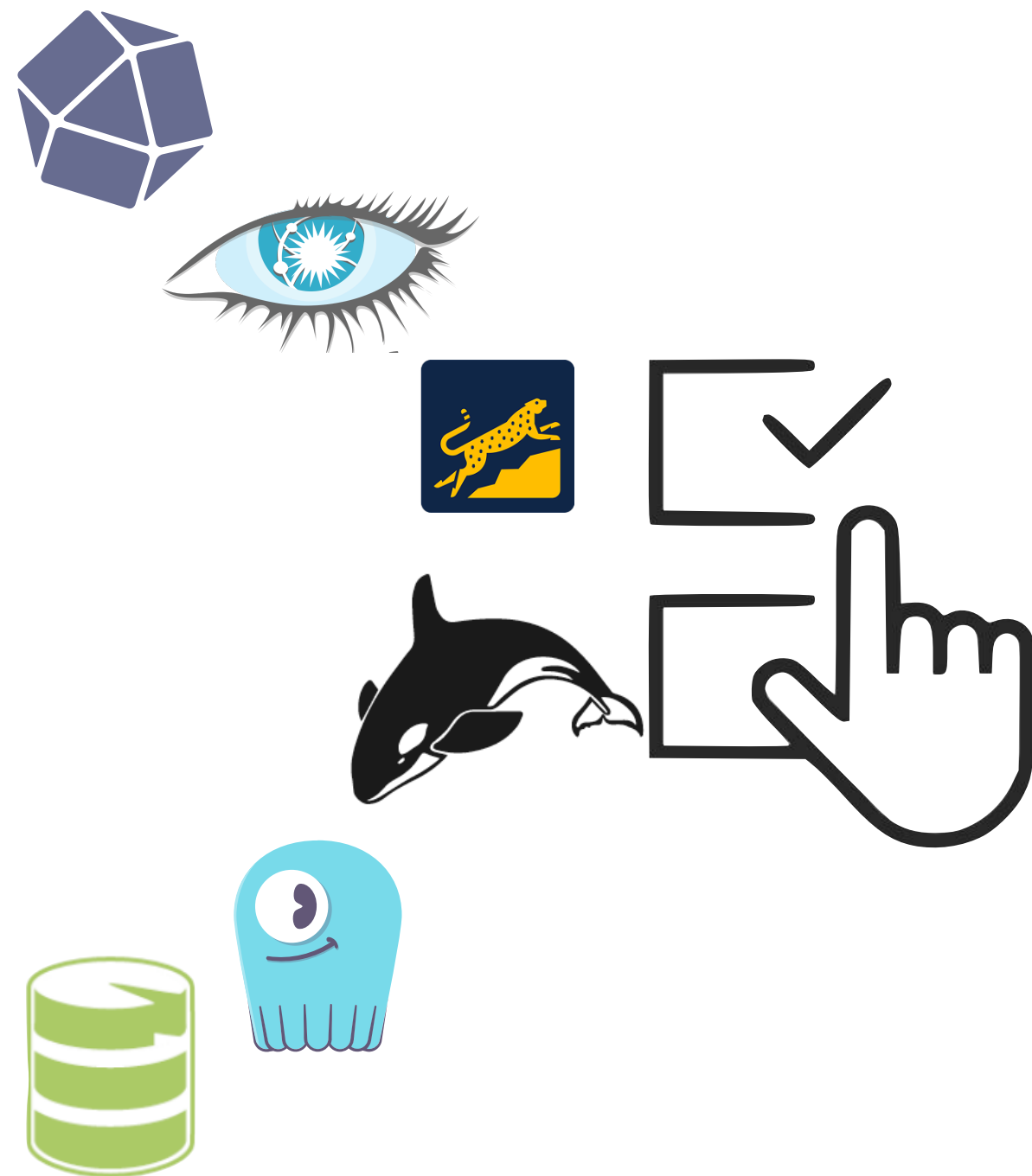
which one fits for
the scenario at hand?



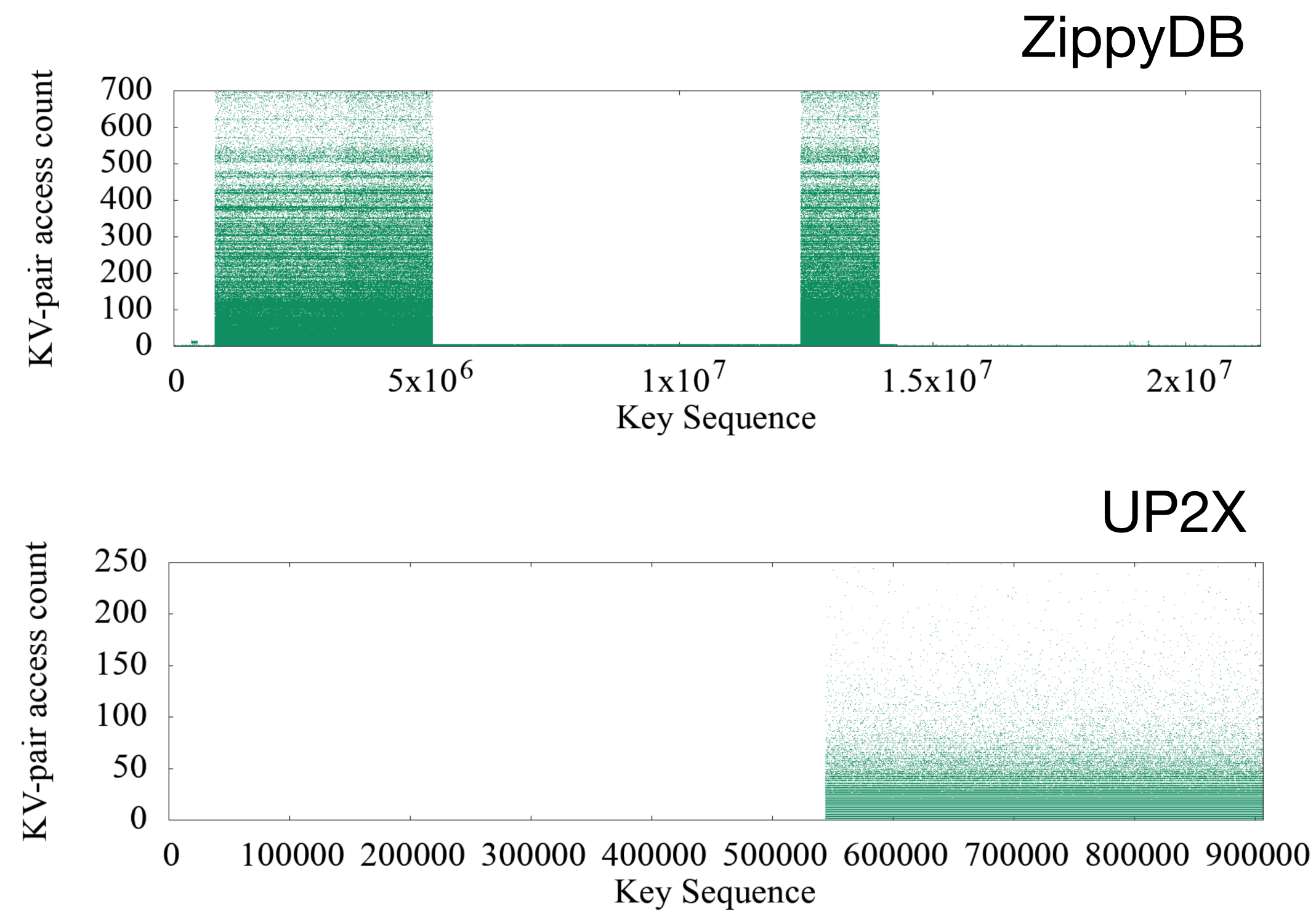
diverse workload scenarios
in real world

How to **choose** the
right database?

How to **tune** a
database?



which one fits for
the scenario at hand?

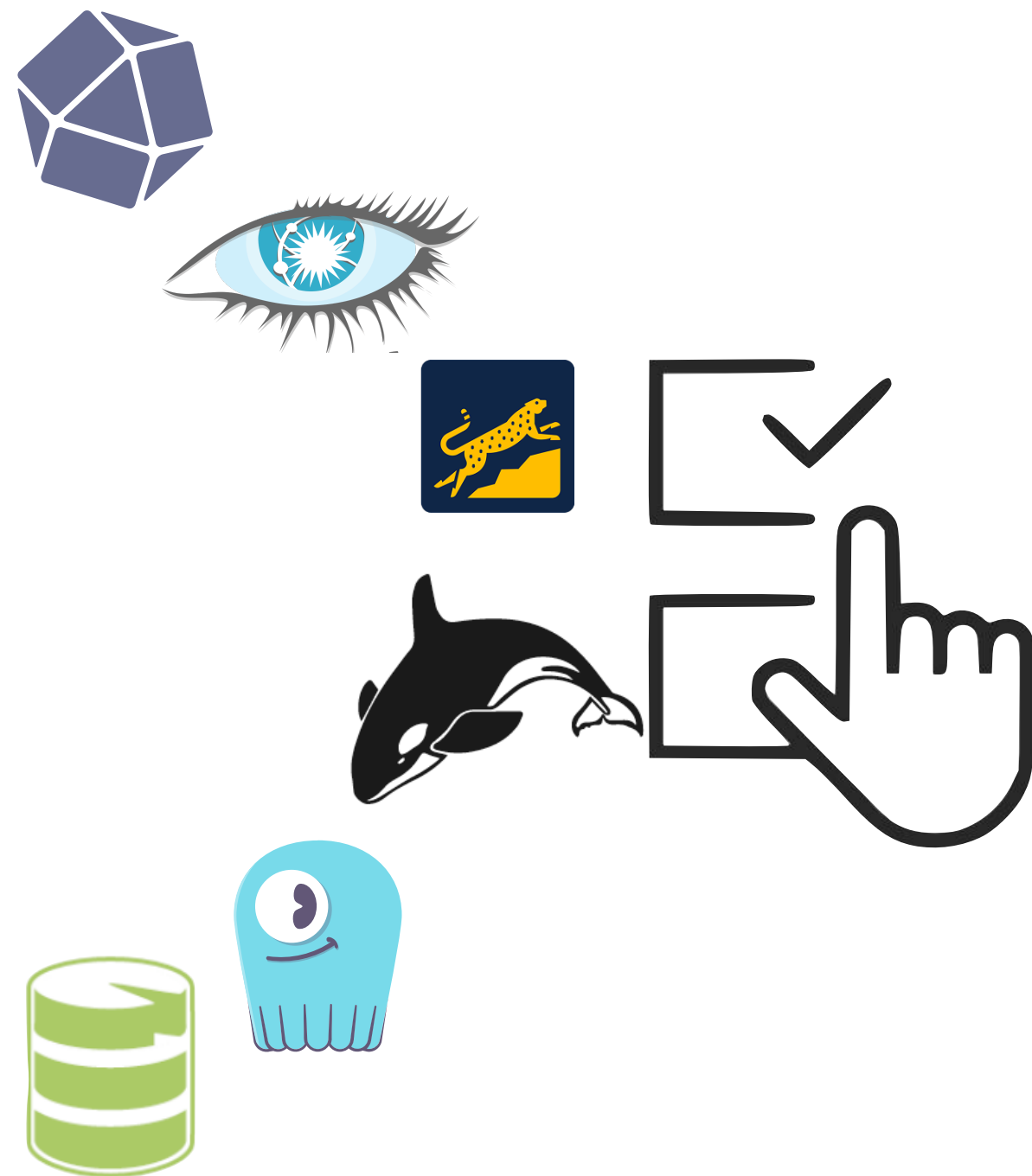


diverse workload scenarios
in real world

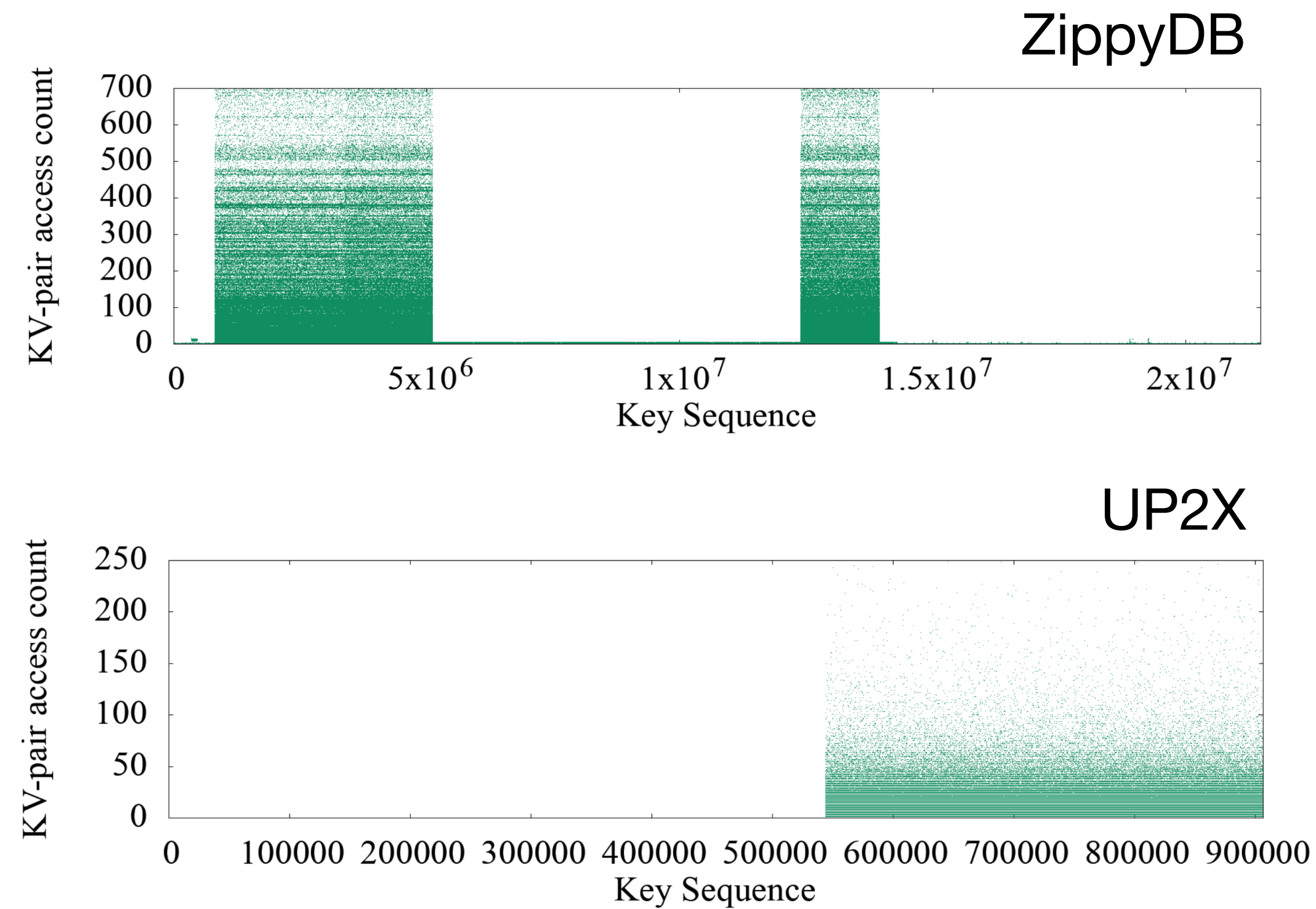
How to **choose** the
right database?

How to **tune** a
database?

How to **compare**
performances?



which one fits for
the scenario at hand?



diverse workload scenarios
in real world

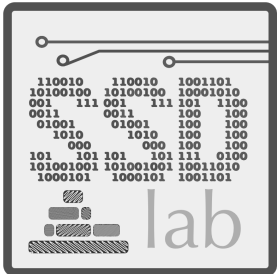
Benchmark databases

How to **choose** the
right database?

How to **tune** a
database?

How to **compare**
performances?

Benchmarks / Workload Generators



Benchmarks / Workload Generators

TPC-C
SIGMOD'93

TPC-DI
VLDB'14

TPCx-HS
TPCTC'14

TPC-E
TPCTC'09

Myriad
ASBD'11

TPC-DS
VLDB'06

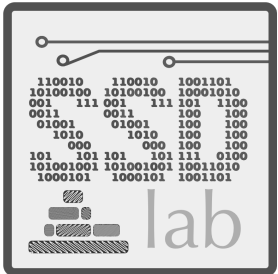
SSB
TPCTC'09

OLTPBench
VLDB'13

MUDD
WOSP'04

PDGF
DBTest'11

relational



Benchmarks / Workload Generators

TPC-C
SIGMOD'93

TPC-DI
VLDB'14

TPCx-HS
TPCTC'14

YCSB
SoCC'10

TPC-E
TPCTC'09

Myriad
ASBD'11

db_bench
RocksDB

TPC-DS
VLDB'06

SSB
TPCTC'09

OLTPBench
VLDB'13

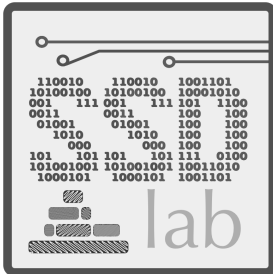
MUDD
WOSP'04

PDGF
DBTest'11

KVBench
DBTest'24

relational

key-value benchmarks



Workload Generators

YCSB

SoCC'10

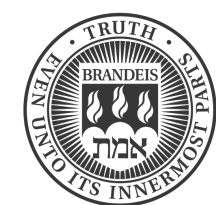
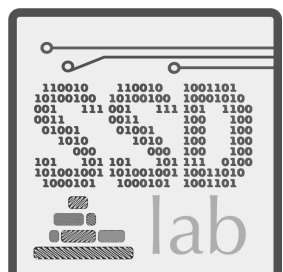
db_bench

RocksDB

KVBench

DBTest'24

key-value benchmarks



Brandeis
UNIVERSITY

Workload Generators

YCSB

SoCC'10

specifications

put

get

scan

delete

...

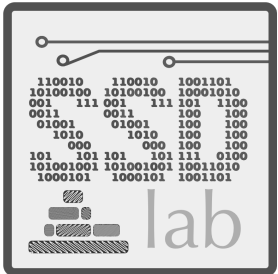
db_bench

RocksDB

KVBench

DBTest'24

key-value benchmarks



Workload Generators

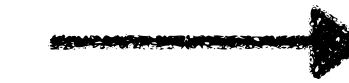
YCSB
SoCC'10

db_bench
RocksDB

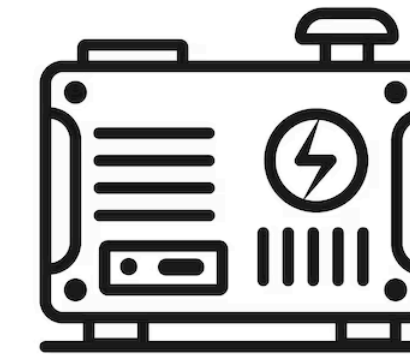
KVBench
DBTest'24

specifications

put	
get	
scan	
delete	
...	

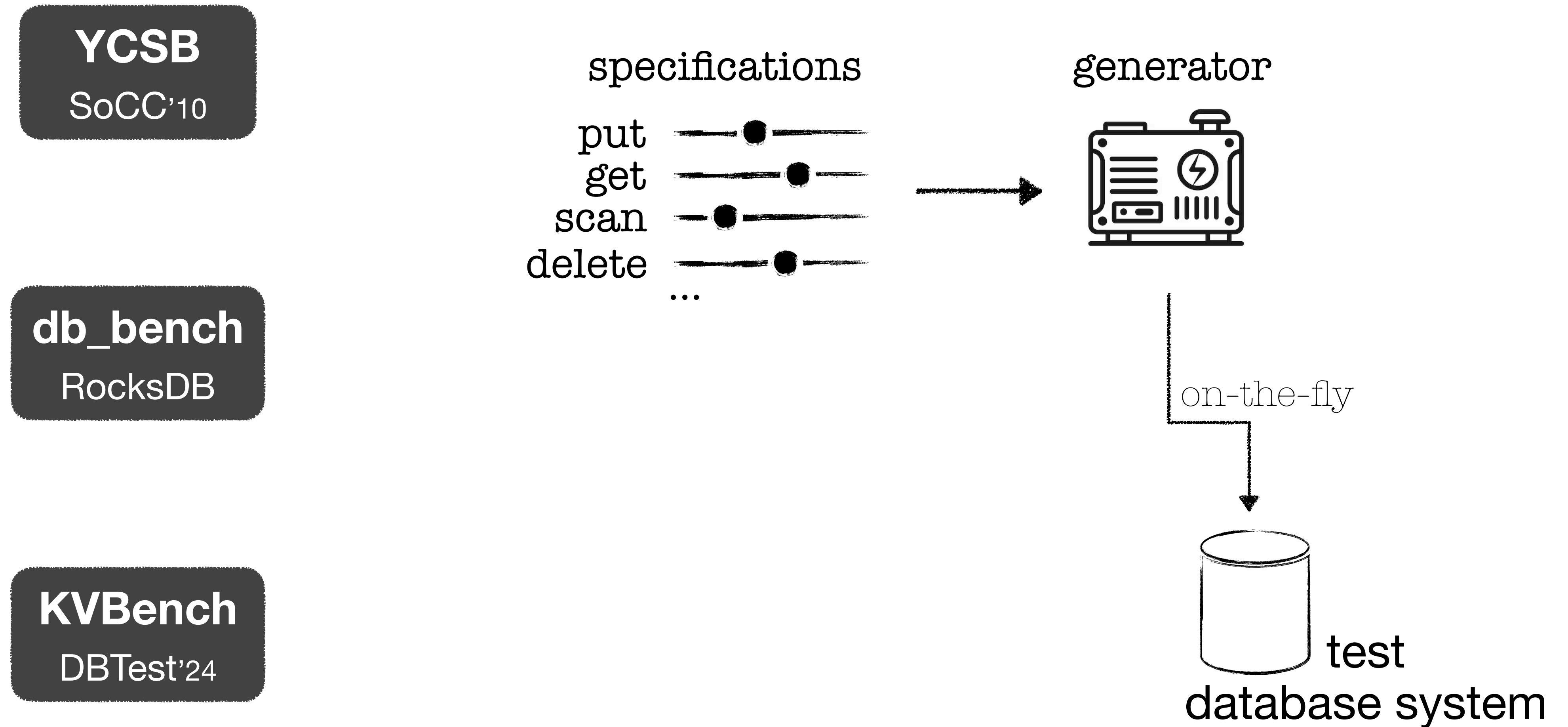


generator



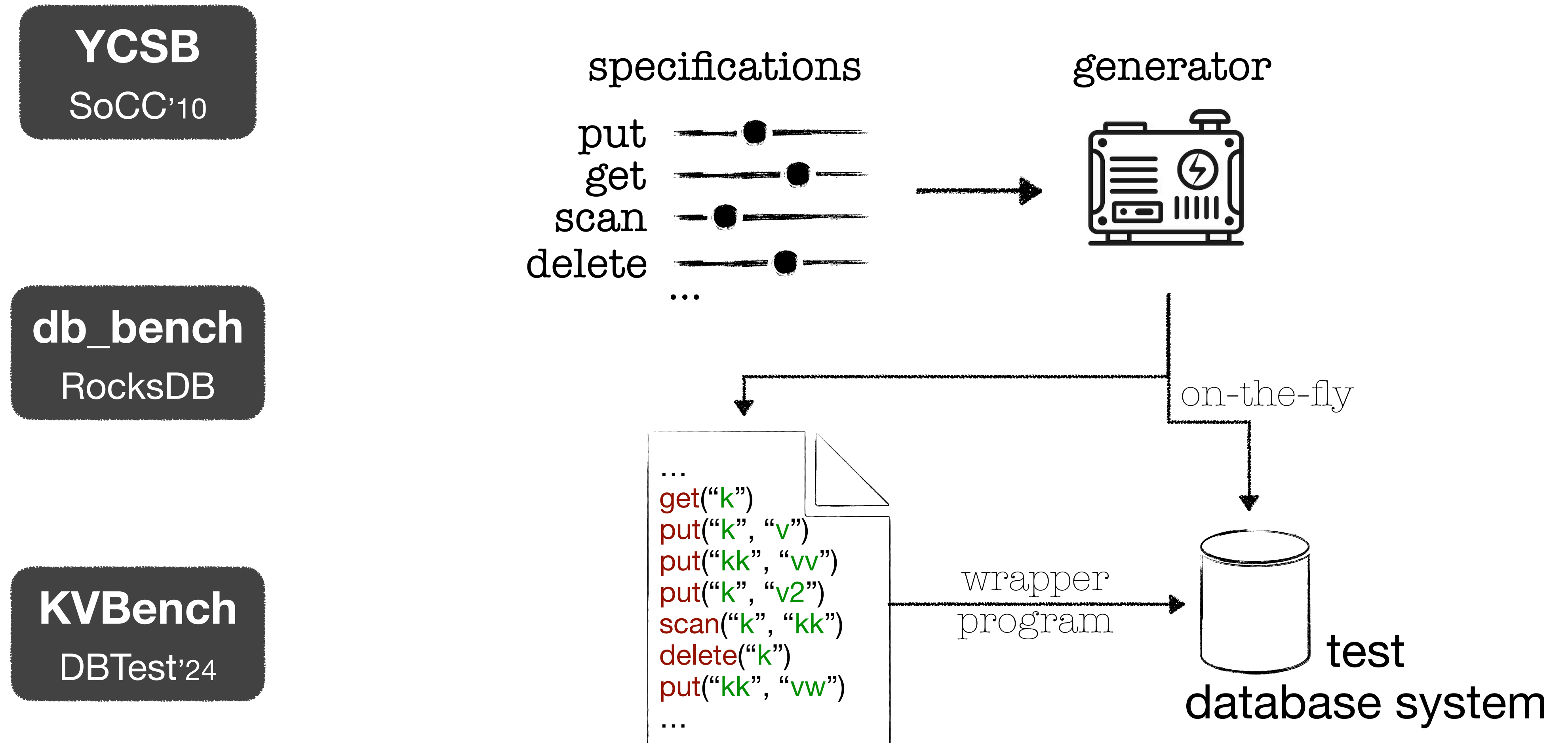
key-value benchmarks

Workload Generators



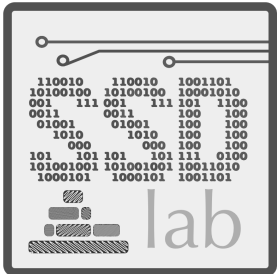
key-value benchmarks

Workload Generators



key-value benchmarks

Workload Generators: **Features**



Workload Generators: **Features**



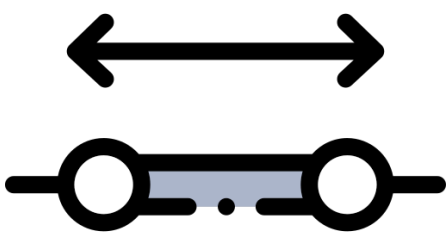
insert



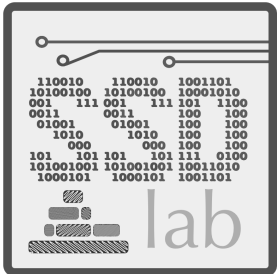
update



point query



range query



Workload Generators: **Features**



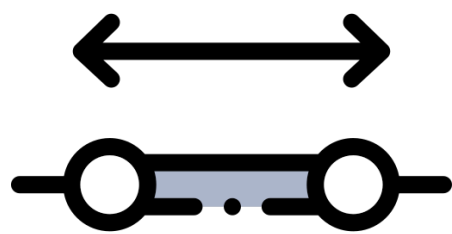
insert



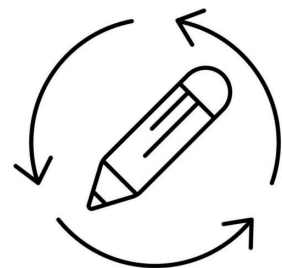
update



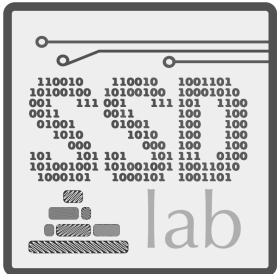
point query



range query



read-modify-write



Workload Generators: **Features**



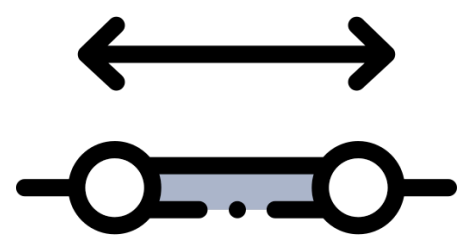
insert



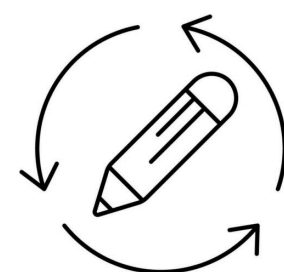
update



point query



range query



read-modify-write

db_bench

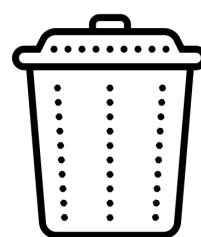
KVBench



delete

db_bench

KVBench



range delete

Workload Generators: **Features**



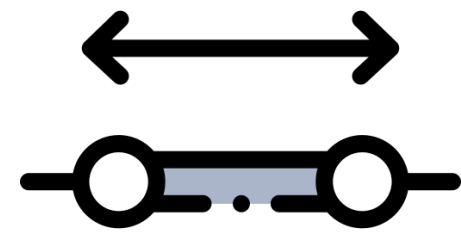
insert



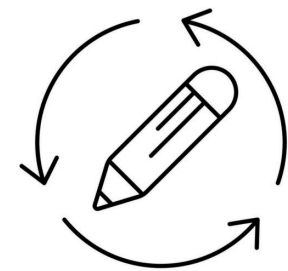
update



point query



range query



read-modify-write

db_bench

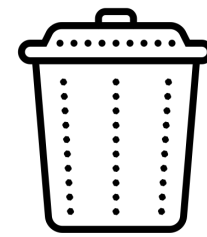
KVBench



delete

db_bench

KVBench



range delete

db_bench

KVBench



empty
point query

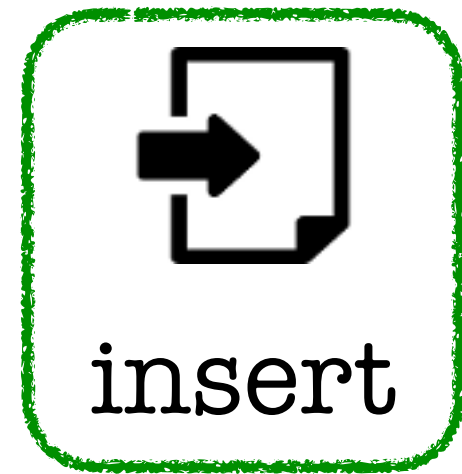
KVBench



empty
point delete

supports **different operations**

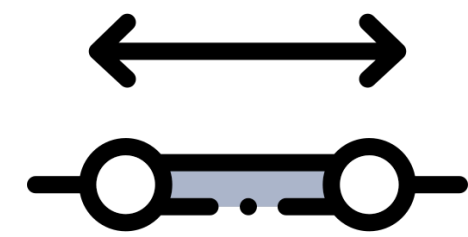
Workload Generators: Features



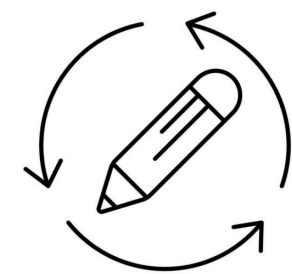
update



point query



range query



read-modify-write

db_bench

KVBench



delete

db_bench

KVBench



range delete

db_bench

KVBench



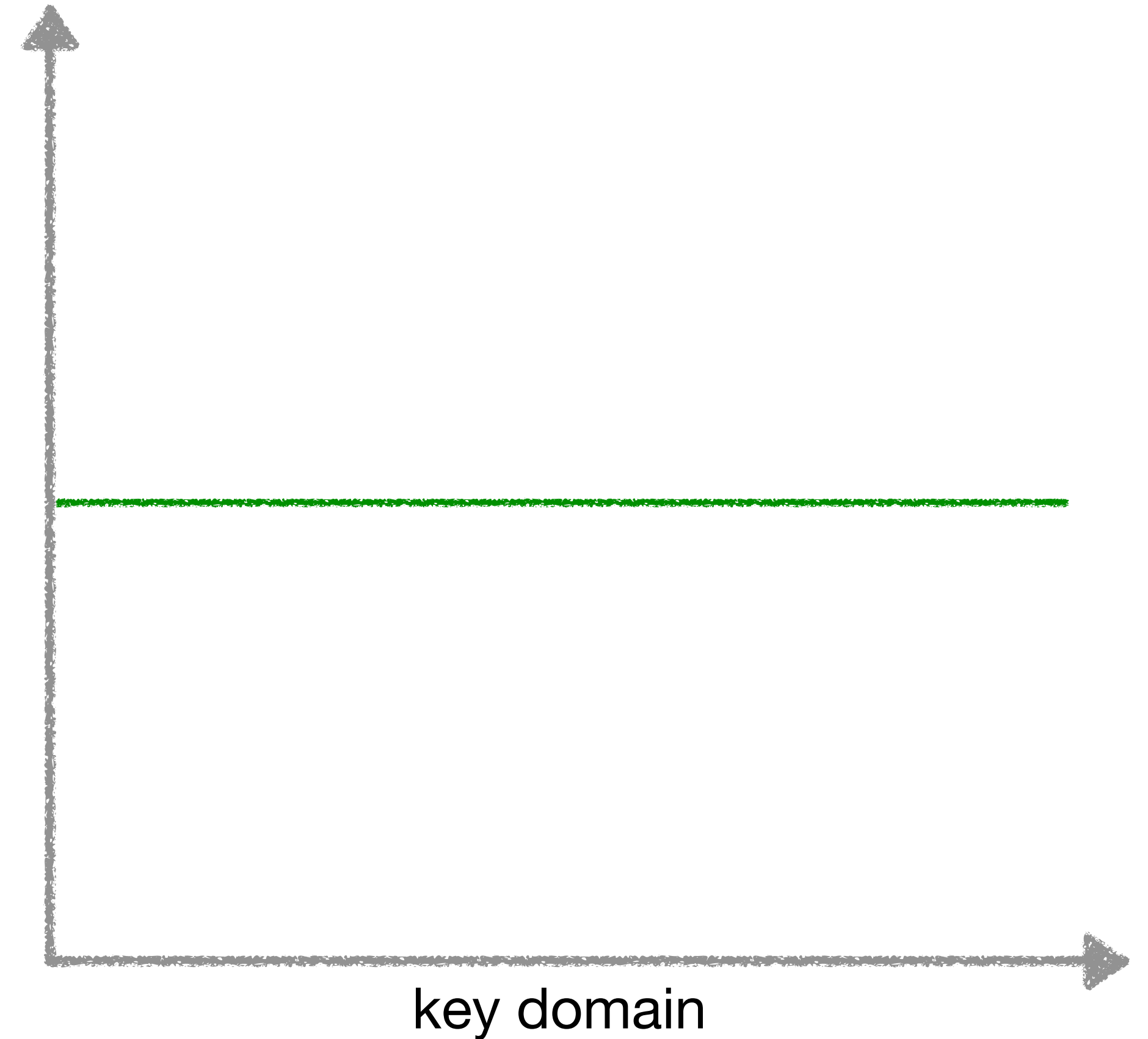
empty
point query

KVBench

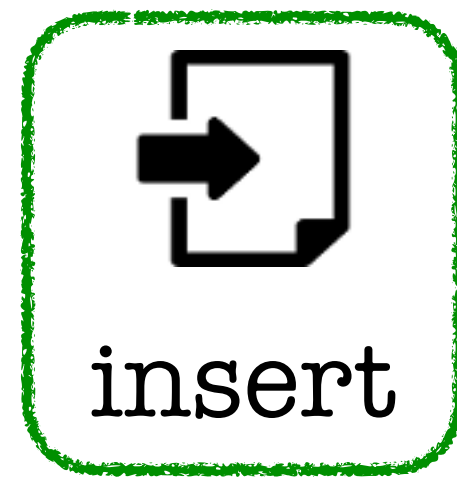


empty
point delete

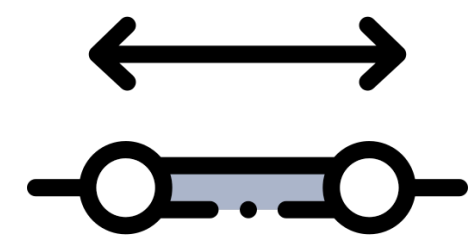
supports **different operations**



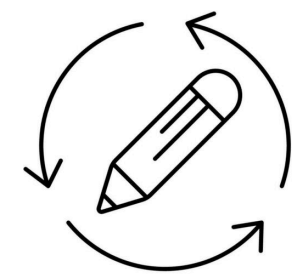
Workload Generators: Features



point query



range query



read-modify-write

db_bench

KVBench



delete

db_bench

KVBench



range delete

db_bench

KVBench



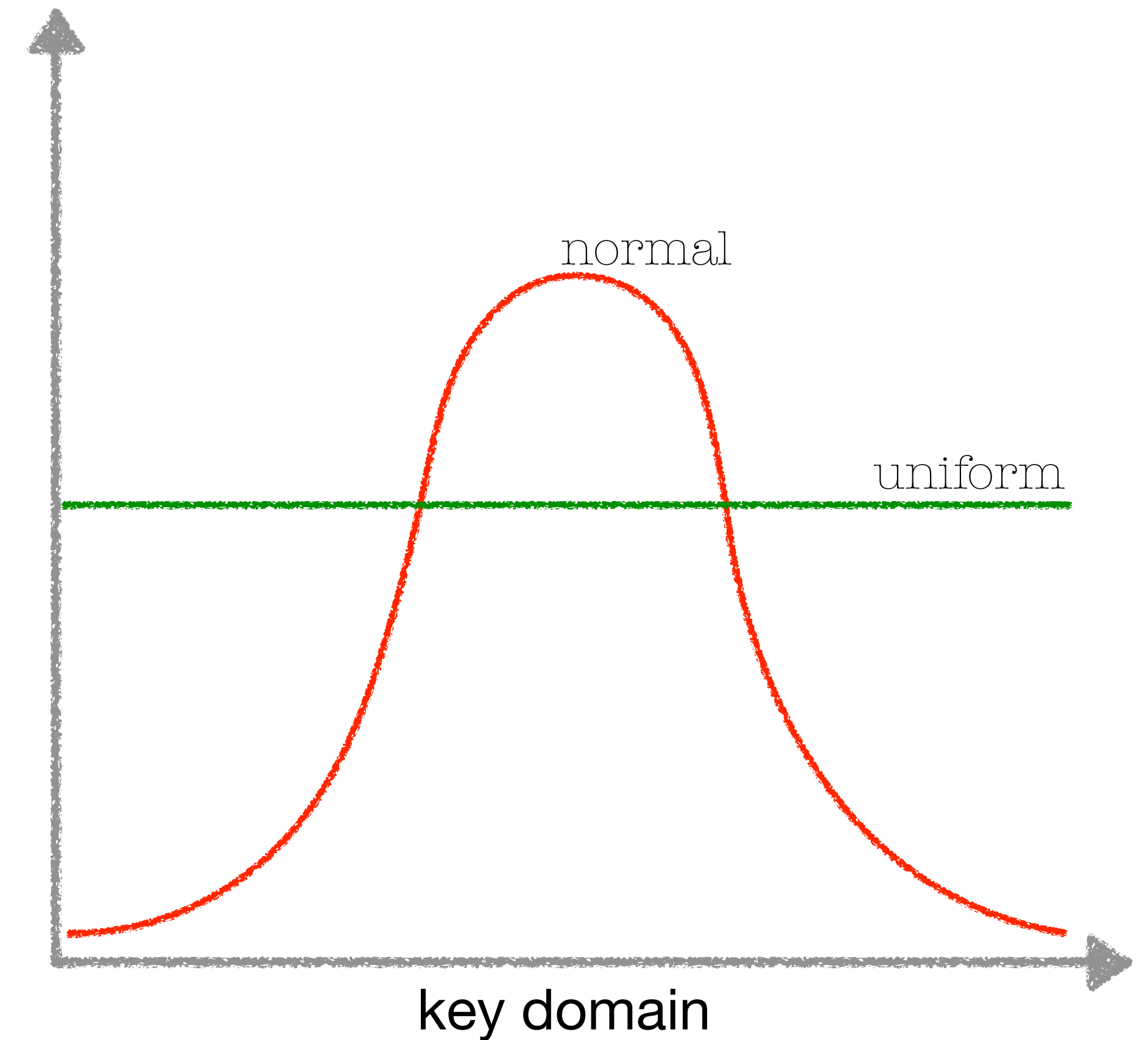
empty
point query

KVBench

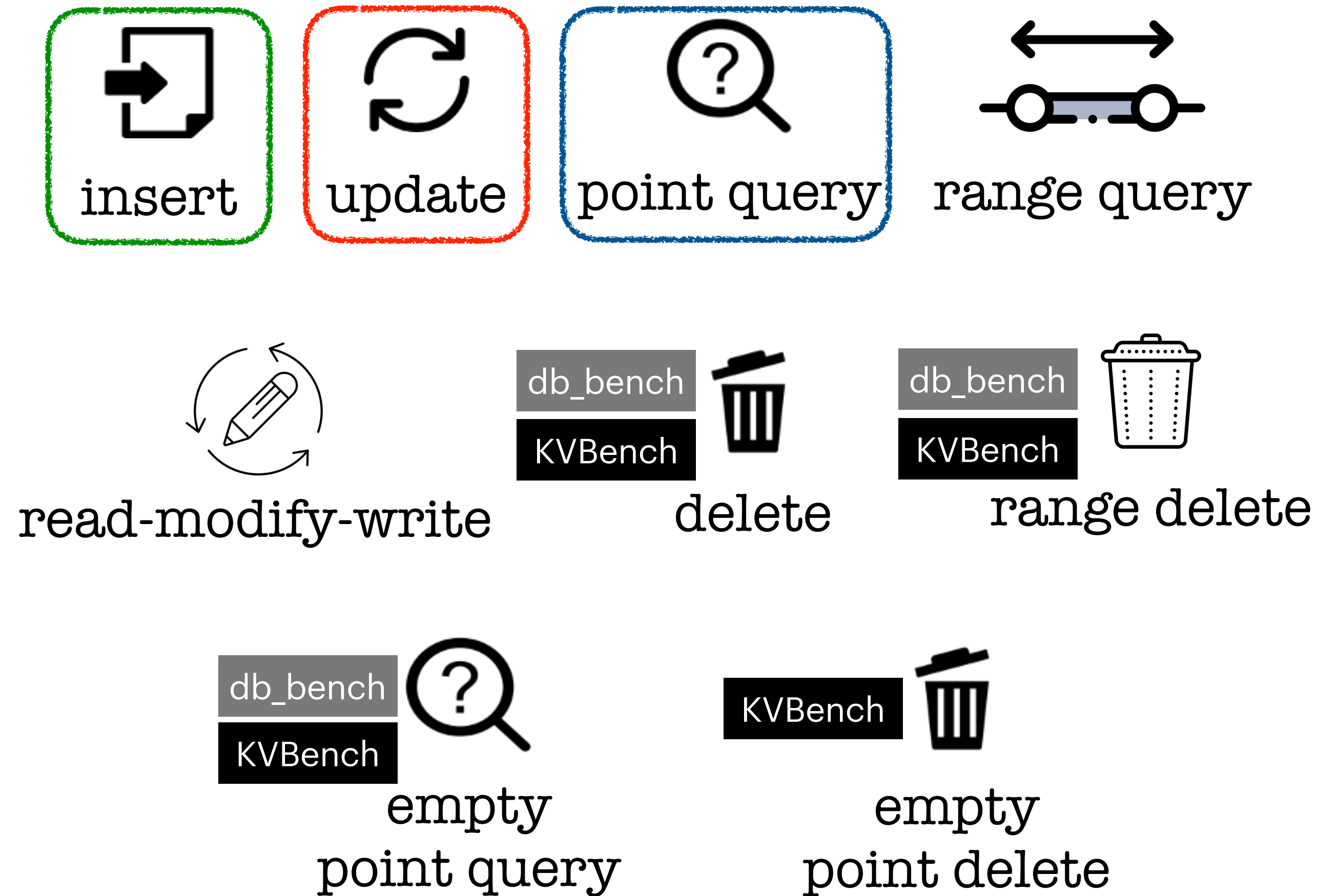


empty
point delete

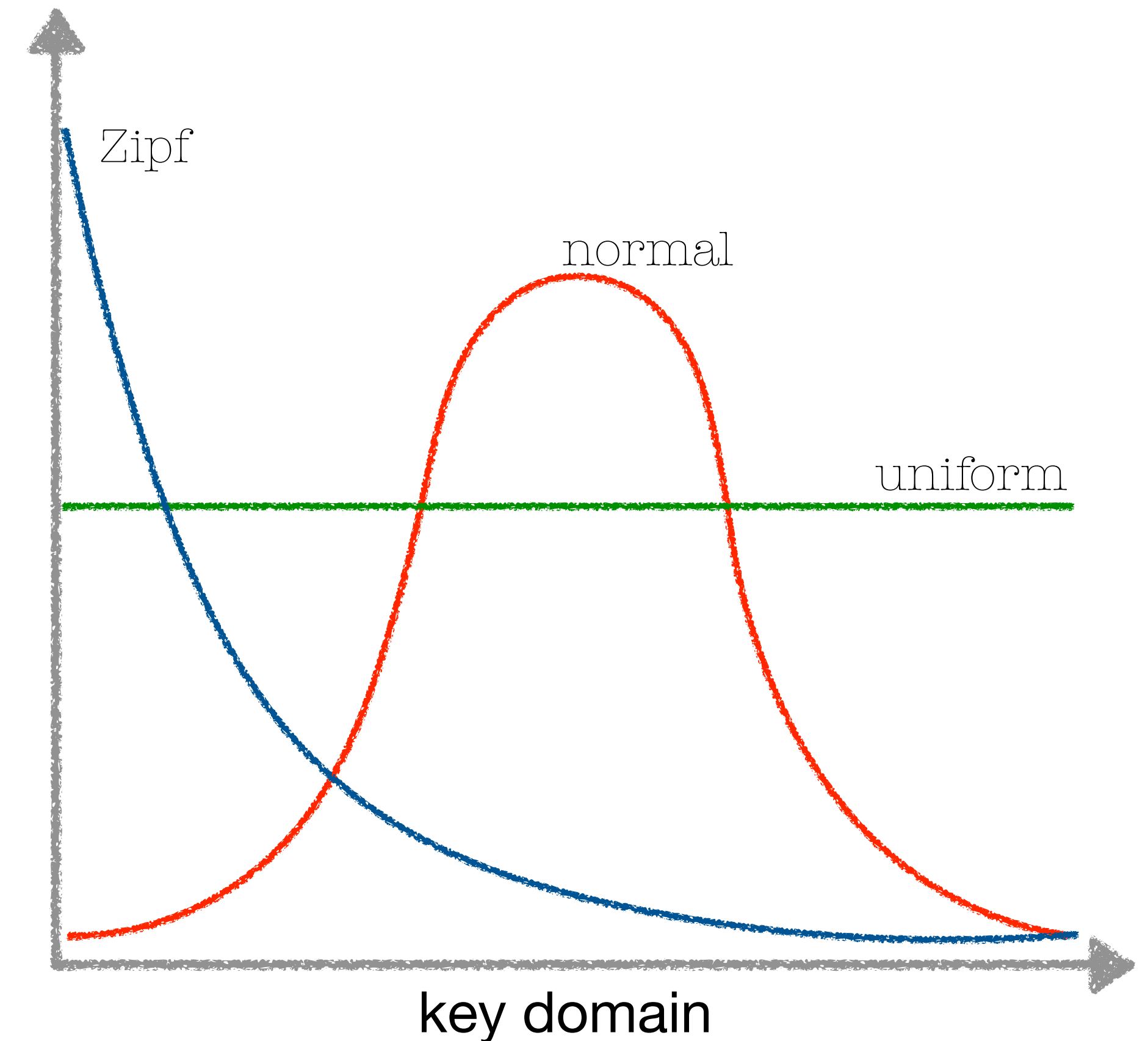
supports **different operations**



Workload Generators: Features

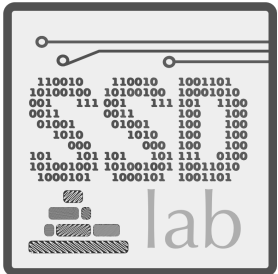
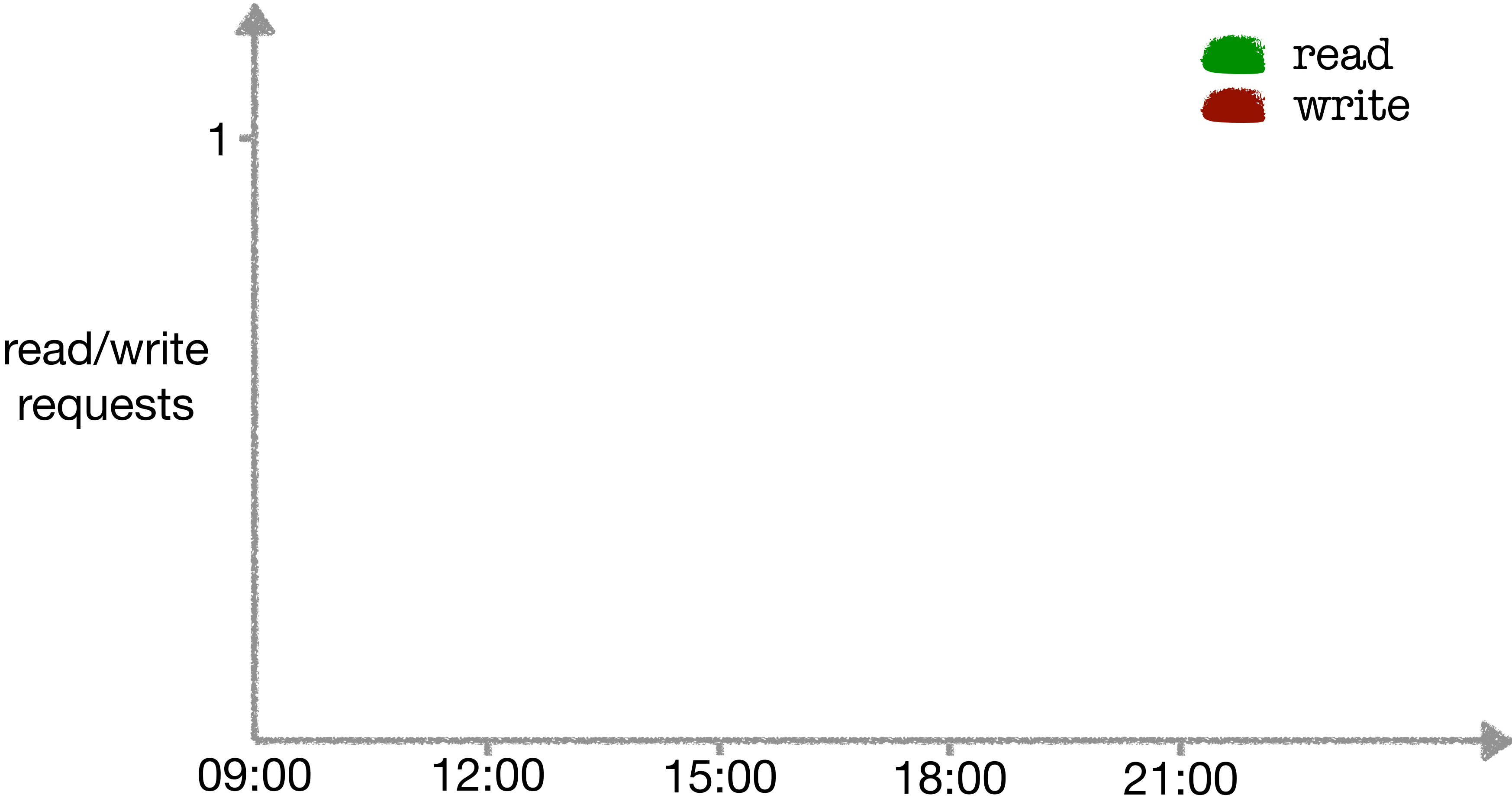


supports **different operations**

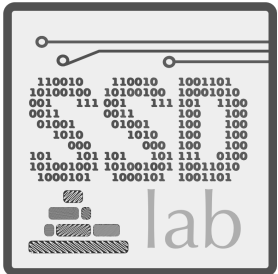
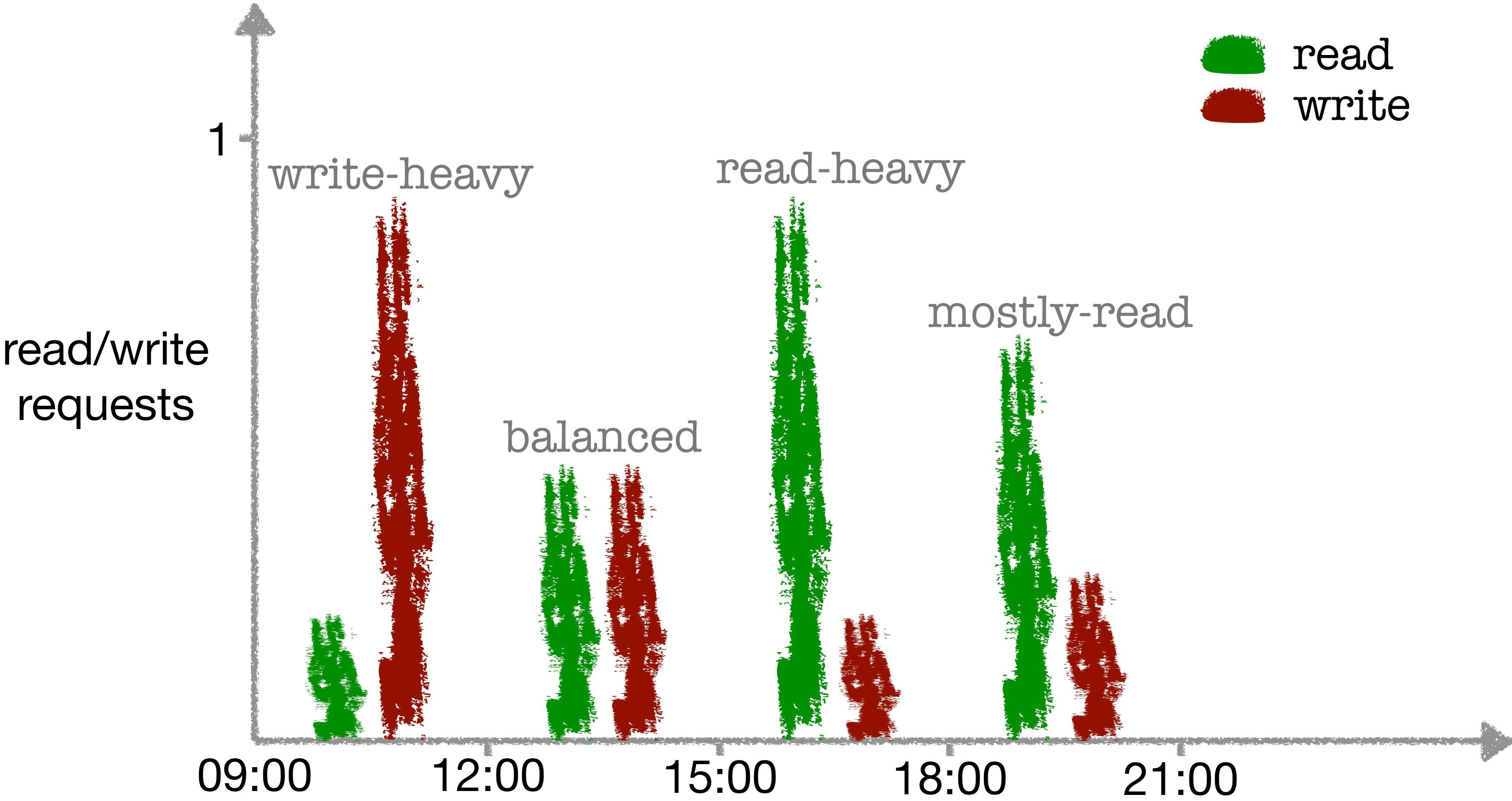


offers **flexible distribution**

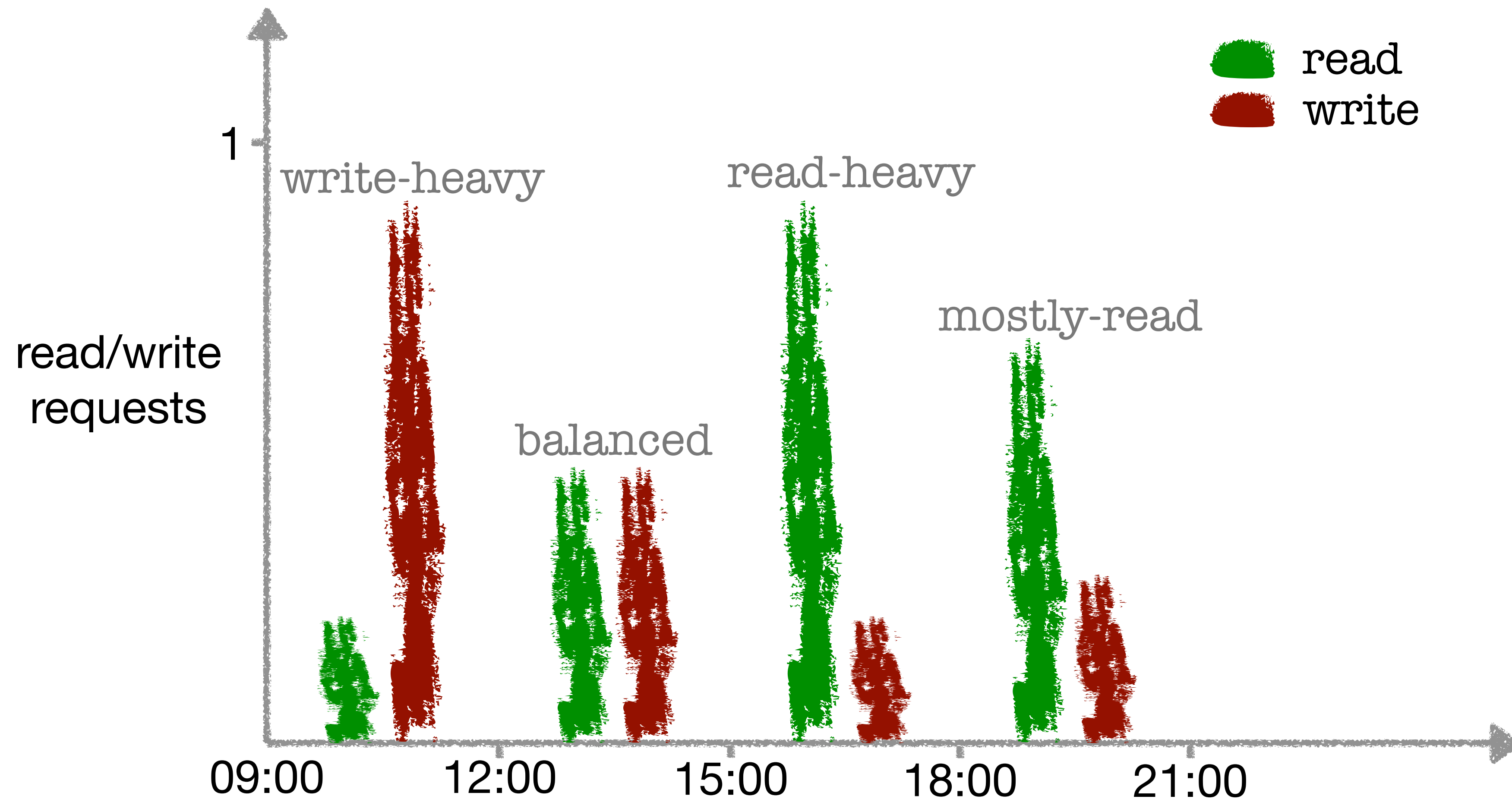
Real-world Workloads: Shifts Dynamically



Real-world Workloads: Shifts Dynamically

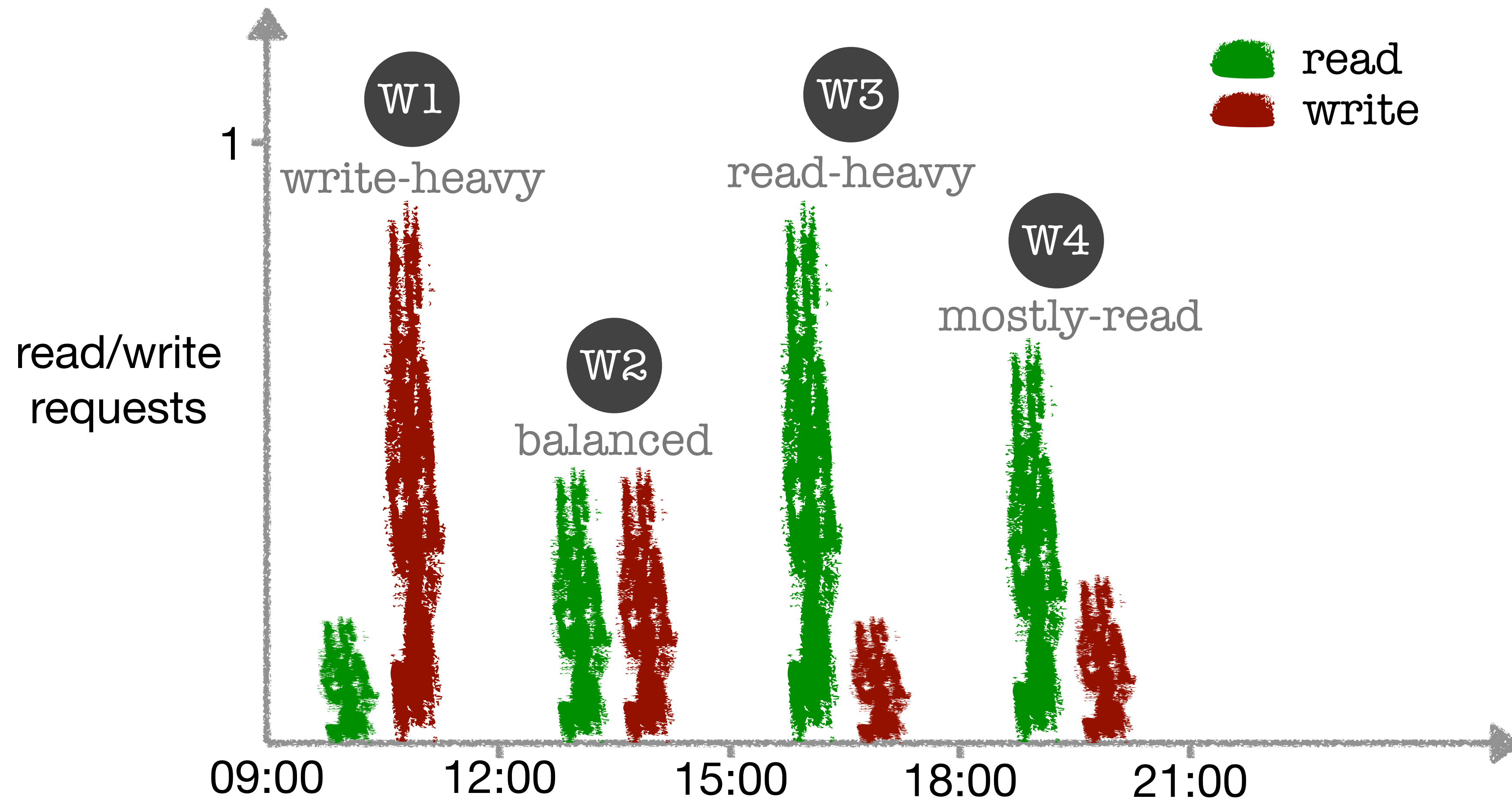


Real-world Workloads: Shifts Dynamically



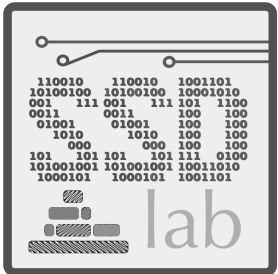
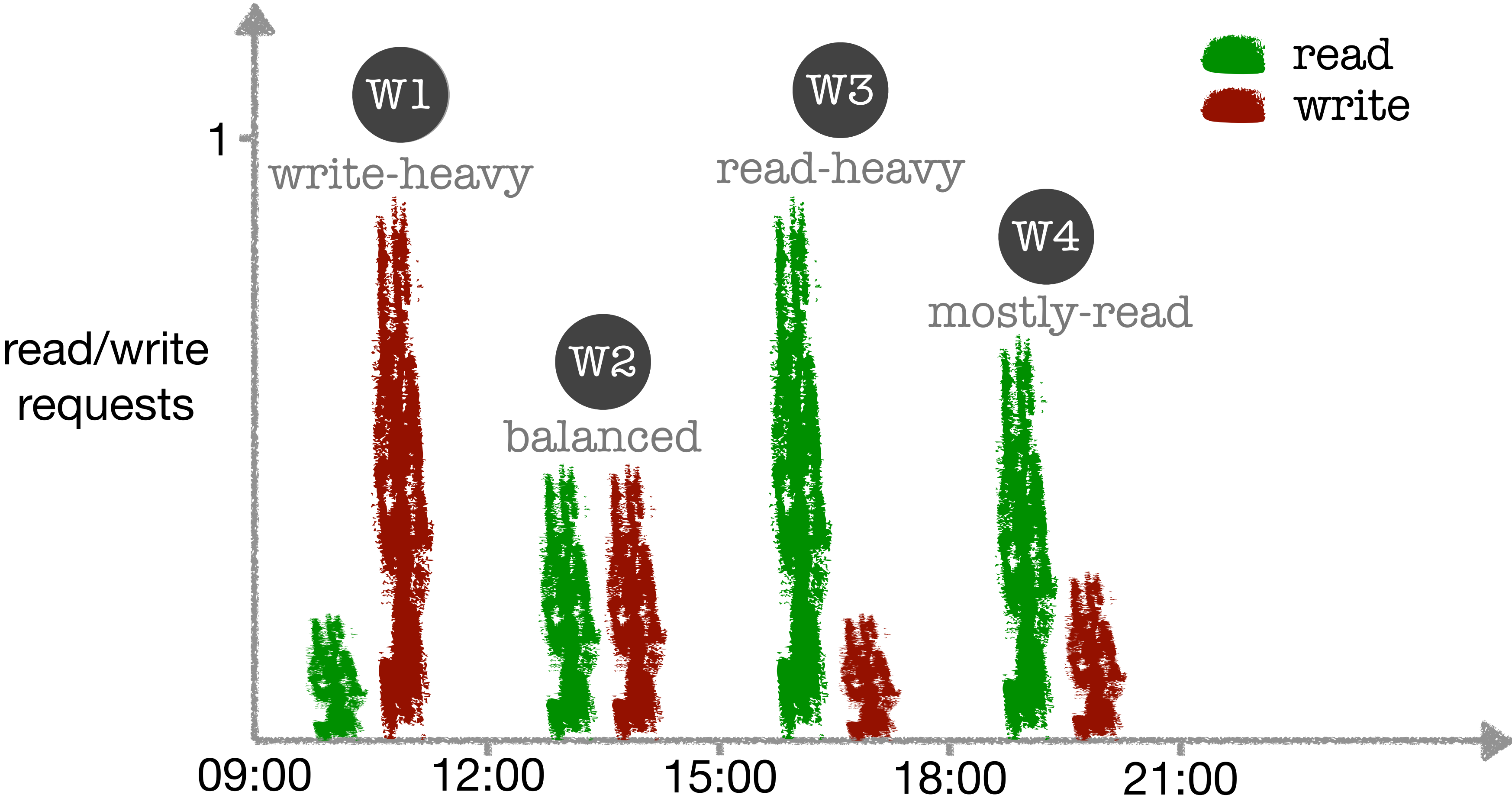
How do we emulate such a workload?

Real-world Workloads: Shifts Dynamically



How do we emulate such a workload?

Real-world Workloads: Shifts Dynamically

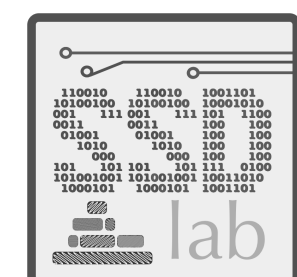
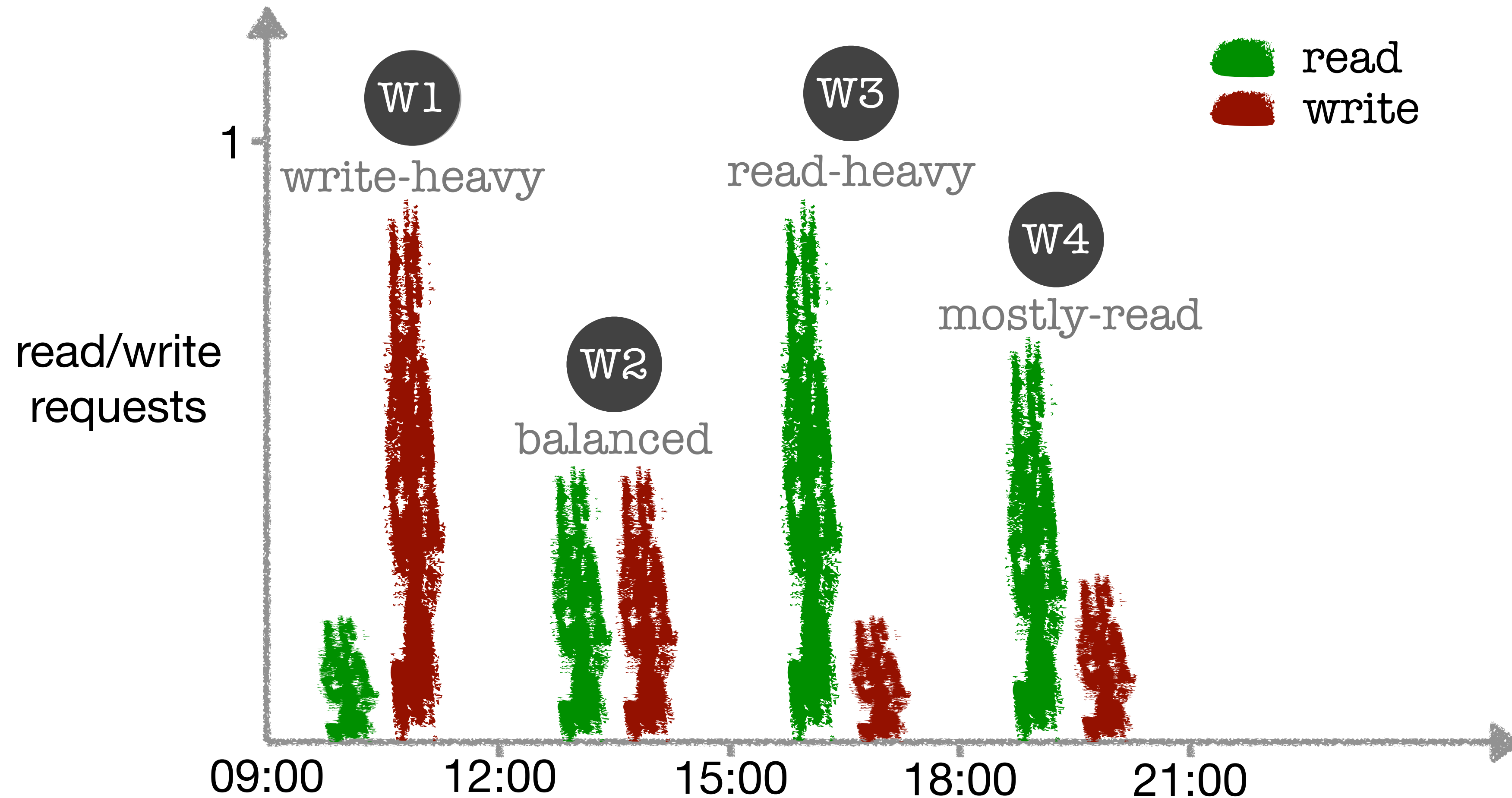


Real-world Workloads: Shifts Dynamically

YCSB
SoCC'10

db_bench
RocksDB

KVBench
DBTest'24

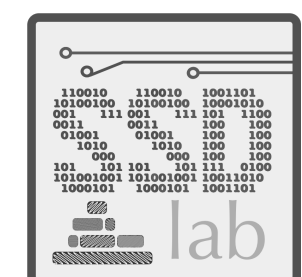
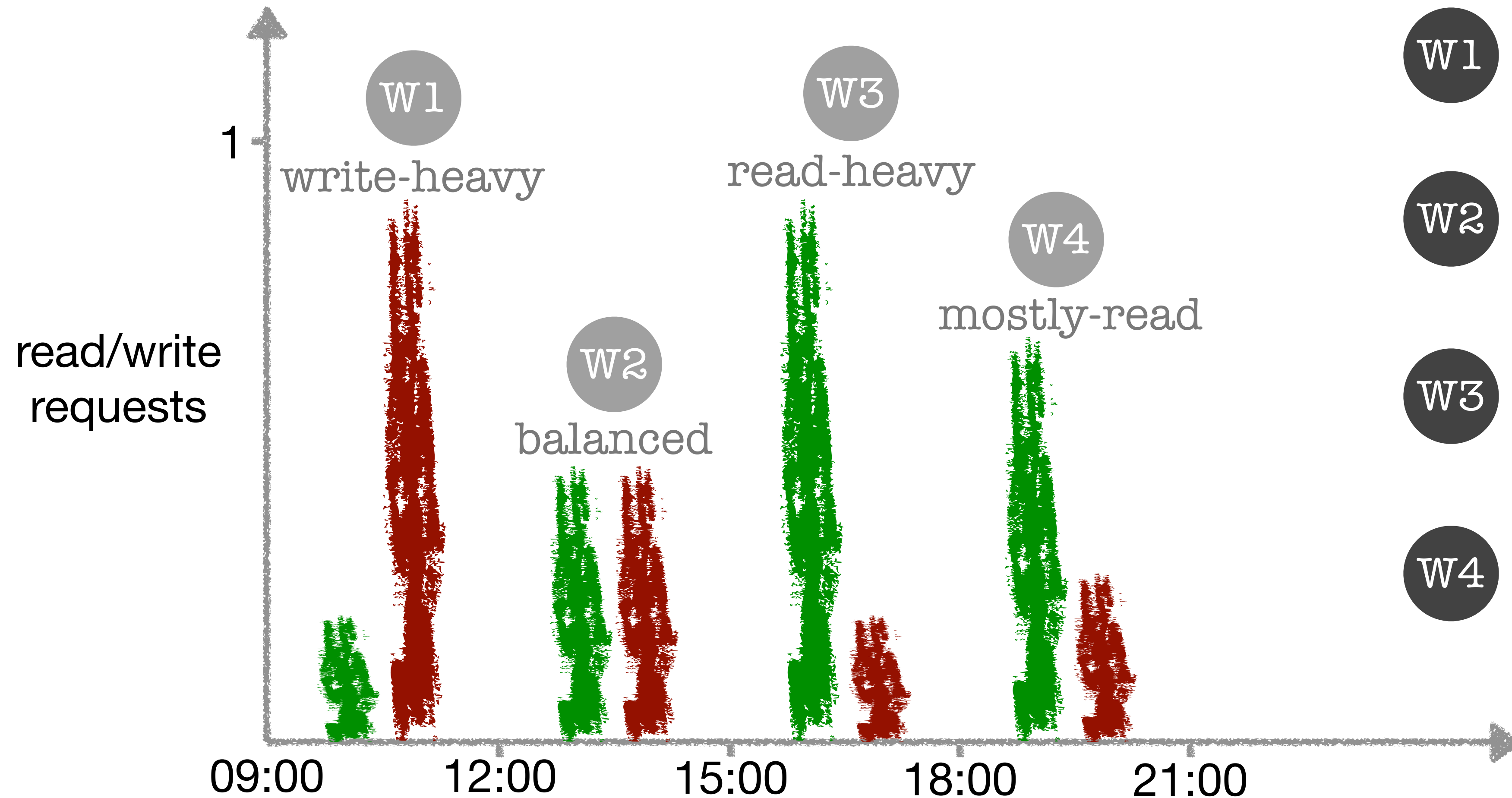


Real-world Workloads: Shifts Dynamically

YCSB
SoCC'10

db_bench
RocksDB

KVBench
DBTest'24

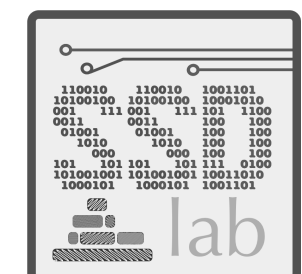
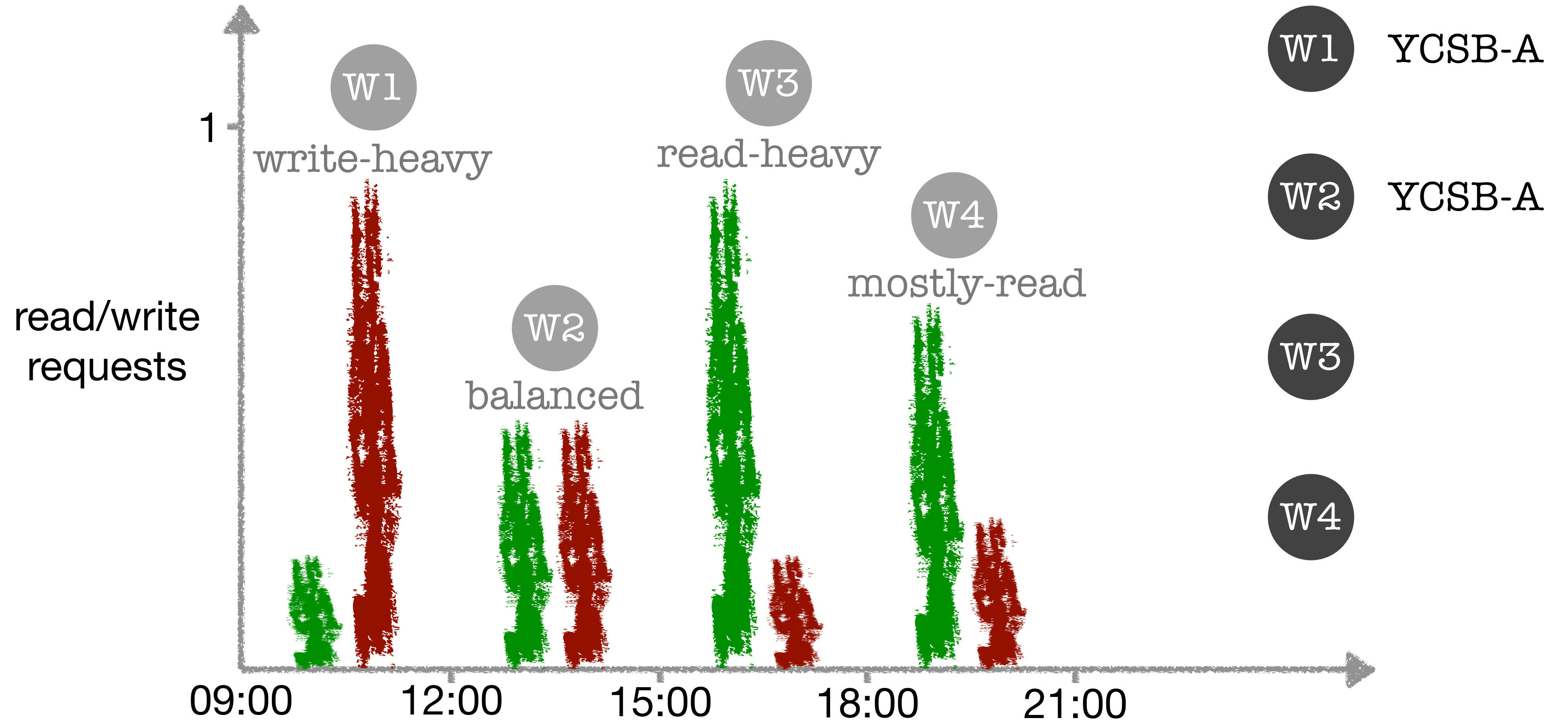


Real-world Workloads: Shifts Dynamically

YCSB
SoCC'10

db_bench
RocksDB

KVBench
DBTest'24

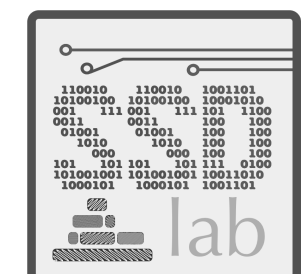
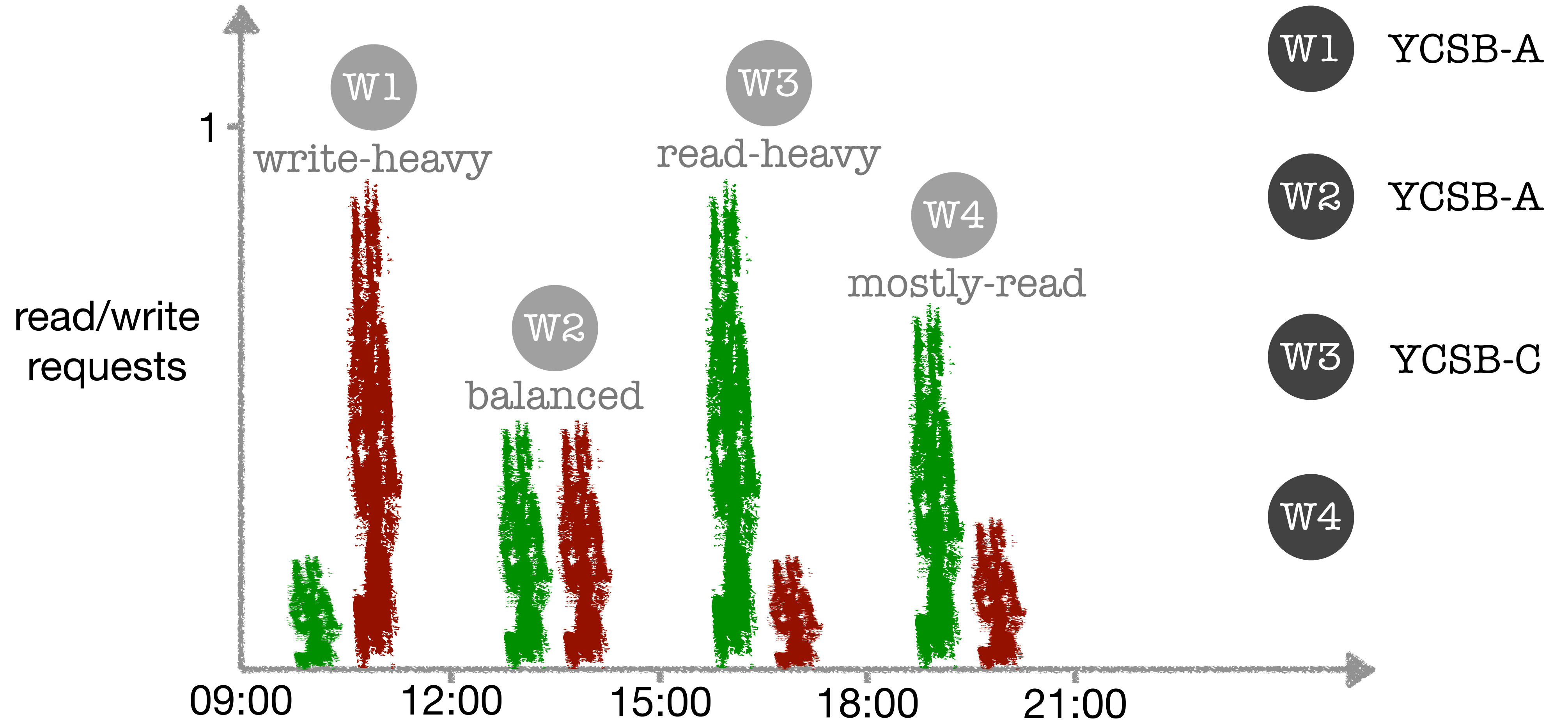


Real-world Workloads: Shifts Dynamically

YCSB
SoCC'10

db_bench
RocksDB

KVBench
DBTest'24

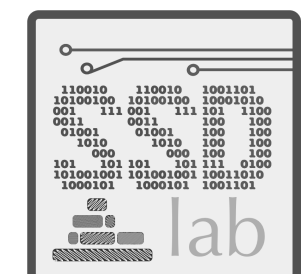
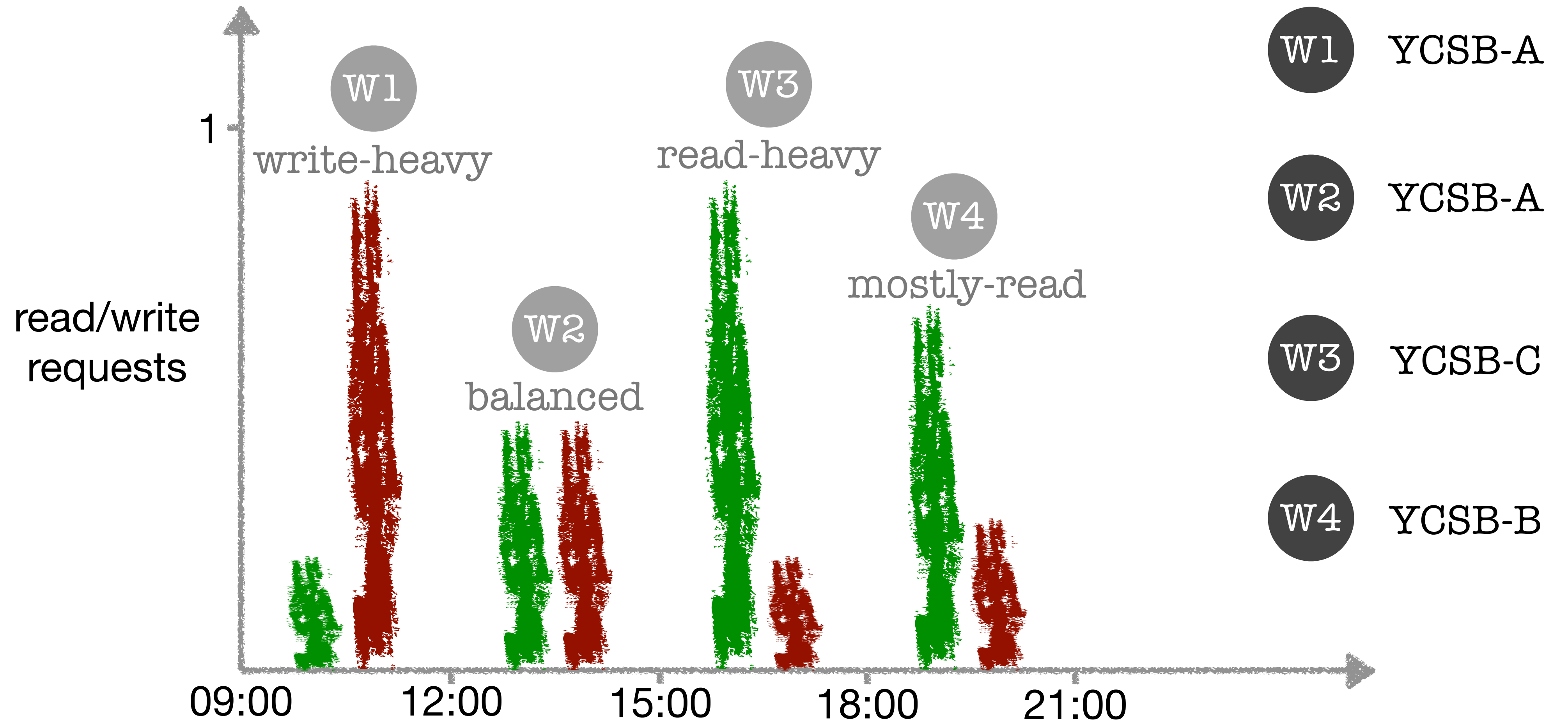


Real-world Workloads: Shifts Dynamically

YCSB
SoCC'10

db_bench
RocksDB

KVBench
DBTest'24

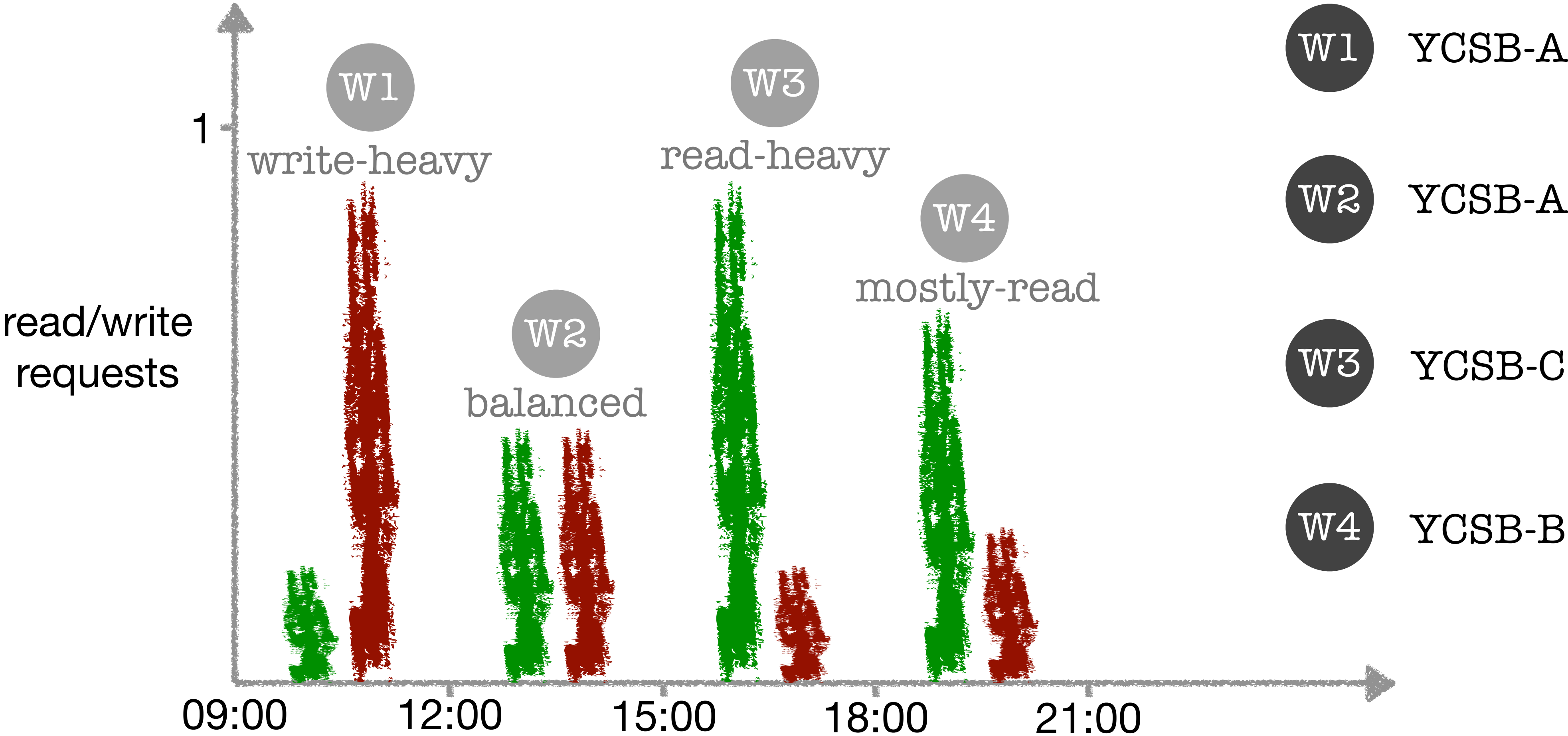


Real-world Workloads: Shifts Dynamically

YCSB
SoCC'10

db_bench
RocksDB

KVBench
DBTest'24



human in the loop

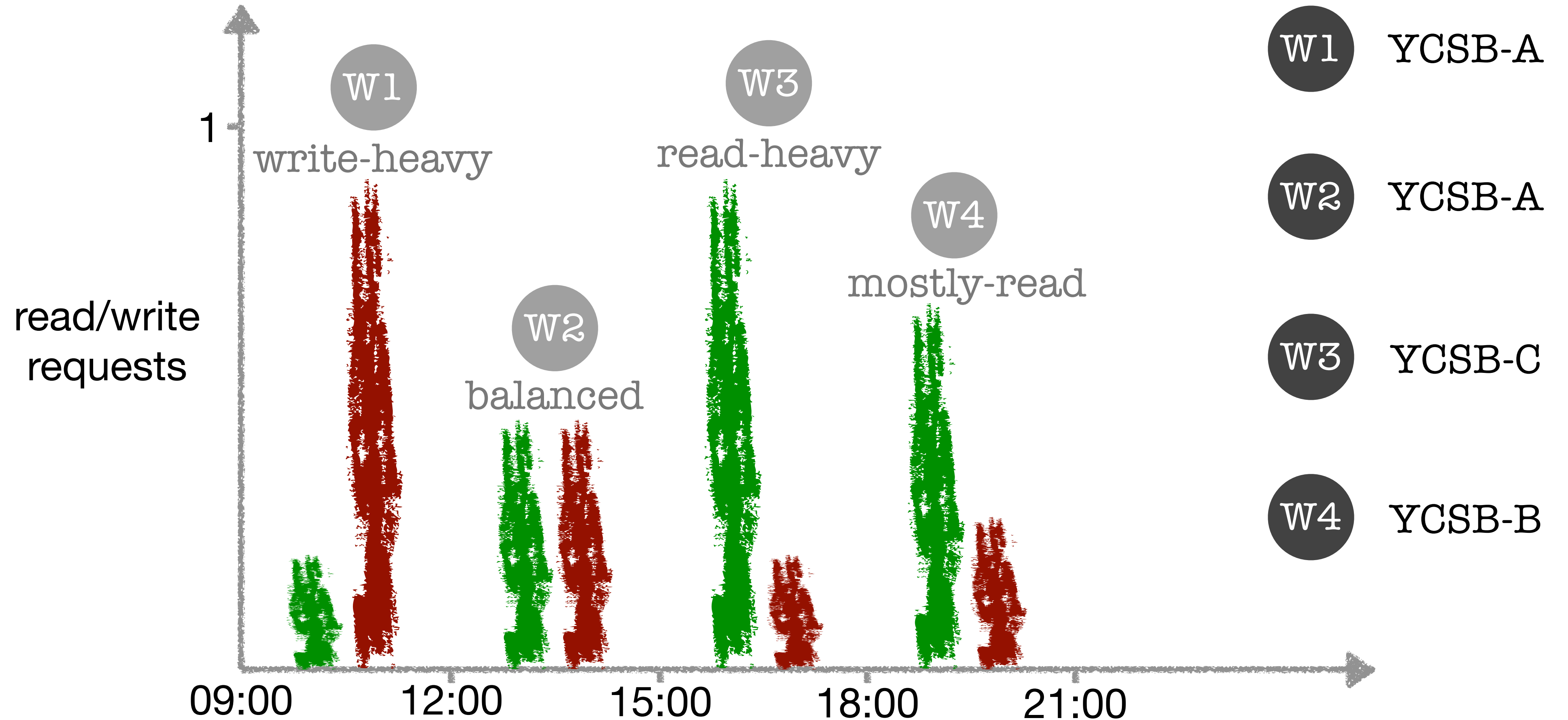


Real-world Workloads: Shifts Dynamically

YCSB
SoCC'10

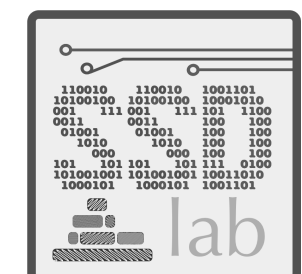
db_bench
RocksDB

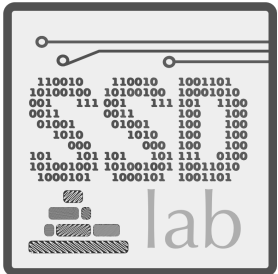
KVBench
DBTest'24



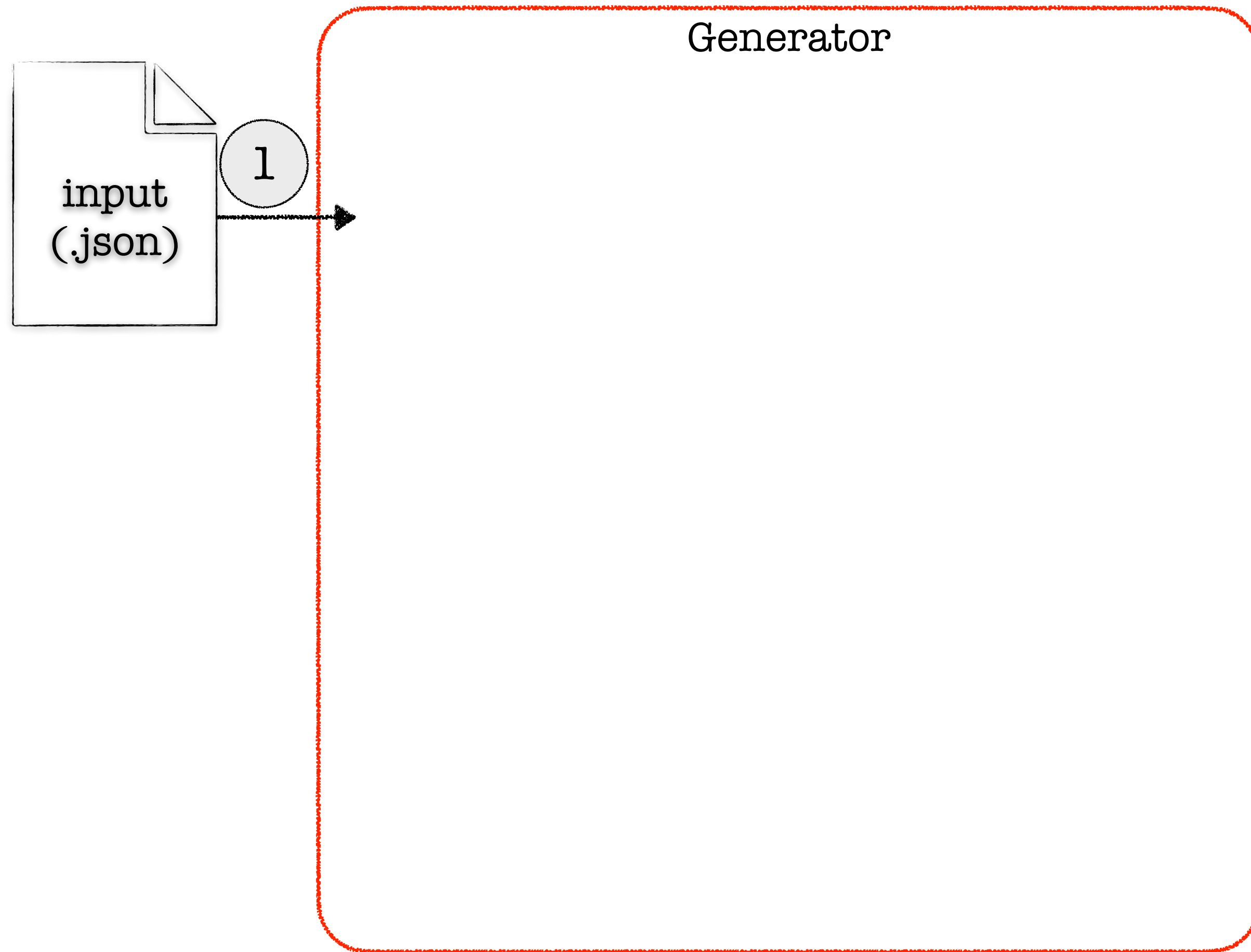
human in the loop

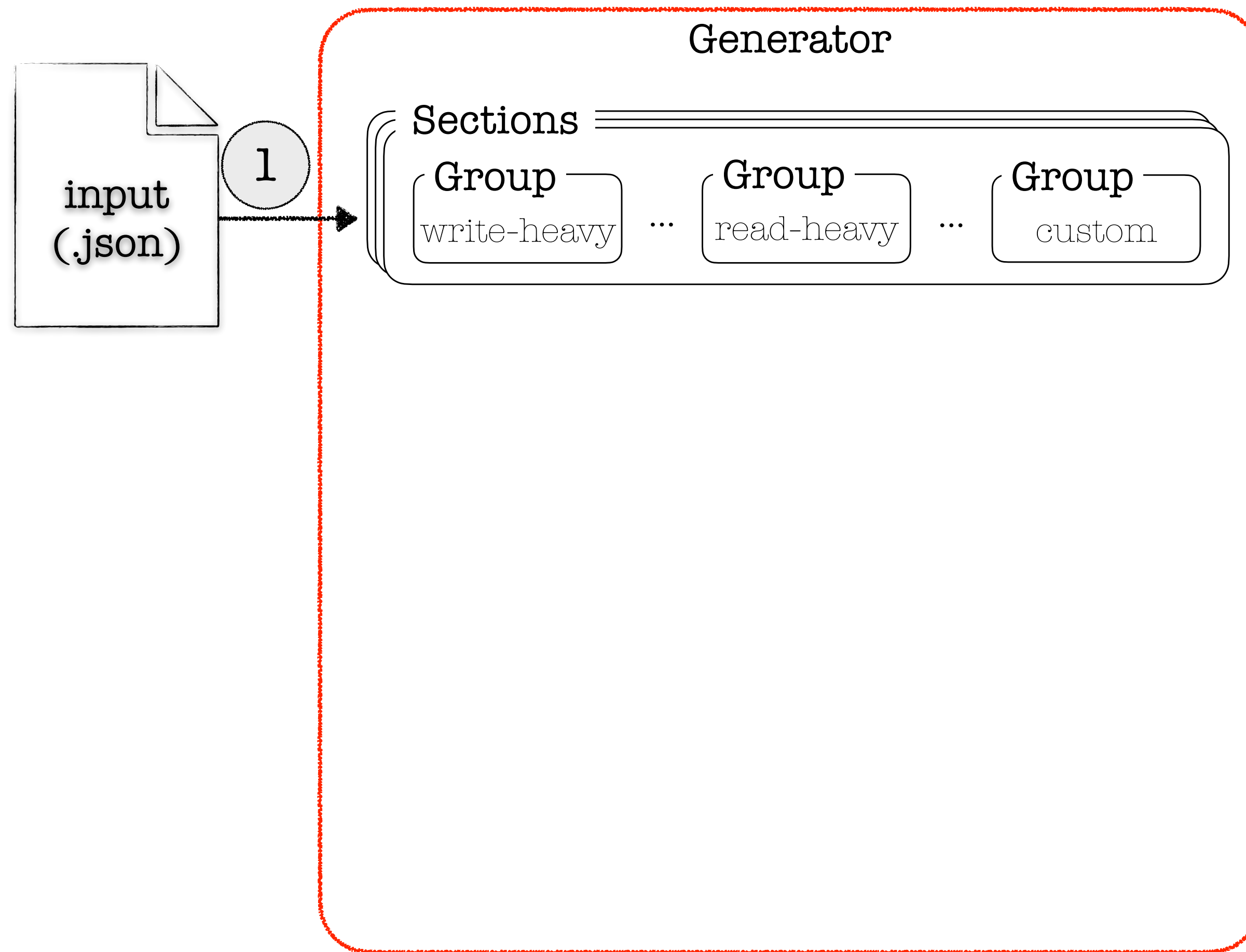
uncorrelated workload





Brandeis
UNIVERSITY



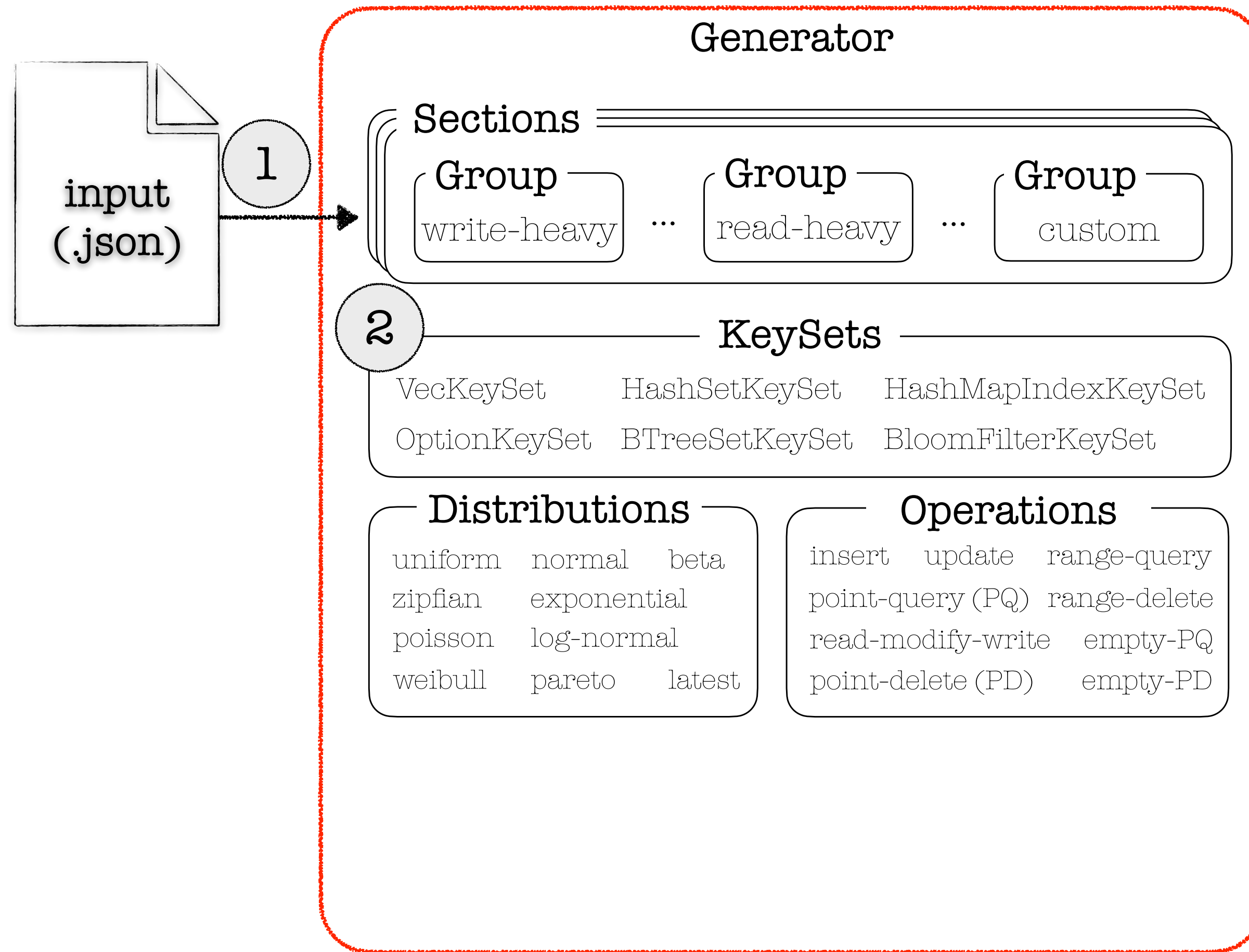


Section

represents a disjoint segment of a workload

Group

represents the phase of a workload



Section

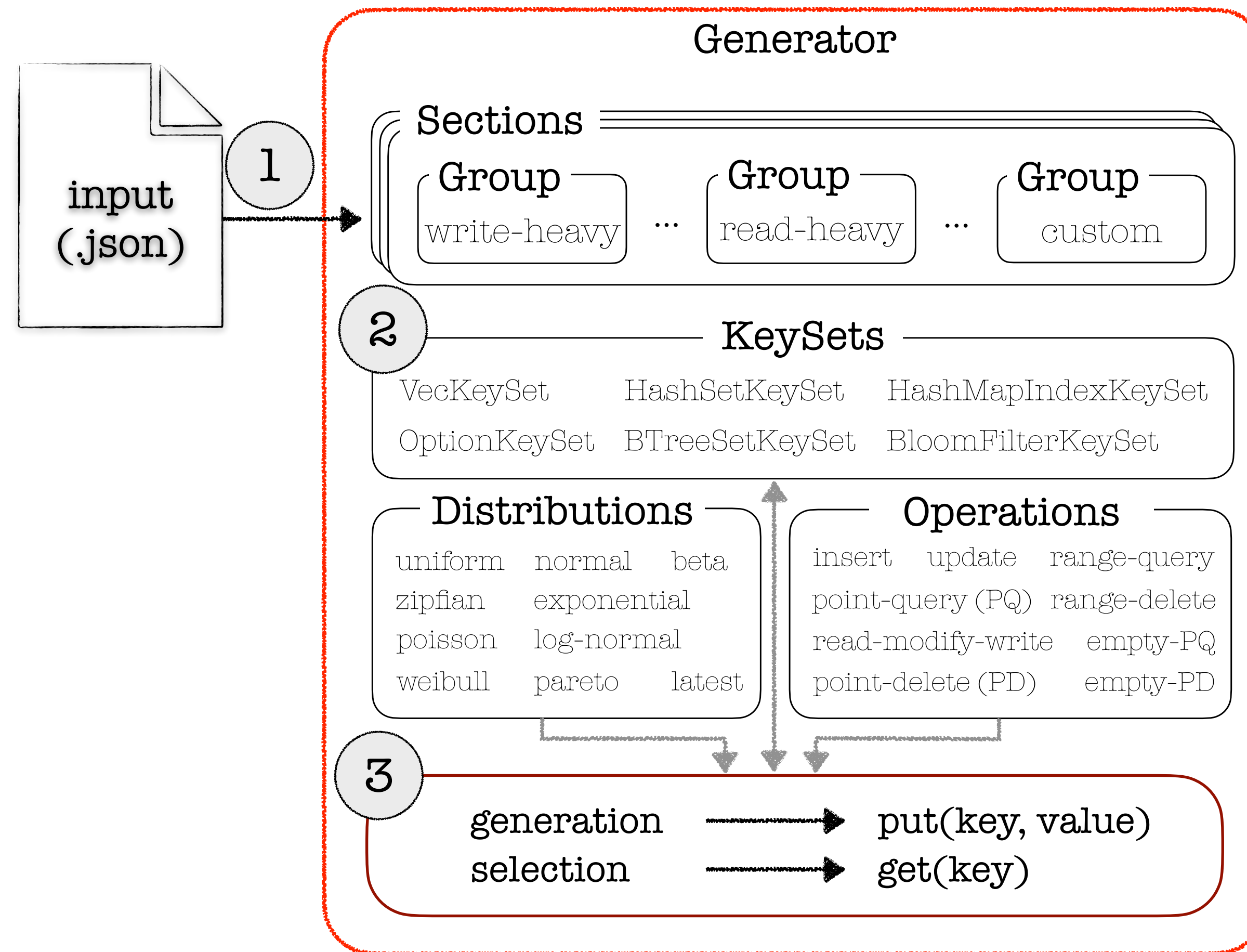
represents a disjoint segment of a workload

Group

represents the phase of a workload

KeySet

a data structure that helps in generating operations more optimally



Section

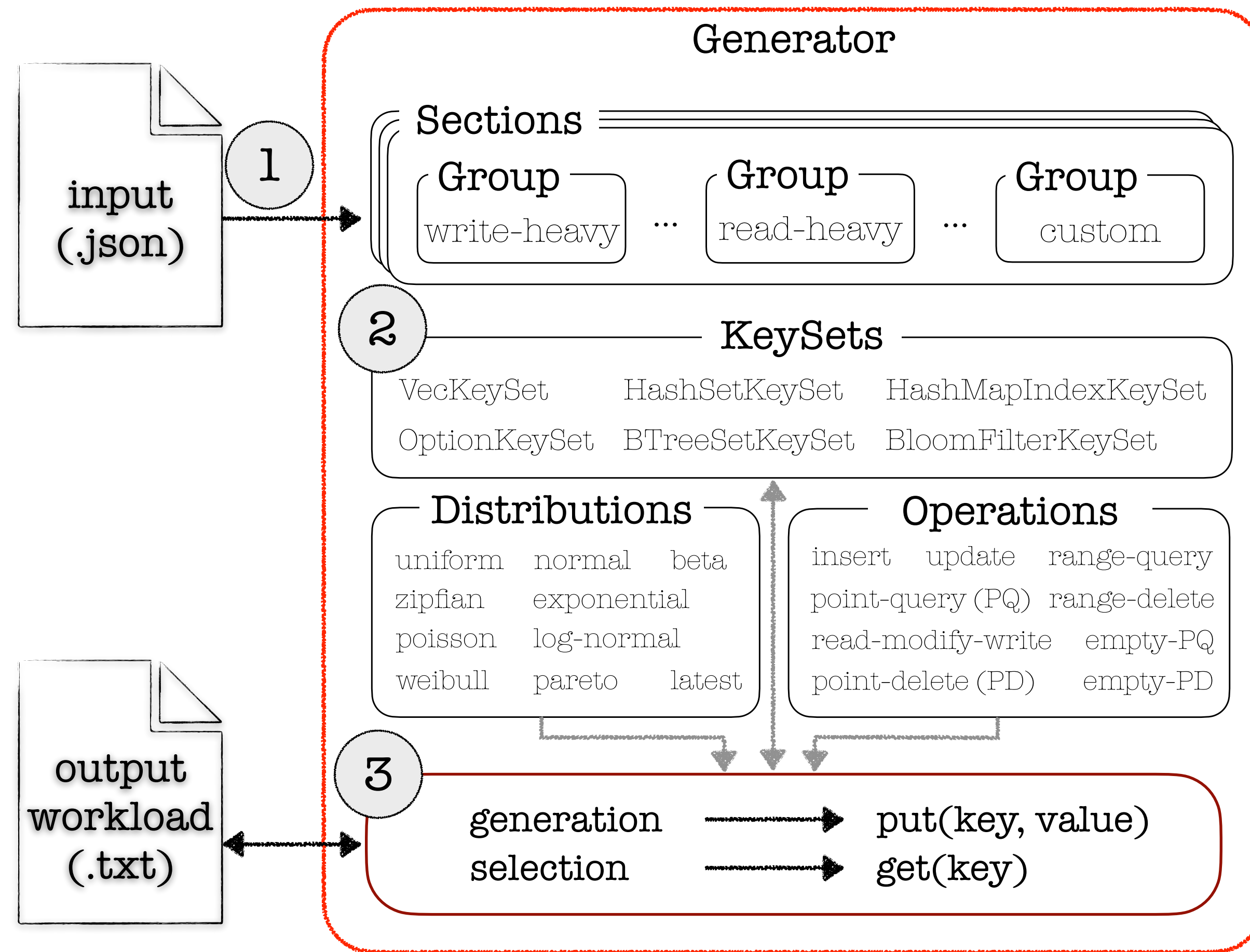
represents a disjoint segment of a workload

Group

represents the phase of a workload

KeySet

a data structure that helps in generating operations more optimally



Section

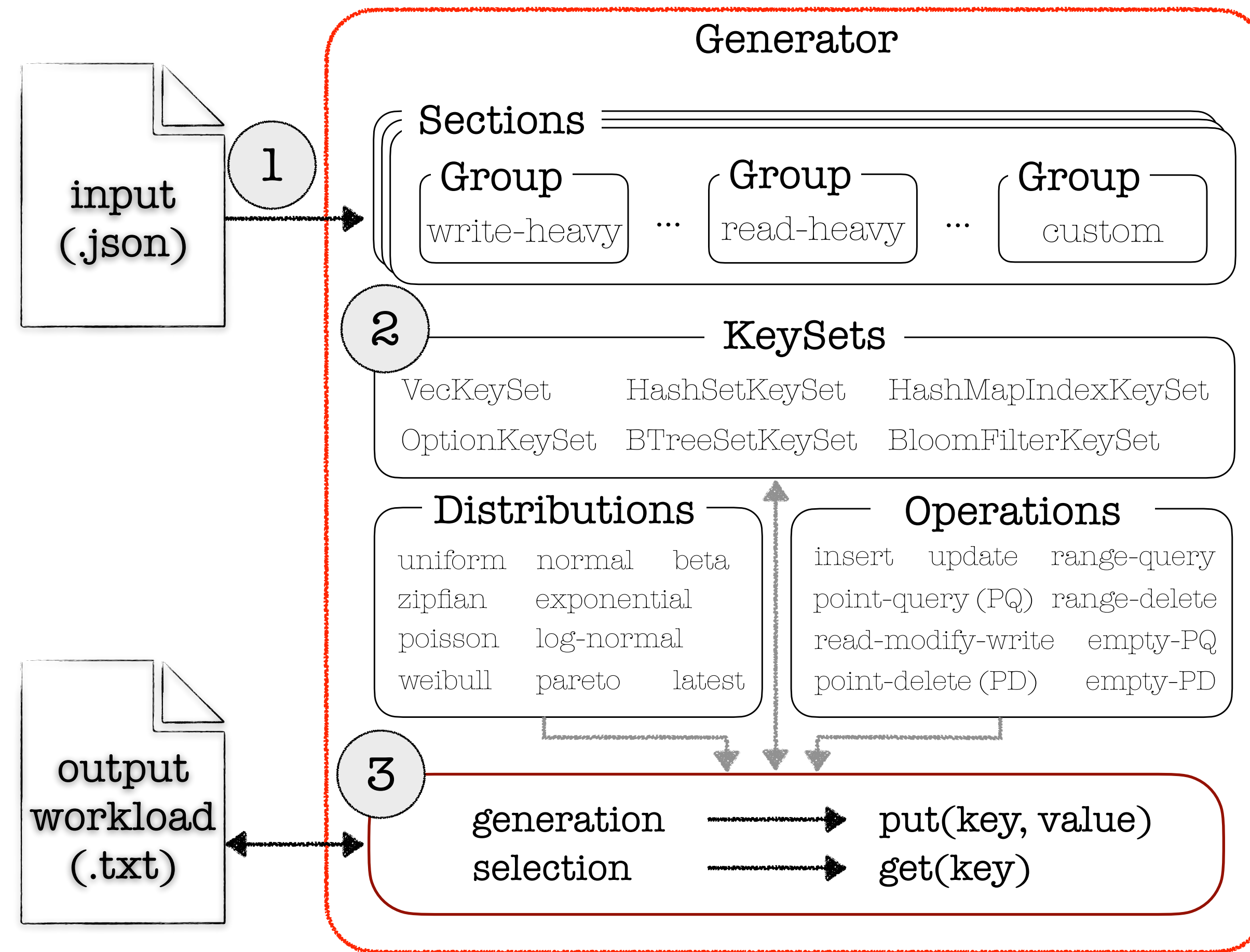
represents a disjoint segment of a workload

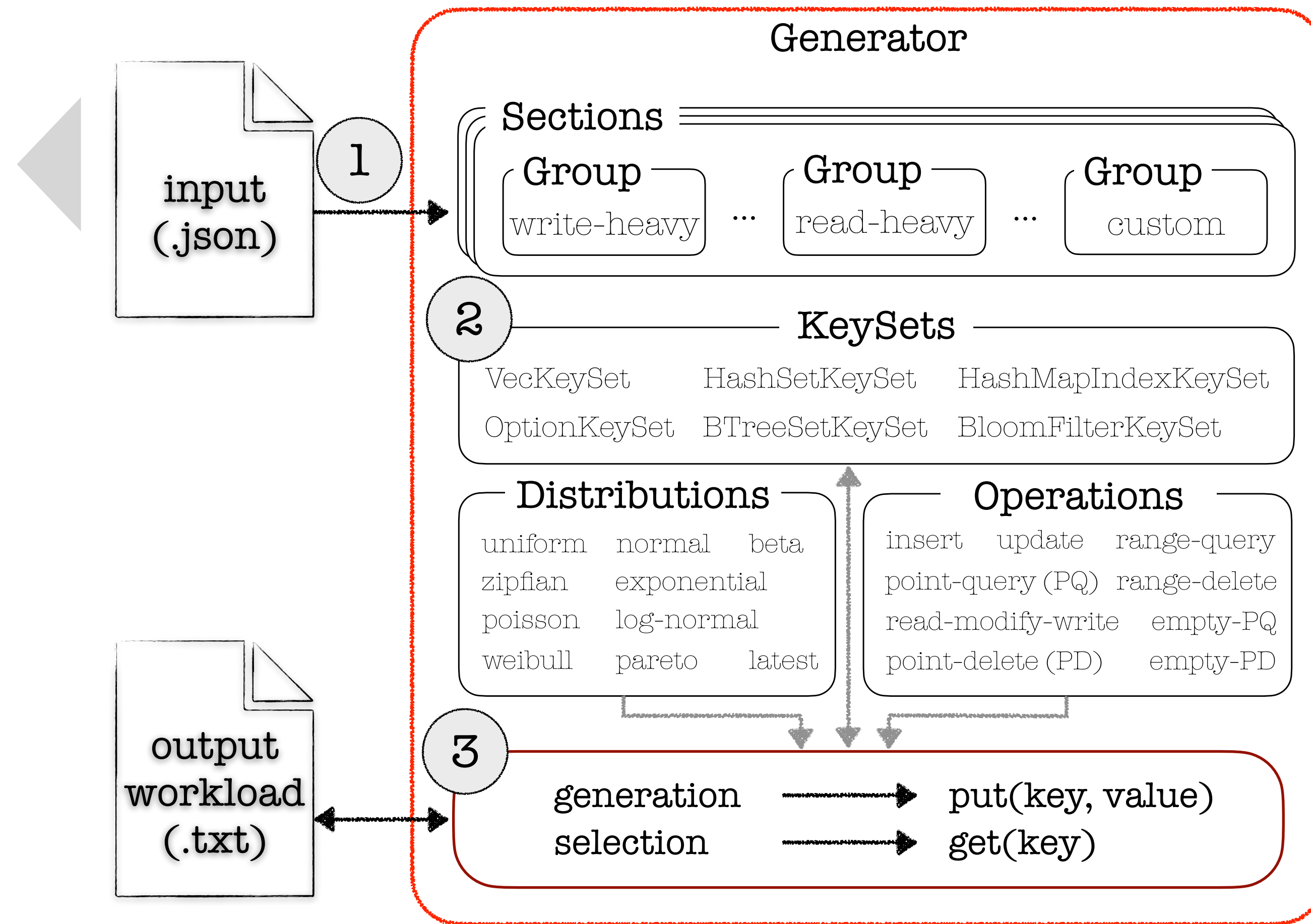
Group

represents the phase of a workload

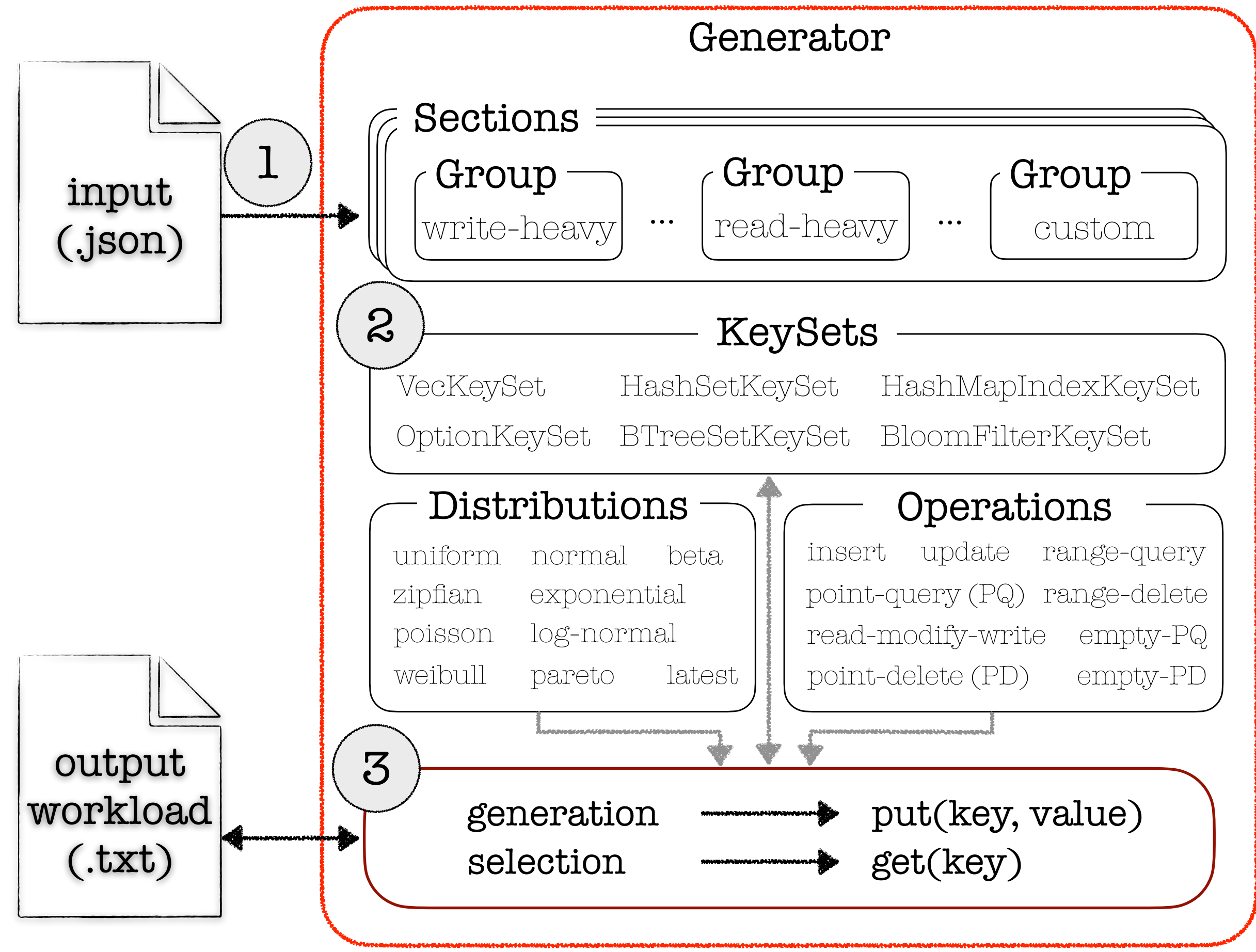
KeySet

a data structure that helps in generating operations more optimally



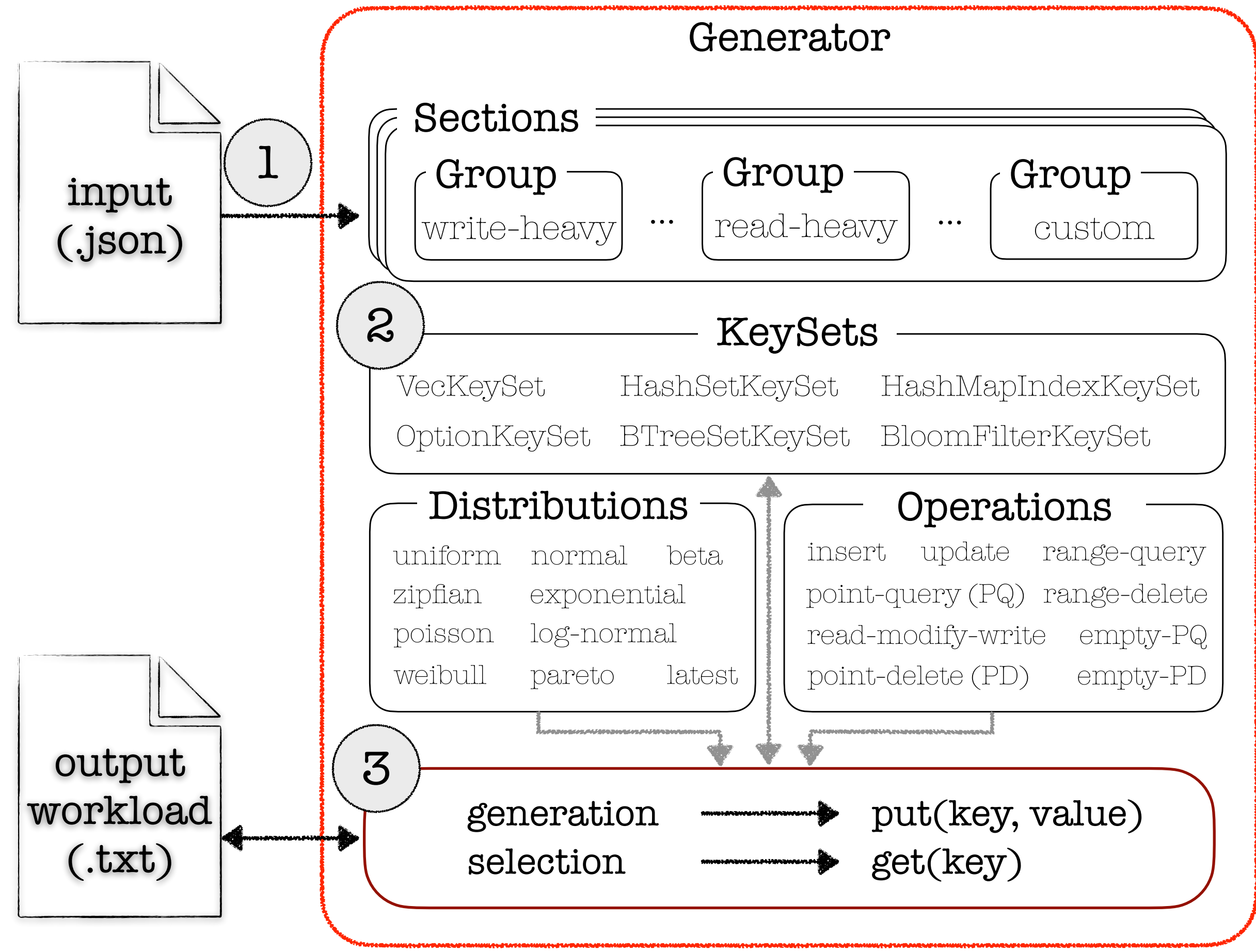


```
{
  "sections": [{
    "groups": [
      {
        // ...
      }
    ]
  }
}]
}]}
```

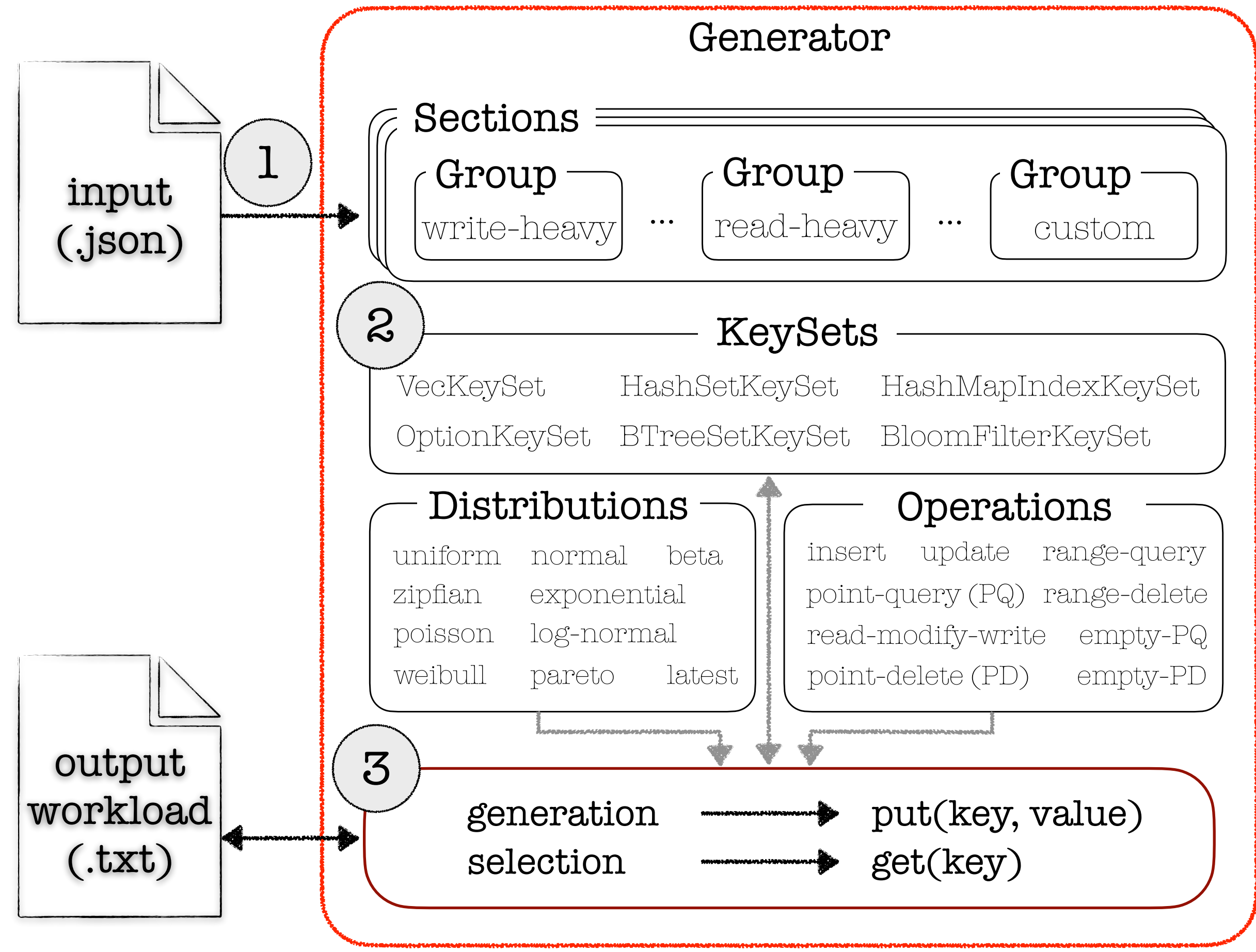


```
{
  "sections": [{
    "groups": [
      {
        "inserts": {

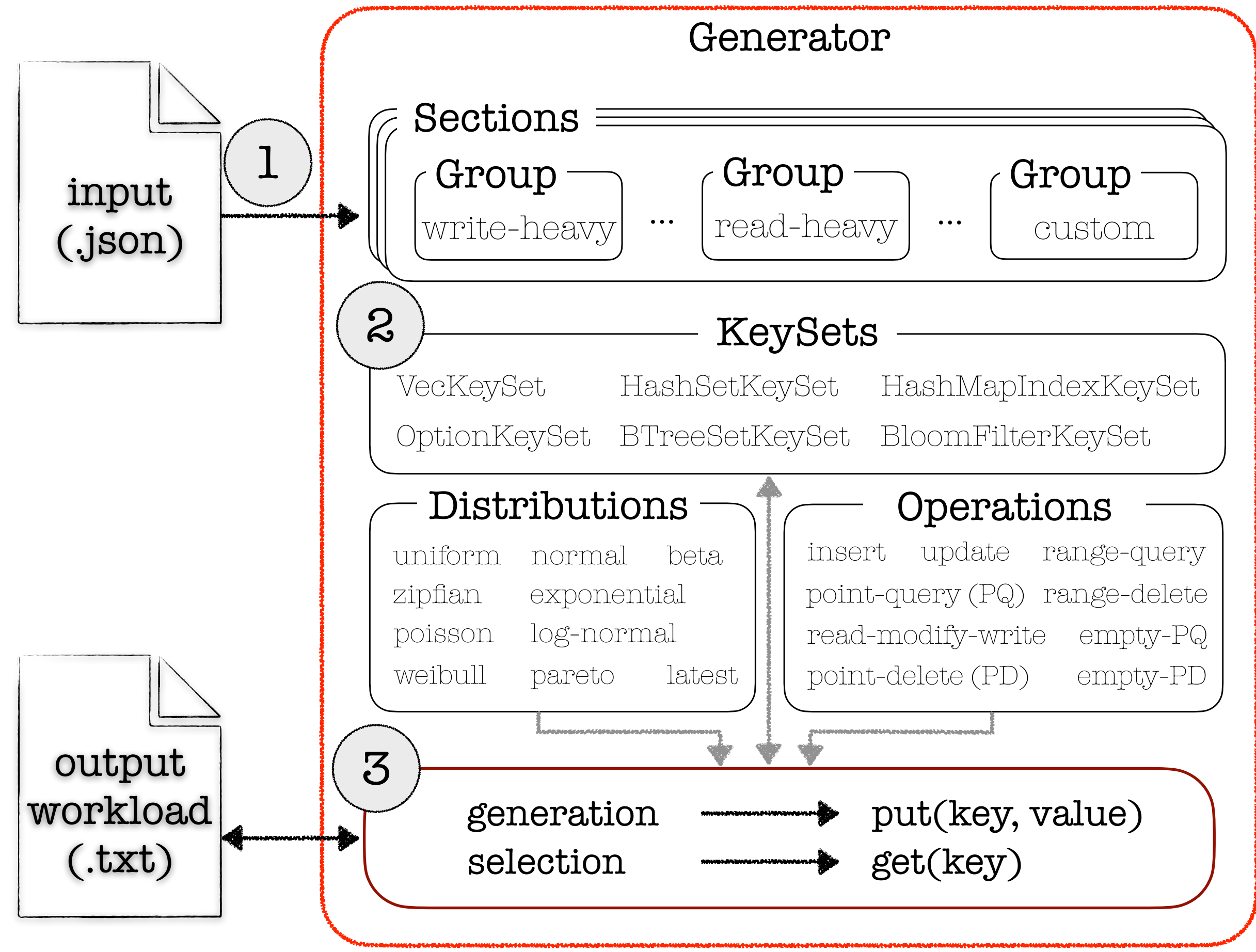
        },
        "point_queries": { ... }
      }
    ]
  }
}]
```



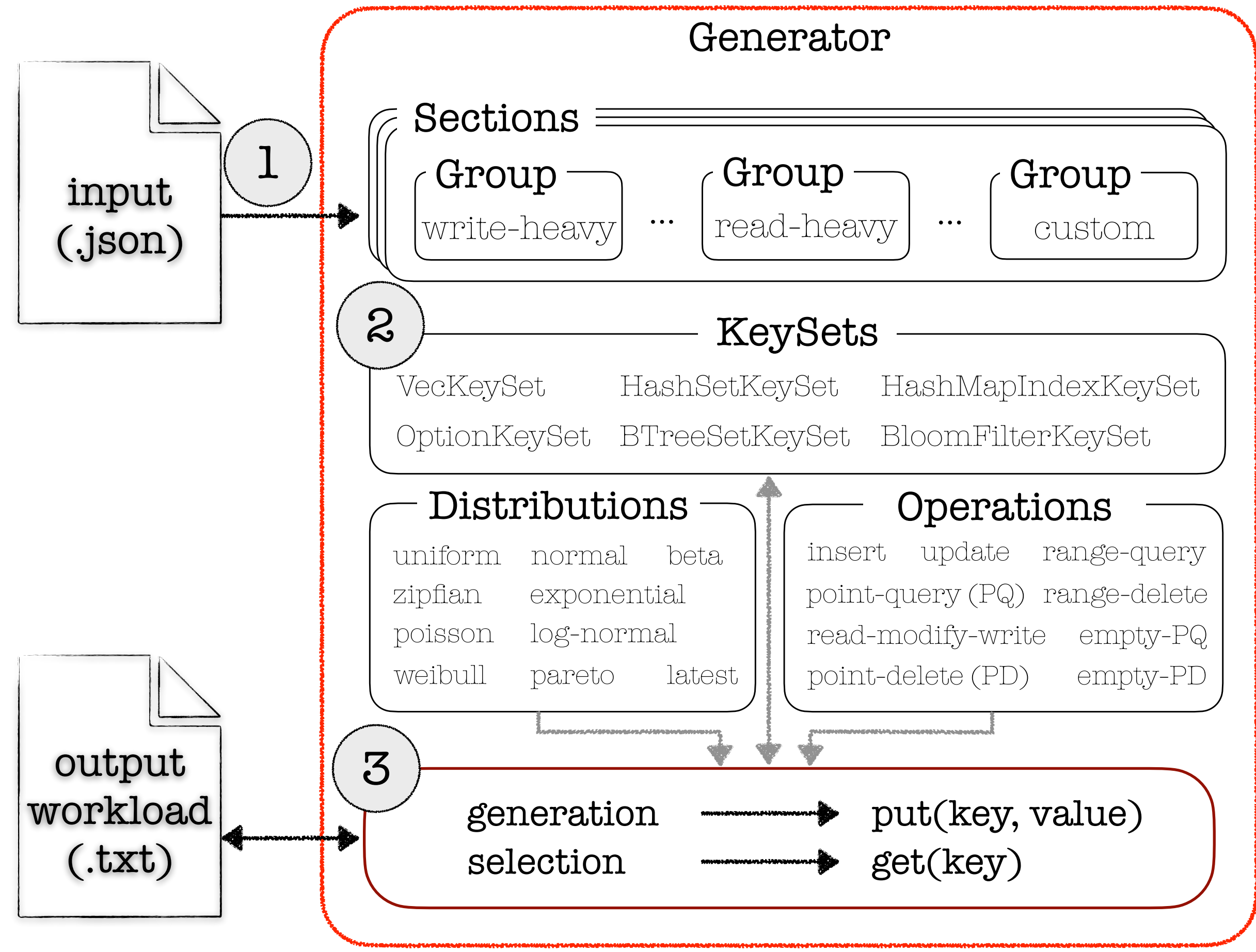

```
{
  "sections": [{
    "groups": [
      {
        "inserts": {
          "op_count": 100M,
          "key": {
            "segmented": {
              "segments": [
                "usertable:app",
                {"uniform": {"len": 9}}
              ],
              "separator": ";",
            },
          },
          "val": {
            "uniform": {
              "len": {
                "uniform": {
                  "min": 32, "max": 1024
                }
              }
            }
          },
          "point_queries": {...}
        }
      ]
    }
  ]
}
```



```
{
  "sections": [{
    "groups": [
      {
        "inserts": {
          "op_count": 100M,
          "key": {
            "segmented": {
              "segments": [
                "usertable:app",
                {"uniform": {"len": 9}}
              ],
              "separator": ";",
            },
          },
          "val": {
            "uniform": {
              "len": {
                "uniform": {
                  "min": 32, "max": 1024
                }
              }
            }
          },
          "point_queries": {...}
        }
      ]
    }
  ]
}
```

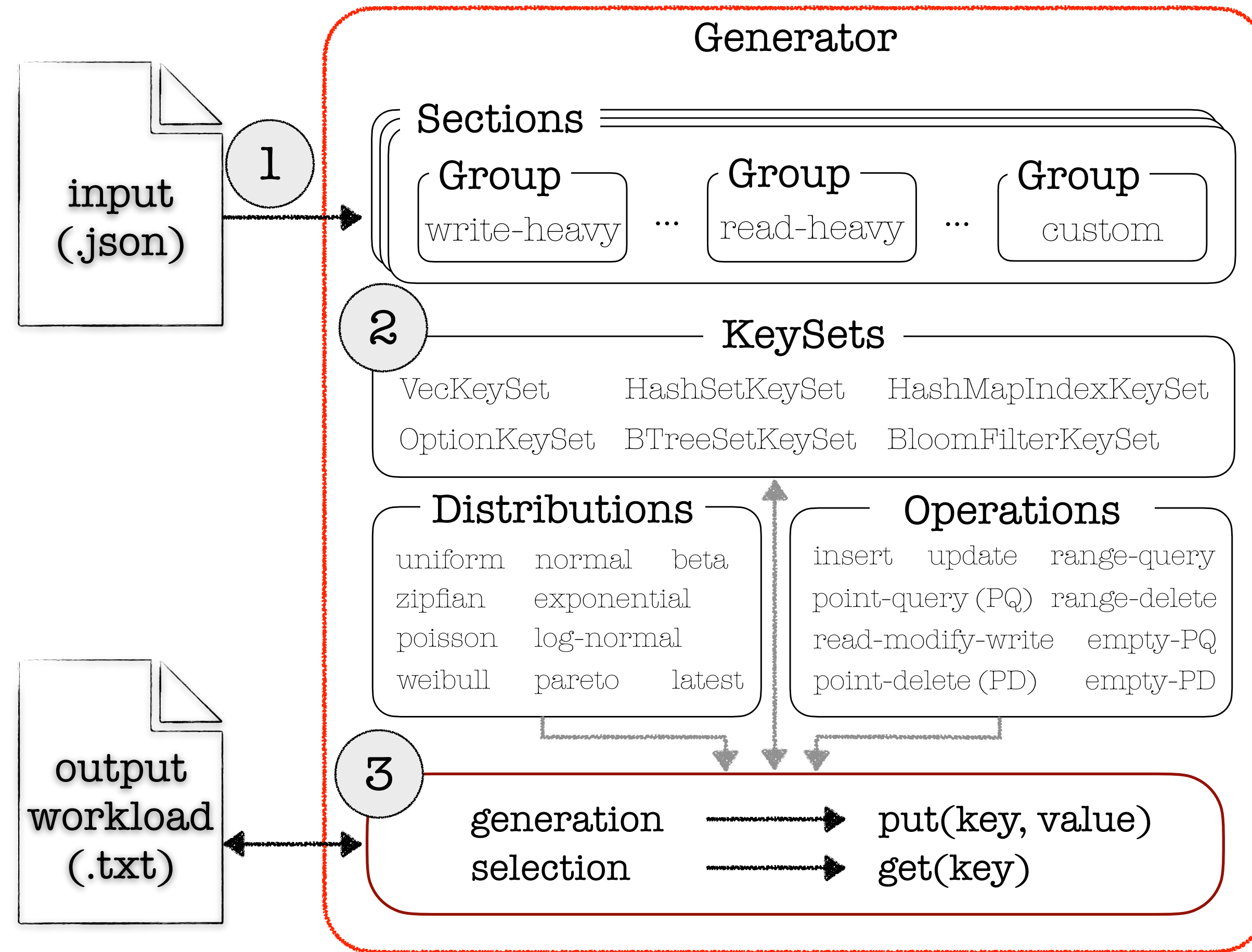



```
{
  "sections": [{
    "groups": [
      {
        "inserts": {
          "op_count": 100M,
          "key": {
            "segmented": {
              "segments": [
                "usertable:app",
                {"uniform": {"len": 9}}
              ],
              "separator": ";",
            },
          },
          "val": {
            "uniform": {
              "len": {
                "uniform": {
                  "min": 32, "max": 1024
                }
              }
            }
          },
          "point_queries": {...}
        }
      ]
    }
  ]
}
```



requires a prior understanding

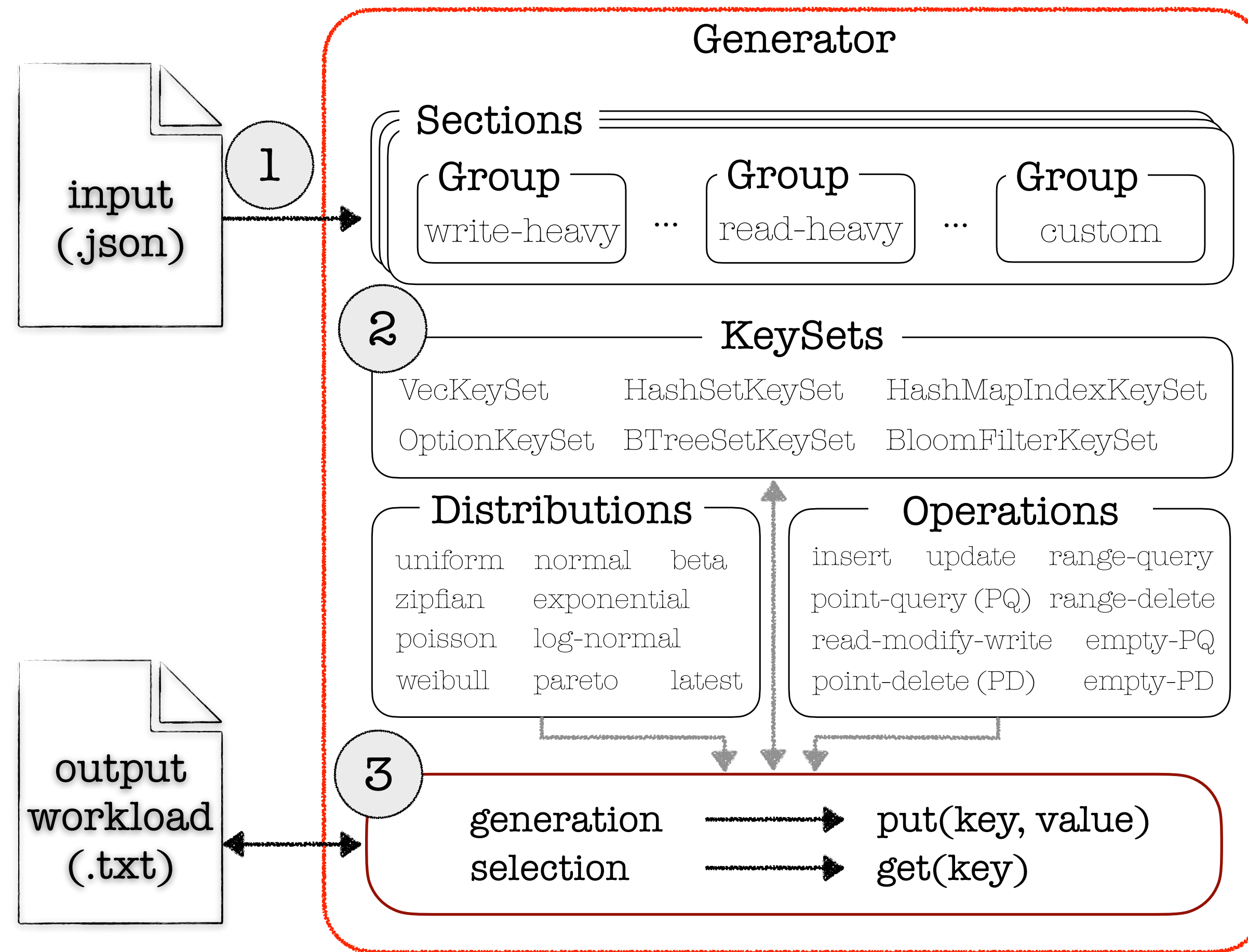
```
{
  "sections": [{
    "groups": [
      {
        "inserts": {
          "op_count": 100M,
          "key": {
            "segmented": {
              "segments": [
                "usertable:app",
                {"uniform": {"len": 9}}
              ],
              "separator": ";",
            },
          },
          "val": {
            "uniform": {
              "len": {
                "uniform": {
                  "min": 32, "max": 1024
                }
              }
            }
          },
          "point_queries": {...}
        }
      ]
    }
  ]
}
```



requires a prior understanding

just a workload generator

```
{
  "sections": [{
    "groups": [
      {
        "inserts": {
          "op_count": 100M,
          "key": {
            "segmented": {
              "segments": [
                "usertable:app",
                {"uniform": {"len": 9}}
              ],
              "separator": ";",
            },
          },
          "val": {
            "uniform": {
              "len": {
                "uniform": {
                  "min": 32, "max": 1024
                }
              }
            },
            "point_queries": {...}
          }
        }
      }
    ]
  }
}]
```

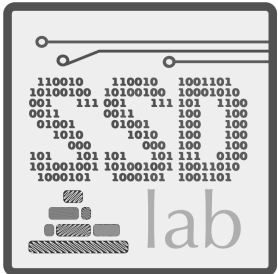


requires a prior understanding

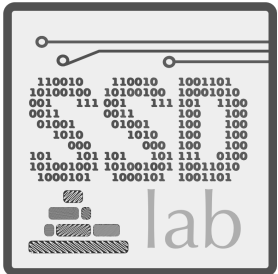
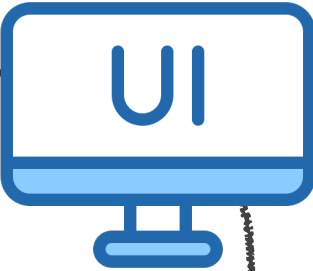
just a workload generator

no side-by-side comparison

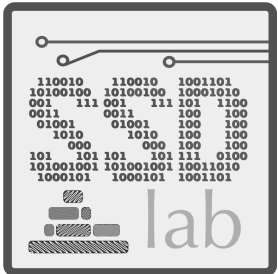
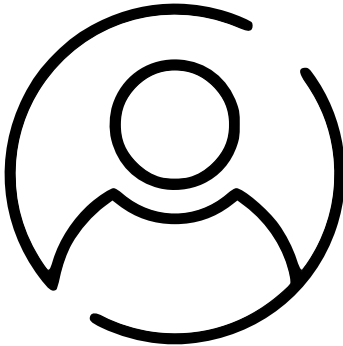
TexBench



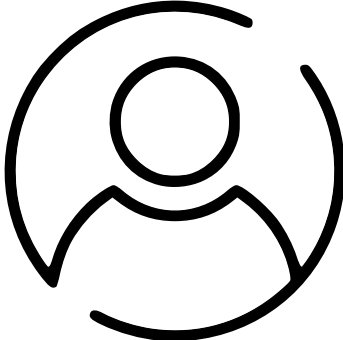
TexBench



TexBench



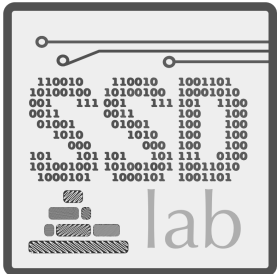
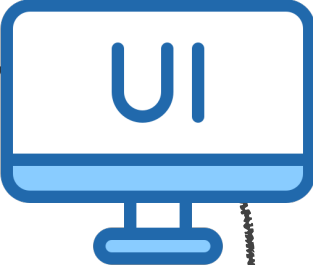
TexBench



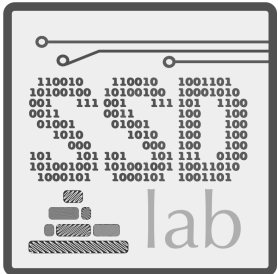
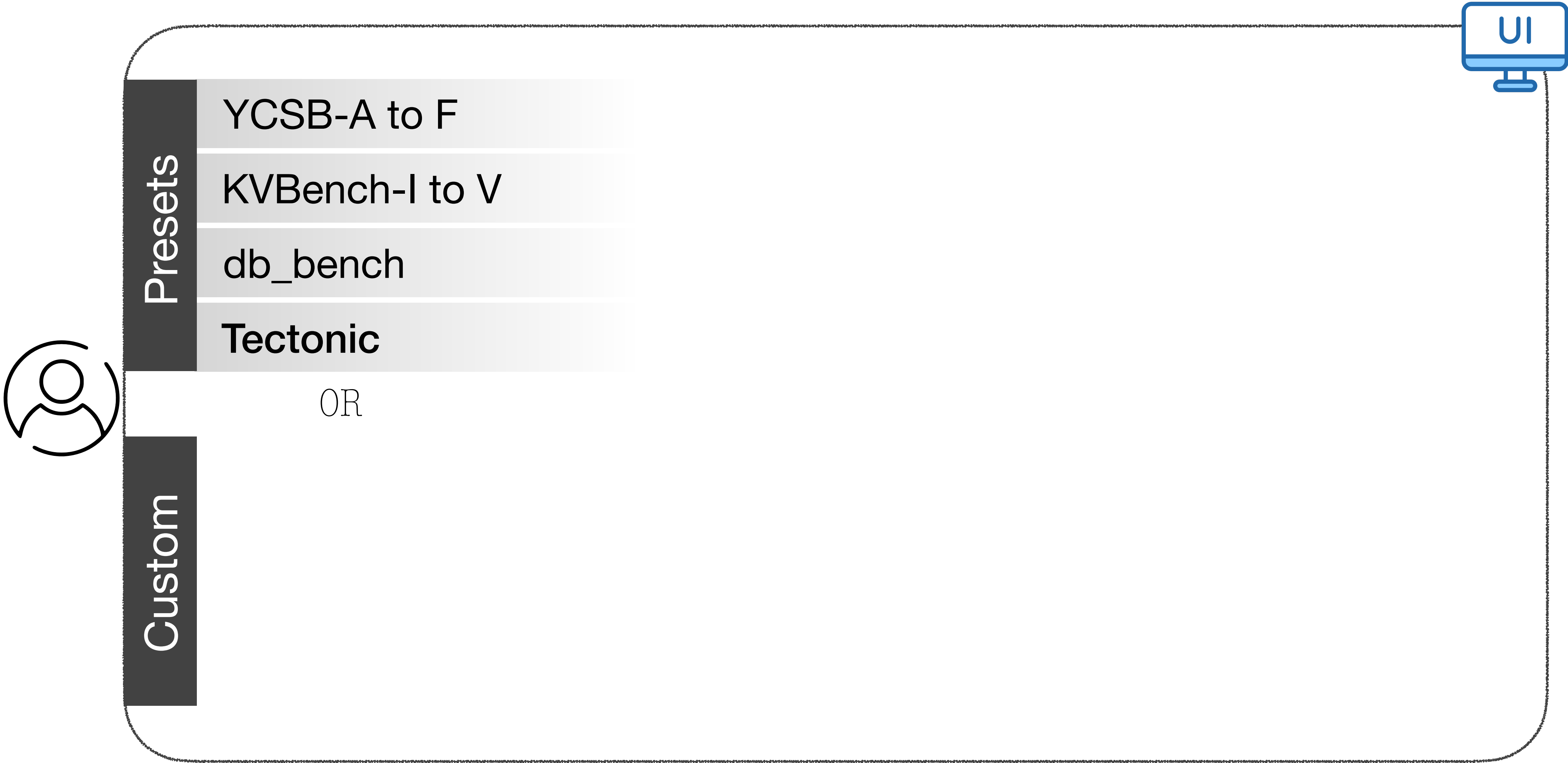
Presets

Custom

OR



TexBench



TexBench



Presets

YCSB-A to F

KVBench-I to V

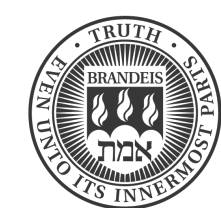
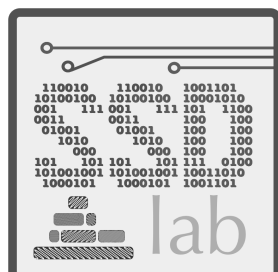
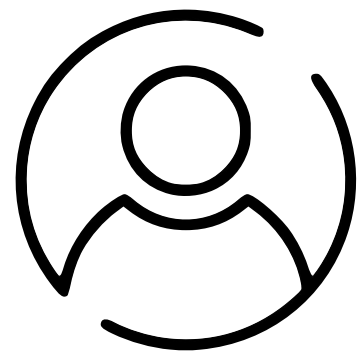
db_bench

Tectonic

OR

Custom

Generate a two-phase workload that starts with an insert-heavy phase and shifts to a read-heavy phase....



TexBench



Presets

YCSB-A to F

KVBench-I to V

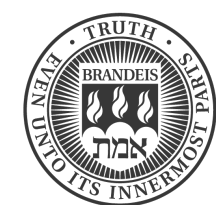
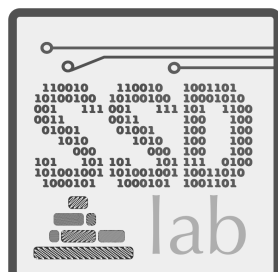
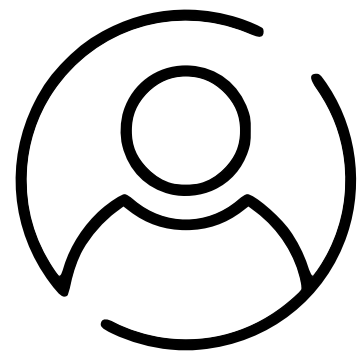
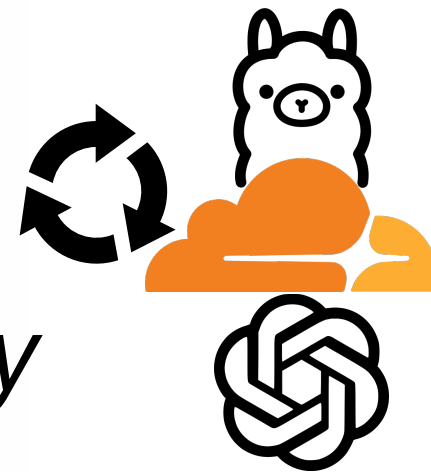
db_bench

Tectonic

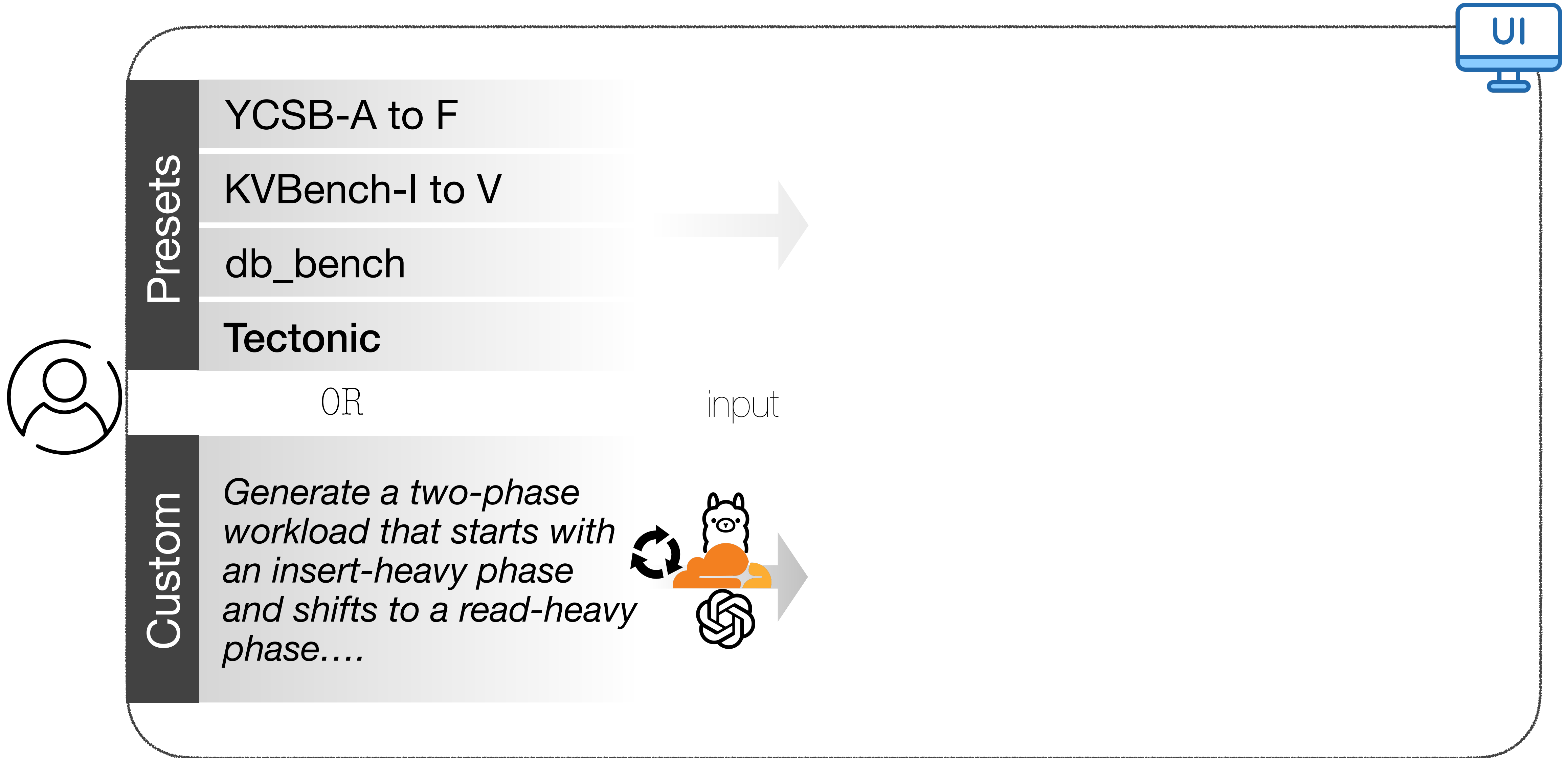
OR

Custom

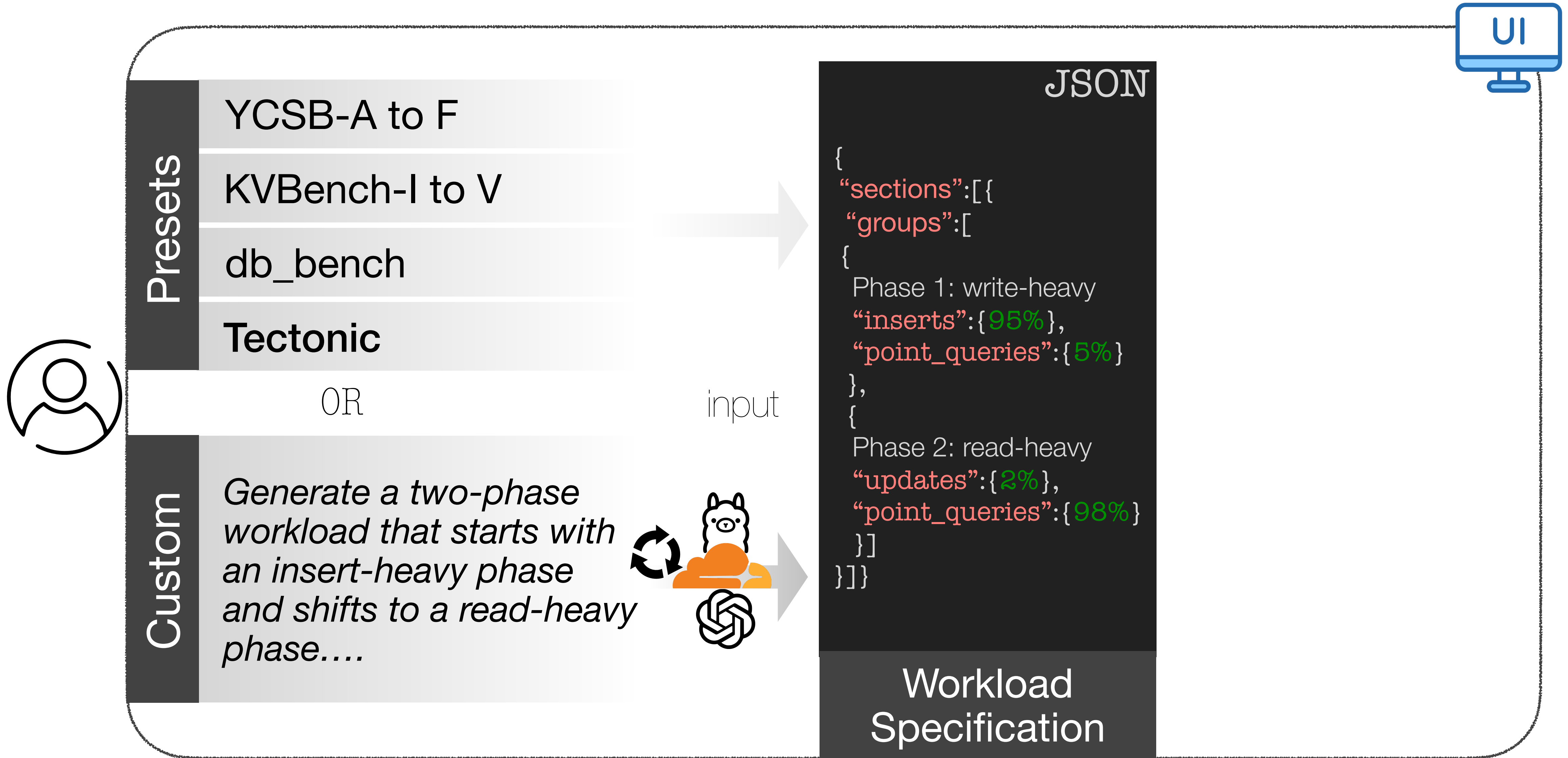
Generate a two-phase workload that starts with an insert-heavy phase and shifts to a read-heavy phase....



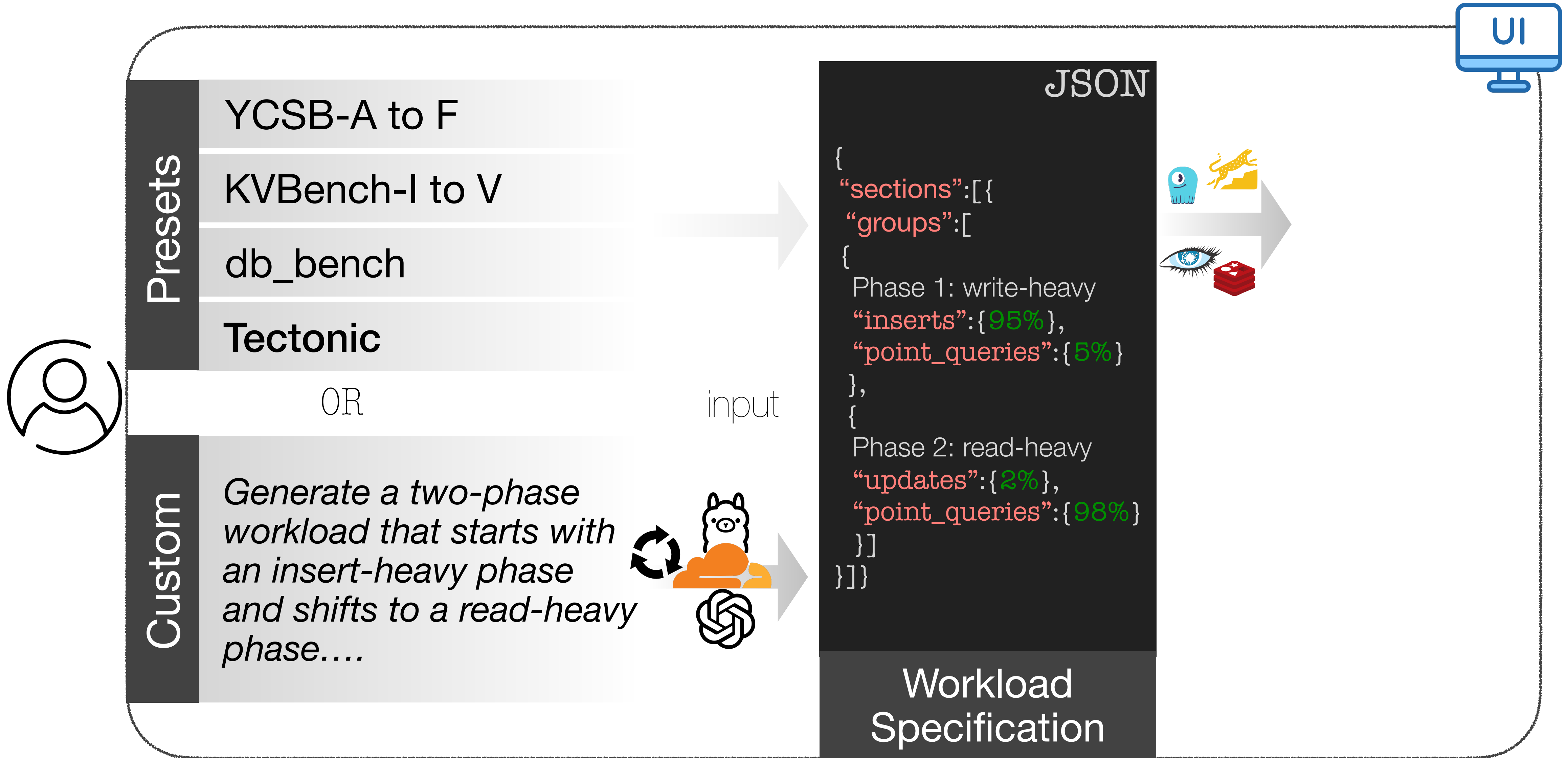
TexBench



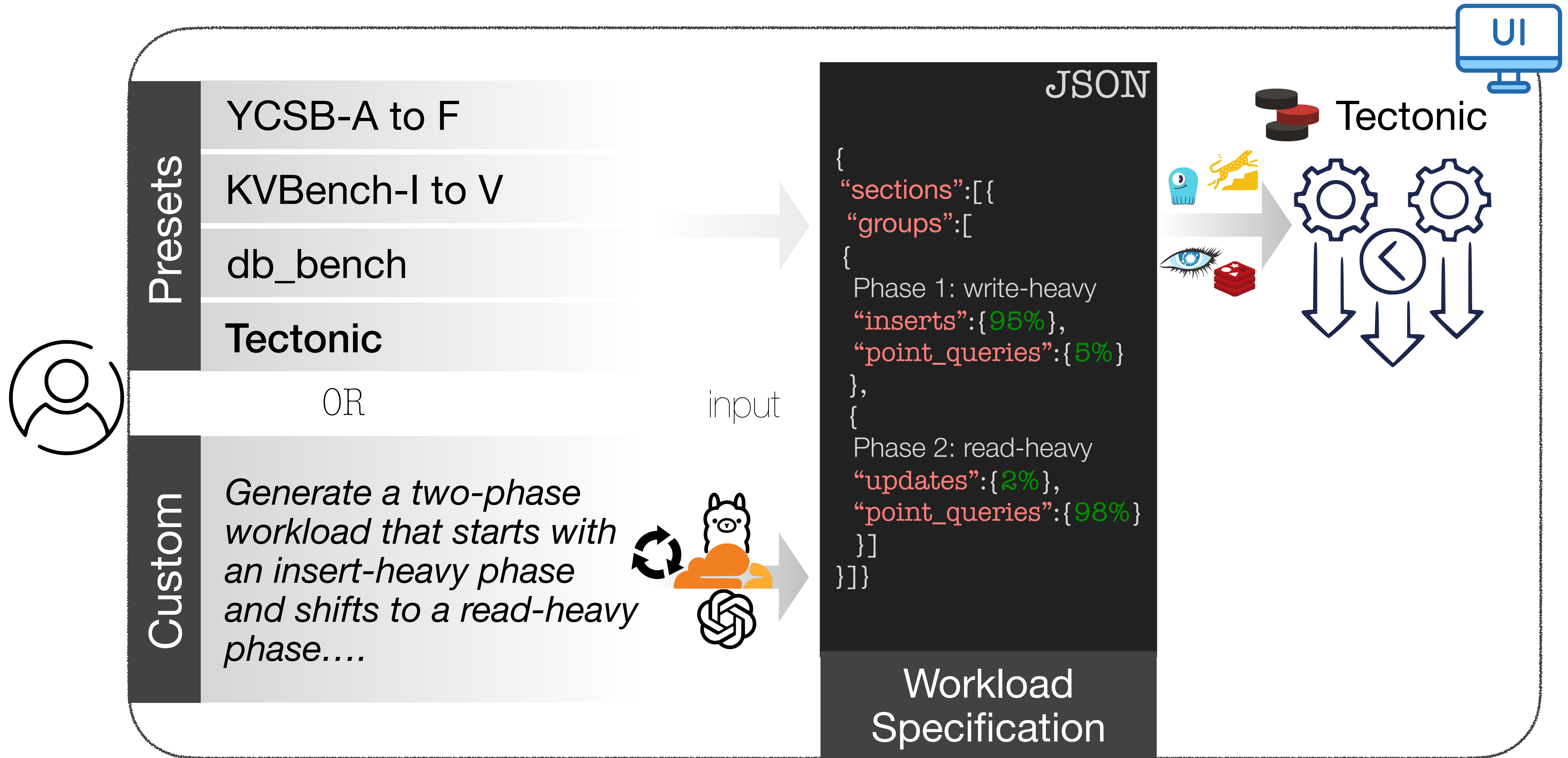
TexBench



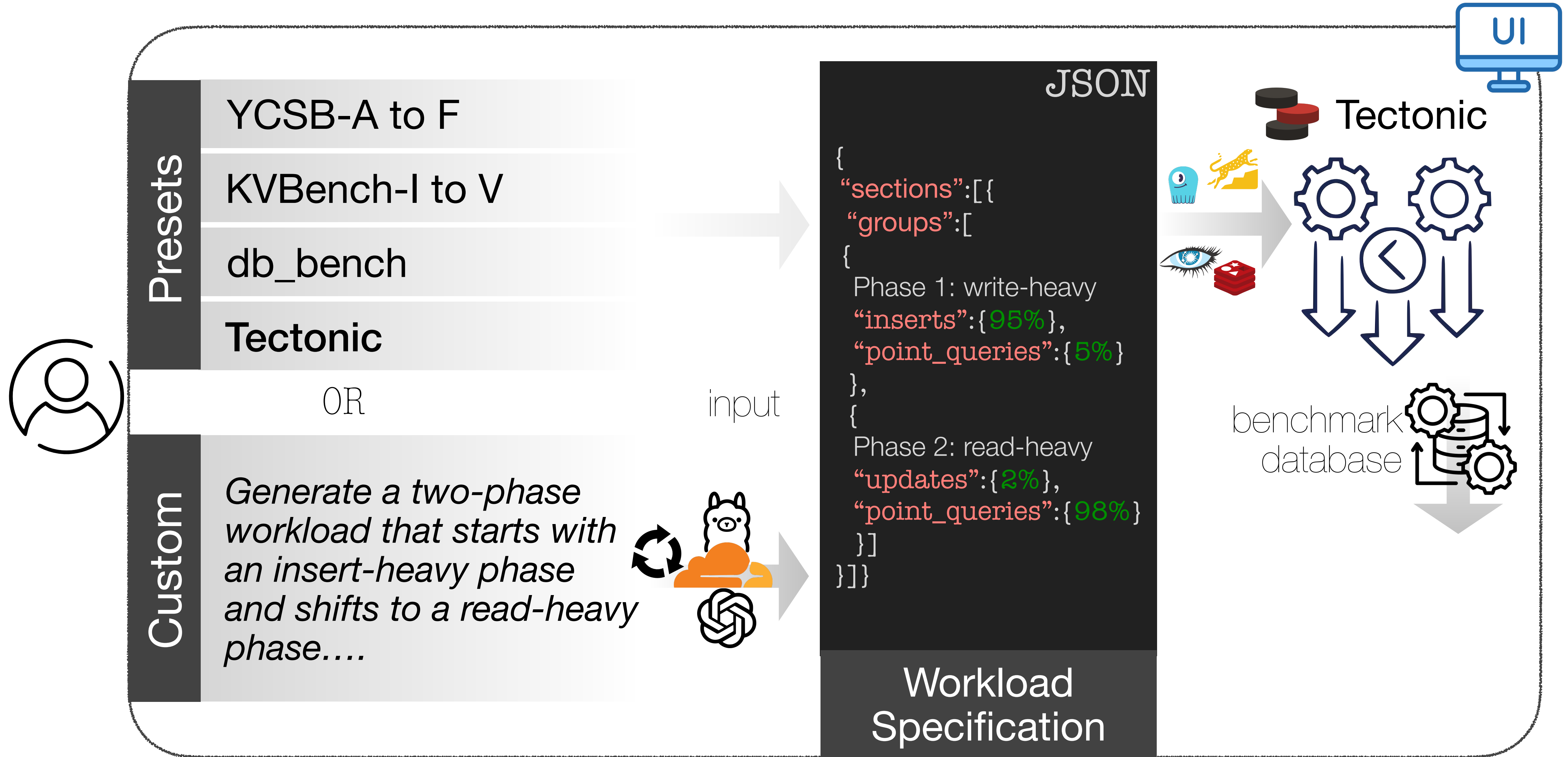
TexBench



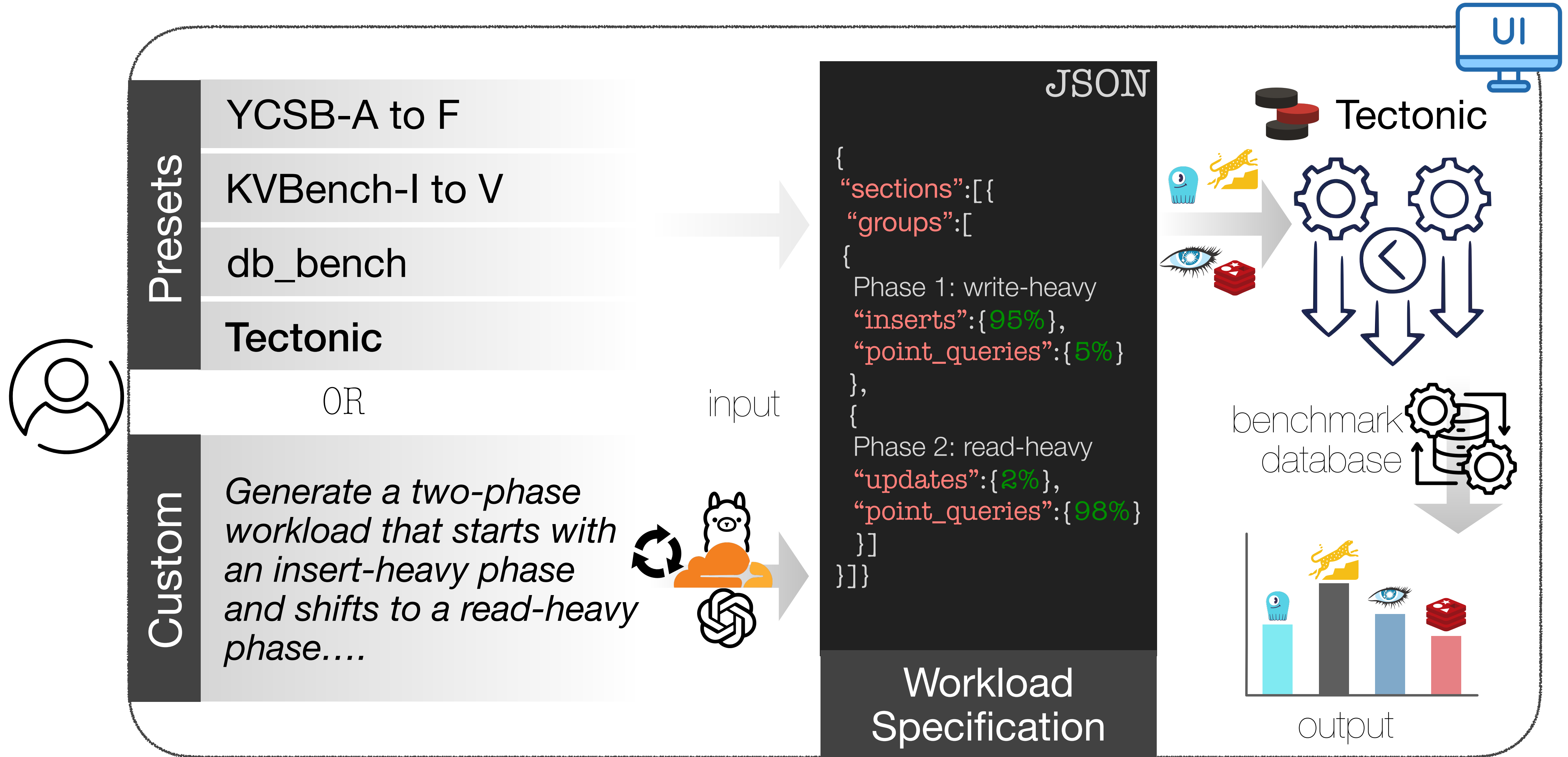
TexBench



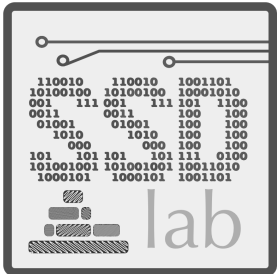
TexBench



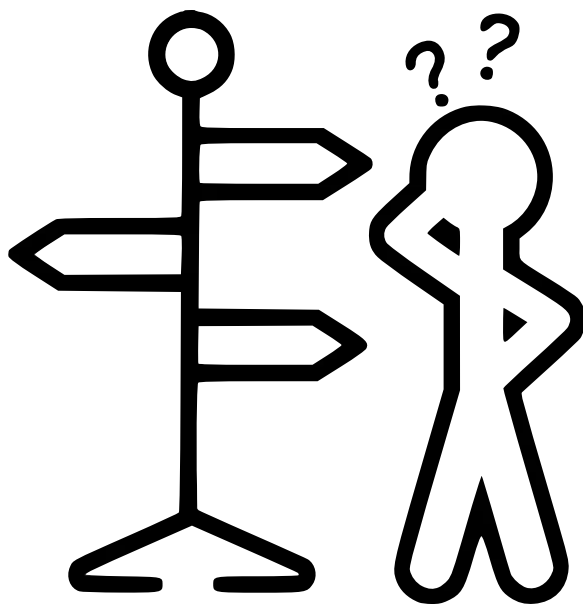
TexBench



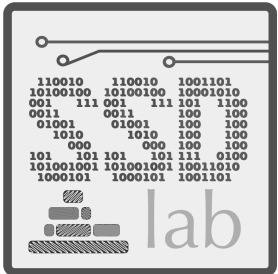
Challenges: LLMs



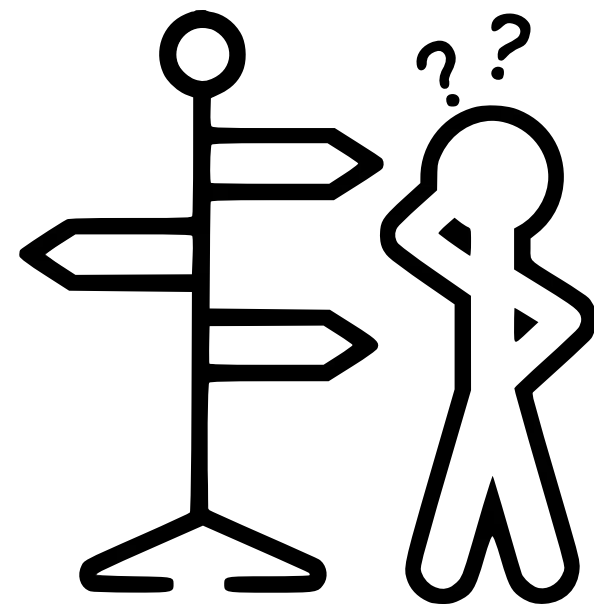
Challenges: LLMs



lost-in-the-middle
effect when the
context is too long



Challenges: LLMs

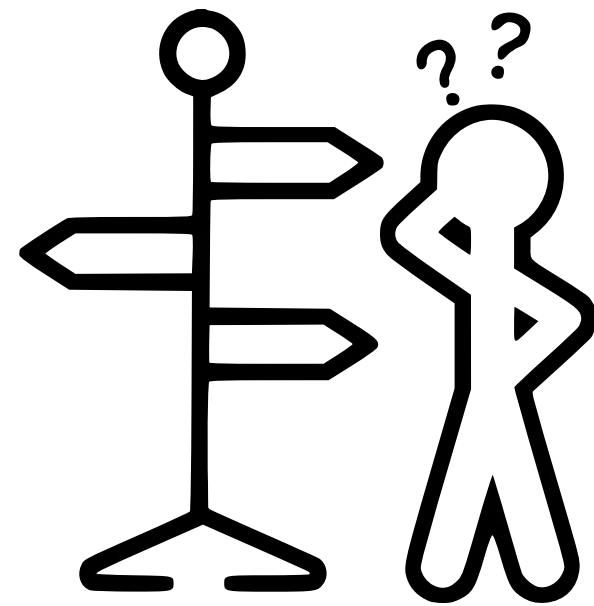


lost-in-the-middle
effect when the
context is too long



hallucination due to
training on a huge
dataset

Challenges: LLMs



lost-in-the-middle
effect when the
context is too long

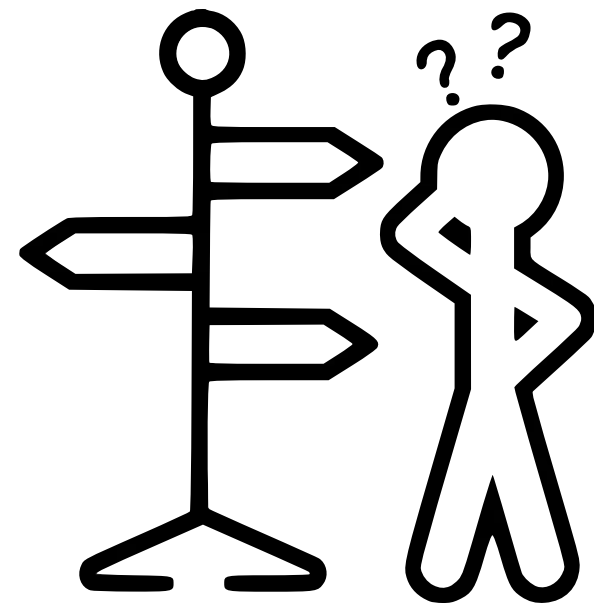


hallucination due to
training on a huge
dataset



invalid results due
to ambiguous human
language

Challenges: LLMs



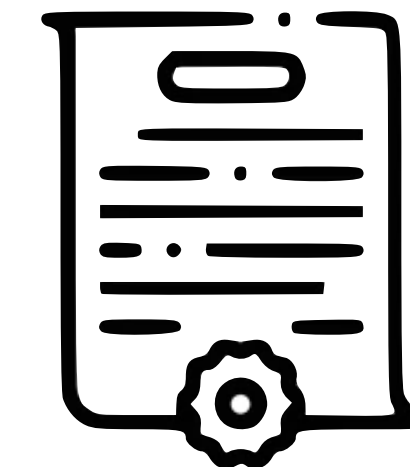
lost-in-the-middle
effect when the
context is too long



hallucination due to
training on a huge
dataset

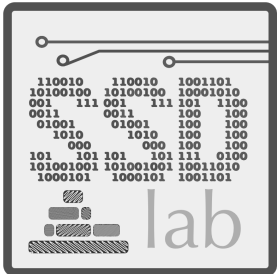


invalid results due
to ambiguous human
language



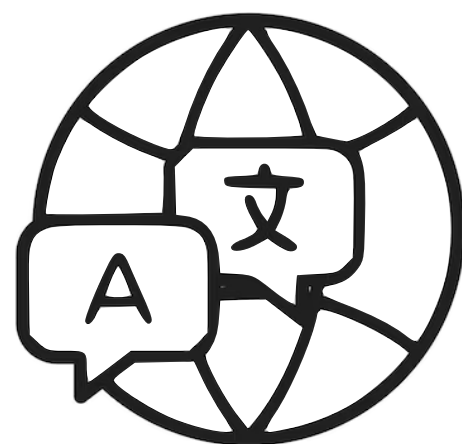
parsing LLMs
textual response to
modify JSON

TexBench’s LLM Core

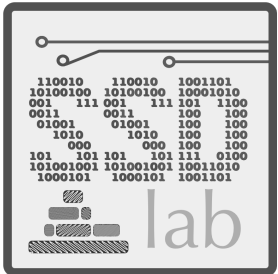


Brandeis
UNIVERSITY

TexBench's LLM Core

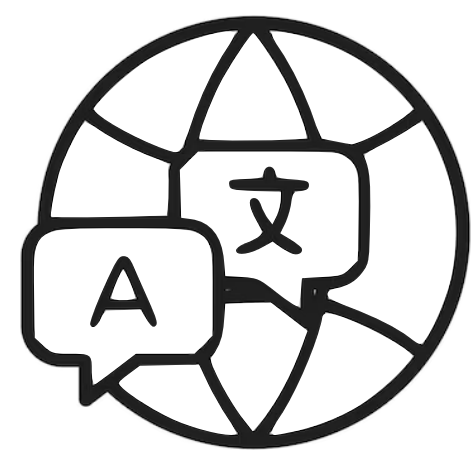


Domain Specific Language



Brandeis
UNIVERSITY

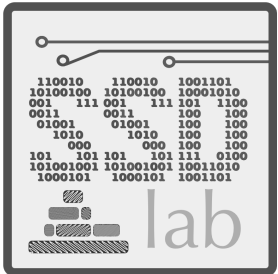
TexBench's LLM Core



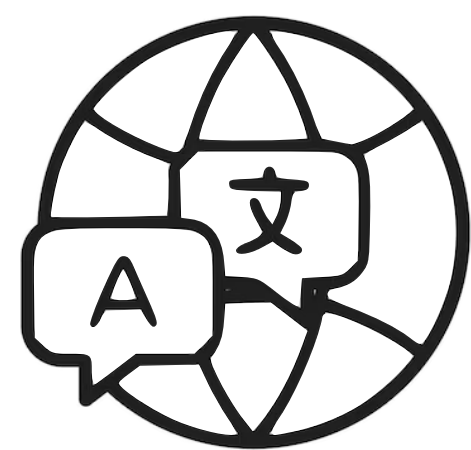
Domain Specific Language



Tectonic vocabulary



TexBench's LLM Core



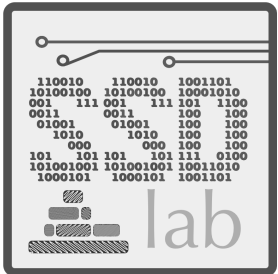
Domain Specific Language



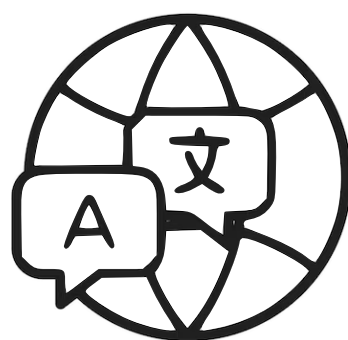
Tectonic vocabulary



System rules



TexBench's LLM Core



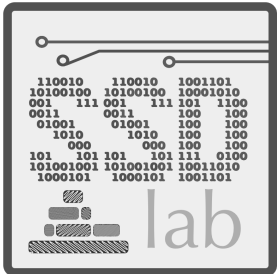
Domain Specific Language



Tectonic vocabulary

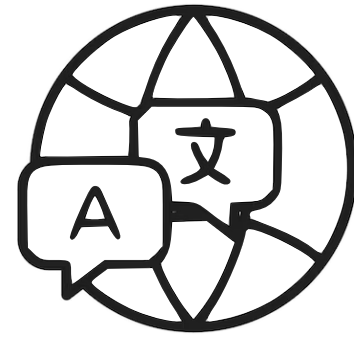


System rules



Brandeis
UNIVERSITY

TexBench's LLM Core



Domain Specific Language



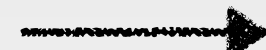
Tectonic vocabulary



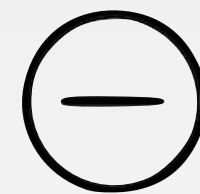
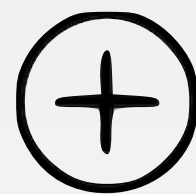
System rules



user intent

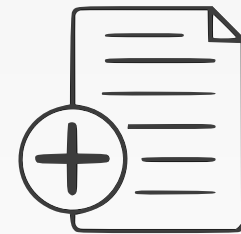


DSL cmds



add or edit operation

set_operation_field

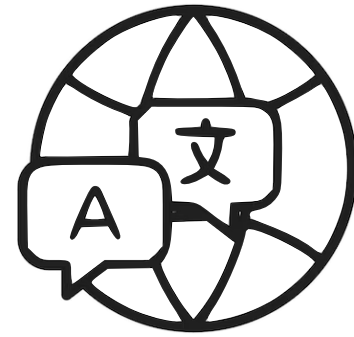


add new phase or group

append_group

8 commands, 30 grammar rules

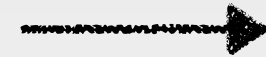
TexBench's LLM Core



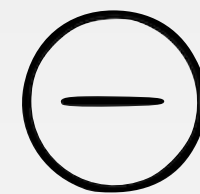
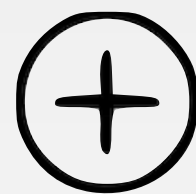
Domain Specific Language



user intent

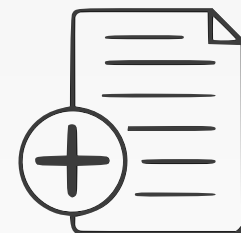


DSL cmds



add or edit operation

`set_operation_field`



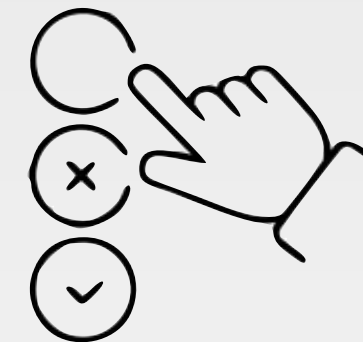
add new phase or group

`append_group`

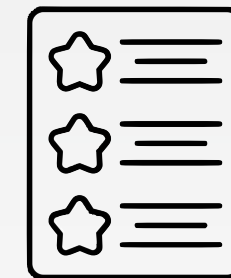
8 commands, 30 grammar rules



Tectonic vocabulary



allowed operations



required properties
per operation

`inserts:`

(op_count, key, value)

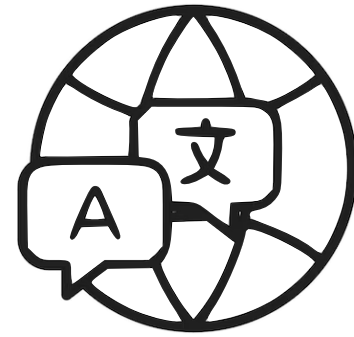
`point_queries:`

(op_count, key)



System rules

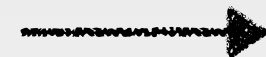
TexBench's LLM Core



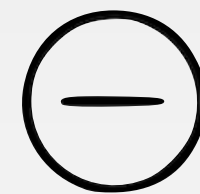
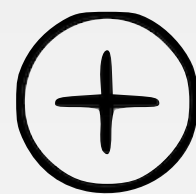
Domain Specific Language



user intent

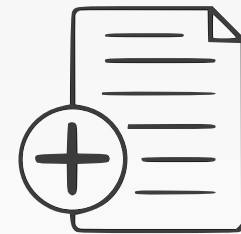


DSL cmds



add or edit operation

set_operation_field



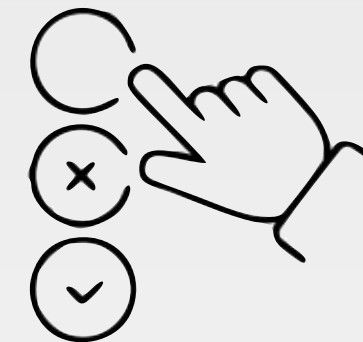
add new phase or group

append_group

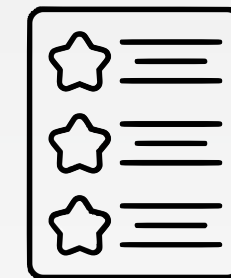
8 commands, 30 grammar rules



Tectonic vocabulary



allowed operations



required properties
per operation

inserts:

(op_count, key, value)

point_queries:

(op_count, key)



System rules

acceptable
format

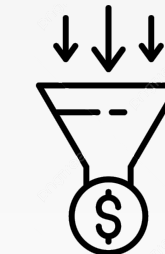


What to
respond?

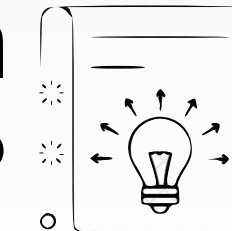


DSL cmds

KB to 1024
K to 1000

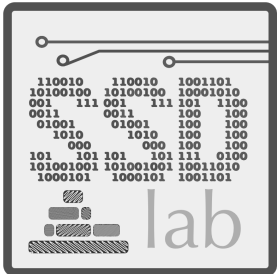


assumption
if any?

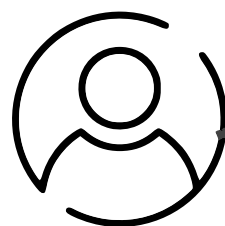


14 rules

TexBench's LLM Prompt_



TexBench's LLM Prompt_

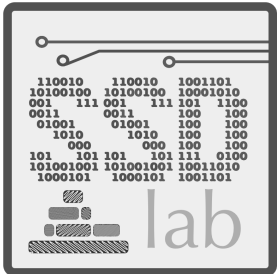


Generate a two-phase workload that starts with an insert-heavy phase...

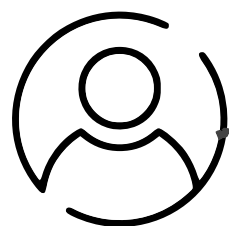
Domain Specific Language

Tectonic vocabulary

System rules



TexBench's LLM Prompt_



Generate a two-phase workload that starts with an insert-heavy phase...

Domain Specific Language

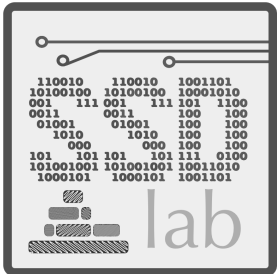
Tectonic vocabulary

System rules

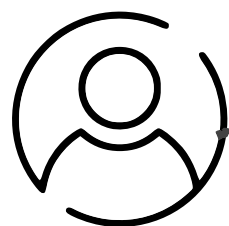
```
JSON
{
  "sections": [{
    "groups": [{
      Phase 1: write-heavy
      "inserts": {95%},
      "point_queries": {5%}
    },
    {
      Phase 2: read-heavy
      "updates": {2%},
      "point_queries": {98%}
    }
  ]
}]
```

initially empty

```
{
  "prompt": "Generate a two-phase ...
             System rules ...
             DSL grammar...",
  "schema_hint": "config schema",
  "current_json": "JSON",
}
```



TexBench's LLM Prompt_



Generate a two-phase workload that starts with an insert-heavy phase...

Domain Specific Language

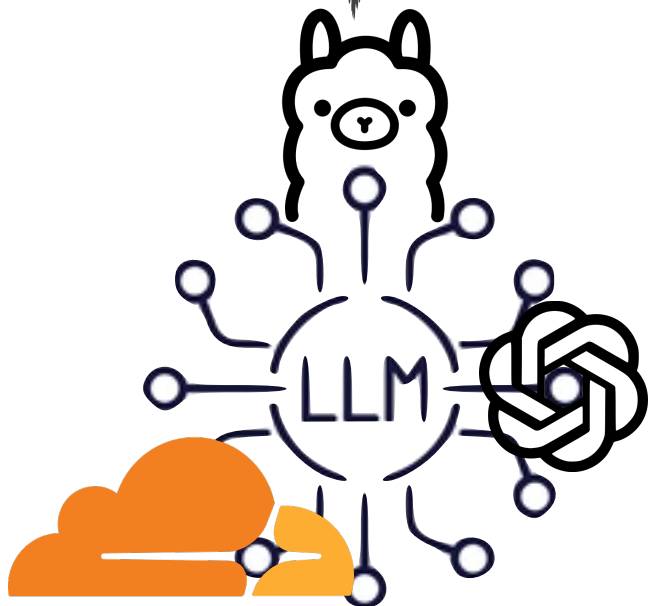
Tectonic vocabulary

System rules

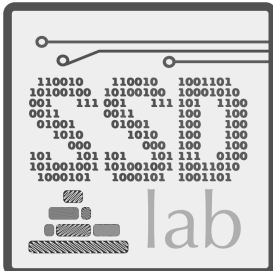
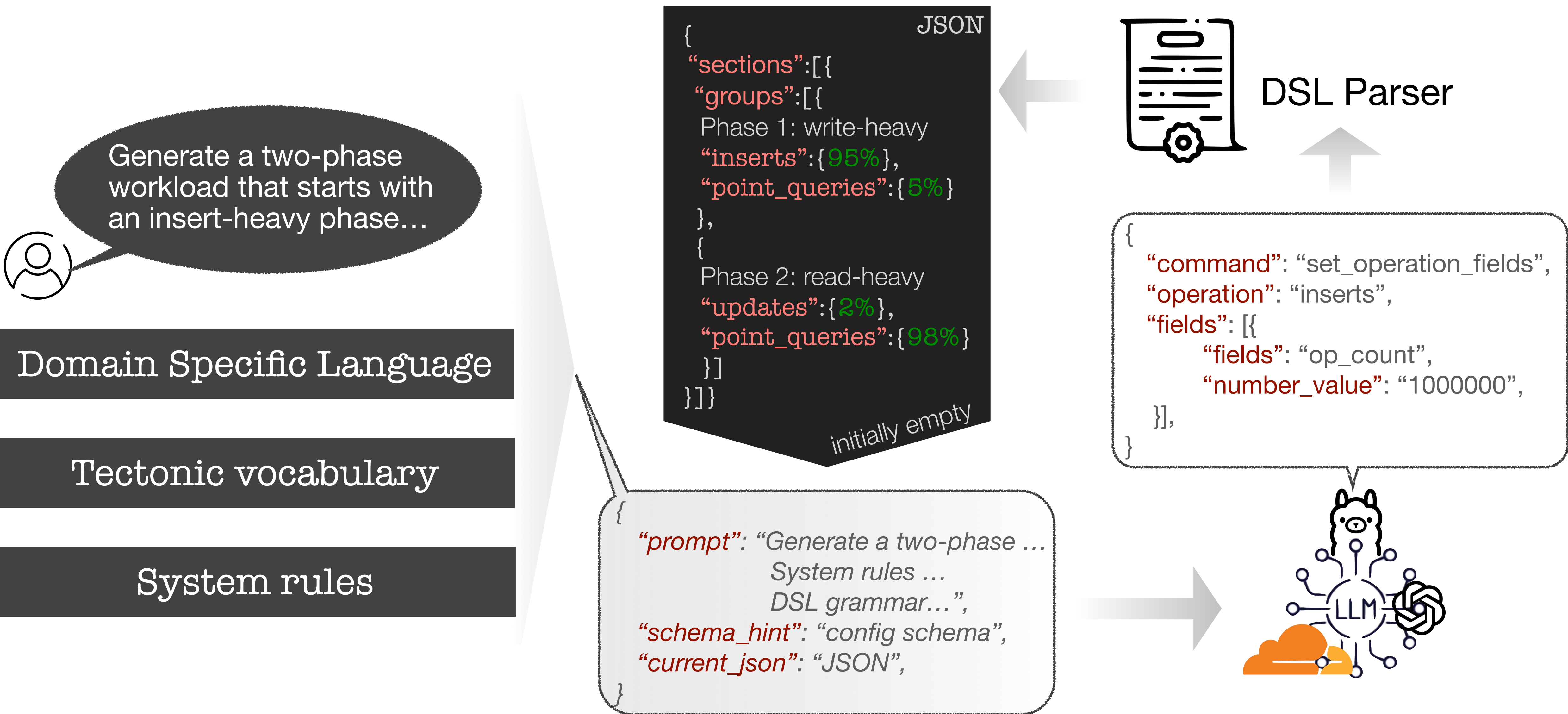
```
JSON
{
  "sections": [{
    "groups": [{
      Phase 1: write-heavy
      "inserts": {95%},
      "point_queries": {5%}
    },
    {
      Phase 2: read-heavy
      "updates": {2%},
      "point_queries": {98%}
    }
  ]
}]
initially empty
```

```
{
  "prompt": "Generate a two-phase ...
             System rules ...
             DSL grammar...",
  "schema_hint": "config schema",
  "current_json": "JSON",
}
```

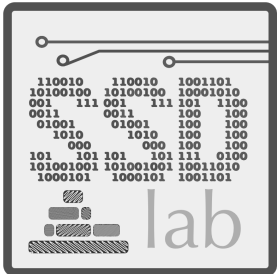
```
{
  "command": "set_operation_fields",
  "operation": "inserts",
  "fields": [{
    "fields": "op_count",
    "number_value": "1000000",
  }],
}
```



TexBench's LLM Prompt_

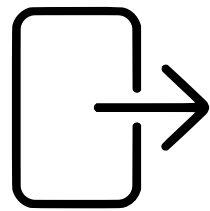


Evaluation

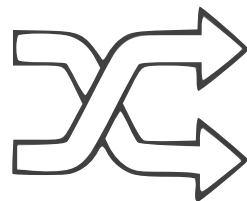


Evaluation

corpus of 50 prompts



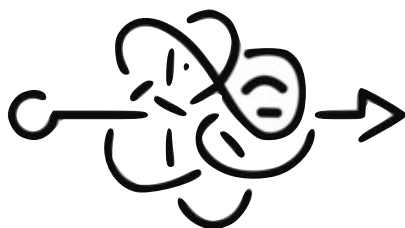
simple



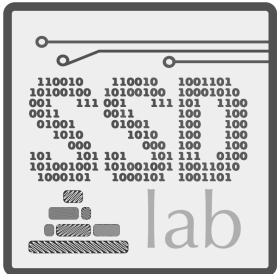
mixed



phased

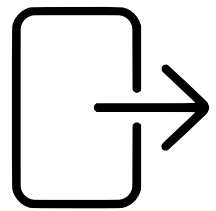


complex



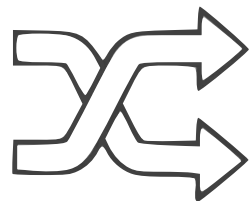
Evaluation

corpus of 50 prompts



simple

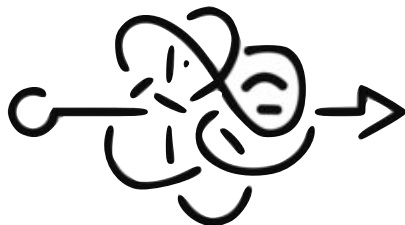
Generate an **insert-only** workload with 100K operations.



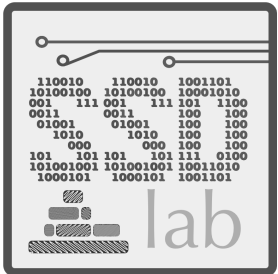
mixed



phased

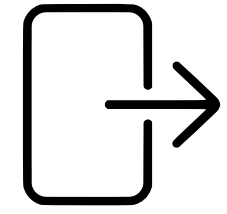


complex



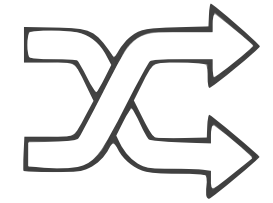
Evaluation

corpus of 50 prompts



simple

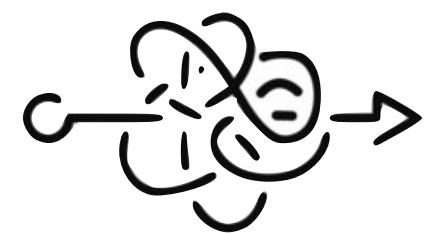
Generate an **insert-only** workload with 100K operations.



mixed

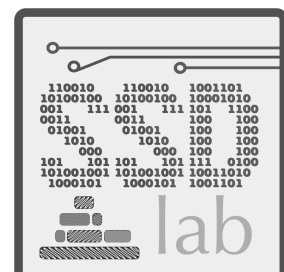


phased



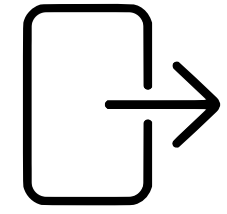
complex

Create a production-like **two-phase workload**: phase 1 inserts 2M records with 64-byte keys and 256-byte values; phase 2 runs 1.8M point queries with a **Zipf selection distribution** ($n=2M$, $s=0.99$) and 200K range queries with 1% selectivity, all interleaved.



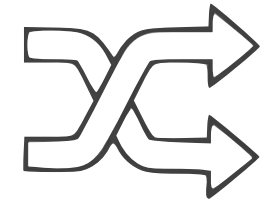
Evaluation

corpus of 50 prompts



simple

Generate an **insert-only** workload with 100K operations.



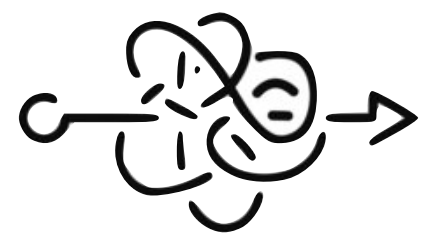
mixed

Generate a workload with 800K **inserts** and 200K **point queries**, totaling 1M operations.



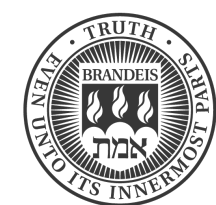
phased

Create a **two-phase workload**: first, insert 1M keys, then run 500K point queries on those keys. The two phases must be separate groups so that phase 2 reads keys written in phase 1.

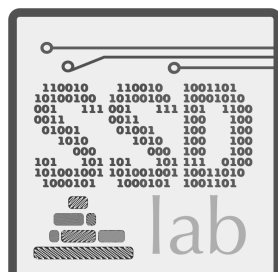


complex

Create a production-like **two-phase workload**: phase 1 inserts 2M records with 64-byte keys and 256-byte values; phase 2 runs 1.8M point queries with a **Zipf selection distribution** ($n=2M$, $s=0.99$) and 200K range queries with 1% selectivity, all interleaved.

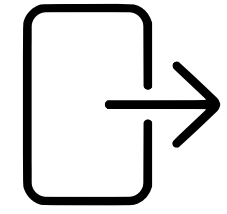


Brandeis
UNIVERSITY



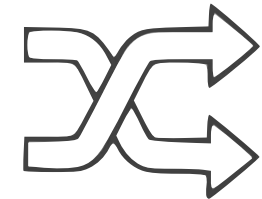
Evaluation

corpus of 50 prompts



simple

Generate an **insert-only** workload with 100K operations.



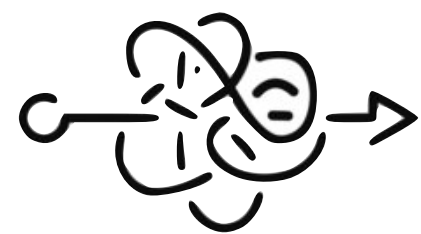
mixed

Generate a workload with 800K **inserts** and 200K **point queries**, totaling 1M operations.



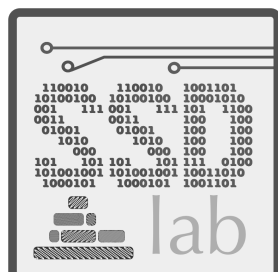
phased

Create a **two-phase workload**: first, insert 1M keys, then run 500K point queries on those keys. The two phases must be separate groups so that phase 2 reads keys written in phase 1.



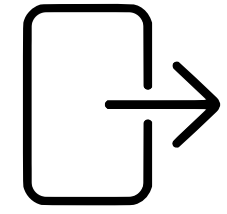
complex

Create a production-like **two-phase workload**: phase 1 inserts 2M records with 64-byte keys and 256-byte values; phase 2 runs 1.8M point queries with a **Zipf selection distribution** ($n=2M$, $s=0.99$) and 200K range queries with 1% selectivity, all interleaved.



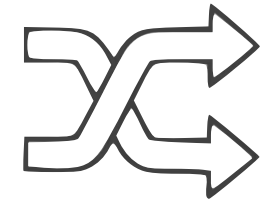
Evaluation

corpus of 50 prompts



simple

Generate an **insert-only** workload with 100K operations.



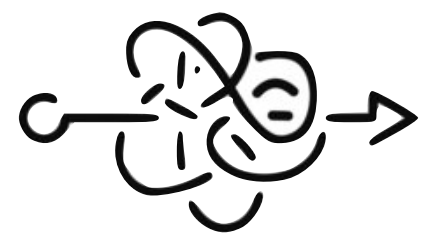
mixed

Generate a workload with 800K **inserts** and 200K **point queries**, totaling 1M operations.



phased

Create a **two-phase workload**: first, insert 1M keys, then run 500K point queries on those keys. The two phases must be separate groups so that phase 2 reads keys written in phase 1.



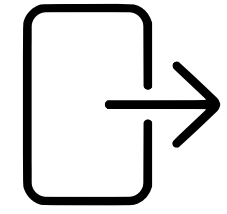
complex

Create a production-like **two-phase workload**: phase 1 inserts 2M records with 64-byte keys and 256-byte values; phase 2 runs 1.8M point queries with a **Zipf selection distribution** ($n=2M$, $s=0.99$) and 200K range queries with 1% selectivity, all interleaved.



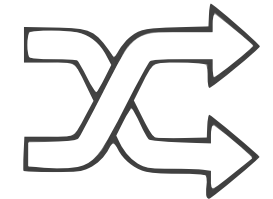
Evaluation

corpus of 50 prompts



simple

Generate an **insert-only** workload with 100K operations.



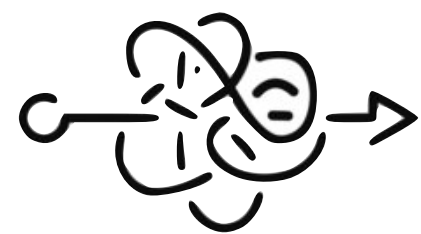
mixed

Generate a workload with 800K **inserts** and 200K **point queries**, totaling 1M operations.



phased

Create a **two-phase workload**: first, insert 1M keys, then run 500K point queries on those keys. The two phases must be separate groups so that phase 2 reads keys written in phase 1.

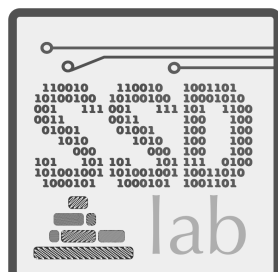


complex

Create a production-like **two-phase workload**: phase 1 inserts 2M records with 64-byte keys and 256-byte values; phase 2 runs 1.8M point queries with a **Zipf selection distribution** ($n=2M$, $s=0.99$) and 200K range queries with 1% selectivity, all interleaved.

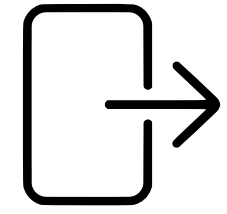


Intel Xeon
Platinum 8380 CPUs
2.30GHz, 80 cores
NVIDIA A100 PCIe
GPU 40GB, 256GB
DDR4 RAM



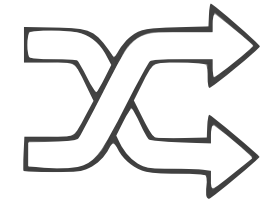
Evaluation

corpus of 50 prompts



simple

Generate an **insert-only** workload with 100K operations.



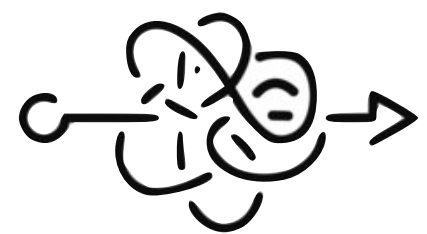
mixed

Generate a workload with 800K **inserts** and 200K **point queries**, totaling 1M operations.



phased

Create a **two-phase workload**: first, insert 1M keys, then run 500K point queries on those keys. The two phases must be separate groups so that phase 2 reads keys written in phase 1.

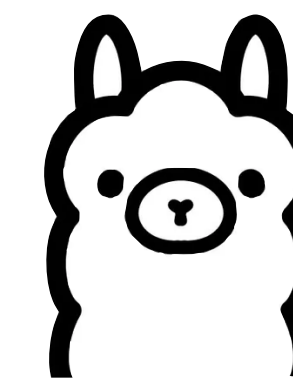


complex

Create a production-like **two-phase workload**: phase 1 inserts 2M records with 64-byte keys and 256-byte values; phase 2 runs 1.8M point queries with a **Zipf selection distribution** ($n=2M$, $s=0.99$) and 200K range queries with 1% selectivity, all interleaved.



Intel Xeon
Platinum 8380 CPUs
2.30GHz, 80 cores
NVIDIA A100 PCIe
GPU 40GB, 256GB
DDR4 RAM

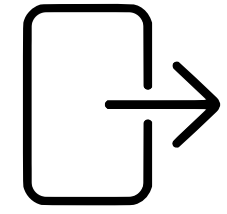


Llama 3.2: 3b



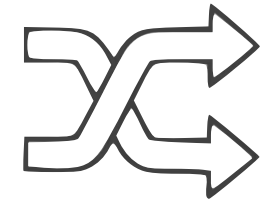
Evaluation

corpus of 50 prompts



simple

Generate an **insert-only** workload with 100K operations.



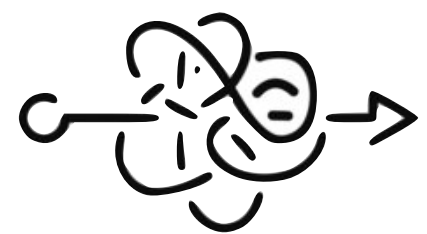
mixed

Generate a workload with 800K **inserts** and 200K **point queries**, totaling 1M operations.



phased

Create a **two-phase workload**: first, insert 1M keys, then run 500K point queries on those keys. The two phases must be separate groups so that phase 2 reads keys written in phase 1.

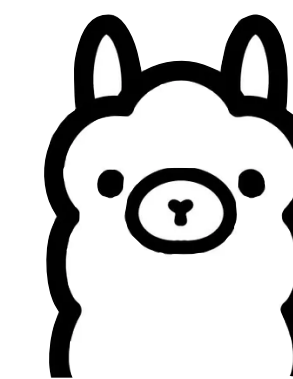


complex

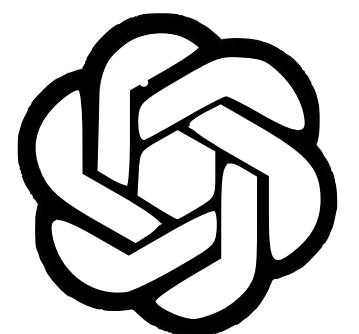
Create a production-like **two-phase workload**: phase 1 inserts 2M records with 64-byte keys and 256-byte values; phase 2 runs 1.8M point queries with a **Zipf selection distribution** ($n=2M$, $s=0.99$) and 200K range queries with 1% selectivity, all interleaved.



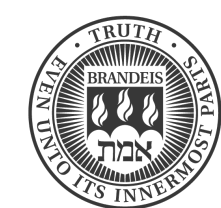
Intel Xeon
Platinum 8380 CPUs
2.30GHz, 80 cores
NVIDIA A100 PCIe
GPU 40GB, 256GB
DDR4 RAM



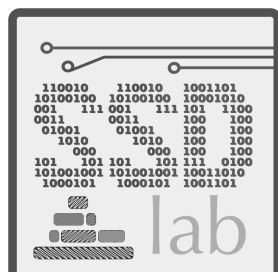
Llama 3.2: 3b



GPT-4o

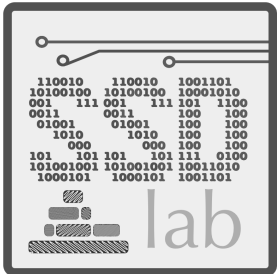
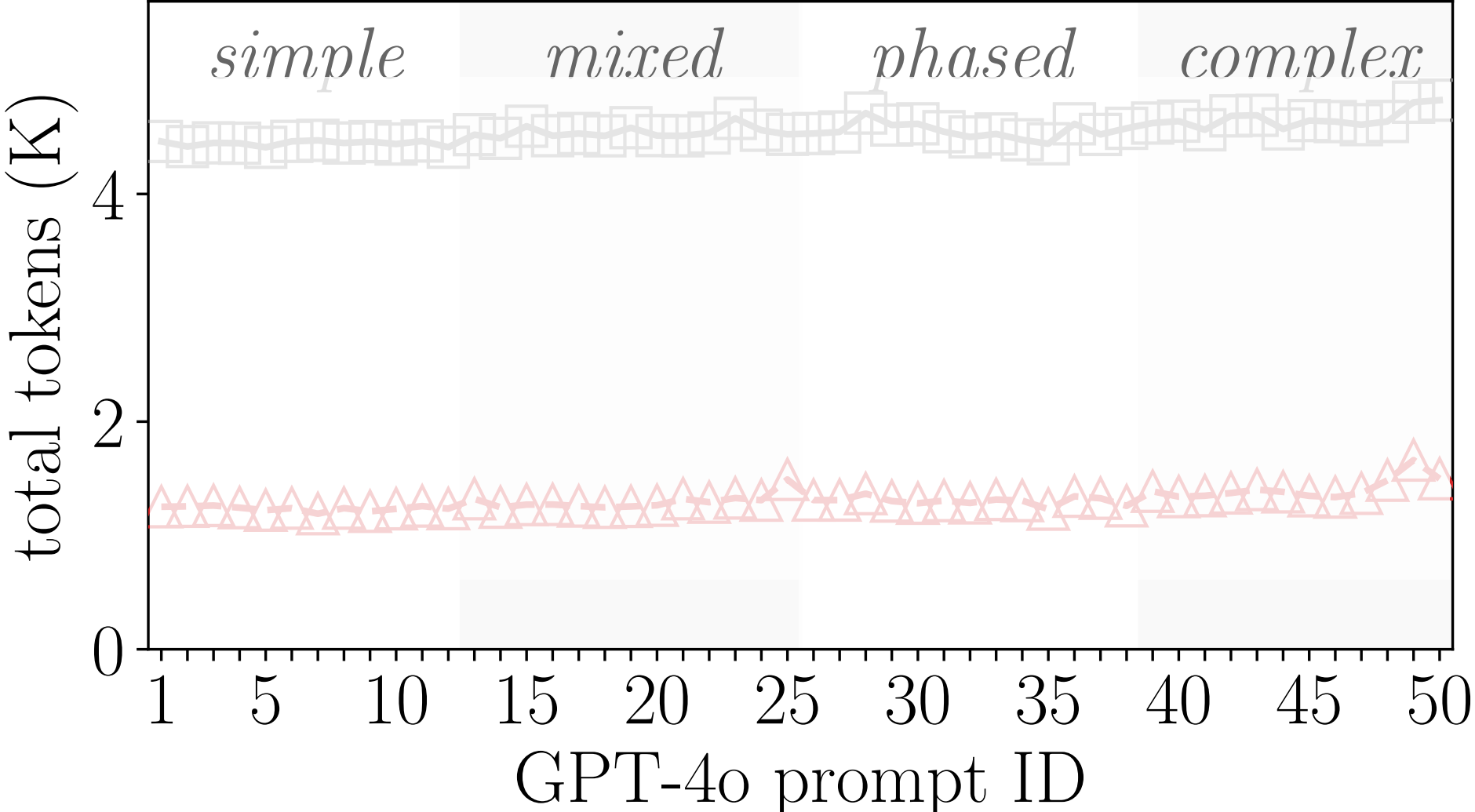
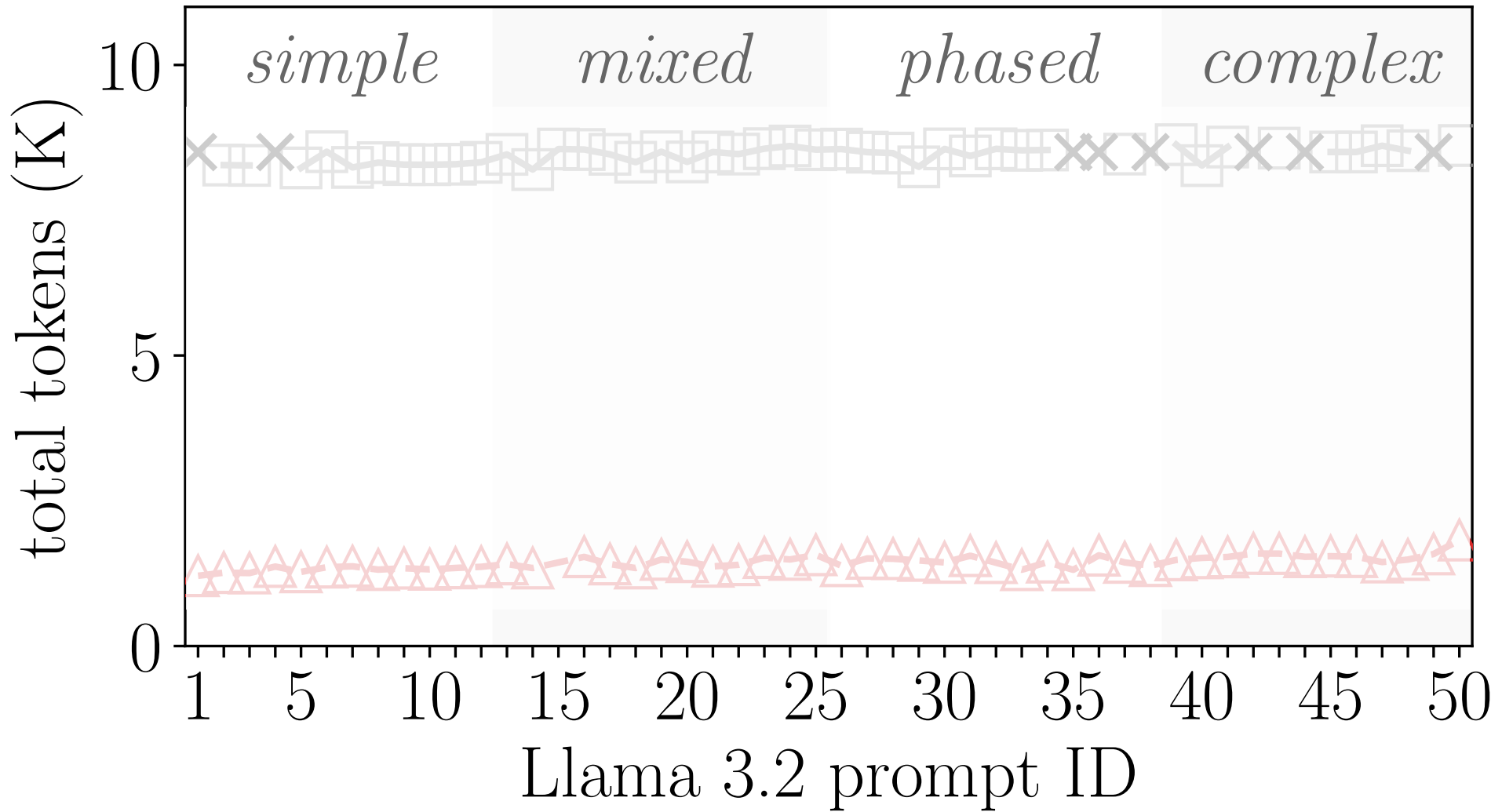


Brandeis
UNIVERSITY



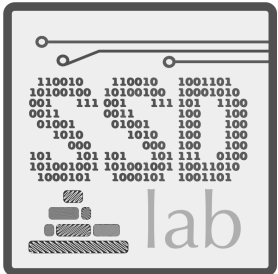
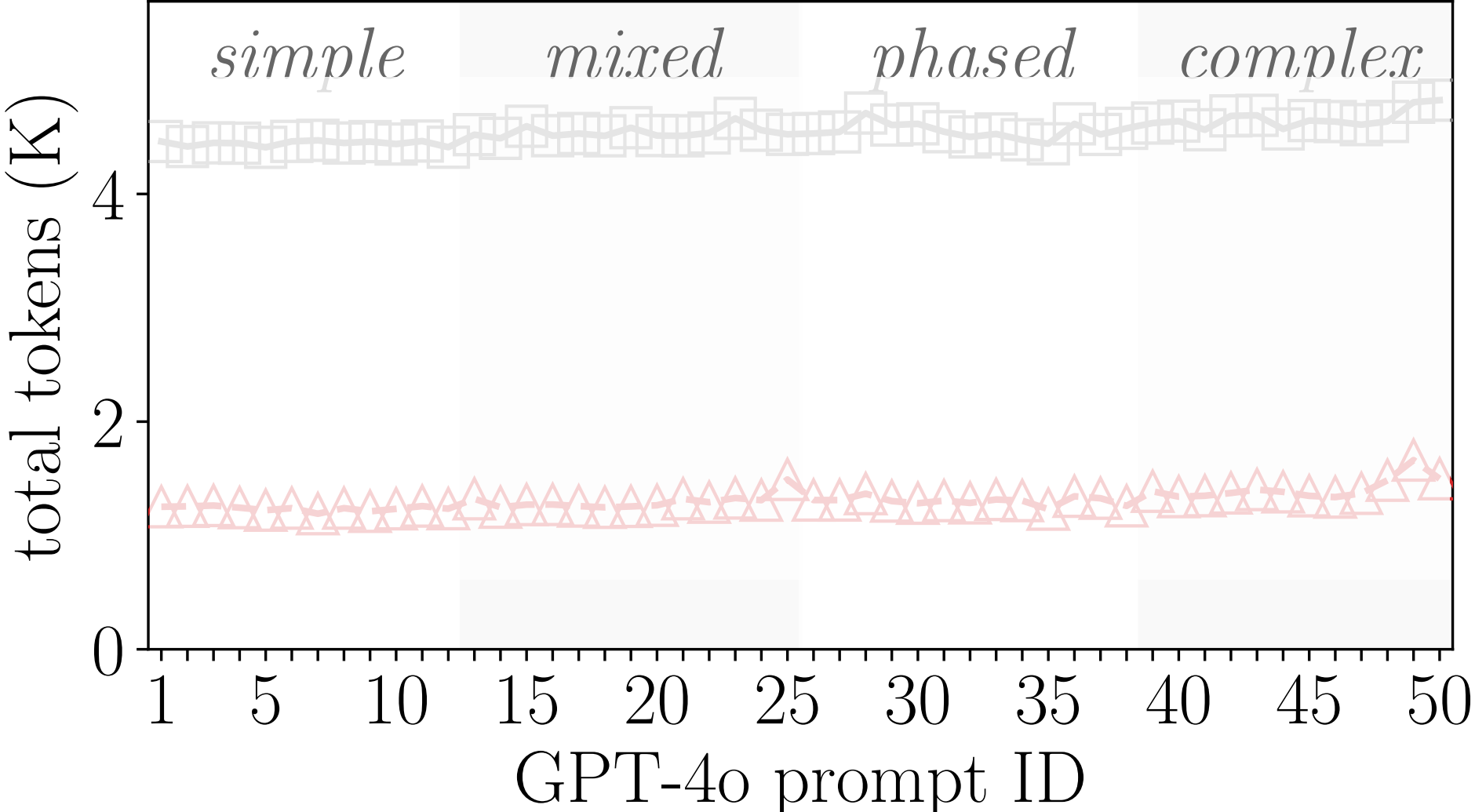
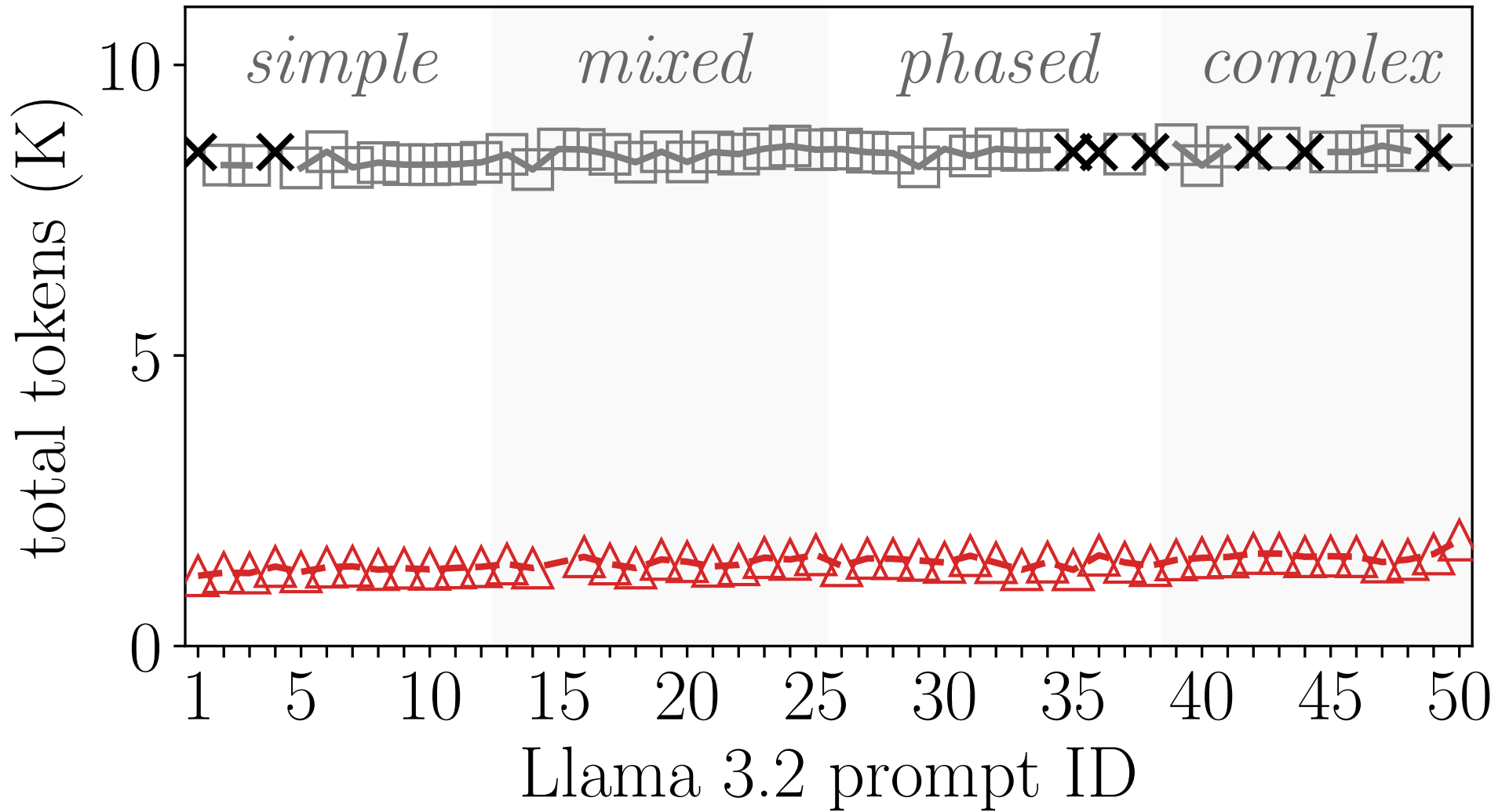
Evaluation

--△-- *TexBench* —□— raw prompt



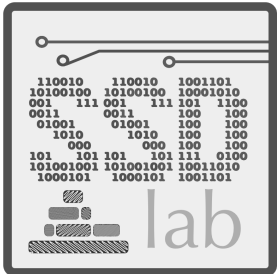
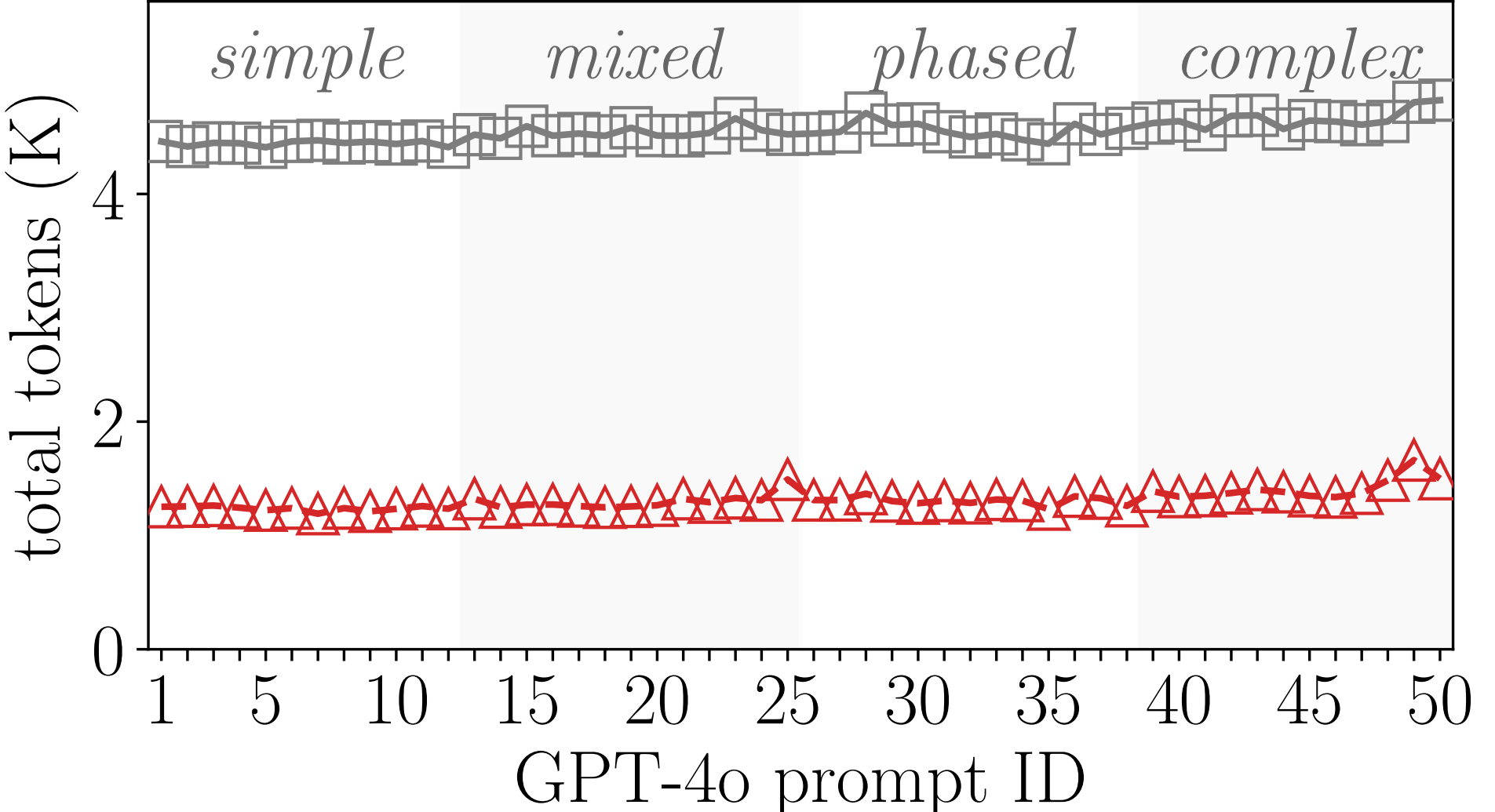
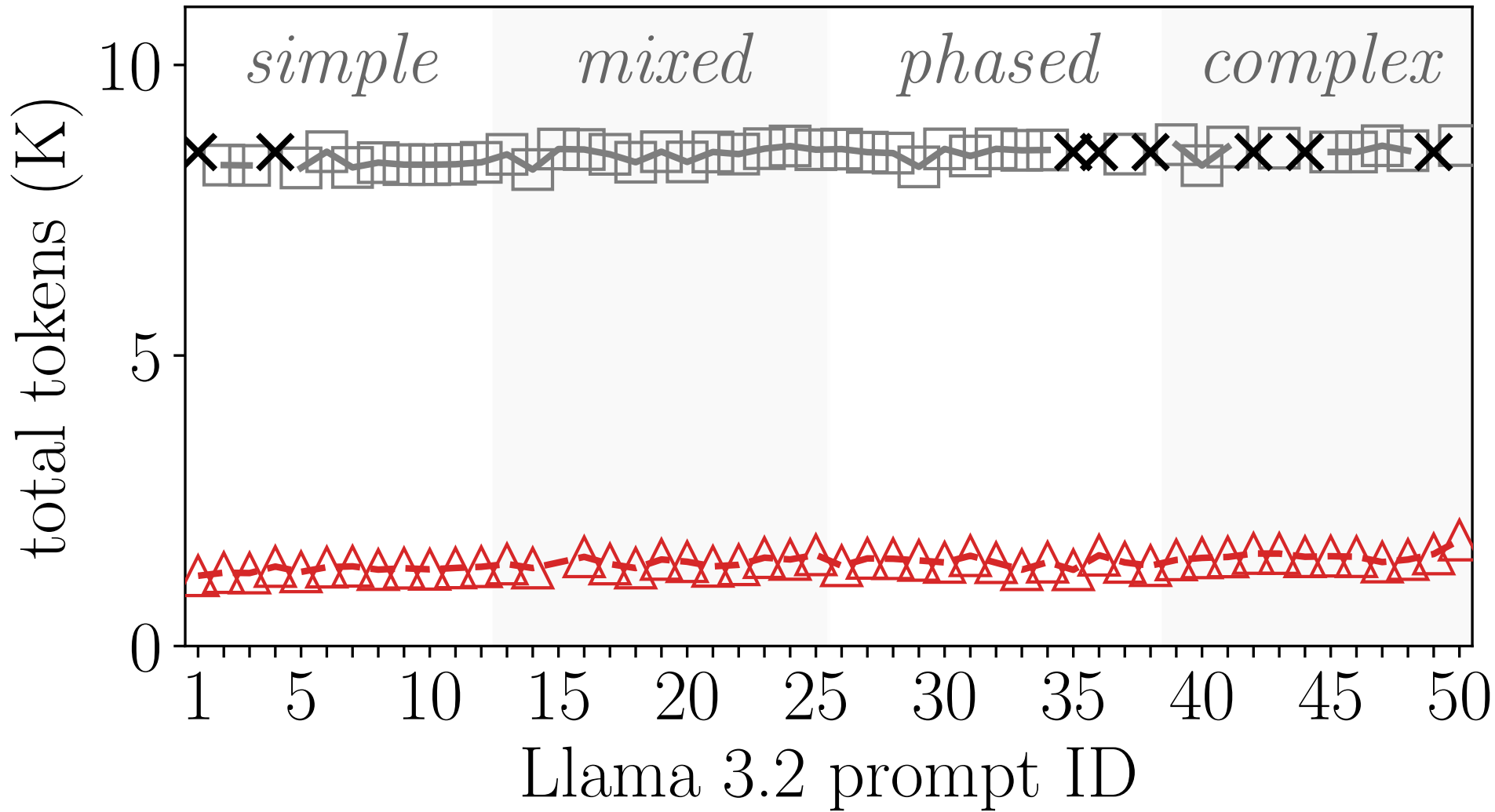
Evaluation

--△-- *TexBench* —□— raw prompt



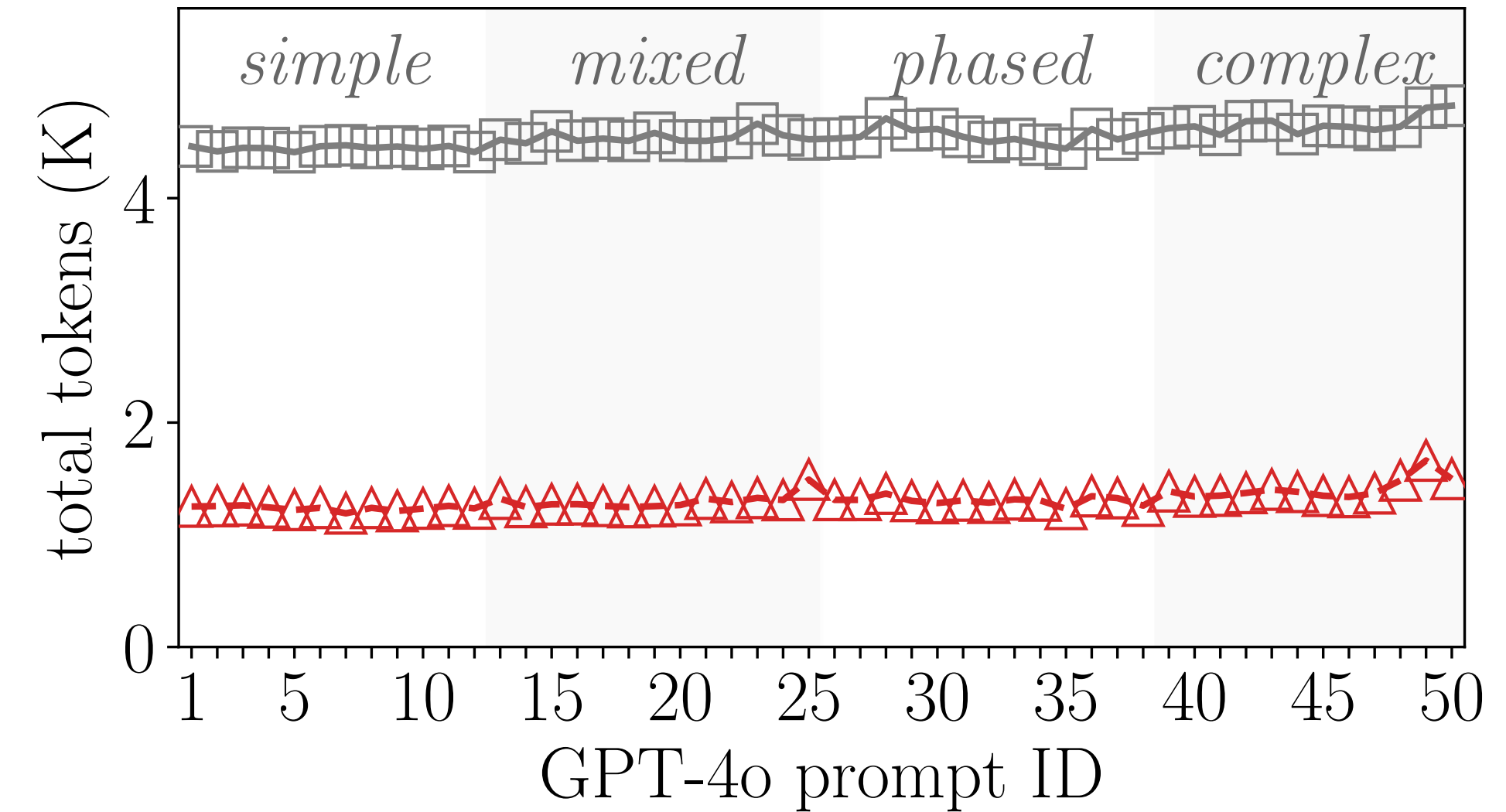
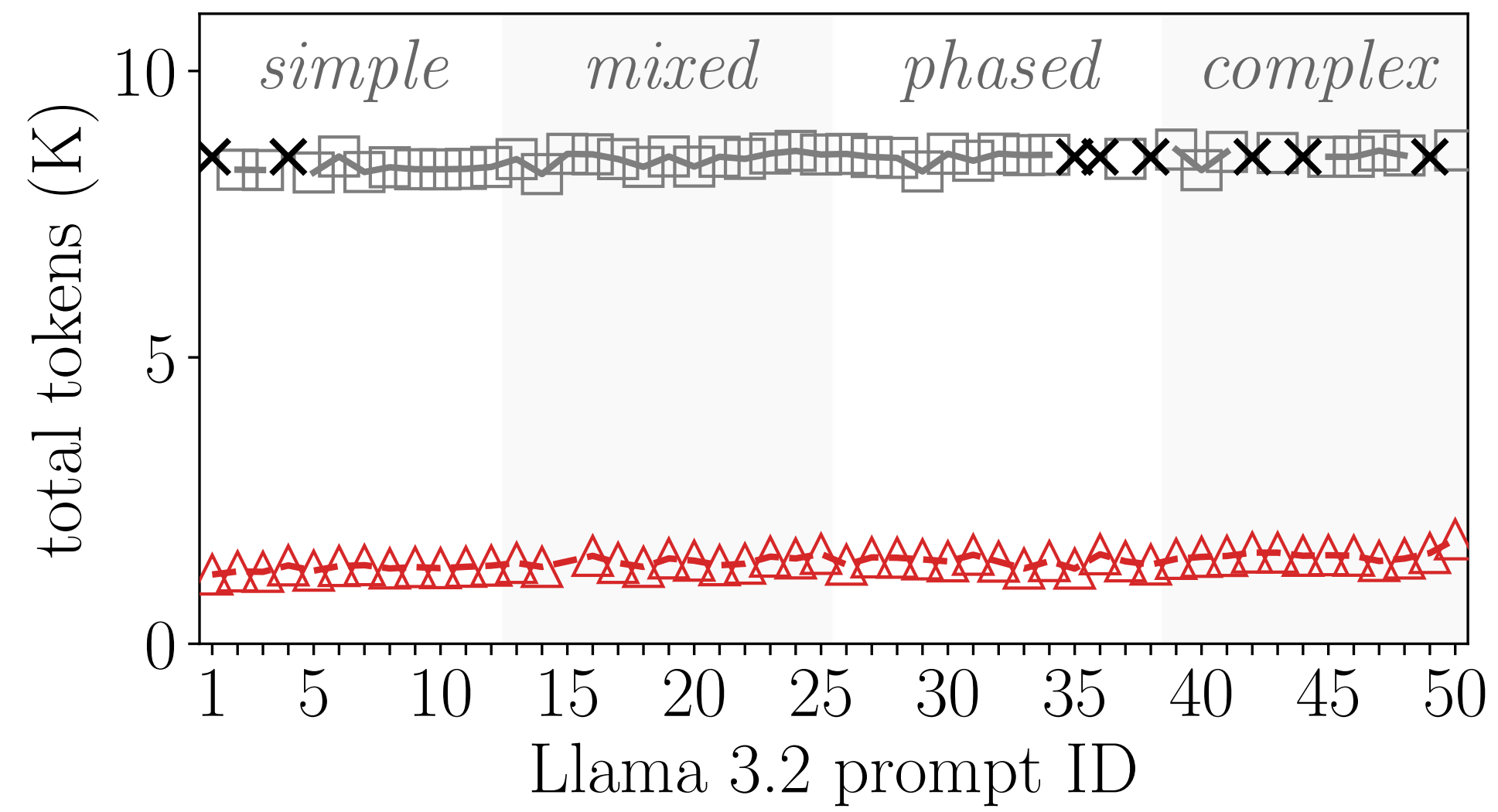
Evaluation

--△-- *TexBench* —□— raw prompt



Evaluation

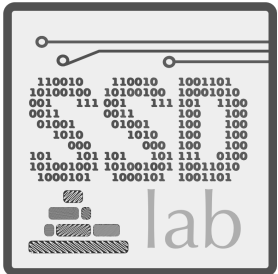
--△-- *TexBench* —□— raw prompt



TexBench consumes fewer tokens compared to raw prompting across models and saves cost

Evaluation

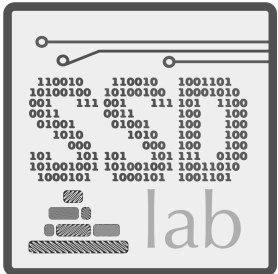
category	Llama 3.2		GPT-4o	
	raw prompt	<i>TexBench</i>	raw prompt	<i>TexBench</i>
simple	4.8	79.2	95.2	100
mixed	5.8	65.4	91.9	96
phased	4.1	56.3	88.1	93.7
complex	3.9	44.4	90.2	95.7
overall	4.7	61.3	91.3	96.3



Evaluation

category	Llama 3.2		GPT-4o	
	raw prompt	<i>TexBench</i>	raw prompt	<i>TexBench</i>
simple	4.8	79.2	95.2	100
mixed	5.8	65.4	91.9	96
phased	4.1	56.3	88.1	93.7
complex	3.9	44.4	90.2	95.7
overall	4.7	61.3	91.3	96.3

raw prompting completely fails for the free models like Llama 3.2

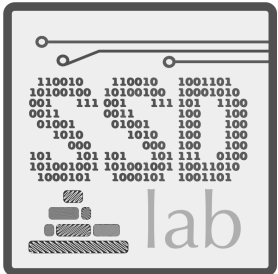


Evaluation

category	Llama 3.2		GPT-4o	
	raw prompt	<i>TexBench</i>	raw prompt	<i>TexBench</i>
simple	4.8	79.2	95.2	100
mixed	5.8	65.4	91.9	96
phased	4.1	56.3	88.1	93.7
complex	3.9	44.4	90.2	95.7
overall	4.7	61.3	91.3	96.3

raw prompting completely fails for the free models like Llama 3.2

TexBench improves accuracy for paid models like GPT-4o



Evaluation

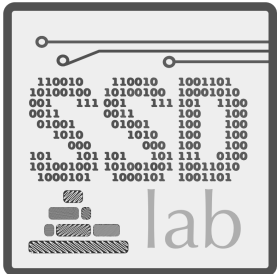
category	Llama 3.2		GPT-4o	
	raw prompt	<i>TexBench</i>	raw prompt	<i>TexBench</i>
simple	4.8	79.2	95.2	100
mixed	5.8	65.4	91.9	96
phased	4.1	56.3	88.1	93.7
complex	3.9	44.4	90.2	95.7
overall	4.7	61.3	91.3	96.3

One of the complex prompt with GPT-4o

dimension/score	maximum	raw prompt	<i>TexBench</i>
syntactic validity	5	5	5
section cardinality	5	5	5
phase cardinality	10	5	10
key-domain	5	5	5
phase ordering	10	0.0	10
operation composition	20	8	20
operation properties	35	18.7	35
schema correctness	10	10	10
total	100	56.7	100

raw prompting completely fails for the free models like Llama 3.2

TexBench improves accuracy for paid models like GPT-4o



Evaluation

category	Llama 3.2		GPT-4o	
	raw prompt	<i>TexBench</i>	raw prompt	<i>TexBench</i>
simple	4.8	79.2	95.2	100
mixed	5.8	65.4	91.9	96
phased	4.1	56.3	88.1	93.7
complex	3.9	44.4	90.2	95.7
overall	4.7	61.3	91.3	96.3

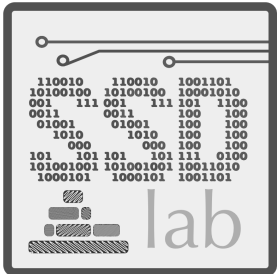
One of the complex prompt with GPT-4o

dimension/score	maximum	raw prompt	<i>TexBench</i>
syntactic validity	5	5	5
section cardinality	5	5	5
phase cardinality	10	5	10
key-domain	5	5	5
phase ordering	10	0.0	10
operation composition	20	8	20
operation properties	35	18.7	35
schema correctness	10	10	10
total	100	56.7	100

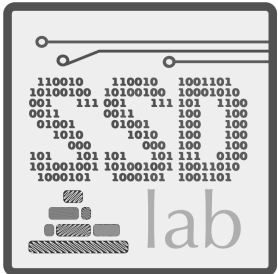
raw prompting completely fails for the free models like Llama 3.2

TexBench improves accuracy for paid models like GPT-4o

even paid models fails to emulate phased workload configuration with raw-prompting



The Key Takeaways

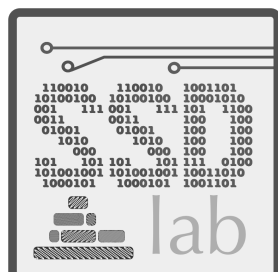


The Key Takeaways

Real-world workloads exhibit complex access patterns

Existing suites are incapable of mimicking modern workloads

Lacking a framework with a simple side-by-side comparison



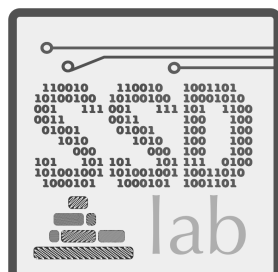
The Key Takeaways

Real-world workloads exhibit complex access patterns

Existing suites are incapable of mimicking modern workloads

Lacking a framework with a simple side-by-side comparison

TexBench: a unified UI-based LLM-powered benchmarking suite that mimics real-world workloads and offers a side-by-side comparison



TexBench: Harnessing LLMs for Efficient Key-value Benchmarking

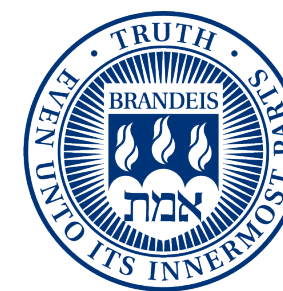
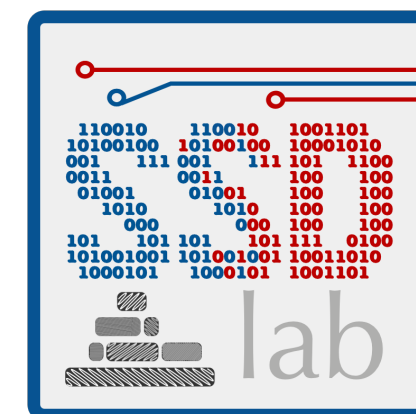
Shubham Kaushik

<https://shubhamkaushik.com>

Abhishek Chanda

Subhadeep Sarkar

<https://subhadeep.net/>



Brandeis
UNIVERSITY



TexBench: Harnessing LLMs for Efficient Key-value Benchmarking



Shubham Kaushik

<https://shubhamkaushik.com>

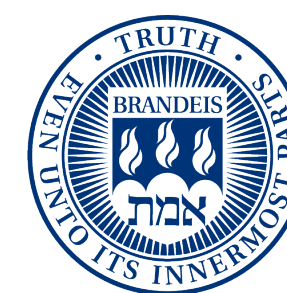
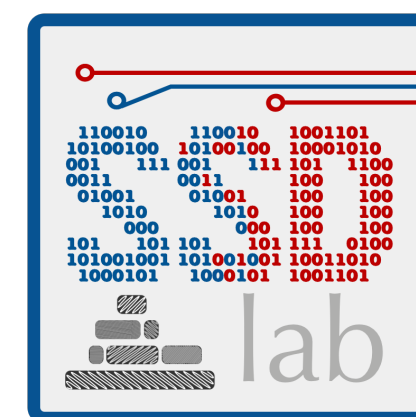


Abhishek Chanda



Subhadeep Sarkar

<https://subhadeep.net/>



Brandeis
UNIVERSITY



See TexBench in Action

Workload Structure

+ Add Section

Editing Section 1 / Phase 1

Section 1

Phase 1: inserts • point queries ×

Phase 2: updates • point queries ×

Click a phase to edit its configuration

+ Add Phase

Character Set ⓘ

alphanumeric

▼

Skip Key Contains Check ⓘ

☐ Enable across sections

Operations ⓘ

☒ Inserts ⓘ

☐ Updates ⓘ

☒ Point Queries ⓘ

☐ Range Queries ⓘ

☐ Merges ⓘ

☐ Point Deletes ⓘ

☐ Range Deletes ⓘ

☐ Sorted

☐ Empty Point Deletes ⓘ

☐ Empty Point Queries ⓘ

Inserts settings

Apply defaults

Inserts specification.

Operation Character Set

Use workload default (alphanumeric)

▼

Op Count ⓘ

80000

Advanced expression editors

Point Queries settings

Apply defaults

Non-empty point queries specification.

Op Count ⓘ

20000

Selection Distribution ⓘ

uniform

▼

Selection Min ⓘ

0

Selection Max ⓘ

1

☒ RocksDB

☒ Cassandra

☐ ScyllaDB

☒ Redis

☐ PrintDB

Run Workload ▶

Valid

Hide JSON

Download

Copy

Describe

Ready

Close

```
{
  "character_set": "alphanumeric",
  "sections": [
    {
      "groups": [
        {
          "enable_granular_stats": true,
          "inserts": {
            "op_count": 80000,
            "key": {
              "uniform": {
                "len": 20,
                "character_set": "alphanumeric"
              }
            },
            "val": {
              "uniform": {
                "len": 1024,
                "character_set": "alphanumeric"
              }
            }
          },
          "point_queries": {
            "op_count": 20000,
            "selection": {
              "zipf": {
                "n": 1000000,
                "s": 1.5
              }
            }
          }
        },
        {
          "enable_granular_stats": true,
          "updates": {
            "op_count": 2000,
            "val": {
              "uniform": {
                "len": 1024,
                "character_set": "alphanumeric"
              }
            }
          }
        }
      ]
    }
  ]
}
```

Generate an insert-heavy workload with 1KB key-value size and 100K operations, with 80% inserts and 20% reads using zipfian distribution.

Generated the workload with 80K inserts and 20K point queries.

Assumptions:

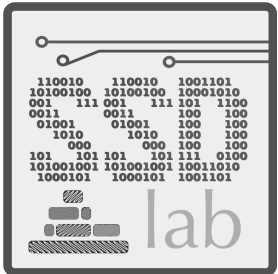
- Using alphanumeric character set
- For point queries, using zipf selection parameters n 1000000, s 1.5 because they were not specified.

Add another read-heavy group with with 20% updates and 80% point queries, where the total number of operations in the phase is 10K and point queries follow a uniform distribution.

Updated the workload by adding 2K updates and changing 8K point queries.

Assumptions:

e.g. Change to 80% reads and 20% writes...



See TexBench in Action

Workload Structure

+ Add Section

Editing Section 1 / Phase 1

Section 1

Phase 1: inserts • point queries

Phase 2: updates • point queries

Click a phase to edit its configuration

+ Add Phase

Character Set ⓘ

alphanumeric

Skip Key Contains Check ⓘ

☐ Enable across sections

Operations ⓘ

☒ Inserts ⓘ☐ Updates ⓘ☒ Point Queries ⓘ☐ Range Queries ⓘ☐ Merges ⓘ☐ Point Deletes ⓘ☐ Range Deletes ⓘ☐ Sorted

☐ Empty Point Deletes ⓘ☐ Empty Point Queries ⓘ

Inserts settings

Inserts specification.

Operation Character Set

Use workload default (alphanumeric)

Op Count ⓘ

80000

Advanced expression editors

Point Queries settings

Non-empty point queries specification.

Op Count ⓘ

20000

Selection Distribution ⓘ

uniform

Selection Min ⓘ

0

Selection Max ⓘ

1

☒ RocksDB☒ Cassandra☐ ScyllaDB☒ Redis☐ PrintDB

Run Workload ▶

Valid

Hide JSON

Download

Copy

Describe

Ready

Close

```
{
  "character_set": "alphanumeric",
  "sections": [
    {
      "groups": [
        {
          "enable_granular_stats": true,
          "inserts": {
            "op_count": 80000,
            "key": {
              "uniform": {
                "len": 20,
                "character_set": "alphanumeric"
              }
            },
            "val": {
              "uniform": {
                "len": 1024,
                "character_set": "alphanumeric"
              }
            }
          },
          "point_queries": {
            "op_count": 20000,
            "selection": {
              "zipf": {
                "n": 1000000,
                "s": 1.5
              }
            }
          }
        },
        {
          "enable_granular_stats": true,
          "updates": {
            "op_count": 2000,
            "val": {
              "uniform": {
                "len": 1024,
                "character_set": "alphanumeric"
              }
            }
          }
        }
      ]
    }
  ]
}
```

Generate an insert-heavy workload with 1KB key-value size and 100K operations, with 80% inserts and 20% reads using zipfian distribution.

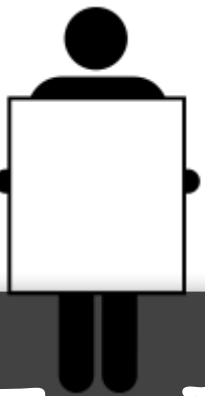
Generated the workload with 80K inserts and 20K point queries.
Assumptions:

- Using alphanumeric character set
- For point queries, using zipf selection parameters n 1000000, s 1.5 because they were not specified.

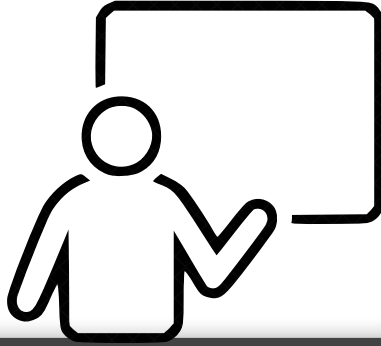
Add another read-heavy group with with 20% updates and 80% point queries, where the total number of operations in the phase is 10K and point queries follow a uniform distribution.

Updated the workload by adding 2K updates and changing 8K point queries.
Assumptions:

e.g. Change to 80% reads and 20% writes...



Today
5:15 - 6:15

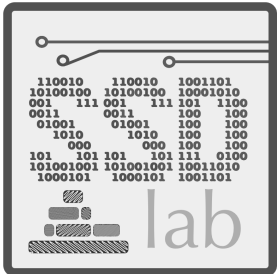


Tuesday
3:45 - 5:15

Thursday
1:45 - 3:15



shubhamkaushik.com/projects/texbench/



Brandeis
UNIVERSITY