Greptime

Online Meetup

Oct 10th 2023

Greptime

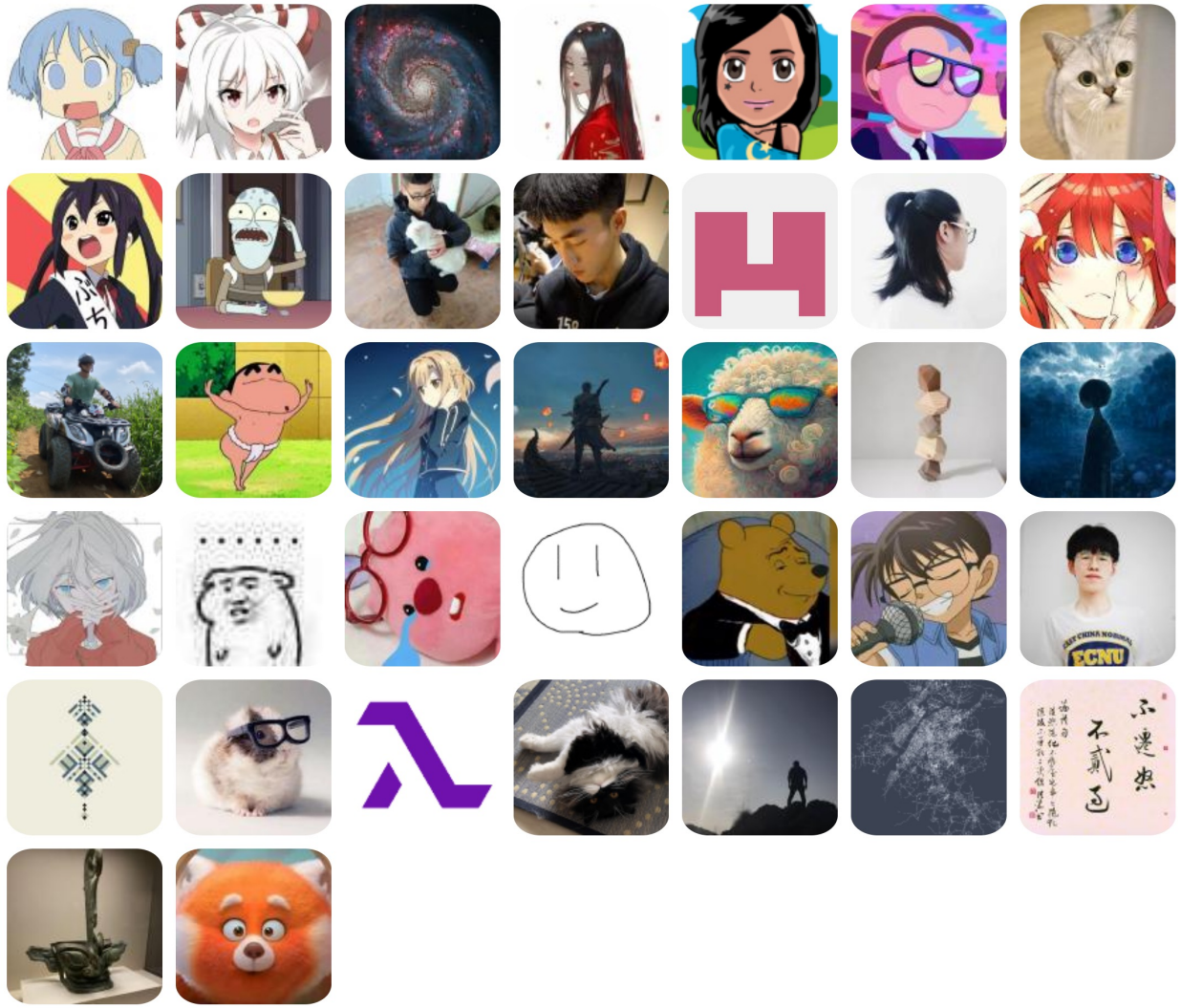
Contents

01

GreptimeDB v0.4

GreptimeDB v0.4

Greptime



700+ PRs

270+ Feats

160+ Fixes

100+ Refactors

Highlights

Greptime

- New Features
  - **New Engine Mito2: 5x~20x faster than v0.3**
  - Inspired by Cassandra, now we support TWCS (Time Window Compaction Strategy)
  - Range Select, now supporting nested use with regular functions.

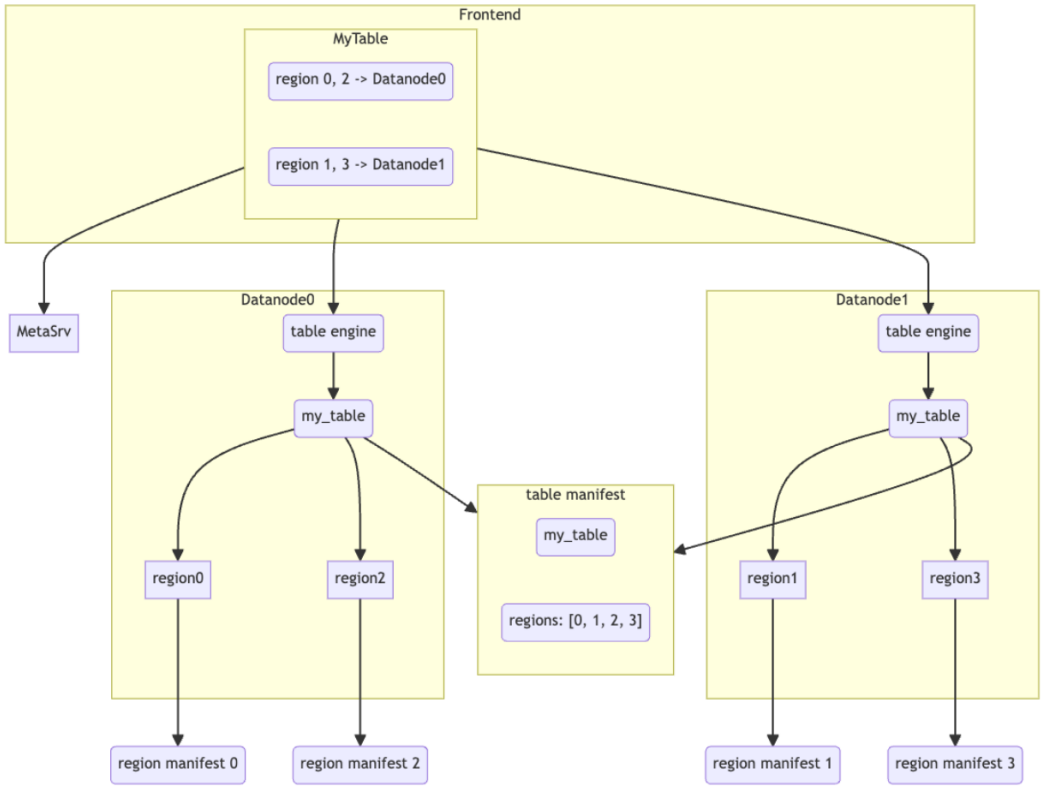
- Reliability
  - Enrich the SQLness testing scenarios
  - Relying on the continuous improvement of chaos testing cases, v0.4 will have more reliable distributed robustness.
  - Implement Create/Alter/Drop Table and other DDL operations based on the Procedure framework
  - **Refactor distributed architecture**

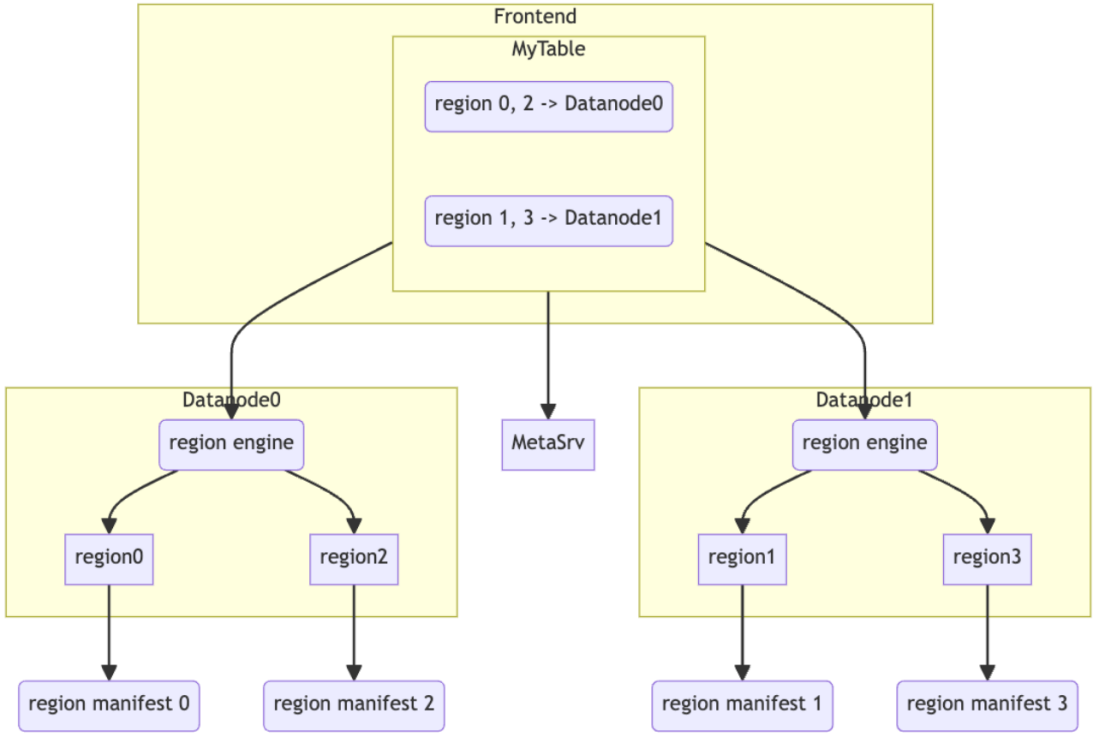
•User Experience Improvements
  - Launched an entirely new Dashboard.
  - Support external tables in the Apache ORC format.
  - Many bugfix

Refactor Distributed Architecture

Greptime

v0.3 -> v0.4

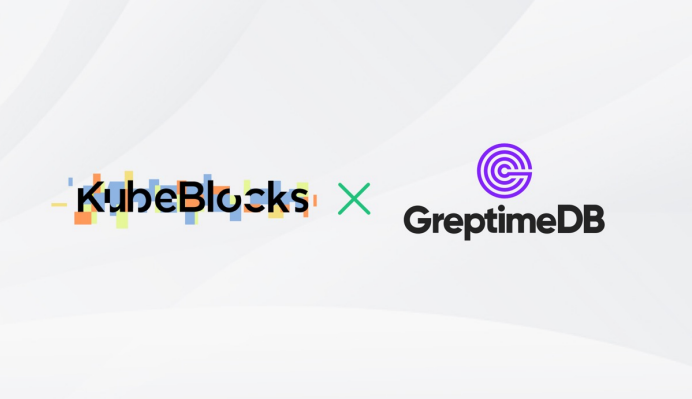




Integrations and others

Greptime

- GreptimeDB Sink for Vector
- Support OTLP metrics writing
- Support EMQX & KubeBlocks
- JS/Erlang/C++ Client
- Builds for CentOS, Windows, and Android.

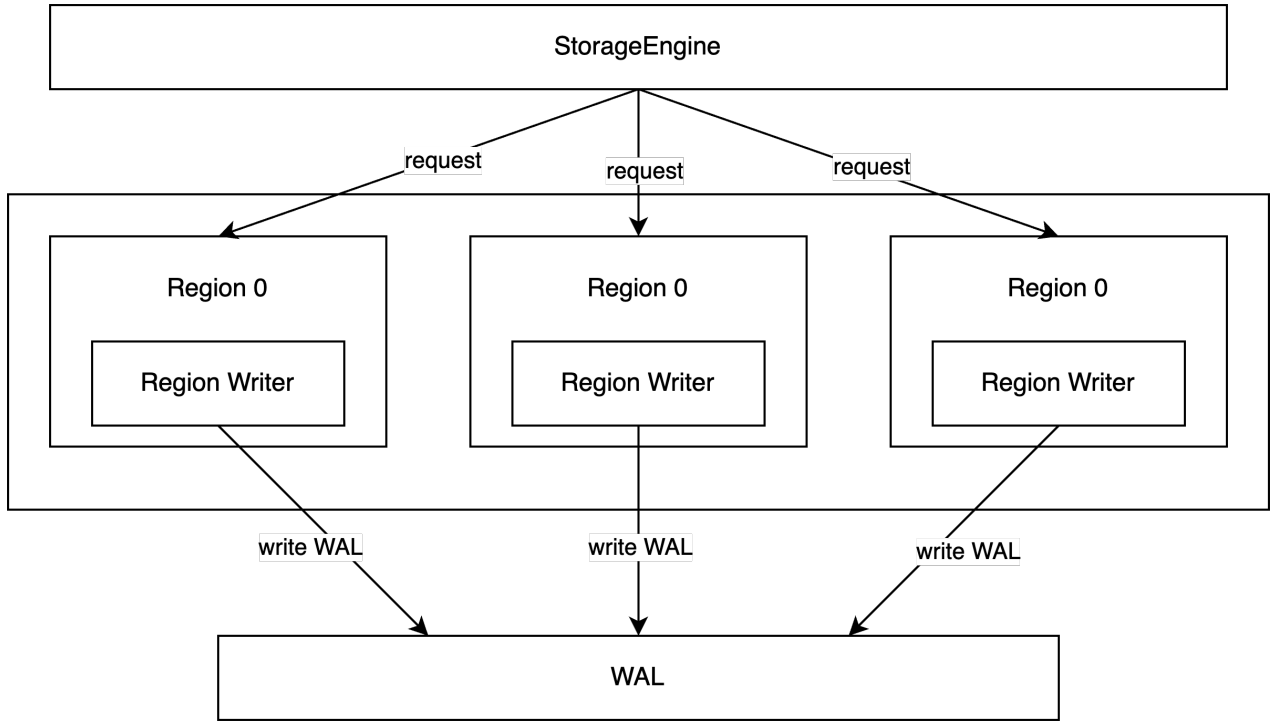


02

Mito2

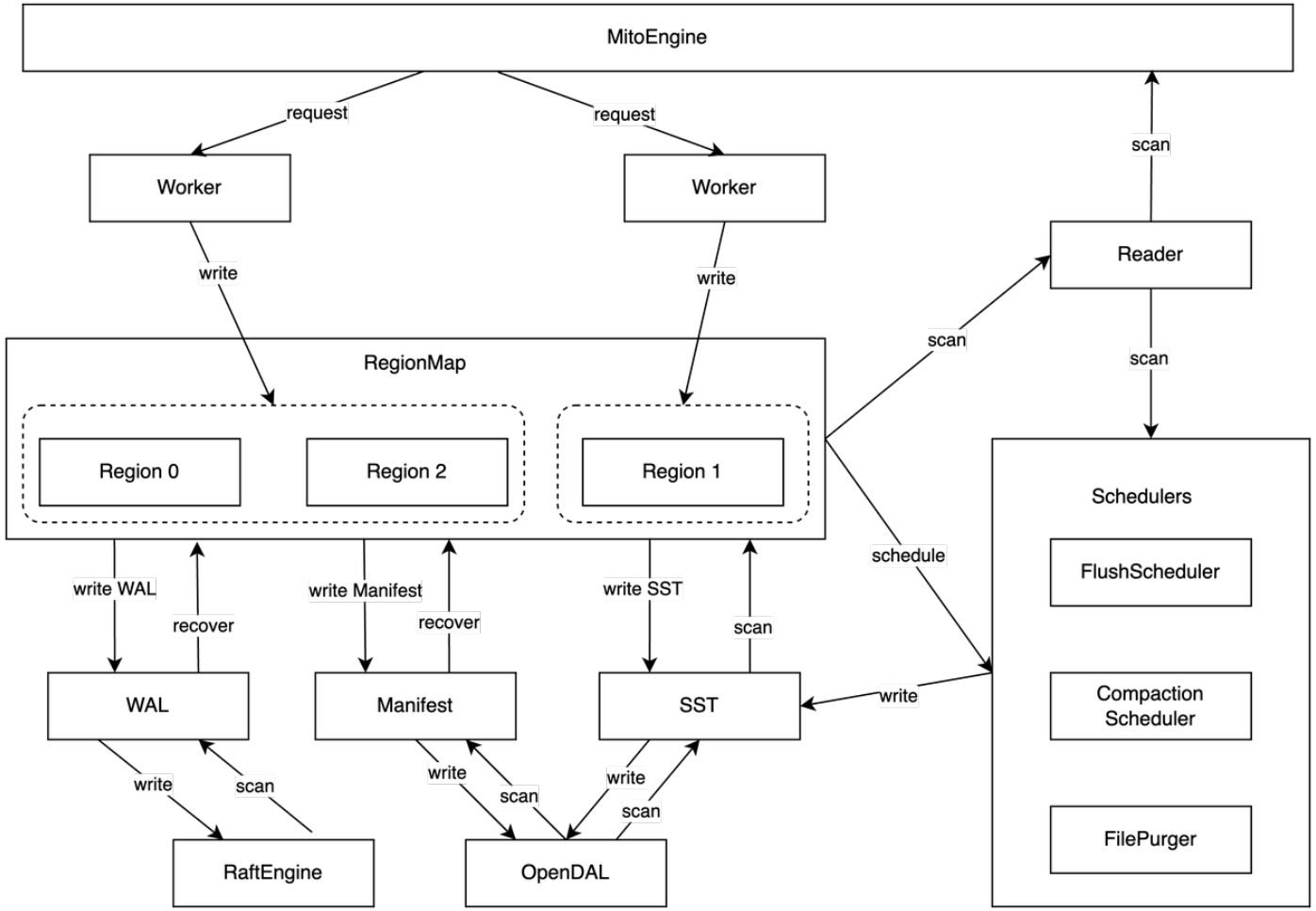
New Engine

Old Storage Engine



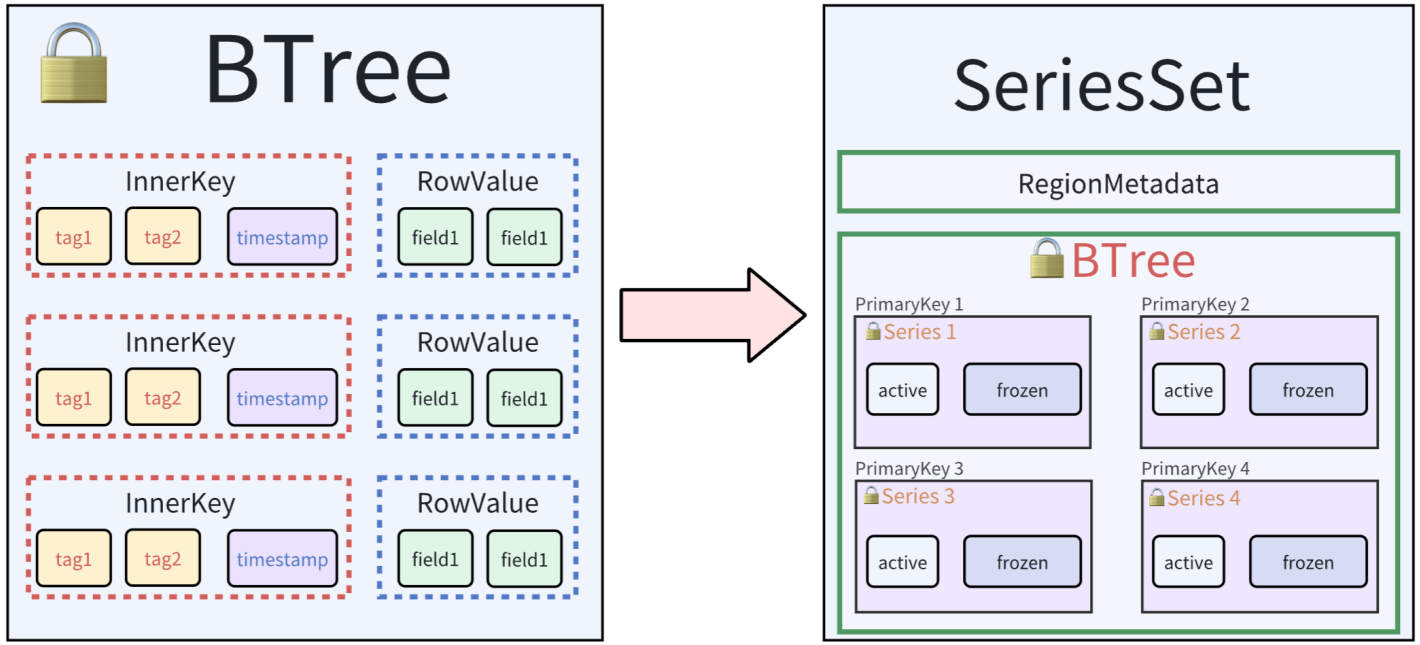
- Each region has a thread-safe region writer
- No batching
- Many locks on the write path and hard to acquire locks in a correct order
- Low memory efficiency of the old memtable

New Engine Architecture



- The engine allocates several workers (thread-per-core), and **sends requests directly to workers**
- Each region belongs to a worker and **is only written by that worker**
- Worker supports **batching**
- There is no restriction on **read threads**, so reader can be arbitrary threads

New Memtable



- **Concurrency** Optimization: Reduce Lock Granularity
- **Memory** Optimization: Group data by time series to reduce memory overhead
- **Storage** Optimization : Compact in-memory columnar format

03

Benchmark

Greptime

Environment

CPU	Ryzen 7 7735HS @ 3.200GHz
Memory	32G
OS	Ubuntu 22.04.3
Kernel	5.15.0-84-generic
Disk	SOLIDIGM SSDPFKNU010TZ 1TB
File System	ext4

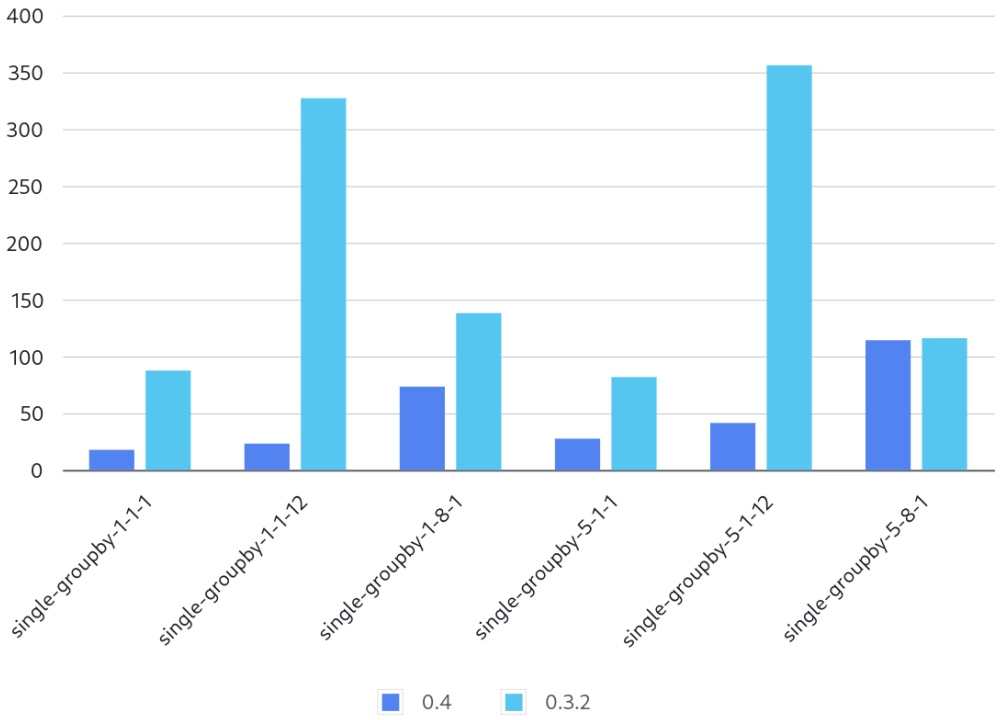
Data Configuration Generated via TSBS

Tags	String * 10
Fields	Int64 * 10
Timestamp	TimestampMillisecond
Series	4000
Time Span	3 days

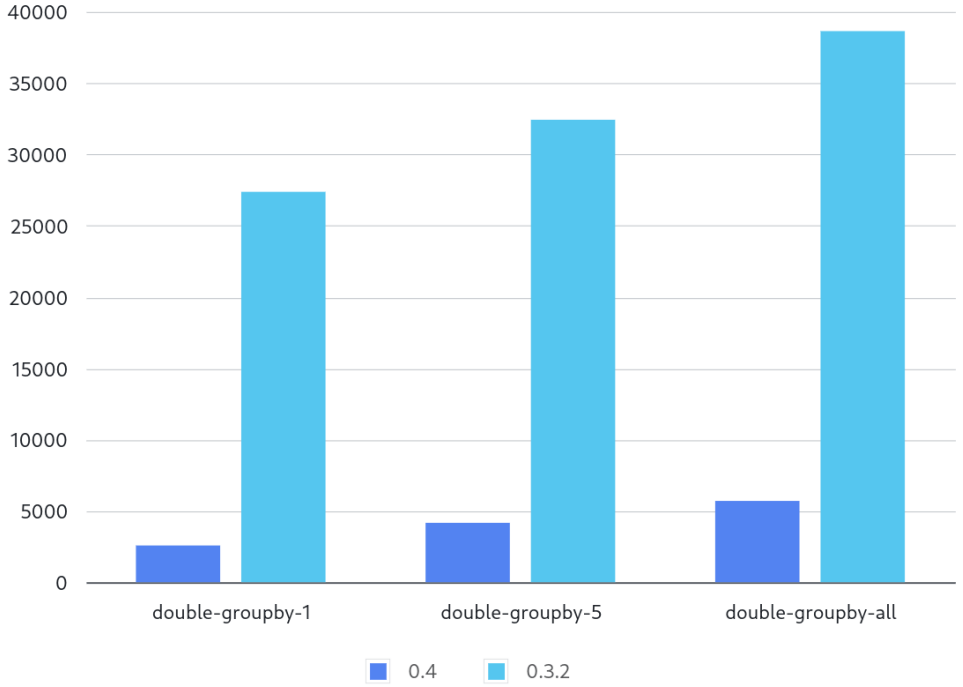
Benchmark

Greptime

Single Groupby (ms)



Double groupby(ms)

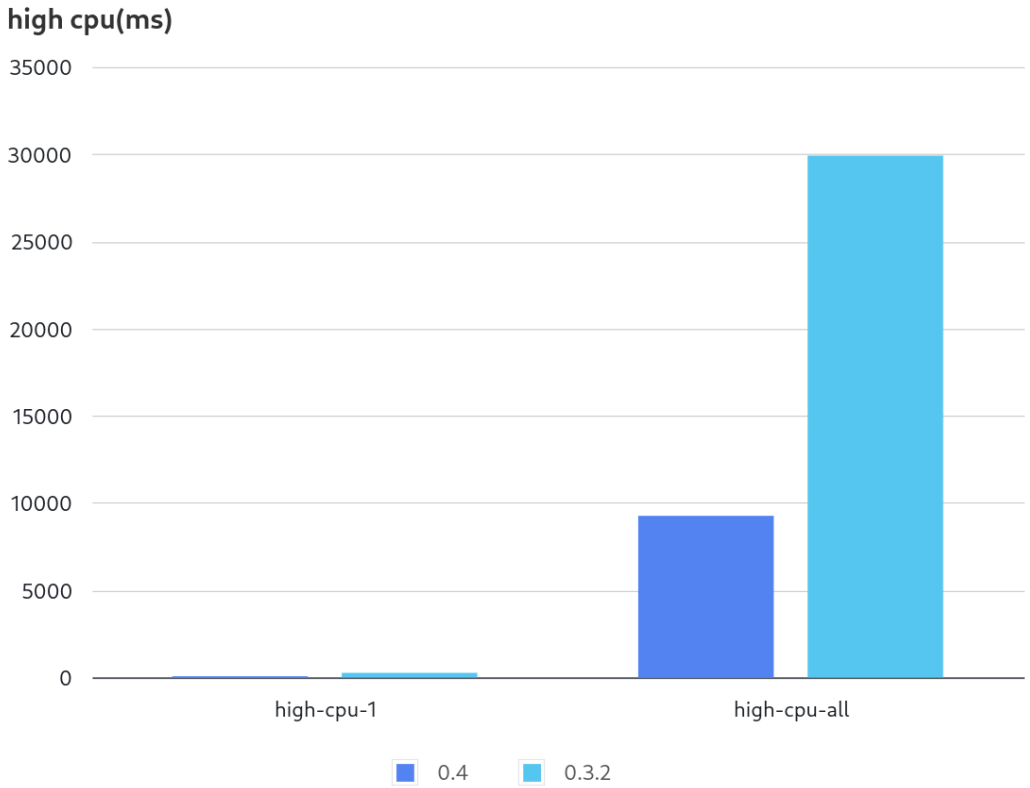


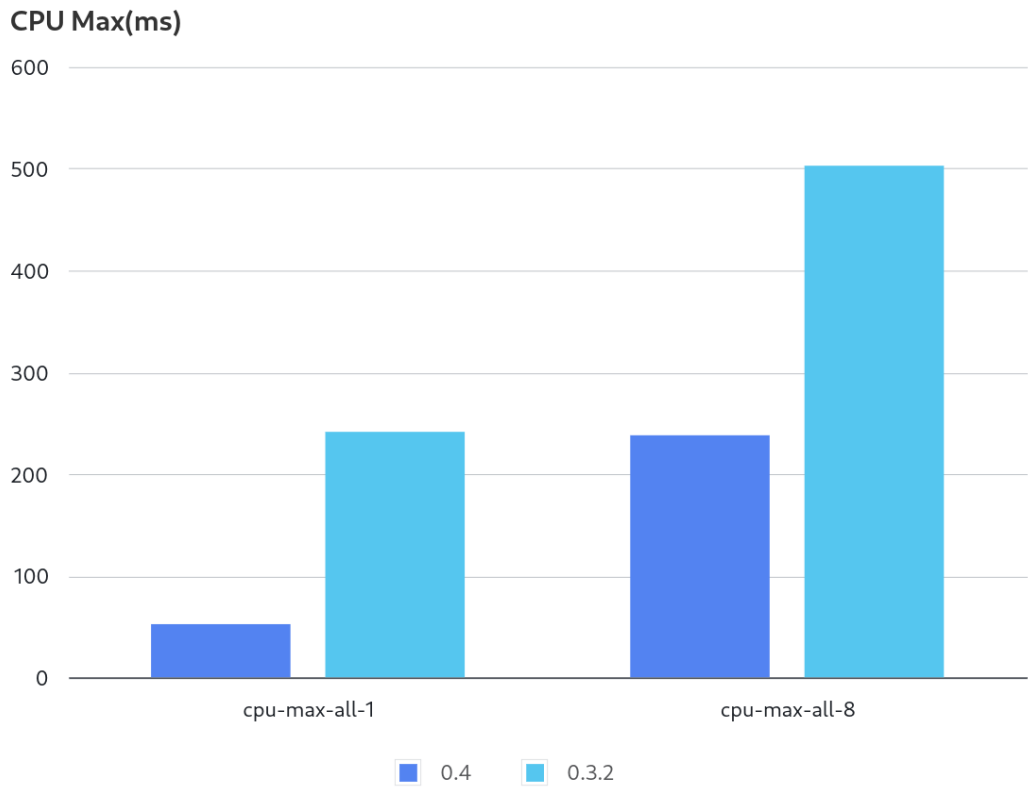

```
SELECT minute, max(metric1), ..., max(metricN)
FROM cpu
WHERE (hostname = '$HOSTNAME_1' OR ... OR hostname = '$HOSTNAME_N')
AND time >= '$HOUR_START' AND time < '$HOUR_END'
GROUP BY minute ORDER BY minute ASC
```

```
SELECT AVG(metric1), ..., AVG(metricN)
FROM cpu
WHERE time >= '$HOUR_START' AND time < '$HOUR_END'
GROUP BY hour, hostname ORDER BY hour, hostname
```

Benchmark

Greptime



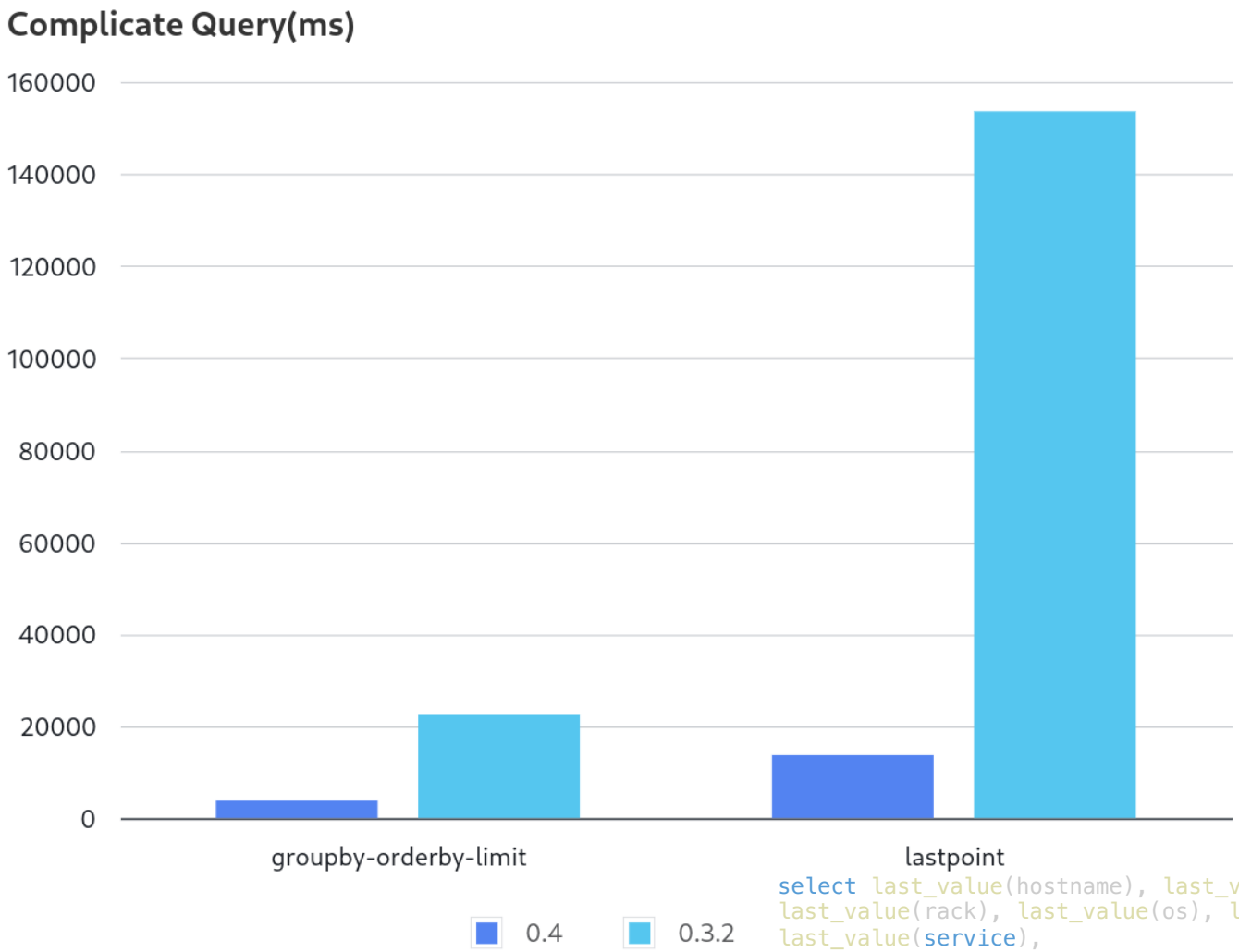



```
SELECT * FROM cpu
WHERE usage_user > 90.0
AND time >= '$TIME_START' AND time < '$TIME_END'
AND (hostname = '$HOST' OR hostname = '$HOST2'...)
```

```
SELECT MAX(metric1), ..., MAX(metricN)
FROM cpu WHERE (hostname = '$HOSTNAME_1' OR ... OR hostname = '$HOSTNAME_N')
AND time >= '$HOUR_START' AND time < '$HOUR_END'
GROUP BY hour ORDER BY hour
```

Benchmark

Greptime



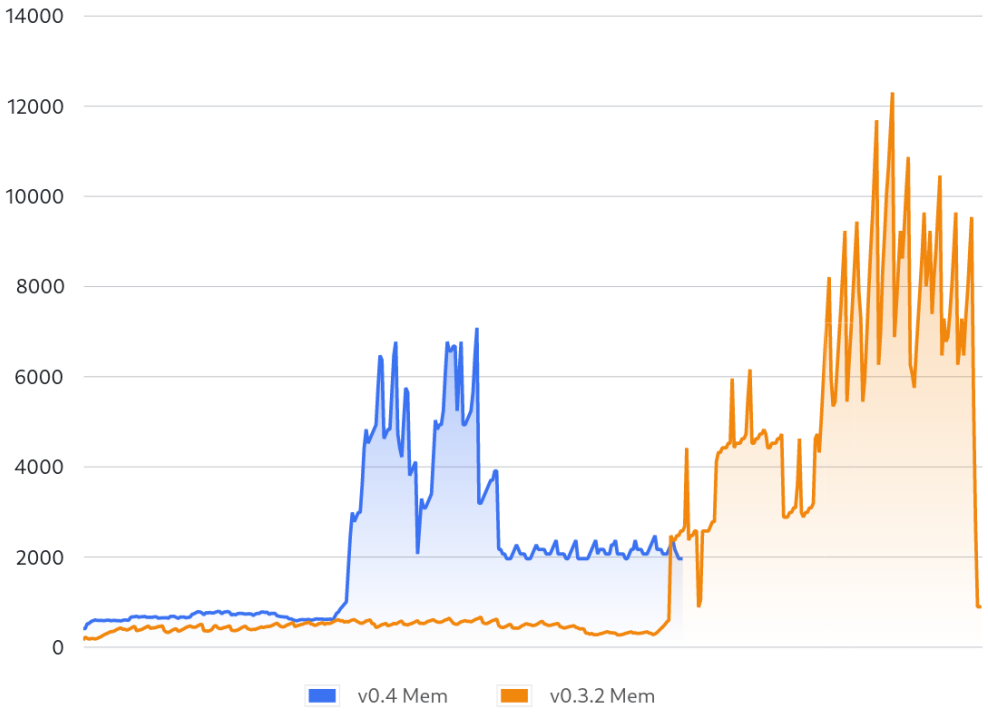

```
SELECT date_trunc('minute', time) AS t, MAX(cpu) FROM cpu
WHERE time < '$TIME'
GROUP BY t ORDER BY t DESC
LIMIT $LIMIT
```

```
select last_value(hostname), last_value(region), last_value(datacenter),
last_value(rack), last_value(os), last_value(arch), last_value(team),
last_value(service),
last_value(service_version), last_value(service_environment),
last_value(usage_user), last_value(usage_system), last_value(usage_idle),
last_value(usage_nice),
last_value(usage_iowait), last_value(usage_irq), last_value(usage_softirq),
last_value(usage_steal), last_value(usage_guest), last_value(usage_guest_nice )
from (select * from cpu order by ts) group by hostname;
```

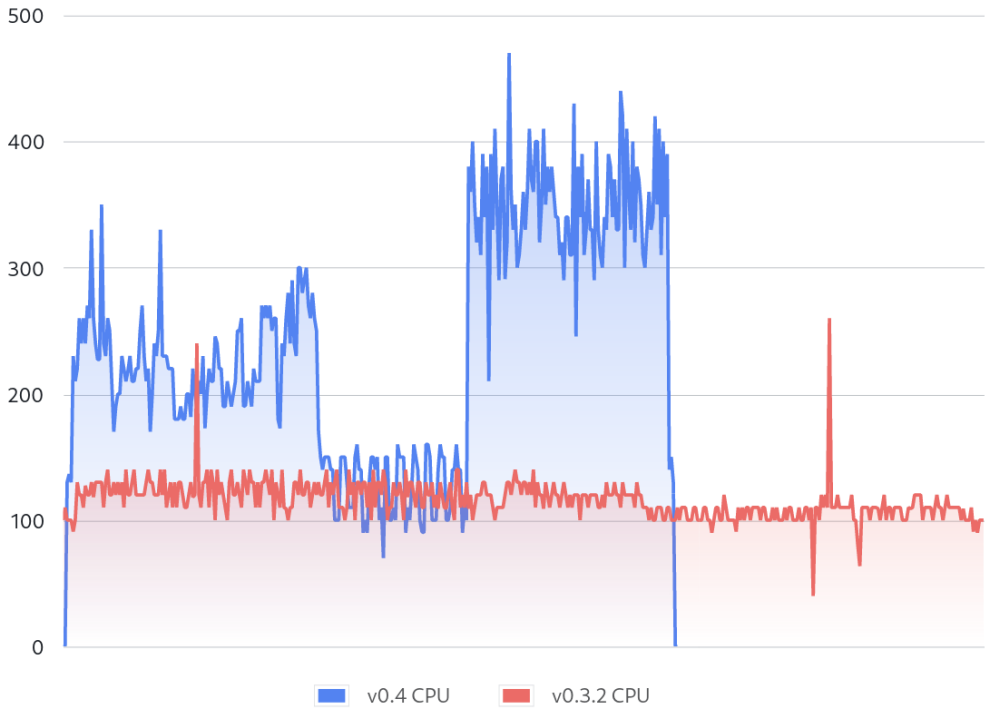
Resource Usage

Greptime

v0.4 Mem,v0.3.2 Mem

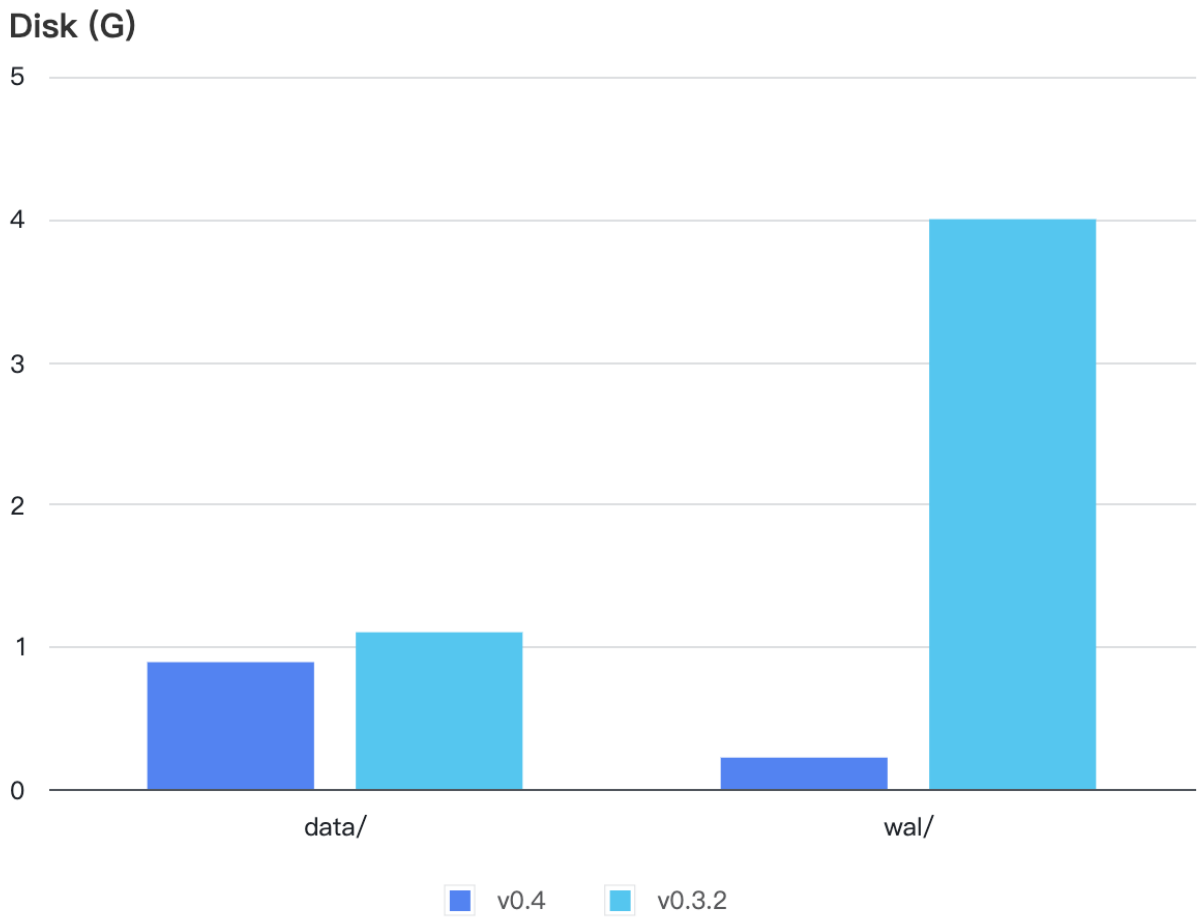


v0.4 CPU,v0.3.2 CPU



Benchmark

Greptime



Demo

Greptime

Full script is available at https://github.com/**GreptimeTeam/tsbs**/blob/master/**scripts/greptimedb.sh**

```
 # generate data
 ./bin/tsbs_generate_data \
 --use-case=cpu-only \
 --seed=123 \
 --scale=4000 \
 --timestamp-start="2023-06-11T00:00:00Z" \
 --timestamp-end="2023-06-14T00:00:00Z" \
 --log-interval=10s \
 --format=influx > ./$BENCH_DIR/bench-data.lp

 # load data
 ./bin/tsbs_load_greptime \
 --urls=http://localhost:4000 \
 --file=./$BENCH_DIR/influx-data.lp \
 --batch-size=3000 \
 --gzip=false \
 --workers=6
```

```
# generate query
./tsbs_generate_queries \
--use-case="devops" --seed=123 --scale=4000 \
--timestamp-start="2023-06-11T00:00:00Z" \
--timestamp-end="2023-06-14T00:00:01Z" \
--queries=10 \
--query-type ${QUERY_TYPE} \
--format="greptime" \
> ./queries/greptime-queries-${QUERY_TYPE}.dat
```

```
# run query
tsbs/bin/tsbs_run_queries_influx --file=./queries/greptime-queries-${QUERY_TYPE}.dat \
--db-name=benchmark \
--urls="http://localhost:4000"
```

04

Q & A

Join Us

Greptime

Slack

JiachunFeng

Thanks for joining us

Greptime