HOW TO:

BENCHMARK TIMESERIES DATABASES WITH TSBS

BY ELLIOT TAN & ZHENGHAO ZHAO

Experiments on Time Series Benchmarking Suite (TSBS) are challenging, mainly because the introductions from TSBS website are out-of-dated, which is our biggest motivation to draft this technical report, as a assistance for students who will work on TSBS in the future.

We used AWS EC2 instance consisting of:
CPU: Intel(R) Xeon(R) CPU E5-2676 v3 @ 2.40GHz
Memory: 1 GiB
Disk: 30 GiB SSD
OS: Ubuntu 20.04

The version of software used in experiment:
InfluxDB v1.8 (Important! TSBS hasn't supported InfluxDB by 04/27/2021)
PostgreSQL v13
TimescaleDB v2.2.0

1. Install Go

1) Download Go (v1.16.3 linux/amd64)

```
$ wget https://golang.org/dl/go1.16.3.linux-amd64.tar.gz
$ tar -C /usr/local -xzf go1.16.3.linux-amd64.tar.gz
```

2) Add Go to $PATH

```
$ cd $HOME/.profile or /etc/profile
$ export PATH=$PATH:/usr/local/go/bin
```

3) Check if it is successfully installed

```
$ go version
```

2. Install TSBS

1) Fetch TSBS and its dependencies

```
$ go get github.com/timescale/tsbs
$ cd $GOPATH/pkg/mod/github.com/timescale/tsbs/cmd
$ go get ./...
```

2) Install desired binaries

```
$ cd $GOPATH/src/github.com/timescale/tsbs/cmd
$ cd tsbs_generate_data && go install
$ cd ../tsbs_generate_queries && go install
$ cd ../tsbs_load_timescaledb && go install
$ cd ../tsbs_run_queries_timescaledb && go install
$ cd ../tsbs_load_Influx && go install
$ cd ../tsbs_run_queries_Influx && go install
```

3) Optionally, install all binaries

```
$ cd $GOPATH/src/github.com/timescale/tsbs/cmd
$ go install ./…
```

4) Add TSBS to $PATH (Important!)

```
$ cd $HOME/.profile or $ cd /etc/profile
$ export PATH=$PATH:/home/ubuntu/go/bin
```

3. Using TSBS

Data Generation

```
$ tsbs_generate_data --use-case="devops" --seed=108 --scale=50 \
    --timestamp-start="2018-01-01T00:00:00Z" \
    --timestamp-end="2018-01-04T00:00:00Z" \
    --log-interval="10s" --format="timescaledb" \
    | gzip > …/tsbs_data/timescaledb-data.gz
```

Query Generation

```
$ tsbs_generate_queries --use-case="iot" --seed=123 --scale=100 \
    --timestamp-start="2018-01-01T00:00:00Z" \
    --timestamp-end="2018-01-04T00:00:01Z" \
    --queries=1000 \
    --query-type=“single-groupby-1-1-1" --format="timescaledb" \
    | gzip > …/bulk_queries/timescaledb-last-loc-queries.gz
```

Query Generation (using scripts)

1) Go to TSBS script folder

```
$ cd /home/ubuntu/go/pkg/mod/github.com/timescale/tsbs@v…/scripts
```

2) Change the authority of scripts

```
$ sudo chmod a+x generate_queries.sh
```

3) Run scripts

```
$ FORMATS=“timescaledb” SCALE=50 SEED=108 \
  TS_START=“2018-01-01T00:00:00Z” \
  TS_END="2018-01-04T00:00:00Z" \
  QUERIES=15000 \
  QUERY_TYPES=“single-groupby-1-1-1 single-groupby-1-1-12” \
  BULK_DATA_DIR=“…/bulk_queries” ./generate_queries.sh
```

Benchmarking insert/write performance (using scripts)

1) Set a password for user postgres

```
$ sudo -u postgres psql
$ ALTER USER postgres PASSWORD “password”
```

2) Go to TSBS script folder

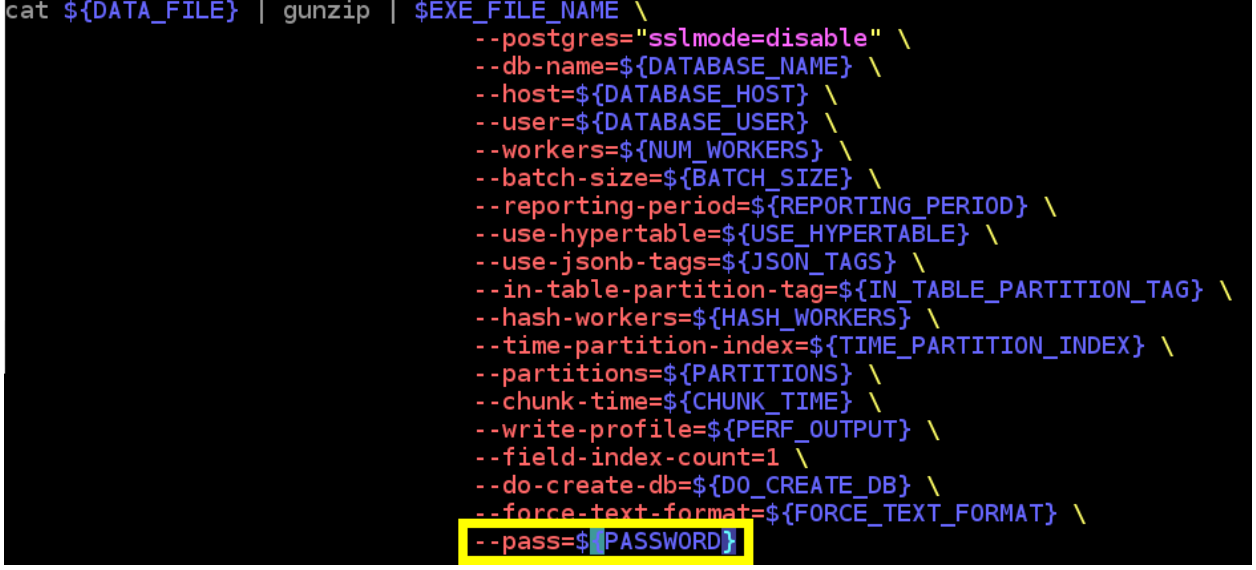
```
$ cd /home/ubuntu/go/pkg/mod/github.com/timescale/tsbs@v…/scripts/load
```

3) Change the authority of scripts

```
$ sudo chmod a+rwx load_timescaledb.sh
```

4) Open load_timescaledb.sh and add PASSWORD argument as the figure shown below

```
$ vi load_timescaledb.sh
```



```
cat ${DATA_FILE} | gunzip | $EXE_FILE_NAME \
                                --postgres="sslmode=disable" \
                                --db-name=${DATABASE_NAME} \
                                --host=${DATABASE_HOST} \
                                --user=${DATABASE_USER} \
                                --workers=${NUM_WORKERS} \
                                --batch-size=${BATCH_SIZE} \
                                --reporting-period=${REPORTING_PERIOD} \
                                --use-hypertable=${USE_HYPERTABLE} \
                                --use-jsonb-tags=${JSON_TAGS} \
                                --in-table-partition-tag=${IN_TABLE_PARTITION_TAG} \
                                --hash-workers=${HASH_WORKERS} \
                                --time-partition-index=${TIME_PARTITION_INDEX} \
                                --partitions=${PARTITIONS} \
                                --chunk-time=${CHUNK_TIME} \
                                --write-profile=${PERF_OUTPUT} \
                                --field-index-count=1 \
                                --do-create-db=${DO_CREATE_DB} \
                                --force-text-format=${FORCE_TEXT_FORMAT} \
                                --pass=${PASSWORD}
```

5) Run script

```
$ NUM_WORKERS=2 BATCH_SIZE=10000 PASSWORD=password \
  BULK_DATA_DIR="…/bulk_queries" ./load_timescaledb.sh
```

For InfluxDB you can skip step 4).

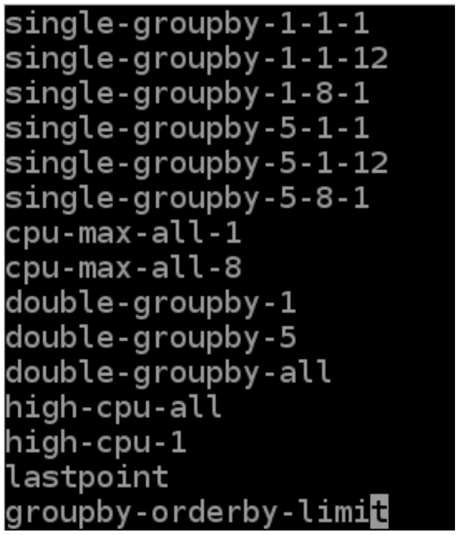
Benchmarking query execution performance (using scripts)

1) Go to TSBS script folder

```
$ cd /home/ubuntu/go/pkg/mod/github.com/timescale/tsbs@v…/scripts
```

2) Create a new text file and add the types of queries to be executed to it as the figure shown below

```
$ sudo vim queries.txt
```



```
single-groupby-1-1-1
single-groupby-1-1-12
single-groupby-1-8-1
single-groupby-5-1-1
single-groupby-5-1-12
single-groupby-5-8-1
cpu-max-all-1
cpu-max-all-8
double-groupby-1
double-groupby-5
double-groupby-all
high-cpu-all
high-cpu-1
lastpoint
groupby-orderby-limit
```

3) Change the authority of the text file

```
$ sudo chmod a+rwx queries.txt
```

4) Run script named generate_run_script.py to generate script for query execution (requires python installed)

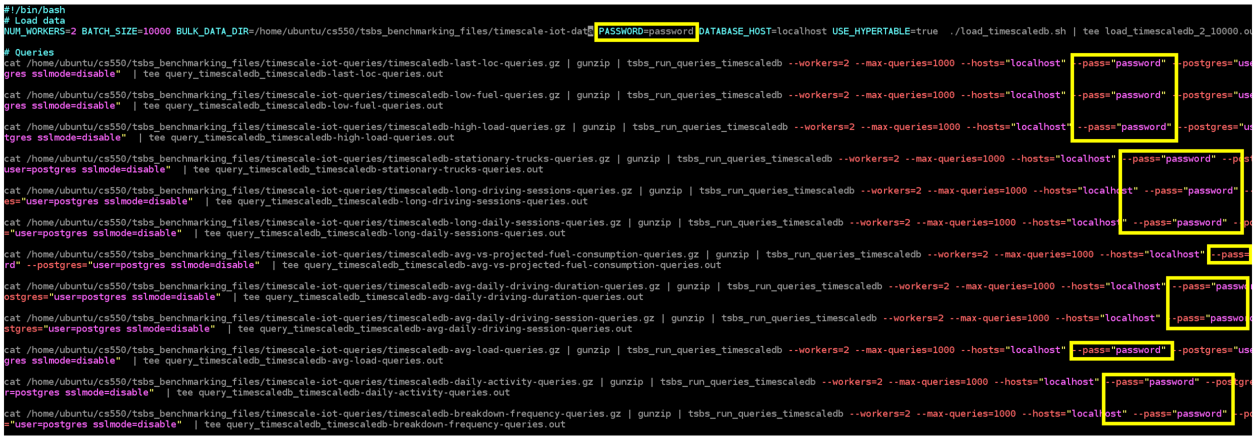
```
$ python3 -d timescaledb -f queries.txt -l …/tsbs_data \
  -o …/bulk_queries -w 2 > query_text.sh
```

5) Change the authority of the generated script

```
$ sudo chmod a+rwx query_text.sh
```

6) Edit the generated script, add password argument as the figure shown below

```
$ sudo vi query_text.sh
```



```
#!/bin/bash
# Load data
NUM_WORKERS=2 BATCH_SIZE=10000 BULK_DATA_DIR=/home/ubuntu/cs550/tsbs_benchmarking_files/timescale-iot-data PASSWORD=password DATABASE_HOST=localhost USE_HYPERTABLE=true  ./load_timescaledb.sh | tee load_timescaledb_2_10000.ou

# Queries
cat /home/ubuntu/cs550/tsbs_benchmarking_files/timescale-iot-queries/timescaledb-last-loc-queries.gz | gunzip | tsbs_run_queries_timescaledb --workers=2 --max-queries=1000 --hosts="localhost" --pass="password" --postgres="use
gres sslmode=disable"  | tee query_timescaledb_timescaledb-last-loc-queries.out

cat /home/ubuntu/cs550/tsbs_benchmarking_files/timescale-iot-queries/timescaledb-low-fuel-queries.gz | gunzip | tsbs_run_queries_timescaledb --workers=2 --max-queries=1000 --hosts="localhost" --pass="password" --postgres="use
gres sslmode=disable"  | tee query_timescaledb_timescaledb-low-fuel-queries.out

cat /home/ubuntu/cs550/tsbs_benchmarking_files/timescale-iot-queries/timescaledb-high-load-queries.gz | gunzip | tsbs_run_queries_timescaledb --workers=2 --max-queries=1000 --hosts="localhost" --pass="password" --postgres="us
tgres sslmode=disable"  | tee query_timescaledb_timescaledb-high-load-queries.out

cat /home/ubuntu/cs550/tsbs_benchmarking_files/timescale-iot-queries/timescaledb-stationary-trucks-queries.gz | gunzip | tsbs_run_queries_timescaledb --workers=2 --max-queries=1000 --hosts="localhost" --pass="password" --post
user=postgres sslmode=disable"  | tee query_timescaledb_timescaledb-stationary-trucks-queries.out

cat /home/ubuntu/cs550/tsbs_benchmarking_files/timescale-iot-queries/timescaledb-long-driving-sessions-queries.gz | gunzip | tsbs_run_queries_timescaledb --workers=2 --max-queries=1000 --hosts="localhost" --pass="password" --
es="user=postgres sslmode=disable"  | tee query_timescaledb_timescaledb-long-driving-sessions-queries.out

cat /home/ubuntu/cs550/tsbs_benchmarking_files/timescale-iot-queries/timescaledb-long-daily-sessions-queries.gz | gunzip | tsbs_run_queries_timescaledb --workers=2 --max-queries=1000 --hosts="localhost" --pass="password" --po
="user=postgres sslmode=disable"  | tee query_timescaledb_timescaledb-long-daily-sessions-queries.out

cat /home/ubuntu/cs550/tsbs_benchmarking_files/timescale-iot-queries/timescaledb-avg-vs-projected-fuel-consumption-queries.gz | gunzip | tsbs_run_queries_timescaledb --workers=2 --max-queries=1000 --hosts="localhost" --pass=
rd" --postgres="user=postgres sslmode=disable"  | tee query_timescaledb_timescaledb-avg-vs-projected-fuel-consumption-queries.out

cat /home/ubuntu/cs550/tsbs_benchmarking_files/timescale-iot-queries/timescaledb-avg-daily-driving-duration-queries.gz | gunzip | tsbs_run_queries_timescaledb --workers=2 --max-queries=1000 --hosts="localhost" --pass="passwor
ostgres="user=postgres sslmode=disable"  | tee query_timescaledb_timescaledb-avg-daily-driving-duration-queries.out

cat /home/ubuntu/cs550/tsbs_benchmarking_files/timescale-iot-queries/timescaledb-avg-daily-driving-session-queries.gz | gunzip | tsbs_run_queries_timescaledb --workers=2 --max-queries=1000 --hosts="localhost" --pass="passwor
stgres="user=postgres sslmode=disable"  | tee query_timescaledb_timescaledb-avg-daily-driving-session-queries.out

cat /home/ubuntu/cs550/tsbs_benchmarking_files/timescale-iot-queries/timescaledb-avg-load-queries.gz | gunzip | tsbs_run_queries_timescaledb --workers=2 --max-queries=1000 --hosts="localhost" --pass="password" --postgres="use
gres sslmode=disable"  | tee query_timescaledb_timescaledb-avg-load-queries.out

cat /home/ubuntu/cs550/tsbs_benchmarking_files/timescale-iot-queries/timescaledb-daily-activity-queries.gz | gunzip | tsbs_run_queries_timescaledb --workers=2 --max-queries=1000 --hosts="localhost" --pass="password" --postgre
r=postgres sslmode=disable"  | tee query_timescaledb_timescaledb-daily-activity-queries.out

cat /home/ubuntu/cs550/tsbs_benchmarking_files/timescale-iot-queries/timescaledb-breakdown-frequency-queries.gz | gunzip | tsbs_run_queries_timescaledb --workers=2 --max-queries=1000 --hosts="localhost" --pass="password" --po
="user=postgres sslmode=disable"  | tee query_timescaledb_timescaledb-breakdown-frequency-queries.out
```

7) Run generated script

```
$ ./query_text.sh
```

For InfluxDB you can skip step 6).