Metrics and Queries

Write Performance: Data Loading Time
Read Performance: Query Performance (Multiple queries)
Data Storage FootPrint: Size of the data file

Queries

double-groupby-5 : This query does multiple group-by by time and host_id .Returns the average of 5 metrics per host per day

cpu-max-all-8 : This query finds the maximum value for all metrics for 1 hour for 8 hosts

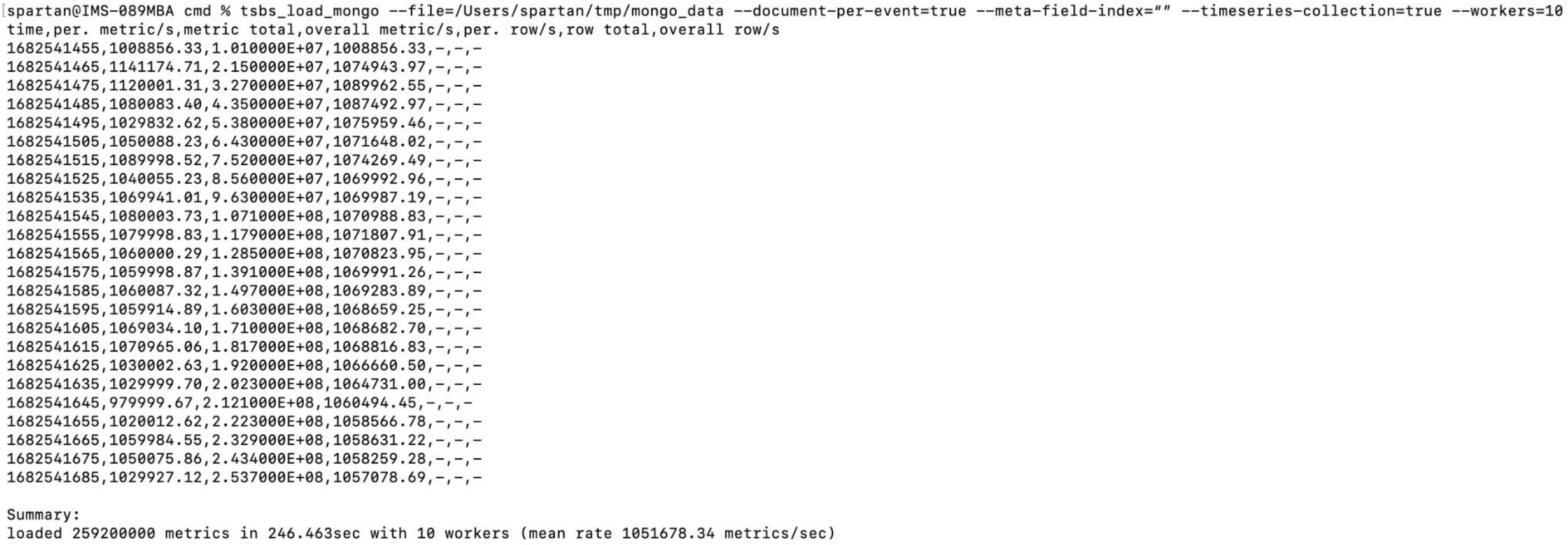
lastpoint : This query finds the latest reading for every device in the dataset.

groupby-orderby-limit : This query does a single rollup on time to get the MAX reading of a CPU metric on a per-minute basis for the last 5 intervals for which there are readings before a specified end time that is randomly selected.

Performing MongoDB Queries

MongoDB Naive

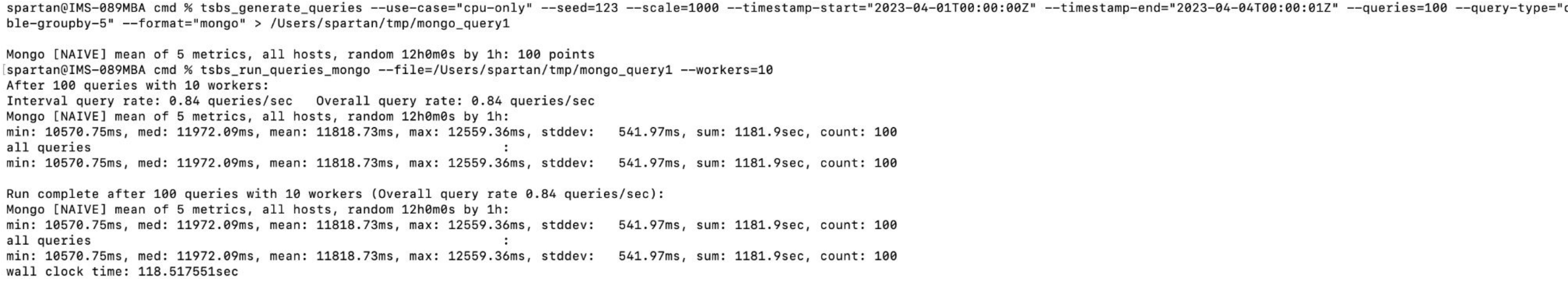
MongoDB: Data Generation and Data Loading



```
[spartan@IMS-089MBA cmd % tsbs_load_mongo --file=/Users/spartan/tmp/mongo_data --document-per-event=true --meta-field-index="" --timeseries-collection=true --workers=10
time,per. metric/s,metric total,overall metric/s,per. row/s,row total,overall row/s
1682541455,1008856.33,1.010000E+07,1008856.33,-,-,-
1682541465,1141174.71,2.150000E+07,1074943.97,-,-,-
1682541475,1120001.31,3.270000E+07,1089962.55,-,-,-
1682541485,1080083.40,4.350000E+07,1087492.97,-,-,-
1682541495,1029832.62,5.380000E+07,1075959.46,-,-,-
1682541505,1050088.23,6.430000E+07,1071648.02,-,-,-
1682541515,1089998.52,7.520000E+07,1074269.49,-,-,-
1682541525,1040055.23,8.560000E+07,1069992.96,-,-,-
1682541535,1069941.01,9.630000E+07,1069987.19,-,-,-
1682541545,1080003.73,1.071000E+08,1070988.83,-,-,-
1682541555,1079998.83,1.179000E+08,1071807.91,-,-,-
1682541565,1060000.29,1.285000E+08,1070823.95,-,-,-
1682541575,1059998.87,1.391000E+08,1069991.26,-,-,-
1682541585,1060087.32,1.497000E+08,1069283.89,-,-,-
1682541595,1059914.89,1.603000E+08,1068659.25,-,-,-
1682541605,1069034.10,1.710000E+08,1068682.70,-,-,-
1682541615,1070965.06,1.817000E+08,1068816.83,-,-,-
1682541625,1030002.63,1.920000E+08,1066660.50,-,-,-
1682541635,1029999.70,2.023000E+08,1064731.00,-,-,-
1682541645,979999.67,2.121000E+08,1060494.45,-,-,-
1682541655,1020012.62,2.223000E+08,1058566.78,-,-,-
1682541665,1059984.55,2.329000E+08,1058631.22,-,-,-
1682541675,1050075.86,2.434000E+08,1058259.28,-,-,-
1682541685,1029927.12,2.537000E+08,1057078.69,-,-,-

Summary:
loaded 259200000 metrics in 246.463sec with 10 workers (mean rate 1051678.34 metrics/sec)
```

Data Loading : 246 seconds

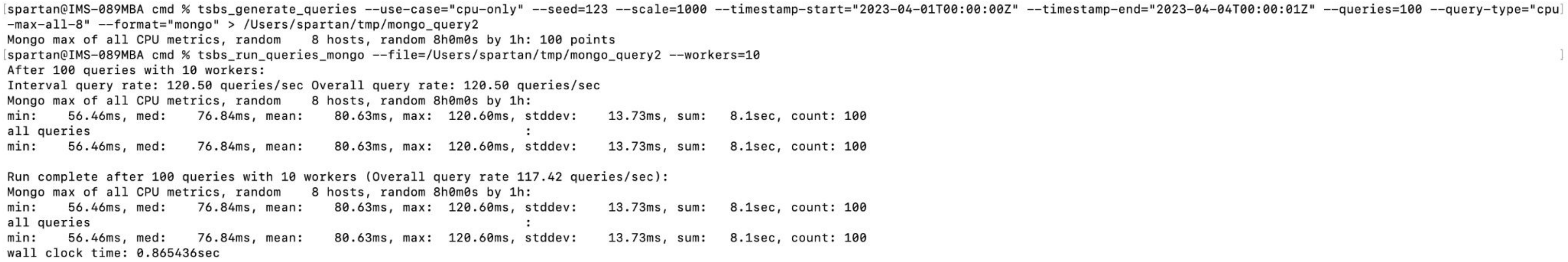
MongoDB:Query–double-groupby-5



```
spartan@IMS-089MBA cmd % tsbs_generate_queries --use-case="cpu-only" --seed=123 --scale=1000 --timestamp-start="2023-04-01T00:00:00Z" --timestamp-end="2023-04-04T00:00:01Z" --queries=100 --query-type="d
ble-groupby-5" --format="mongo" > /Users/spartan/tmp/mongo_query1

Mongo [NAIVE] mean of 5 metrics, all hosts, random 12h0m0s by 1h: 100 points
spartan@IMS-089MBA cmd % tsbs_run_queries_mongo --file=/Users/spartan/tmp/mongo_query1 --workers=10
After 100 queries with 10 workers:
Interval query rate: 0.84 queries/sec   Overall query rate: 0.84 queries/sec
Mongo [NAIVE] mean of 5 metrics, all hosts, random 12h0m0s by 1h:
min: 10570.75ms, med: 11972.09ms, mean: 11818.73ms, max: 12559.36ms, stddev:   541.97ms, sum: 1181.9sec, count: 100
all queries                                                      :
min: 10570.75ms, med: 11972.09ms, mean: 11818.73ms, max: 12559.36ms, stddev:   541.97ms, sum: 1181.9sec, count: 100

Run complete after 100 queries with 10 workers (Overall query rate 0.84 queries/sec):
Mongo [NAIVE] mean of 5 metrics, all hosts, random 12h0m0s by 1h:
min: 10570.75ms, med: 11972.09ms, mean: 11818.73ms, max: 12559.36ms, stddev:   541.97ms, sum: 1181.9sec, count: 100
all queries                                                      :
min: 10570.75ms, med: 11972.09ms, mean: 11818.73ms, max: 12559.36ms, stddev:   541.97ms, sum: 1181.9sec, count: 100
wall clock time: 118.517551sec
```

Total Time: 118 sec

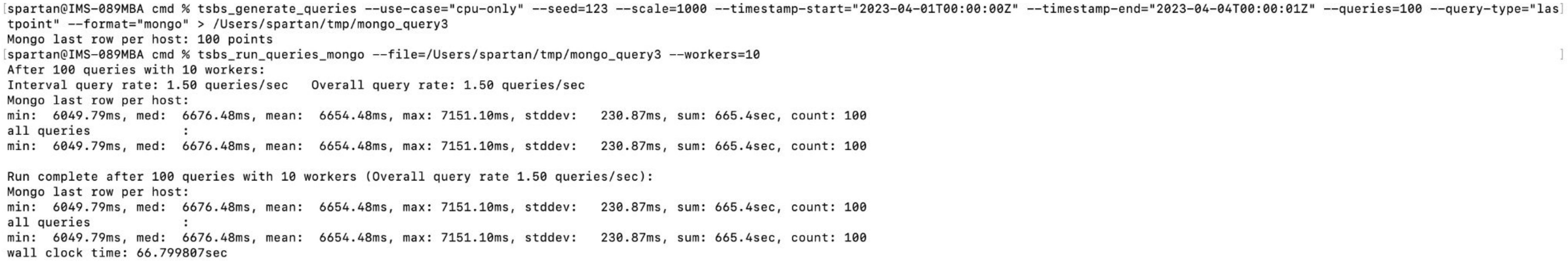
MongoDB: Query–cpu-max-all-8



```
[spartan@IMS-089MBA cmd % tsbs_generate_queries --use-case="cpu-only" --seed=123 --scale=1000 --timestamp-start="2023-04-01T00:00:00Z" --timestamp-end="2023-04-04T00:00:01Z" --queries=100 --query-type="cpu]
-max-all-8" --format="mongo" > /Users/spartan/tmp/mongo_query2
Mongo max of all CPU metrics, random    8 hosts, random 8h0m0s by 1h: 100 points
[spartan@IMS-089MBA cmd % tsbs_run_queries_mongo --file=/Users/spartan/tmp/mongo_query2 --workers=10                                                                                                         ]
After 100 queries with 10 workers:
Interval query rate: 120.50 queries/sec Overall query rate: 120.50 queries/sec
Mongo max of all CPU metrics, random    8 hosts, random 8h0m0s by 1h:
min:    56.46ms, med:    76.84ms, mean:    80.63ms, max:  120.60ms, stddev:    13.73ms, sum:   8.1sec, count: 100
all queries                                                     :
min:    56.46ms, med:    76.84ms, mean:    80.63ms, max:  120.60ms, stddev:    13.73ms, sum:   8.1sec, count: 100

Run complete after 100 queries with 10 workers (Overall query rate 117.42 queries/sec):
Mongo max of all CPU metrics, random    8 hosts, random 8h0m0s by 1h:
min:    56.46ms, med:    76.84ms, mean:    80.63ms, max:  120.60ms, stddev:    13.73ms, sum:   8.1sec, count: 100
all queries                                                     :
min:    56.46ms, med:    76.84ms, mean:    80.63ms, max:  120.60ms, stddev:    13.73ms, sum:   8.1sec, count: 100
wall clock time: 0.865436sec
```

Total Time: 0.86 sec

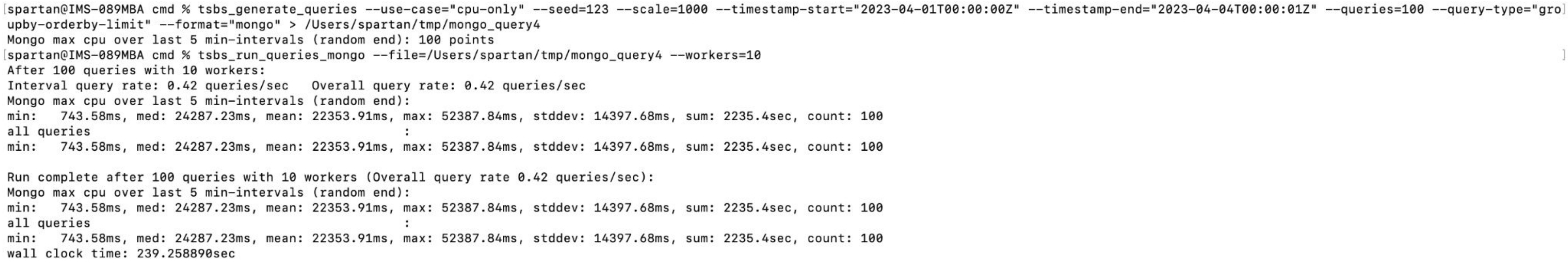
MongoDB: Query–lastpoint



```
[spartan@IMS-089MBA cmd % tsbs_generate_queries --use-case="cpu-only" --seed=123 --scale=1000 --timestamp-start="2023-04-01T00:00:00Z" --timestamp-end="2023-04-04T00:00:01Z" --queries=100 --query-type="las
tpoint" --format="mongo" > /Users/spartan/tmp/mongo_query3
Mongo last row per host: 100 points
[spartan@IMS-089MBA cmd % tsbs_run_queries_mongo --file=/Users/spartan/tmp/mongo_query3 --workers=10
After 100 queries with 10 workers:
Interval query rate: 1.50 queries/sec   Overall query rate: 1.50 queries/sec
Mongo last row per host:
min:  6049.79ms, med:  6676.48ms, mean:  6654.48ms, max: 7151.10ms, stddev:   230.87ms, sum: 665.4sec, count: 100
all queries            :
min:  6049.79ms, med:  6676.48ms, mean:  6654.48ms, max: 7151.10ms, stddev:   230.87ms, sum: 665.4sec, count: 100

Run complete after 100 queries with 10 workers (Overall query rate 1.50 queries/sec):
Mongo last row per host:
min:  6049.79ms, med:  6676.48ms, mean:  6654.48ms, max: 7151.10ms, stddev:   230.87ms, sum: 665.4sec, count: 100
all queries            :
min:  6049.79ms, med:  6676.48ms, mean:  6654.48ms, max: 7151.10ms, stddev:   230.87ms, sum: 665.4sec, count: 100
wall clock time: 66.799807sec
```

Total Time: 66.79 sec

MongoDB: Query–groupby-orderby-limit

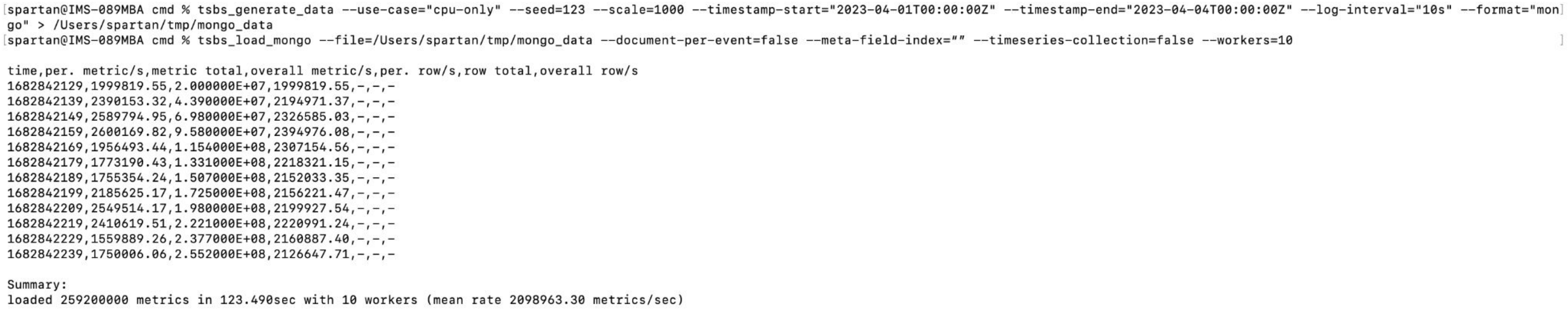


```
[spartan@IMS-089MBA cmd % tsbs_generate_queries --use-case="cpu-only" --seed=123 --scale=1000 --timestamp-start="2023-04-01T00:00:00Z" --timestamp-end="2023-04-04T00:00:01Z" --queries=100 --query-type="gro]
upby-orderby-limit" --format="mongo" > /Users/spartan/tmp/mongo_query4
Mongo max cpu over last 5 min-intervals (random end): 100 points
[spartan@IMS-089MBA cmd % tsbs_run_queries_mongo --file=/Users/spartan/tmp/mongo_query4 --workers=10                                                                                              ]
After 100 queries with 10 workers:
Interval query rate: 0.42 queries/sec   Overall query rate: 0.42 queries/sec
Mongo max cpu over last 5 min-intervals (random end):
min:   743.58ms, med: 24287.23ms, mean: 22353.91ms, max: 52387.84ms, stddev: 14397.68ms, sum: 2235.4sec, count: 100
all queries                                        :
min:   743.58ms, med: 24287.23ms, mean: 22353.91ms, max: 52387.84ms, stddev: 14397.68ms, sum: 2235.4sec, count: 100

Run complete after 100 queries with 10 workers (Overall query rate 0.42 queries/sec):
Mongo max cpu over last 5 min-intervals (random end):
min:   743.58ms, med: 24287.23ms, mean: 22353.91ms, max: 52387.84ms, stddev: 14397.68ms, sum: 2235.4sec, count: 100
all queries                                        :
min:   743.58ms, med: 24287.23ms, mean: 22353.91ms, max: 52387.84ms, stddev: 14397.68ms, sum: 2235.4sec, count: 100
wall clock time: 239.258890sec
```

Total Time: 239 sec

MongoDB Recommended

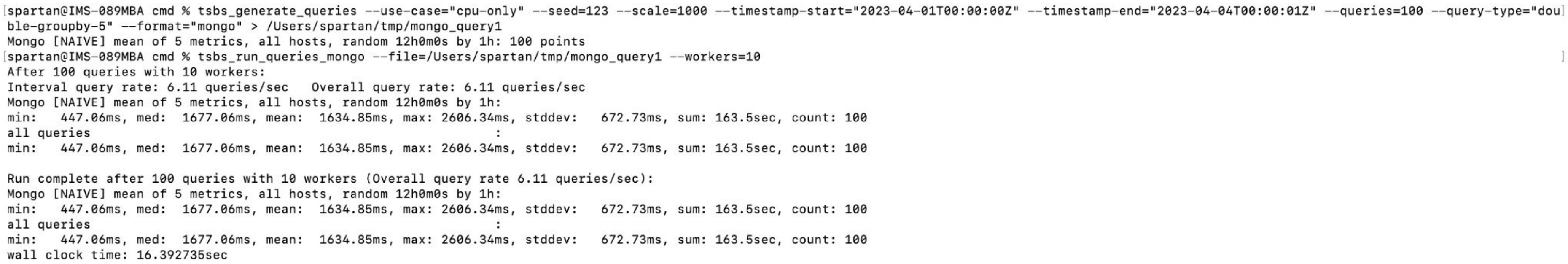
MongoDB: Data Generation and Data Loading



```
[spartan@IMS-089MBA cmd % tsbs_generate_data --use-case="cpu-only" --seed=123 --scale=1000 --timestamp-start="2023-04-01T00:00:00Z" --timestamp-end="2023-04-04T00:00:00Z" --log-interval="10s" --format="mon]
go" > /Users/spartan/tmp/mongo_data
[spartan@IMS-089MBA cmd % tsbs_load_mongo --file=/Users/spartan/tmp/mongo_data --document-per-event=false --meta-field-index=“” --timeseries-collection=false --workers=10                                    ]

time,per. metric/s,metric total,overall metric/s,per. row/s,row total,overall row/s
1682842129,1999819.55,2.000000E+07,1999819.55,-,-,-
1682842139,2390153.32,4.390000E+07,2194971.37,-,-,-
1682842149,2589794.95,6.980000E+07,2326585.03,-,-,-
1682842159,2600169.82,9.580000E+07,2394976.08,-,-,-
1682842169,1956493.44,1.154000E+08,2307154.56,-,-,-
1682842179,1773190.43,1.331000E+08,2218321.15,-,-,-
1682842189,1755354.24,1.507000E+08,2152033.35,-,-,-
1682842199,2185625.17,1.725000E+08,2156221.47,-,-,-
1682842209,2549514.17,1.980000E+08,2199927.54,-,-,-
1682842219,2410619.51,2.221000E+08,2220991.24,-,-,-
1682842229,1559889.26,2.377000E+08,2160887.40,-,-,-
1682842239,1750006.06,2.552000E+08,2126647.71,-,-,-

Summary:
loaded 259200000 metrics in 123.490sec with 10 workers (mean rate 2098963.30 metrics/sec)
```

Data Loading : 123 seconds

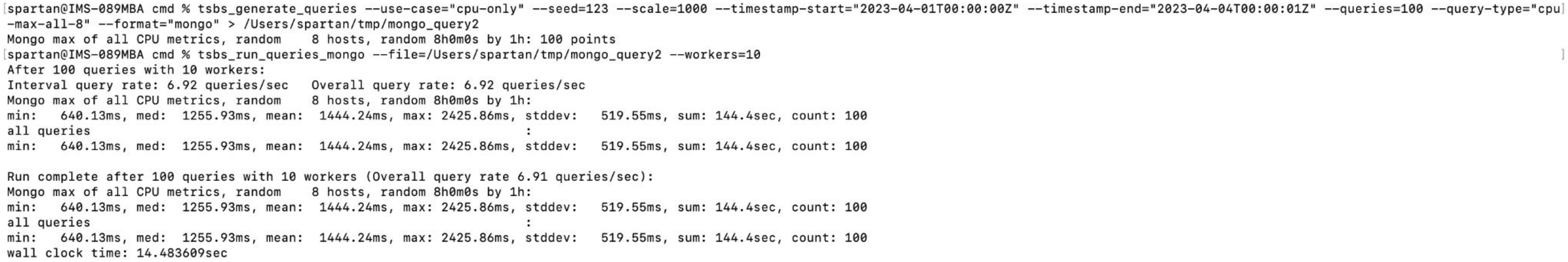
MongoDB:Query–double-groupby-5



```
[spartan@IMS-089MBA cmd % tsbs_generate_queries --use-case="cpu-only" --seed=123 --scale=1000 --timestamp-start="2023-04-01T00:00:00Z" --timestamp-end="2023-04-04T00:00:01Z" --queries=100 --query-type="dou]
ble-groupby-5" --format="mongo" > /Users/spartan/tmp/mongo_query1
Mongo [NAIVE] mean of 5 metrics, all hosts, random 12h0m0s by 1h: 100 points
[spartan@IMS-089MBA cmd % tsbs_run_queries_mongo --file=/Users/spartan/tmp/mongo_query1 --workers=10                                                                                           ]
After 100 queries with 10 workers:
Interval query rate: 6.11 queries/sec   Overall query rate: 6.11 queries/sec
Mongo [NAIVE] mean of 5 metrics, all hosts, random 12h0m0s by 1h:
min:   447.06ms, med:  1677.06ms, mean:  1634.85ms, max: 2606.34ms, stddev:   672.73ms, sum: 163.5sec, count: 100
all queries                                                     :
min:   447.06ms, med:  1677.06ms, mean:  1634.85ms, max: 2606.34ms, stddev:   672.73ms, sum: 163.5sec, count: 100

Run complete after 100 queries with 10 workers (Overall query rate 6.11 queries/sec):
Mongo [NAIVE] mean of 5 metrics, all hosts, random 12h0m0s by 1h:
min:   447.06ms, med:  1677.06ms, mean:  1634.85ms, max: 2606.34ms, stddev:   672.73ms, sum: 163.5sec, count: 100
all queries                                                     :
min:   447.06ms, med:  1677.06ms, mean:  1634.85ms, max: 2606.34ms, stddev:   672.73ms, sum: 163.5sec, count: 100
wall clock time: 16.392735sec
```

Total Time: 16.39 sec

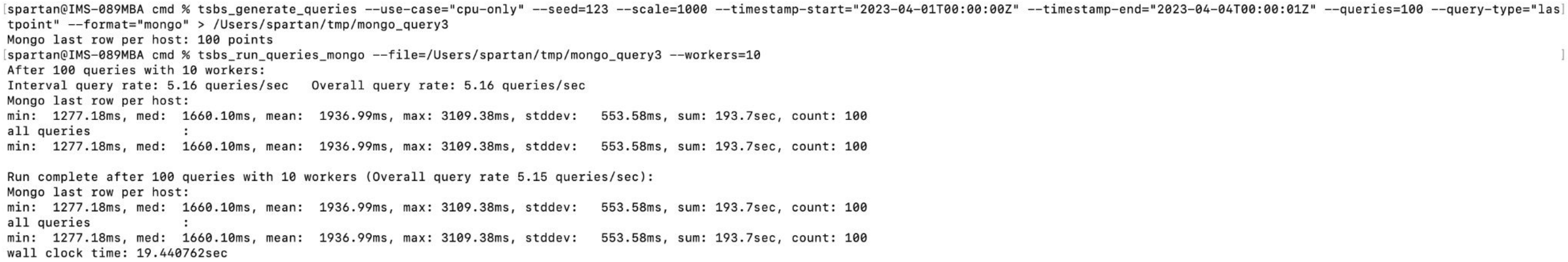
MongoDB: Query–cpu-max-all-8



```
[spartan@IMS-089MBA cmd % tsbs_generate_queries --use-case="cpu-only" --seed=123 --scale=1000 --timestamp-start="2023-04-01T00:00:00Z" --timestamp-end="2023-04-04T00:00:01Z" --queries=100 --query-type="cpu]
-max-all-8" --format="mongo" > /Users/spartan/tmp/mongo_query2
Mongo max of all CPU metrics, random    8 hosts, random 8h0m0s by 1h: 100 points
[spartan@IMS-089MBA cmd % tsbs_run_queries_mongo --file=/Users/spartan/tmp/mongo_query2 --workers=10                                                                                          ]
After 100 queries with 10 workers:
Interval query rate: 6.92 queries/sec   Overall query rate: 6.92 queries/sec
Mongo max of all CPU metrics, random    8 hosts, random 8h0m0s by 1h:
min:   640.13ms, med:  1255.93ms, mean:  1444.24ms, max: 2425.86ms, stddev:   519.55ms, sum: 144.4sec, count: 100
all queries                                                         :
min:   640.13ms, med:  1255.93ms, mean:  1444.24ms, max: 2425.86ms, stddev:   519.55ms, sum: 144.4sec, count: 100

Run complete after 100 queries with 10 workers (Overall query rate 6.91 queries/sec):
Mongo max of all CPU metrics, random    8 hosts, random 8h0m0s by 1h:
min:   640.13ms, med:  1255.93ms, mean:  1444.24ms, max: 2425.86ms, stddev:   519.55ms, sum: 144.4sec, count: 100
all queries                                                         :
min:   640.13ms, med:  1255.93ms, mean:  1444.24ms, max: 2425.86ms, stddev:   519.55ms, sum: 144.4sec, count: 100
wall clock time: 14.483609sec
```

Total Time: 14.48 sec

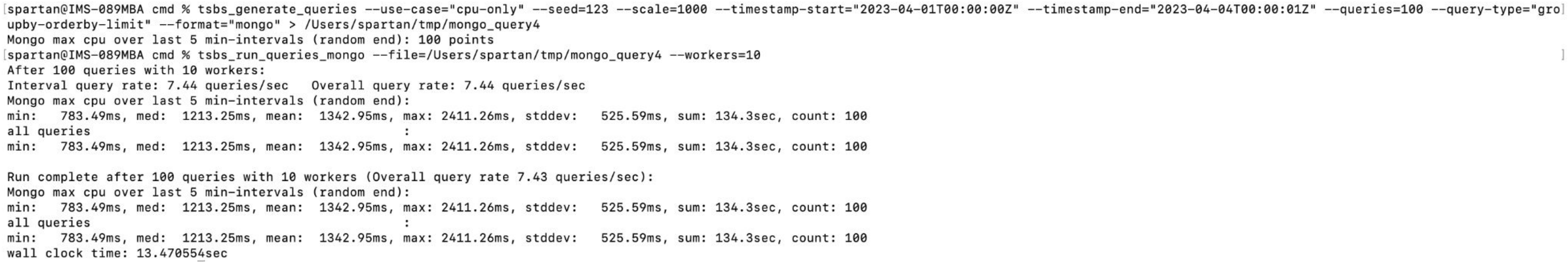
MongoDB: Query–lastpoint



```
[spartan@IMS-089MBA cmd % tsbs_generate_queries --use-case="cpu-only" --seed=123 --scale=1000 --timestamp-start="2023-04-01T00:00:00Z" --timestamp-end="2023-04-04T00:00:01Z" --queries=100 --query-type="las]
tpoint" --format="mongo" > /Users/spartan/tmp/mongo_query3
Mongo last row per host: 100 points
[spartan@IMS-089MBA cmd % tsbs_run_queries_mongo --file=/Users/spartan/tmp/mongo_query3 --workers=10                                                                                             ]
After 100 queries with 10 workers:
Interval query rate: 5.16 queries/sec   Overall query rate: 5.16 queries/sec
Mongo last row per host:
min:  1277.18ms, med:  1660.10ms, mean:  1936.99ms, max: 3109.38ms, stddev:   553.58ms, sum: 193.7sec, count: 100
all queries             :
min:  1277.18ms, med:  1660.10ms, mean:  1936.99ms, max: 3109.38ms, stddev:   553.58ms, sum: 193.7sec, count: 100

Run complete after 100 queries with 10 workers (Overall query rate 5.15 queries/sec):
Mongo last row per host:
min:  1277.18ms, med:  1660.10ms, mean:  1936.99ms, max: 3109.38ms, stddev:   553.58ms, sum: 193.7sec, count: 100
all queries             :
min:  1277.18ms, med:  1660.10ms, mean:  1936.99ms, max: 3109.38ms, stddev:   553.58ms, sum: 193.7sec, count: 100
wall clock time: 19.440762sec
```

Total Time: 19.44 sec

MongoDB: Query–groupby-orderby-limit

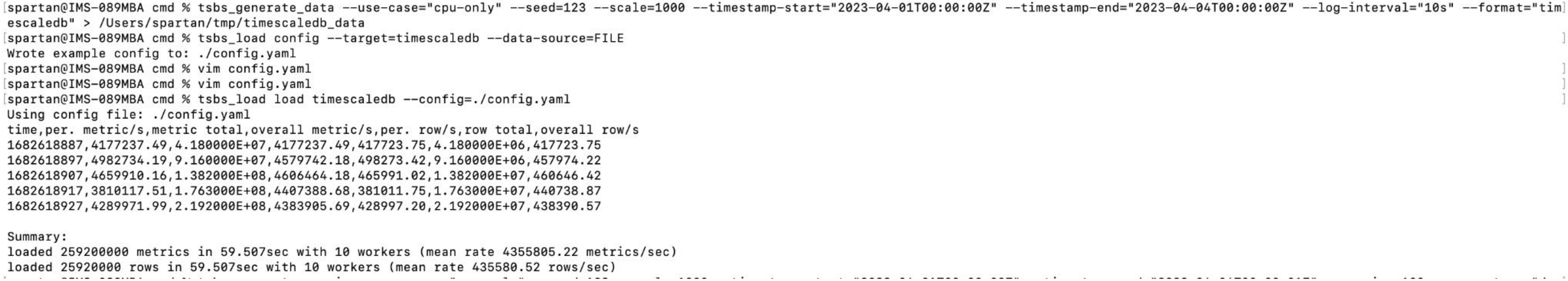


```
spartan@IMS-089MBA cmd % tsbs_generate_queries --use-case="cpu-only" --seed=123 --scale=1000 --timestamp-start="2023-04-01T00:00:00Z" --timestamp-end="2023-04-04T00:00:01Z" --queries=100 --query-type="gro
upby-orderby-limit" --format="mongo" > /Users/spartan/tmp/mongo_query4
Mongo max cpu over last 5 min-intervals (random end): 100 points
spartan@IMS-089MBA cmd % tsbs_run_queries_mongo --file=/Users/spartan/tmp/mongo_query4 --workers=10
After 100 queries with 10 workers:
Interval query rate: 7.44 queries/sec   Overall query rate: 7.44 queries/sec
Mongo max cpu over last 5 min-intervals (random end):
min:   783.49ms, med:  1213.25ms, mean:  1342.95ms, max: 2411.26ms, stddev:   525.59ms, sum: 134.3sec, count: 100
all queries                                            :
min:   783.49ms, med:  1213.25ms, mean:  1342.95ms, max: 2411.26ms, stddev:   525.59ms, sum: 134.3sec, count: 100

Run complete after 100 queries with 10 workers (Overall query rate 7.43 queries/sec):
Mongo max cpu over last 5 min-intervals (random end):
min:   783.49ms, med:  1213.25ms, mean:  1342.95ms, max: 2411.26ms, stddev:   525.59ms, sum: 134.3sec, count: 100
all queries                                            :
min:   783.49ms, med:  1213.25ms, mean:  1342.95ms, max: 2411.26ms, stddev:   525.59ms, sum: 134.3sec, count: 100
wall clock time: 13.470554sec
```

Total Time: 13.47 sec

Performing TimescaleDB Queries

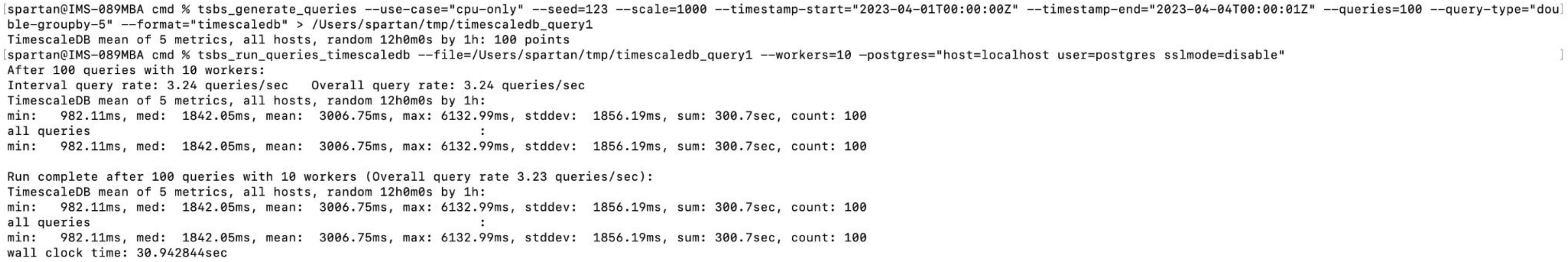
TimescaleDB: Data Generation and Data Loading



```
spartan@IMS-089MBA cmd % tsbs_generate_data --use-case="cpu-only" --seed=123 --scale=1000 --timestamp-start="2023-04-01T00:00:00Z" --timestamp-end="2023-04-04T00:00:00Z" --log-interval="10s" --format="tim
escaledb" > /Users/spartan/tmp/timescaledb_data
spartan@IMS-089MBA cmd % tsbs_load config --target=timescaledb --data-source=FILE
Wrote example config to: ./config.yaml
spartan@IMS-089MBA cmd % vim config.yaml
spartan@IMS-089MBA cmd % vim config.yaml
spartan@IMS-089MBA cmd % tsbs_load load timescaledb --config=./config.yaml
Using config file: ./config.yaml
time,per. metric/s,metric total,overall metric/s,per. row/s,row total,overall row/s
1682618887,4177237.49,4.180000E+07,4177237.49,417723.75,4.180000E+06,417723.75
1682618897,4982734.19,9.160000E+07,4579742.18,498273.42,9.160000E+06,457974.22
1682618907,4659910.16,1.382000E+08,4606464.18,465991.02,1.382000E+07,460646.42
1682618917,3810117.51,1.763000E+08,4407388.68,381011.75,1.763000E+07,440738.87
1682618927,4289971.99,2.192000E+08,4383905.69,428997.20,2.192000E+07,438390.57

Summary:
loaded 259200000 metrics in 59.507sec with 10 workers (mean rate 4355805.22 metrics/sec)
loaded 25920000 rows in 59.507sec with 10 workers (mean rate 435580.52 rows/sec)
```

Data Loading : 59 seconds

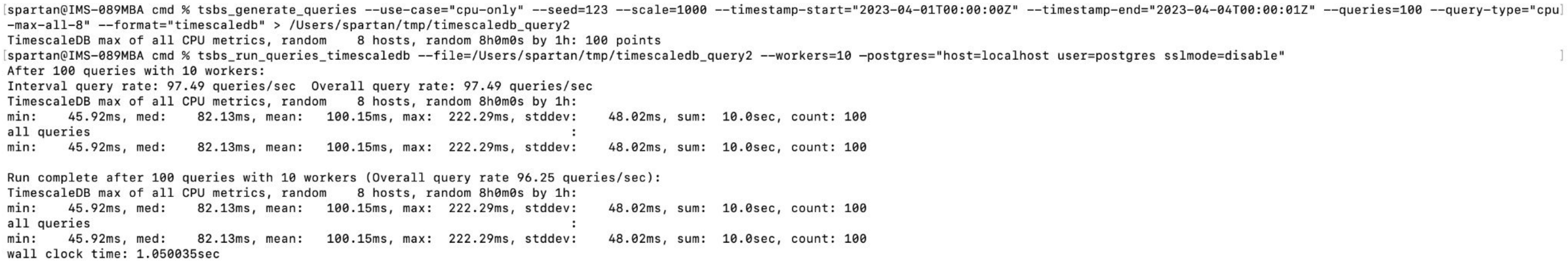
TimescaleDB: Query–double-groupby-5



```
[spartan@IMS-089MBA cmd % tsbs_generate_queries --use-case="cpu-only" --seed=123 --scale=1000 --timestamp-start="2023-04-01T00:00:00Z" --timestamp-end="2023-04-04T00:00:01Z" --queries=100 --query-type="dou]
 ble-groupby-5" --format="timescaledb" > /Users/spartan/tmp/timescaledb_query1
 TimescaleDB mean of 5 metrics, all hosts, random 12h0m0s by 1h: 100 points
[spartan@IMS-089MBA cmd % tsbs_run_queries_timescaledb --file=/Users/spartan/tmp/timescaledb_query1 --workers=10 –postgres="host=localhost user=postgres sslmode=disable"                                        ]
 After 100 queries with 10 workers:
 Interval query rate: 3.24 queries/sec   Overall query rate: 3.24 queries/sec
 TimescaleDB mean of 5 metrics, all hosts, random 12h0m0s by 1h:
 min:   982.11ms, med:  1842.05ms, mean:  3006.75ms, max: 6132.99ms, stddev:  1856.19ms, sum: 300.7sec, count: 100
 all queries                                                   :
 min:   982.11ms, med:  1842.05ms, mean:  3006.75ms, max: 6132.99ms, stddev:  1856.19ms, sum: 300.7sec, count: 100

 Run complete after 100 queries with 10 workers (Overall query rate 3.23 queries/sec):
 TimescaleDB mean of 5 metrics, all hosts, random 12h0m0s by 1h:
 min:   982.11ms, med:  1842.05ms, mean:  3006.75ms, max: 6132.99ms, stddev:  1856.19ms, sum: 300.7sec, count: 100
 all queries                                                   :
 min:   982.11ms, med:  1842.05ms, mean:  3006.75ms, max: 6132.99ms, stddev:  1856.19ms, sum: 300.7sec, count: 100
 wall clock time: 30.942844sec
```

Total Time: 30.94 sec

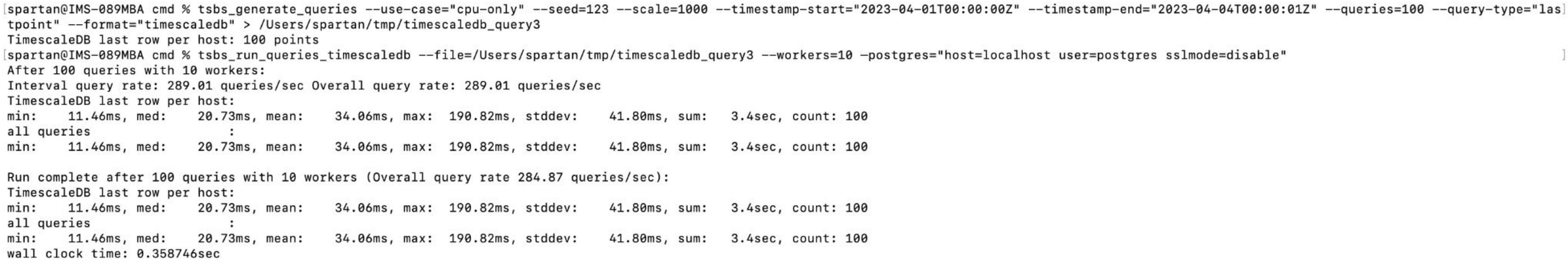
TimescaleDB: Query–cpu-max-all-8



```
spartan@IMS-089MBA cmd % tsbs_generate_queries --use-case="cpu-only" --seed=123 --scale=1000 --timestamp-start="2023-04-01T00:00:00Z" --timestamp-end="2023-04-04T00:00:01Z" --queries=100 --query-type="cpu
-max-all-8" --format="timescaledb" > /Users/spartan/tmp/timescaledb_query2
TimescaleDB max of all CPU metrics, random    8 hosts, random 8h0m0s by 1h: 100 points
spartan@IMS-089MBA cmd % tsbs_run_queries_timescaledb --file=/Users/spartan/tmp/timescaledb_query2 --workers=10 –postgres="host=localhost user=postgres sslmode=disable"
After 100 queries with 10 workers:
Interval query rate: 97.49 queries/sec  Overall query rate: 97.49 queries/sec
TimescaleDB max of all CPU metrics, random    8 hosts, random 8h0m0s by 1h:
min:    45.92ms, med:    82.13ms, mean:   100.15ms, max:  222.29ms, stddev:    48.02ms, sum:  10.0sec, count: 100
all queries                                                             :
min:    45.92ms, med:    82.13ms, mean:   100.15ms, max:  222.29ms, stddev:    48.02ms, sum:  10.0sec, count: 100

Run complete after 100 queries with 10 workers (Overall query rate 96.25 queries/sec):
TimescaleDB max of all CPU metrics, random    8 hosts, random 8h0m0s by 1h:
min:    45.92ms, med:    82.13ms, mean:   100.15ms, max:  222.29ms, stddev:    48.02ms, sum:  10.0sec, count: 100
all queries                                                             :
min:    45.92ms, med:    82.13ms, mean:   100.15ms, max:  222.29ms, stddev:    48.02ms, sum:  10.0sec, count: 100
wall clock time: 1.050035sec
```

Total Time: 1.05 sec

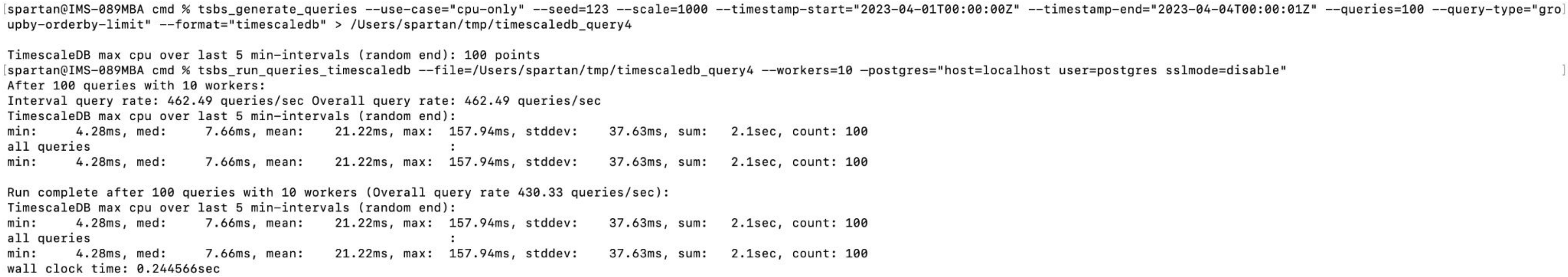
TimescaleDB: Query–lastpoint



```
[spartan@IMS-089MBA cmd % tsbs_generate_queries --use-case="cpu-only" --seed=123 --scale=1000 --timestamp-start="2023-04-01T00:00:00Z" --timestamp-end="2023-04-04T00:00:01Z" --queries=100 --query-type="las]
tpoint" --format="timescaledb" > /Users/spartan/tmp/timescaledb_query3
TimescaleDB last row per host: 100 points
[spartan@IMS-089MBA cmd % tsbs_run_queries_timescaledb --file=/Users/spartan/tmp/timescaledb_query3 --workers=10 -postgres="host=localhost user=postgres sslmode=disable"                                            ]
After 100 queries with 10 workers:
Interval query rate: 289.01 queries/sec Overall query rate: 289.01 queries/sec
TimescaleDB last row per host:
min:    11.46ms, med:    20.73ms, mean:    34.06ms, max:  190.82ms, stddev:    41.80ms, sum:   3.4sec, count: 100
all queries                   :
min:    11.46ms, med:    20.73ms, mean:    34.06ms, max:  190.82ms, stddev:    41.80ms, sum:   3.4sec, count: 100

Run complete after 100 queries with 10 workers (Overall query rate 284.87 queries/sec):
TimescaleDB last row per host:
min:    11.46ms, med:    20.73ms, mean:    34.06ms, max:  190.82ms, stddev:    41.80ms, sum:   3.4sec, count: 100
all queries                   :
min:    11.46ms, med:    20.73ms, mean:    34.06ms, max:  190.82ms, stddev:    41.80ms, sum:   3.4sec, count: 100
wall clock time: 0.358746sec
```

Total Time: 0.35 sec

TimescaleDB: Query–groupby-orderby-limit

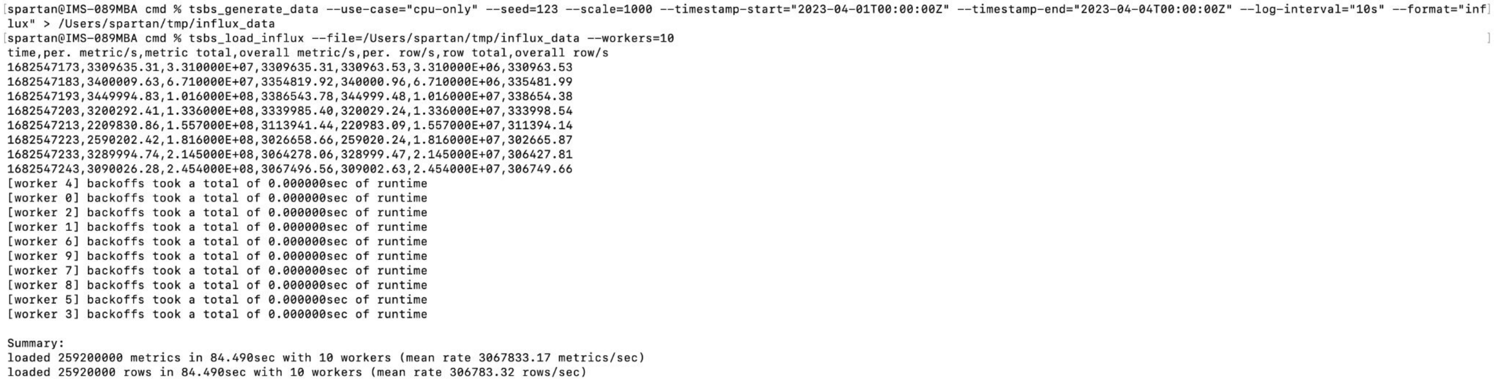


```
[spartan@IMS-089MBA cmd % tsbs_generate_queries --use-case="cpu-only" --seed=123 --scale=1000 --timestamp-start="2023-04-01T00:00:00Z" --timestamp-end="2023-04-04T00:00:01Z" --queries=100 --query-type="gro]
upby-orderby-limit" --format="timescaledb" > /Users/spartan/tmp/timescaledb_query4

TimescaleDB max cpu over last 5 min-intervals (random end): 100 points
[spartan@IMS-089MBA cmd % tsbs_run_queries_timescaledb --file=/Users/spartan/tmp/timescaledb_query4 --workers=10 -postgres="host=localhost user=postgres sslmode=disable"                                        ]
After 100 queries with 10 workers:
Interval query rate: 462.49 queries/sec Overall query rate: 462.49 queries/sec
TimescaleDB max cpu over last 5 min-intervals (random end):
min:     4.28ms, med:     7.66ms, mean:    21.22ms, max:  157.94ms, stddev:    37.63ms, sum:   2.1sec, count: 100
all queries                                                :
min:     4.28ms, med:     7.66ms, mean:    21.22ms, max:  157.94ms, stddev:    37.63ms, sum:   2.1sec, count: 100

Run complete after 100 queries with 10 workers (Overall query rate 430.33 queries/sec):
TimescaleDB max cpu over last 5 min-intervals (random end):
min:     4.28ms, med:     7.66ms, mean:    21.22ms, max:  157.94ms, stddev:    37.63ms, sum:   2.1sec, count: 100
all queries                                                :
min:     4.28ms, med:     7.66ms, mean:    21.22ms, max:  157.94ms, stddev:    37.63ms, sum:   2.1sec, count: 100
wall clock time: 0.244566sec
```

Total Time: 0.24 sec

Performing influxDB Queries

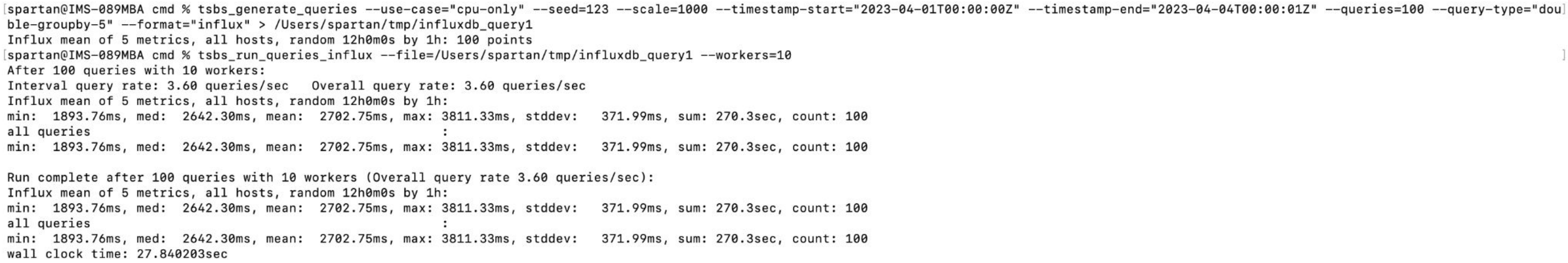
InfluxDB: Data Generation and Data Loading



```
spartan@IMS-089MBA cmd % tsbs_generate_data --use-case="cpu-only" --seed=123 --scale=1000 --timestamp-start="2023-04-01T00:00:00Z" --timestamp-end="2023-04-04T00:00:00Z" --log-interval="10s" --format="inf
lux" > /Users/spartan/tmp/influx_data
spartan@IMS-089MBA cmd % tsbs_load_influx --file=/Users/spartan/tmp/influx_data --workers=10
time,per. metric/s,metric total,overall metric/s,per. row/s,row total,overall row/s
1682547173,3309635.31,3.310000E+07,3309635.31,330963.53,3.310000E+06,330963.53
1682547183,3400009.63,6.710000E+07,3354819.92,340000.96,6.710000E+06,335481.99
1682547193,3449994.83,1.016000E+08,3386543.78,344999.48,1.016000E+07,338654.38
1682547203,3200292.41,1.336000E+08,3339985.40,320029.24,1.336000E+07,333998.54
1682547213,2209830.86,1.557000E+08,3113941.44,220983.09,1.557000E+07,311394.14
1682547223,2590202.42,1.816000E+08,3026658.66,259020.24,1.816000E+07,302665.87
1682547233,3289994.74,2.145000E+08,3064278.06,328999.47,2.145000E+07,306427.81
1682547243,3090026.28,2.454000E+08,3067496.56,309002.63,2.454000E+07,306749.66
[worker 4] backoffs took a total of 0.000000sec of runtime
[worker 0] backoffs took a total of 0.000000sec of runtime
[worker 2] backoffs took a total of 0.000000sec of runtime
[worker 1] backoffs took a total of 0.000000sec of runtime
[worker 6] backoffs took a total of 0.000000sec of runtime
[worker 9] backoffs took a total of 0.000000sec of runtime
[worker 7] backoffs took a total of 0.000000sec of runtime
[worker 8] backoffs took a total of 0.000000sec of runtime
[worker 5] backoffs took a total of 0.000000sec of runtime
[worker 3] backoffs took a total of 0.000000sec of runtime

Summary:
loaded 259200000 metrics in 84.490sec with 10 workers (mean rate 3067833.17 metrics/sec)
loaded 25920000 rows in 84.490sec with 10 workers (mean rate 306783.32 rows/sec)
```

Data Loading : 2.337 seconds

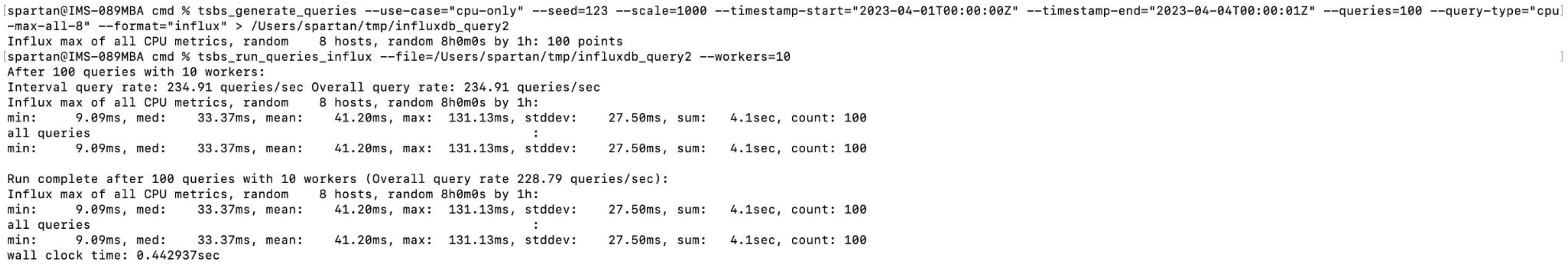
InfluxDB: Query–double-groupby-5



```
[spartan@IMS-089MBA cmd % tsbs_generate_queries --use-case="cpu-only" --seed=123 --scale=1000 --timestamp-start="2023-04-01T00:00:00Z" --timestamp-end="2023-04-04T00:00:01Z" --queries=100 --query-type="dou]
ble-groupby-5" --format="influx" > /Users/spartan/tmp/influxdb_query1
Influx mean of 5 metrics, all hosts, random 12h0m0s by 1h: 100 points
[spartan@IMS-089MBA cmd % tsbs_run_queries_influx --file=/Users/spartan/tmp/influxdb_query1 --workers=10                                                                                            ]
After 100 queries with 10 workers:
Interval query rate: 3.60 queries/sec   Overall query rate: 3.60 queries/sec
Influx mean of 5 metrics, all hosts, random 12h0m0s by 1h:
min:  1893.76ms, med:  2642.30ms, mean:  2702.75ms, max: 3811.33ms, stddev:   371.99ms, sum: 270.3sec, count: 100
all queries                                             :
min:  1893.76ms, med:  2642.30ms, mean:  2702.75ms, max: 3811.33ms, stddev:   371.99ms, sum: 270.3sec, count: 100

Run complete after 100 queries with 10 workers (Overall query rate 3.60 queries/sec):
Influx mean of 5 metrics, all hosts, random 12h0m0s by 1h:
min:  1893.76ms, med:  2642.30ms, mean:  2702.75ms, max: 3811.33ms, stddev:   371.99ms, sum: 270.3sec, count: 100
all queries                                             :
min:  1893.76ms, med:  2642.30ms, mean:  2702.75ms, max: 3811.33ms, stddev:   371.99ms, sum: 270.3sec, count: 100
wall clock time: 27.840203sec
```

Total Time: 27.84 sec

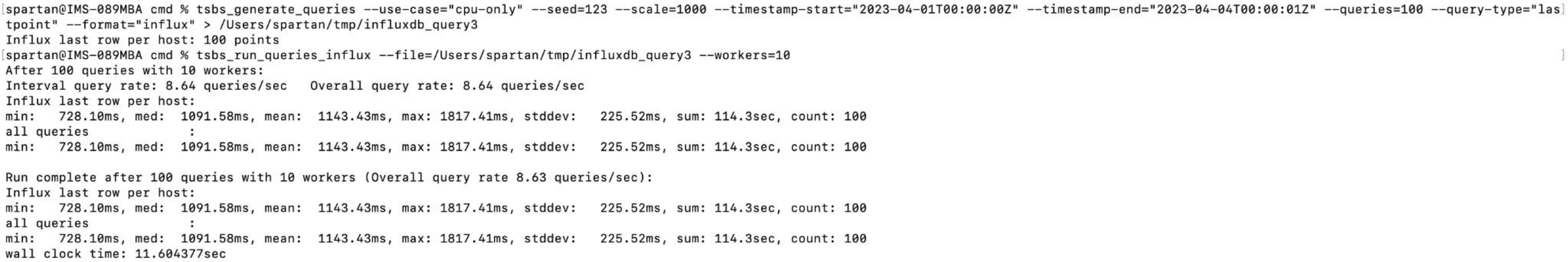
InfluxDB: Query–cpu-max-all-8



```
[spartan@IMS-089MBA cmd % tsbs_generate_queries --use-case="cpu-only" --seed=123 --scale=1000 --timestamp-start="2023-04-01T00:00:00Z" --timestamp-end="2023-04-04T00:00:01Z" --queries=100 --query-type="cpu]
-max-all-8" --format="influx" > /Users/spartan/tmp/influxdb_query2
Influx max of all CPU metrics, random    8 hosts, random 8h0m0s by 1h: 100 points
[spartan@IMS-089MBA cmd % tsbs_run_queries_influx --file=/Users/spartan/tmp/influxdb_query2 --workers=10                                                                                  ]
After 100 queries with 10 workers:
Interval query rate: 234.91 queries/sec Overall query rate: 234.91 queries/sec
Influx max of all CPU metrics, random    8 hosts, random 8h0m0s by 1h:
min:     9.09ms, med:    33.37ms, mean:    41.20ms, max:  131.13ms, stddev:    27.50ms, sum:   4.1sec, count: 100
all queries                                                          :
min:     9.09ms, med:    33.37ms, mean:    41.20ms, max:  131.13ms, stddev:    27.50ms, sum:   4.1sec, count: 100

Run complete after 100 queries with 10 workers (Overall query rate 228.79 queries/sec):
Influx max of all CPU metrics, random    8 hosts, random 8h0m0s by 1h:
min:     9.09ms, med:    33.37ms, mean:    41.20ms, max:  131.13ms, stddev:    27.50ms, sum:   4.1sec, count: 100
all queries                                                          :
min:     9.09ms, med:    33.37ms, mean:    41.20ms, max:  131.13ms, stddev:    27.50ms, sum:   4.1sec, count: 100
wall clock time: 0.442937sec
```

Total Time: 0.44 sec

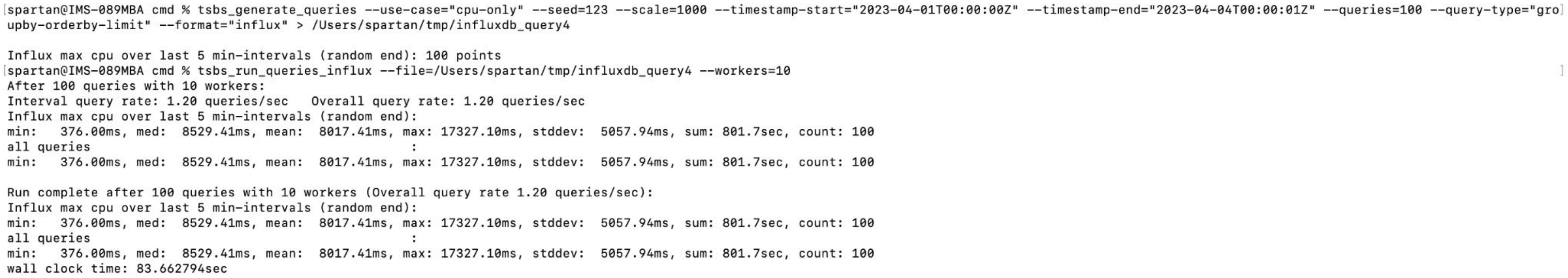
InfluxDB: Query–lastpoint



```
[spartan@IMS-089MBA cmd % tsbs_generate_queries --use-case="cpu-only" --seed=123 --scale=1000 --timestamp-start="2023-04-01T00:00:00Z" --timestamp-end="2023-04-04T00:00:01Z" --queries=100 --query-type="las]
tpoint" --format="influx" > /Users/spartan/tmp/influxdb_query3
Influx last row per host: 100 points
[spartan@IMS-089MBA cmd % tsbs_run_queries_influx --file=/Users/spartan/tmp/influxdb_query3 --workers=10                                                                                                  ]
After 100 queries with 10 workers:
Interval query rate: 8.64 queries/sec   Overall query rate: 8.64 queries/sec
Influx last row per host:
min:   728.10ms, med:  1091.58ms, mean:  1143.43ms, max: 1817.41ms, stddev:   225.52ms, sum: 114.3sec, count: 100
all queries             :
min:   728.10ms, med:  1091.58ms, mean:  1143.43ms, max: 1817.41ms, stddev:   225.52ms, sum: 114.3sec, count: 100

Run complete after 100 queries with 10 workers (Overall query rate 8.63 queries/sec):
Influx last row per host:
min:   728.10ms, med:  1091.58ms, mean:  1143.43ms, max: 1817.41ms, stddev:   225.52ms, sum: 114.3sec, count: 100
all queries             :
min:   728.10ms, med:  1091.58ms, mean:  1143.43ms, max: 1817.41ms, stddev:   225.52ms, sum: 114.3sec, count: 100
wall clock time: 11.604377sec
```

Total Time: 11.60 sec

InfluxDB: Query–groupby-orderby-limit

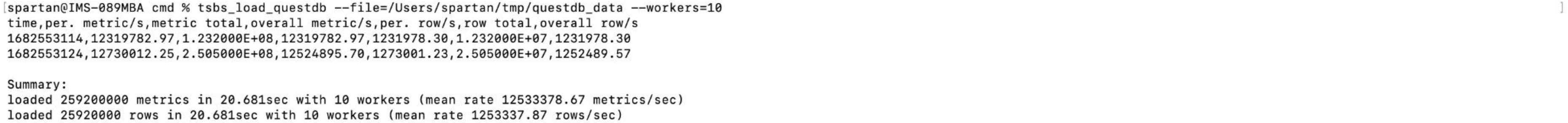


```
[spartan@IMS-089MBA cmd % tsbs_generate_queries --use-case="cpu-only" --seed=123 --scale=1000 --timestamp-start="2023-04-01T00:00:00Z" --timestamp-end="2023-04-04T00:00:01Z" --queries=100 --query-type="gro]
 upby-orderby-limit" --format="influx" > /Users/spartan/tmp/influxdb_query4

 Influx max cpu over last 5 min-intervals (random end): 100 points
[spartan@IMS-089MBA cmd % tsbs_run_queries_influx --file=/Users/spartan/tmp/influxdb_query4 --workers=10                                                                                  ]
 After 100 queries with 10 workers:
 Interval query rate: 1.20 queries/sec   Overall query rate: 1.20 queries/sec
 Influx max cpu over last 5 min-intervals (random end):
 min:   376.00ms, med:  8529.41ms, mean:  8017.41ms, max: 17327.10ms, stddev:  5057.94ms, sum: 801.7sec, count: 100
 all queries                                 :
 min:   376.00ms, med:  8529.41ms, mean:  8017.41ms, max: 17327.10ms, stddev:  5057.94ms, sum: 801.7sec, count: 100

 Run complete after 100 queries with 10 workers (Overall query rate 1.20 queries/sec):
 Influx max cpu over last 5 min-intervals (random end):
 min:   376.00ms, med:  8529.41ms, mean:  8017.41ms, max: 17327.10ms, stddev:  5057.94ms, sum: 801.7sec, count: 100
 all queries                                 :
 min:   376.00ms, med:  8529.41ms, mean:  8017.41ms, max: 17327.10ms, stddev:  5057.94ms, sum: 801.7sec, count: 100
 wall clock time: 83.662794sec
```

Total Time: 83.66 sec

Performing QuestDB Queries

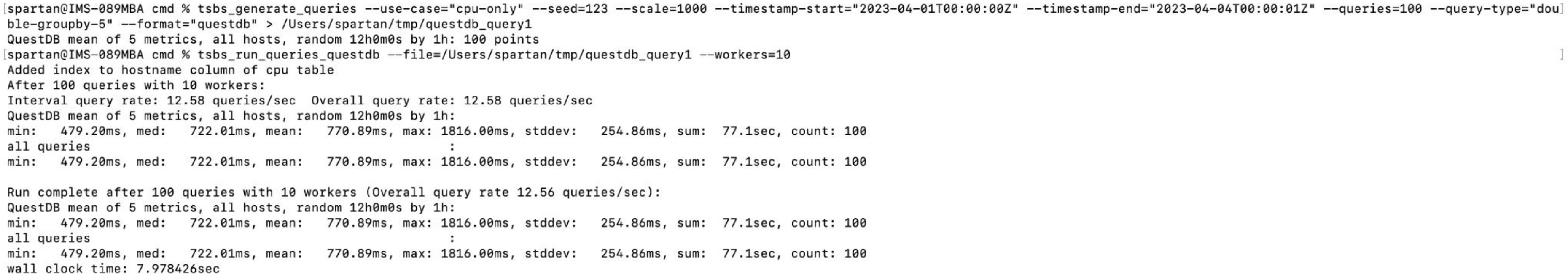
QuestDB: Data Generation and Data Loading



```
[spartan@IMS-089MBA cmd % tsbs_load_questdb --file=/Users/spartan/tmp/questdb_data --workers=10
time,per. metric/s,metric total,overall metric/s,per. row/s,row total,overall row/s
1682553114,12319782.97,1.232000E+08,12319782.97,1231978.30,1.232000E+07,1231978.30
1682553124,12730012.25,2.505000E+08,12524895.70,1273001.23,2.505000E+07,1252489.57

Summary:
loaded 259200000 metrics in 20.681sec with 10 workers (mean rate 12533378.67 metrics/sec)
loaded 25920000 rows in 20.681sec with 10 workers (mean rate 1253337.87 rows/sec)
```

Data Loading : 20.68 sec

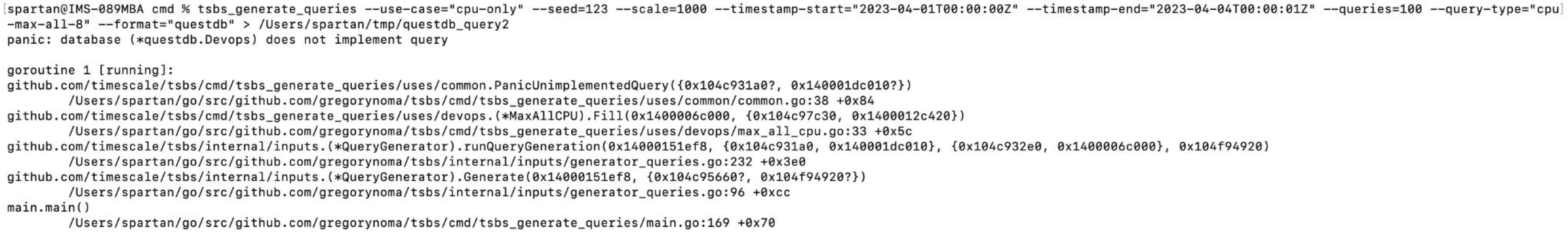
QuestDB:Query–double-groupby-5



```
[spartan@IMS-089MBA cmd % tsbs_generate_queries --use-case="cpu-only" --seed=123 --scale=1000 --timestamp-start="2023-04-01T00:00:00Z" --timestamp-end="2023-04-04T00:00:01Z" --queries=100 --query-type="dou]
ble-groupby-5" --format="questdb" > /Users/spartan/tmp/questdb_query1
QuestDB mean of 5 metrics, all hosts, random 12h0m0s by 1h: 100 points
[spartan@IMS-089MBA cmd % tsbs_run_queries_questdb --file=/Users/spartan/tmp/questdb_query1 --workers=10                                                          ]
Added index to hostname column of cpu table
After 100 queries with 10 workers:
Interval query rate: 12.58 queries/sec  Overall query rate: 12.58 queries/sec
QuestDB mean of 5 metrics, all hosts, random 12h0m0s by 1h:
min:   479.20ms, med:   722.01ms, mean:   770.89ms, max: 1816.00ms, stddev:   254.86ms, sum:  77.1sec, count: 100
all queries                                             :
min:   479.20ms, med:   722.01ms, mean:   770.89ms, max: 1816.00ms, stddev:   254.86ms, sum:  77.1sec, count: 100

Run complete after 100 queries with 10 workers (Overall query rate 12.56 queries/sec):
QuestDB mean of 5 metrics, all hosts, random 12h0m0s by 1h:
min:   479.20ms, med:   722.01ms, mean:   770.89ms, max: 1816.00ms, stddev:   254.86ms, sum:  77.1sec, count: 100
all queries                                             :
min:   479.20ms, med:   722.01ms, mean:   770.89ms, max: 1816.00ms, stddev:   254.86ms, sum:  77.1sec, count: 100
wall clock time: 7.978426sec
```

Total Time: 7.97sec

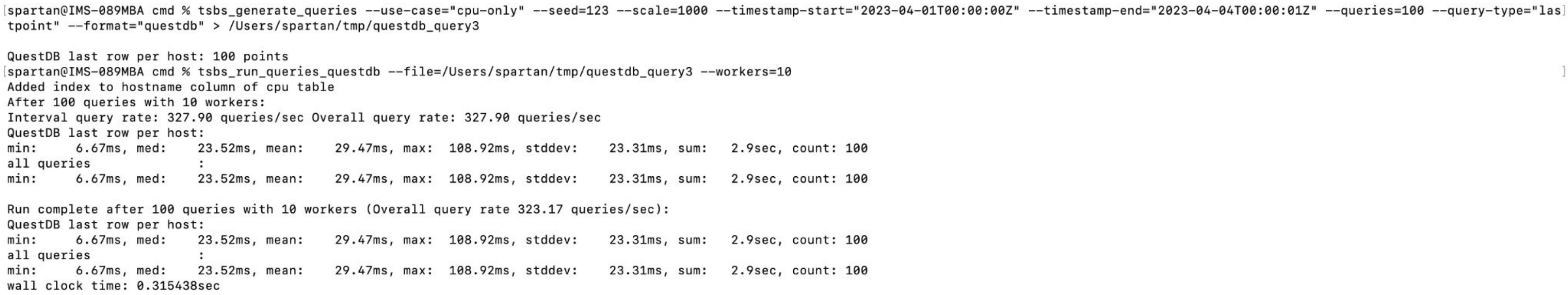
QuestDB: Query–cpu-max-all-8



```
spartan@IMS-089MBA cmd % tsbs_generate_queries --use-case="cpu-only" --seed=123 --scale=1000 --timestamp-start="2023-04-01T00:00:00Z" --timestamp-end="2023-04-04T00:00:01Z" --queries=100 --query-type="cpu
-max-all-8" --format="questdb" > /Users/spartan/tmp/questdb_query2
panic: database (*questdb.Devops) does not implement query

goroutine 1 [running]:
github.com/timescale/tsbs/cmd/tsbs_generate_queries/uses/common.PanicUnimplementedQuery({0x104c931a0?, 0x140001dc010?})
        /Users/spartan/go/src/github.com/gregorynoma/tsbs/cmd/tsbs_generate_queries/uses/common/common.go:38 +0x84
github.com/timescale/tsbs/cmd/tsbs_generate_queries/uses/devops.(*MaxAllCPU).Fill(0x1400006c000, {0x104c97c30, 0x1400012c420})
        /Users/spartan/go/src/github.com/gregorynoma/tsbs/cmd/tsbs_generate_queries/uses/devops/max_all_cpu.go:33 +0x5c
github.com/timescale/tsbs/internal/inputs.(*QueryGenerator).runQueryGeneration(0x14000151ef8, {0x104c931a0, 0x140001dc010}, {0x104c932e0, 0x1400006c000}, 0x104f94920)
        /Users/spartan/go/src/github.com/gregorynoma/tsbs/internal/inputs/generator_queries.go:232 +0x3e0
github.com/timescale/tsbs/internal/inputs.(*QueryGenerator).Generate(0x14000151ef8, {0x104c95660?, 0x104f94920?})
        /Users/spartan/go/src/github.com/gregorynoma/tsbs/internal/inputs/generator_queries.go:96 +0xcc
main.main()
        /Users/spartan/go/src/github.com/gregorynoma/tsbs/cmd/tsbs_generate_queries/main.go:169 +0x70
```

No Output

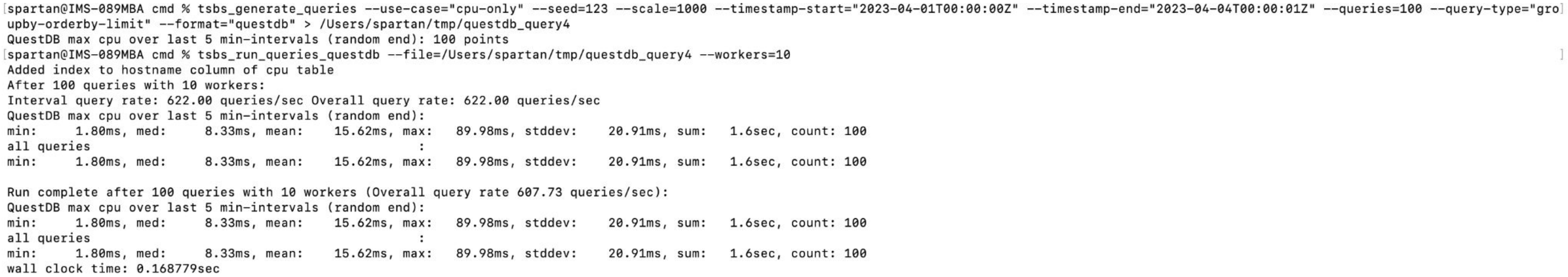
QuestDB: Query–lastpoint



```
spartan@IMS-089MBA cmd % tsbs_generate_queries --use-case="cpu-only" --seed=123 --scale=1000 --timestamp-start="2023-04-01T00:00:00Z" --timestamp-end="2023-04-04T00:00:01Z" --queries=100 --query-type="las
tpoint" --format="questdb" > /Users/spartan/tmp/questdb_query3

QuestDB last row per host: 100 points
spartan@IMS-089MBA cmd % tsbs_run_queries_questdb --file=/Users/spartan/tmp/questdb_query3 --workers=10
Added index to hostname column of cpu table
After 100 queries with 10 workers:
Interval query rate: 327.90 queries/sec Overall query rate: 327.90 queries/sec
QuestDB last row per host:
min:     6.67ms, med:    23.52ms, mean:    29.47ms, max:  108.92ms, stddev:    23.31ms, sum:   2.9sec, count: 100
all queries              :
min:     6.67ms, med:    23.52ms, mean:    29.47ms, max:  108.92ms, stddev:    23.31ms, sum:   2.9sec, count: 100

Run complete after 100 queries with 10 workers (Overall query rate 323.17 queries/sec):
QuestDB last row per host:
min:     6.67ms, med:    23.52ms, mean:    29.47ms, max:  108.92ms, stddev:    23.31ms, sum:   2.9sec, count: 100
all queries              :
min:     6.67ms, med:    23.52ms, mean:    29.47ms, max:  108.92ms, stddev:    23.31ms, sum:   2.9sec, count: 100
wall clock time: 0.315438sec
```

Total Time: 0.31 sec

QuestDB: Query–groupby-orderby-limit



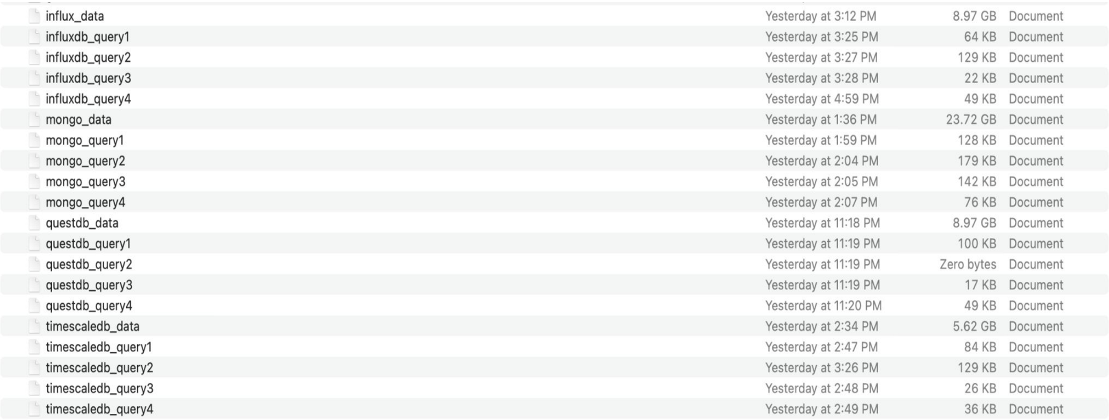
```
[spartan@IMS-089MBA cmd % tsbs_generate_queries --use-case="cpu-only" --seed=123 --scale=1000 --timestamp-start="2023-04-01T00:00:00Z" --timestamp-end="2023-04-04T00:00:01Z" --queries=100 --query-type="gro]
upby-orderby-limit" --format="questdb" > /Users/spartan/tmp/questdb_query4
QuestDB max cpu over last 5 min-intervals (random end): 100 points
[spartan@IMS-089MBA cmd % tsbs_run_queries_questdb --file=/Users/spartan/tmp/questdb_query4 --workers=10
Added index to hostname column of cpu table
After 100 queries with 10 workers:
Interval query rate: 622.00 queries/sec Overall query rate: 622.00 queries/sec
QuestDB max cpu over last 5 min-intervals (random end):
min:     1.80ms, med:     8.33ms, mean:    15.62ms, max:   89.98ms, stddev:    20.91ms, sum:   1.6sec, count: 100
all queries                                           :
min:     1.80ms, med:     8.33ms, mean:    15.62ms, max:   89.98ms, stddev:    20.91ms, sum:   1.6sec, count: 100

Run complete after 100 queries with 10 workers (Overall query rate 607.73 queries/sec):
QuestDB max cpu over last 5 min-intervals (random end):
min:     1.80ms, med:     8.33ms, mean:    15.62ms, max:   89.98ms, stddev:    20.91ms, sum:   1.6sec, count: 100
all queries                                           :
min:     1.80ms, med:     8.33ms, mean:    15.62ms, max:   89.98ms, stddev:    20.91ms, sum:   1.6sec, count: 100
wall clock time: 0.168779sec
```

Total Time: 0.16 sec

Result Summary

Data Storage Size

MongoDB: 23.72 GB
TimescaleDB : 5.62 GB
InfluxDB : 8.97 GB
QuestDB: 8.97 GB



influx_data	Yesterday at 3:12 PM	8.97 GB	Document
influxdb_query1	Yesterday at 3:25 PM	64 KB	Document
influxdb_query2	Yesterday at 3:27 PM	129 KB	Document
influxdb_query3	Yesterday at 3:28 PM	22 KB	Document
influxdb_query4	Yesterday at 4:59 PM	49 KB	Document
mongo_data	Yesterday at 1:36 PM	23.72 GB	Document
mongo_query1	Yesterday at 1:59 PM	128 KB	Document
mongo_query2	Yesterday at 2:04 PM	179 KB	Document
mongo_query3	Yesterday at 2:05 PM	142 KB	Document
mongo_query4	Yesterday at 2:07 PM	76 KB	Document
questdb_data	Yesterday at 11:18 PM	8.97 GB	Document
questdb_query1	Yesterday at 11:19 PM	100 KB	Document
questdb_query2	Yesterday at 11:19 PM	Zero bytes	Document
questdb_query3	Yesterday at 11:19 PM	17 KB	Document
questdb_query4	Yesterday at 11:20 PM	49 KB	Document
timescaledb_data	Yesterday at 2:34 PM	5.62 GB	Document
timescaledb_query1	Yesterday at 2:47 PM	84 KB	Document
timescaledb_query2	Yesterday at 3:26 PM	129 KB	Document
timescaledb_query3	Yesterday at 2:48 PM	26 KB	Document
timescaledb_query4	Yesterday at 2:49 PM	36 KB	Document

Data Loading(Write Performance)

MongoDB(Naive): 246 sec
MongoDB(Recommended): 123 sec
TimescaleDB : 59 sec
InfluxDB : 2.33 sec
QuestDB: 20.68 sec

Query Execution (Read Performance)

	double-groupby-5	cpu-max-all-8	lastpoint	groupby-orderby-limit
MongoDB (Naive)	118	0.86	66.79	239
MongoDB(Recommended)	16.39	14.48	19.44	13.47
TimeScaleDB	30.94	1.05	0.35	0.24
InfluxDB	27.84	0.44	11.60	83.66
QuestDB	7.97	N/A	0.32	0.16

Data Storage Comparison

- TimeScaleDB wins as it uses compression technique to compress the generated data.

- InfluxDB and QuestDB has the same data size for two datasets. So the reason might the similarity in their storage structure as both uses SQL tables for storing data ,resulting in similar size.

- Both MongoDB methods resulted in 24 gb of data generated .Its prominent that storing every data in a different document and also chunking will result in the same size of data .But the data loading time will be affected as the chunking the data means that data can be uploaded quickly.

Write Performance

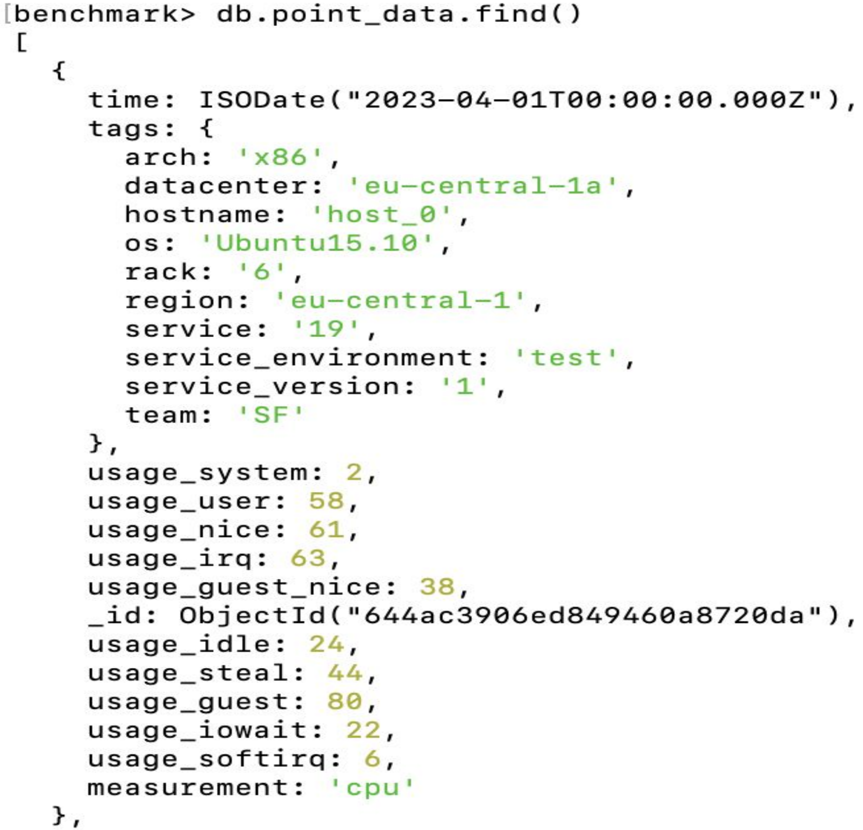
InfluxDB uses storage engine called the Time-Structured Merge Tree (TSM), which is designed to write data quickly and compactly. Also, InfluxDB works with very well with datasets having low cardinality and so it wins in write performance as the data used in this benchmark is relatively small .

QuestDB is designed for high performance and offers a number of features to optimize write performance, such as vectorization and zero-garbage collection.

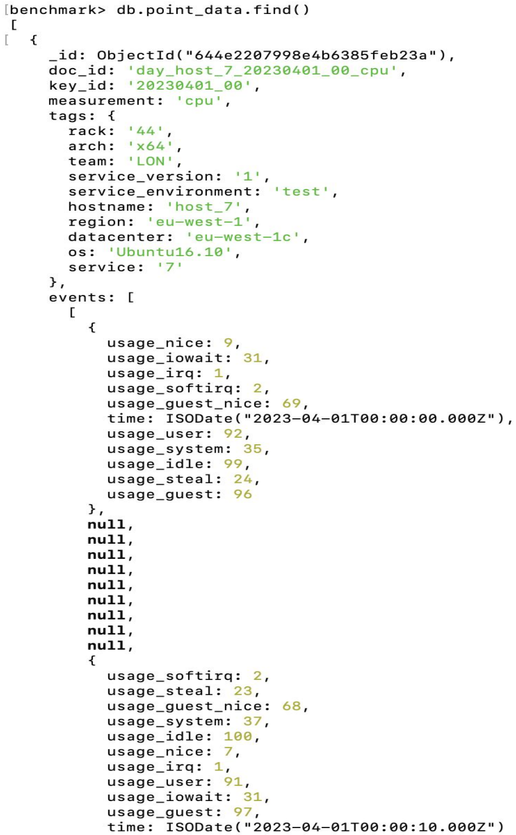
TimescaleDB offers chunking and indexing thus grouping data helps in importing data quickly, but its little slow than QuestDB.

MongoDB doesn’t provide any special feature for write performance and so Mongo-naive approach takes minutes to insert the data . But changing ‘document-per-event’ to false, helps the data to be grouped together and result in half time as MongoDb naive.

MongoDB Comparison



```
[benchmark> db.point_data.find()
 [
   {
     time: ISODate("2023-04-01T00:00:00.000Z"),
     tags: {
       arch: 'x86',
       datacenter: 'eu-central-1a',
       hostname: 'host_0',
       os: 'Ubuntu15.10',
       rack: '6',
       region: 'eu-central-1',
       service: '19',
       service_environment: 'test',
       service_version: '1',
       team: 'SF'
     },
     usage_system: 2,
     usage_user: 58,
     usage_nice: 61,
     usage_irq: 63,
     usage_guest_nice: 38,
     _id: ObjectId("644ac3906ed849460a8720da"),
     usage_idle: 24,
     usage_steal: 44,
     usage_guest: 80,
     usage_iowait: 22,
     usage_softirq: 6,
     measurement: 'cpu'
   },
```



```
[benchmark> db.point_data.find()
 [
[  {
     _id: ObjectId("644e2207998e4b6385feb23a"),
     doc_id: 'day_host_7_20230401_00_cpu',
     key_id: '20230401_00',
     measurement: 'cpu',
     tags: {
       rack: '44',
       arch: 'x64',
       team: 'LON',
       service_version: '1',
       service_environment: 'test',
       hostname: 'host_7',
       region: 'eu-west-1',
       datacenter: 'eu-west-1c',
       os: 'Ubuntu16.10',
       service: '7'
     },
     events: [
       [
         {
           usage_nice: 9,
           usage_iowait: 31,
           usage_irq: 1,
           usage_softirq: 2,
           usage_guest_nice: 69,
           time: ISODate("2023-04-01T00:00:00.000Z"),
           usage_user: 92,
           usage_system: 35,
           usage_idle: 99,
           usage_steal: 24,
           usage_guest: 96
         },
         null,
         null,
         null,
         null,
         null,
         null,
         null,
         null,
         null,
         {
           usage_softirq: 2,
           usage_steal: 23,
           usage_guest_nice: 68,
           usage_system: 37,
           usage_idle: 100,
           usage_nice: 7,
           usage_irq: 1,
           usage_user: 91,
           usage_iowait: 31,
           usage_guest: 97,
           time: ISODate("2023-04-01T00:00:10.000Z")
```

Naive vs Recommended

The default and straightforward method in storing data in MongoDB is to store them in document as MongoDB is a document database . So, MongoDB naive is where each reading is stored in a document.

The results of data write and queries were very bad compared to other databases . I found a better approach to store time series data in MongoDB.The better approach is to aggregate the time series data in a group based on a time.For E.g. : for each device, a document is created for every hour. So, this contains is a matrix with 60*60 (minutes and seconds) as the data is updated constantly.These data for an hour is stored as one document for a particular device.The document is updated accordingly when a reading is done and so there is no need to make a new document for every reading.

This allows the overall structure of data to be simple and efficient.We can see from the result that there is huge difference in efficiency in the recommended method. The data loading took almost half a time (246 vs 123) as because of aggregation, the data can be quickly inserted into the database .This grouped data can also be filtered quickly as we don't have to go to every document for query execution. So lesser documents result in the recommended method outperforming the naive method in most of the queries by a big margin.

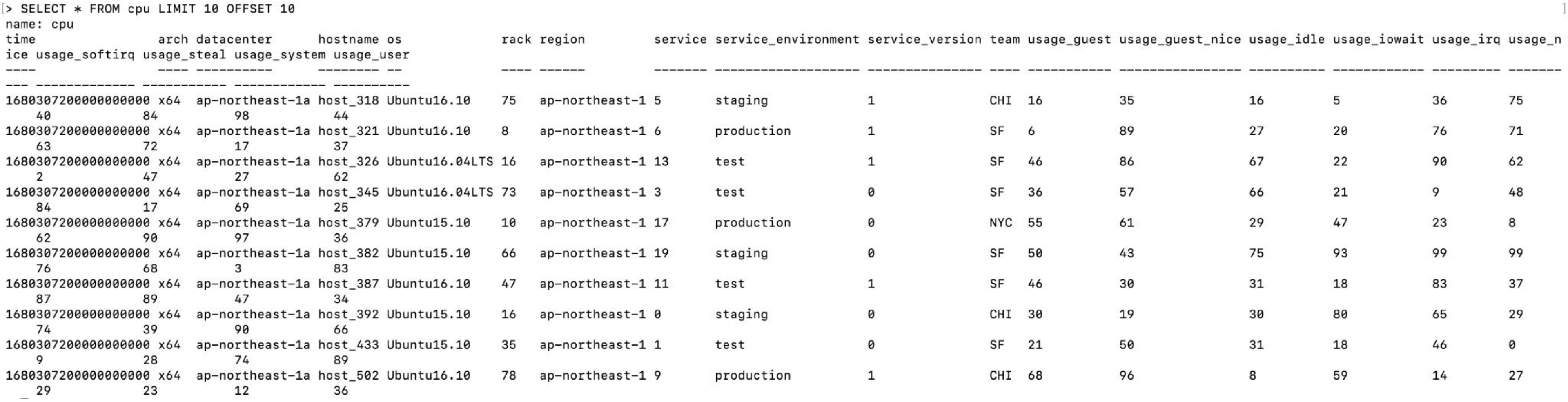
TimeScaleDB

- TimeScaleDB is built on PostgreSQL and is specifically designed for handling time series data.
- Timescale DB supports SQL and has a wide range of features like Group By functions and JOINS.
- This makes timescableDB a good choice for storing and querying time series data.
- Timescale DB uses a unique hypertable concept to partition data across time, which allows for efficient querying and analysis of time series data.

TimeScaleDB

- In my benchmarking , I have used chunk size as 12 hours and total duration of data is 3 days . So there will be 6 hyper tables created of duration 12 hrs and all the data will be divided and stored according to the tampstamp .
- Because of this TimeScale DB performed better than MongoDb and influxDB for queries like lastpoint and groupby as the grouping of data in chunks make it faster and efficient to query .
- But, the performance was worse for double-groupby query and comparatively bad for cpu-max query. But overall it beats majority of the databases in the benchmarking.

InfluxDB



```
> SELECT * FROM cpu LIMIT 10 OFFSET 10
name: cpu
time                arch datacenter      hostname os             rack region         service service_environment service_version team usage_guest usage_guest_nice usage_idle usage_iowait usage_irq usage_n
ice usage_softirq usage_steal usage_system usage_user
----                ---- ----------      -------- --             ---- ------         ------- ------------------- --------------- ---- ----------- ---------------- ---------- ------------ --------- -------
--- ------------- ----------- ------------ ----------
1680307200000000000 x64  ap-northeast-1a host_318 Ubuntu16.10    75   ap-northeast-1 5       staging             1               CHI  16          35               16         5            36        75
    40            84          98           44
1680307200000000000 x64  ap-northeast-1a host_321 Ubuntu16.10    8    ap-northeast-1 6       production          1               SF   6           89               27         20           76        71
    63            72          17           37
1680307200000000000 x64  ap-northeast-1a host_326 Ubuntu16.04LTS 16   ap-northeast-1 13      test                1               SF   46          86               67         22           90        62
    2             47          27           62
1680307200000000000 x64  ap-northeast-1a host_345 Ubuntu16.04LTS 73   ap-northeast-1 3       test                0               SF   36          57               66         21           9         48
    84            17          69           25
1680307200000000000 x64  ap-northeast-1a host_379 Ubuntu15.10    10   ap-northeast-1 17      production          0               NYC  55          61               29         47           23        8
    62            90          97           36
1680307200000000000 x64  ap-northeast-1a host_382 Ubuntu15.10    66   ap-northeast-1 19      staging             0               SF   50          43               75         93           99        99
    76            68          3            83
1680307200000000000 x64  ap-northeast-1a host_387 Ubuntu16.10    47   ap-northeast-1 11      test                1               SF   46          30               31         18           83        37
    87            89          47           34
1680307200000000000 x64  ap-northeast-1a host_392 Ubuntu15.10    16   ap-northeast-1 0       staging             0               CHI  30          19               30         80           65        29
    74            39          90           66
1680307200000000000 x64  ap-northeast-1a host_433 Ubuntu15.10    35   ap-northeast-1 1       test                0               SF   21          50               31         18           46        0
    9             28          74           89
1680307200000000000 x64  ap-northeast-1a host_502 Ubuntu16.10    78   ap-northeast-1 9       production          1               CHI  68          96               8          59           14        27
  _ 29            23          12           36
```

InfluxDB

- InfluxDB is built around the concept of time-stamped data, which makes it easy to store and query time series data.
- InfluxDB includes features such as retention policies, which allow you to automatically expire old data, and continuous queries, which allow you to pre-aggregate data for faster queries.
- The TSM Tree storage engine of InfluxDB helps in storing the data by data points in a chronological order.
- This data is organized in sorted key value pairs with each each pair corresponds to a specific timestamp. Depending on the query, the data can be retrieved for the time range by looping over the key-value points.

InfluxDB

- But, InfluxDB does not allow joins, which makes queries using joins perform worse as the code has to be made without Joins to get the desired result.
- Also, InfluxDB works only with recent data (depending on the retention rate) and so grouping can be done with recent time only,which makes it slow for groubpy queries.
- I used InfluxDB version 1 for this benchmarking as that's the only supported one, which may have impacted the results as the later version have optimizations.
- While InfluxDB read performance is the best amongst the databases, its performance is bad for query performance.
- InfluxDB performs worse for the groupby-orderby queries and that is because of not using joins, so it has to take a longer route to loop the data and get the results.
- It also took InfluxDB more time to find the latest reading of the data as it performed worse than TimeScaleDB and QuestDB.

QuestDB

- Like TimeScaleDB, QuestDB also uses SQL and other operations that are specifically designed for aggregation and querying time series data.
- Questdb has special functions for time series data :
  ASOF JOIN: Allows you to join two tables based on a specific timestamp.
  SAMPLE BY: Allows you to aggregate data by time intervals.
  LATEST ON: Allows you to get the most recent value for a given timestamp.
  AGGREGATES: Supports aggregates like MIN, MAX, AVG, and SUM.
- Another important advantage of QuestDB is that the data is stored in column and not rows and fields like timestamp are accessed by column, so this makes querying fast.
- Questdb partitions data by time, allowing efficient querying.

QuestDB

- QuestDB outperforms other Databases because of its special features. ‘Latest On’ operator lets Quest DB win in the last point query .
- QuestDB wins by a big ratio in other group by queries ,showing how the storage and partitioning system with other features designed specially for data series data makes QuestDb a better choice for time series data.

Conclusion

- All four databases have their own advantages and disadvantages and so choosing the right database for time series data depends on factors like data size ,simplicity,time and resources.
- While MongoDb is the worst performant ,it is a good choice if the developers have expertise in NoSQL database and familiar with MongoDB.
- InfluxDB is not open source ,so for multi node requirements it can be little costly. Also, it will take time to learn InfluxDB as it's not a simple database.
- TimeScaleDB is a optimal choice as its support for SQL and have better read and write performance as seen in this benchmarking
- Questdb also uses SQL and other special operators and is specially designed for time series data making it a optimal choice based on the the results of this benchmark and other databases used in performance evaluation.
- Comparing NoSQL database i.e. MongoDB with specific time series databases, we can conclude that its a good choice to use time series database depending on the ease of use and functionality than MongoDB.
- But ,we the application is already using MongoDB for non-time series storage ,then it can be a good option to use MongoDB for time-series data and so the developers don't have to spent time learning new database and storing data in it .
- Also , if the developers are proficient in SQl queries ,then its better to use time series data as they can run SQL series as we can see from the database and this benchmarking .