Analysis and Design of Information Systems

Timeseries Databases : CrateDB and QuestDB

Github link : here - Project in LaTeX: here

Team17
Dimitrios-David Gerokonstantis (03119209)
Athanasios Tsoukleidis-Karydakis (03119009)
Filippos Sevastakis (03119183)

National Technical University of Athens
School of Electrical and Computer Engineering

Fall 2023

Table of Contents

Table of Contents

Purpose

A comparative study of two time-series database systems:

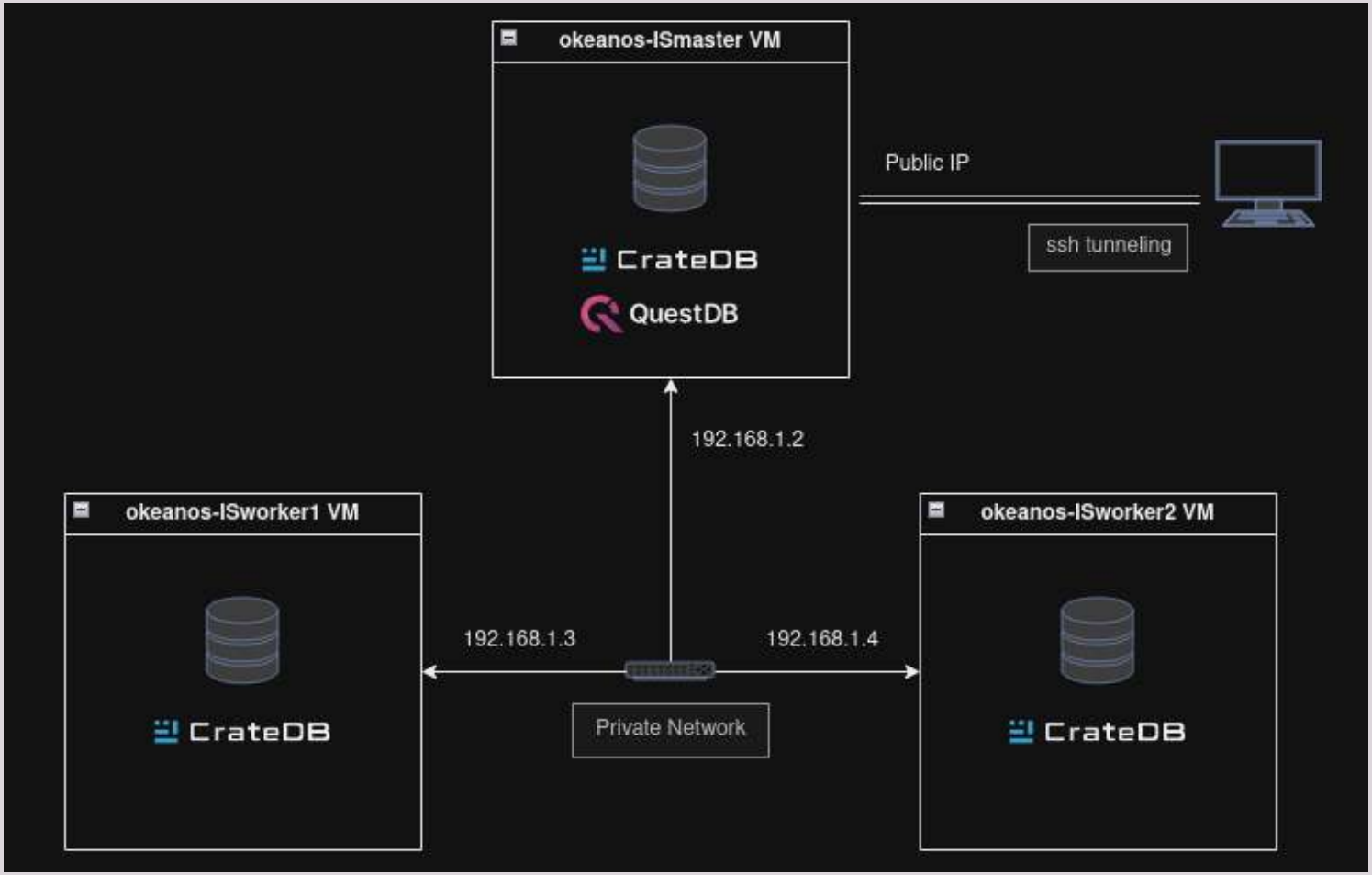
- QuestDB
- CrateDB

Taking into consideration their characteristics, the purpose is to study how these characteristics can affect the performance of the two DBs in two cases:

- Data Ingestion under different data load and DB configurations
- Query Execution under different DB configurations
  1. Query batches
  2. Single queries

Table of Contents

Cluster Overview



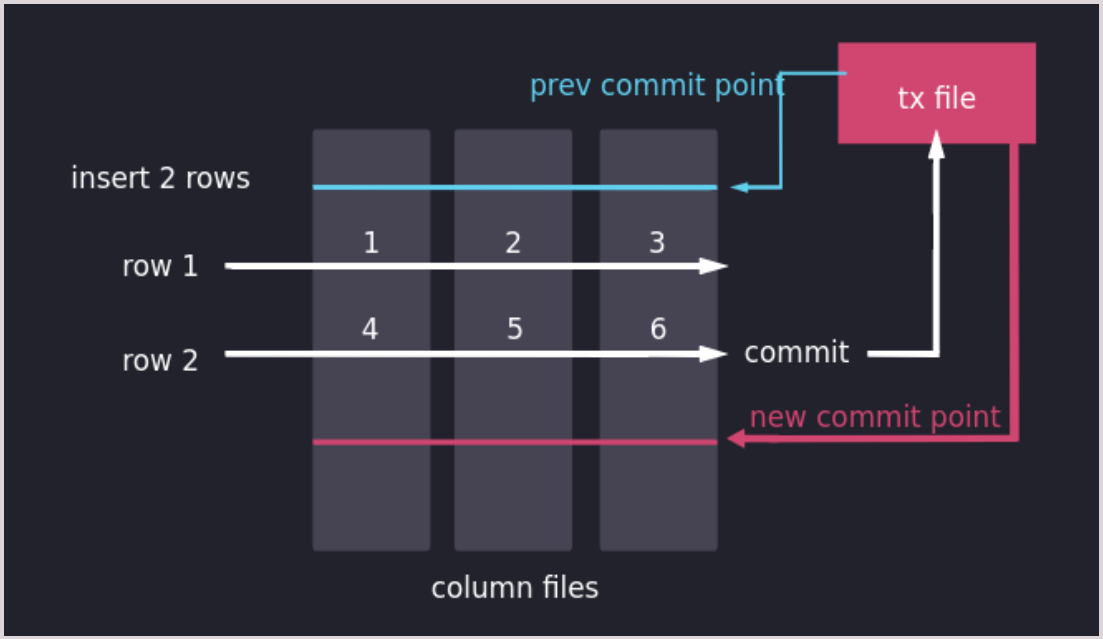
QuestDB Overview

Single-node Database specialized in time-series data

- SQL interface and useful SQL extensions
- High performance in data ingestion from multiple sources (WAL)
- Columnar Storage
- Supports time-based partitions
- Indexing (limited)
- Interval Scan : effective execution of range queries

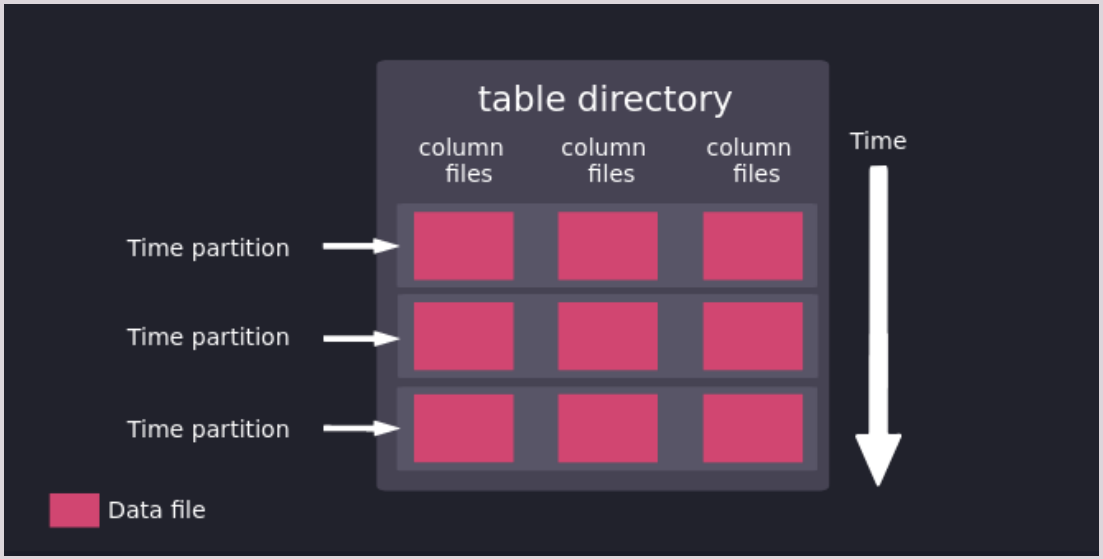
Storage architecture

- Columnar Storage : useful for aggregations
- Column files and append model
- ACID properties maintained
- No performance degradation with high cardinality



Partitioning

- By : HOUR, DAY, WEEK, MONTH, YEAR
- Reduces disk I/O and search time
- Enables handling of out-of-order insertions reducing Write Amplification (partition split and squash)

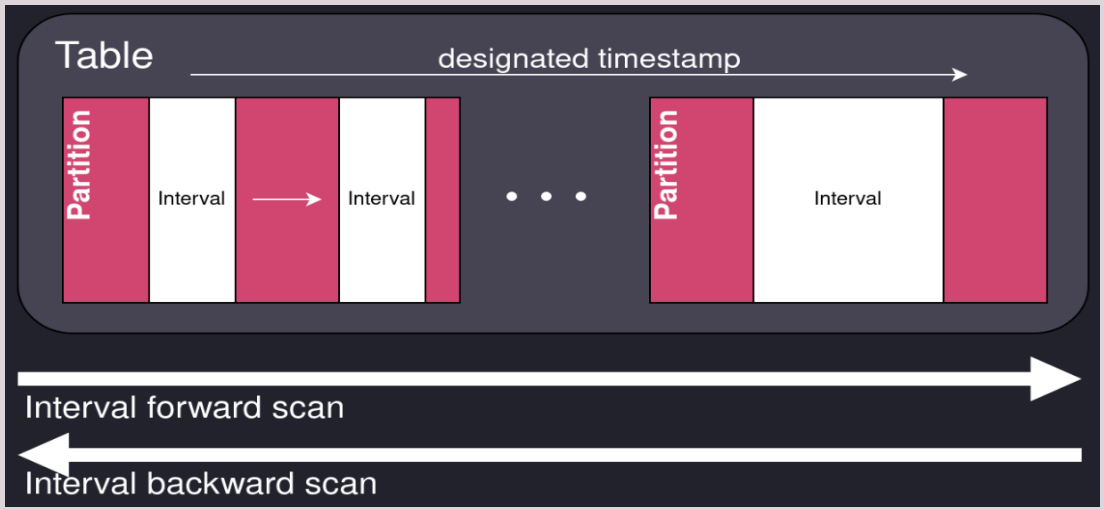


Indexing - Interval Scan

Supports Indexing only for SYMBOL data type

Interval Scan : An effective way to search for the time intervals of a range query

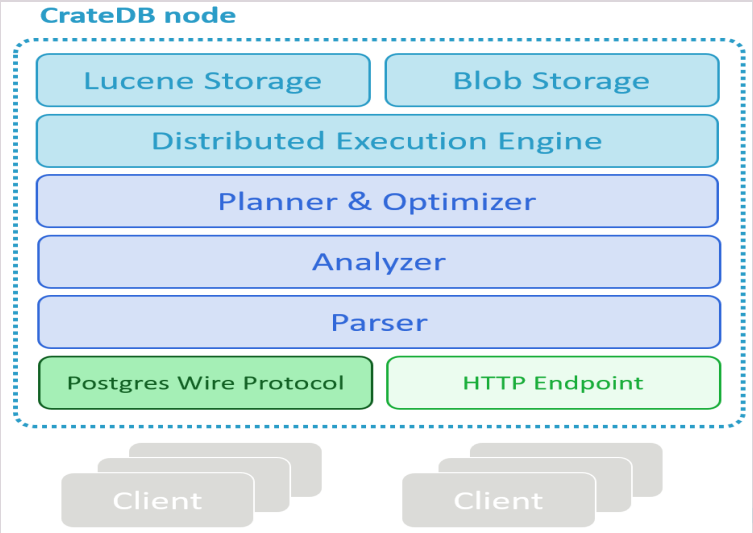
- Define time condition, identify time intervals, find boundaries of the intervals with binary search, scan through the discovered intervals



CrateDB Overview

Distributed Database specialized in time-series data (no spof)

- SQL interface and NoSQL characteristics
- High scalability, columnar storage
- Apache Lucene Engine : Indexes all columns
- High availability :
  1. Replication : self-healing process
  2. Shared-nothing architecture : built-in-load balancing

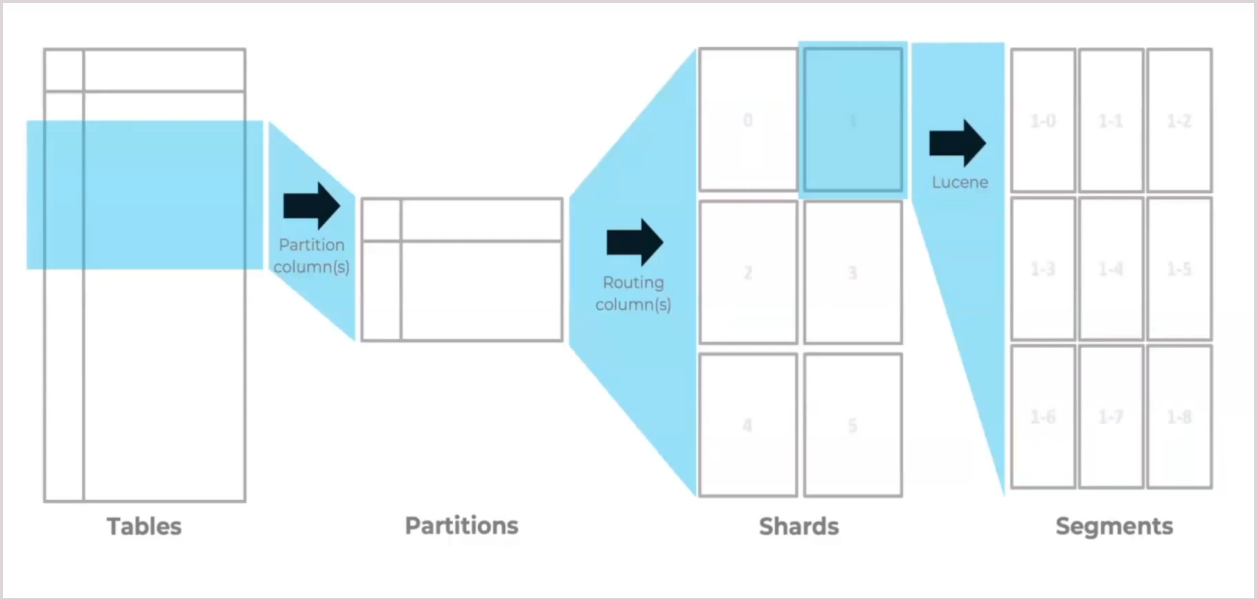


Dynamic Schema - NoSQL characteristics

- Handles all types of data [(un)structured, semi-structured]
- Stores JSON files : OBJECT datatype
  1. Arbitrary number of attributes and nesting levels
  2. Flexibility in updating the Schema (DYNAMIC)
- On-the-fly update of a table's structure

Shards - Partitions - Segments I

- Allows data distribution accross CrateDB nodes
- Sharding and partitioning benefit query execution (especially highly parallelizable queries)
- Append-Only Lucene Segments
- Optimizer : Merges Lucene Segments



Shards - Partitions - Segments II

- Supports partitioning by any column (not only time-based like QuestDB)
- Lucene Index into each shard's segments
- Avoid too many or too little shards/partitions!

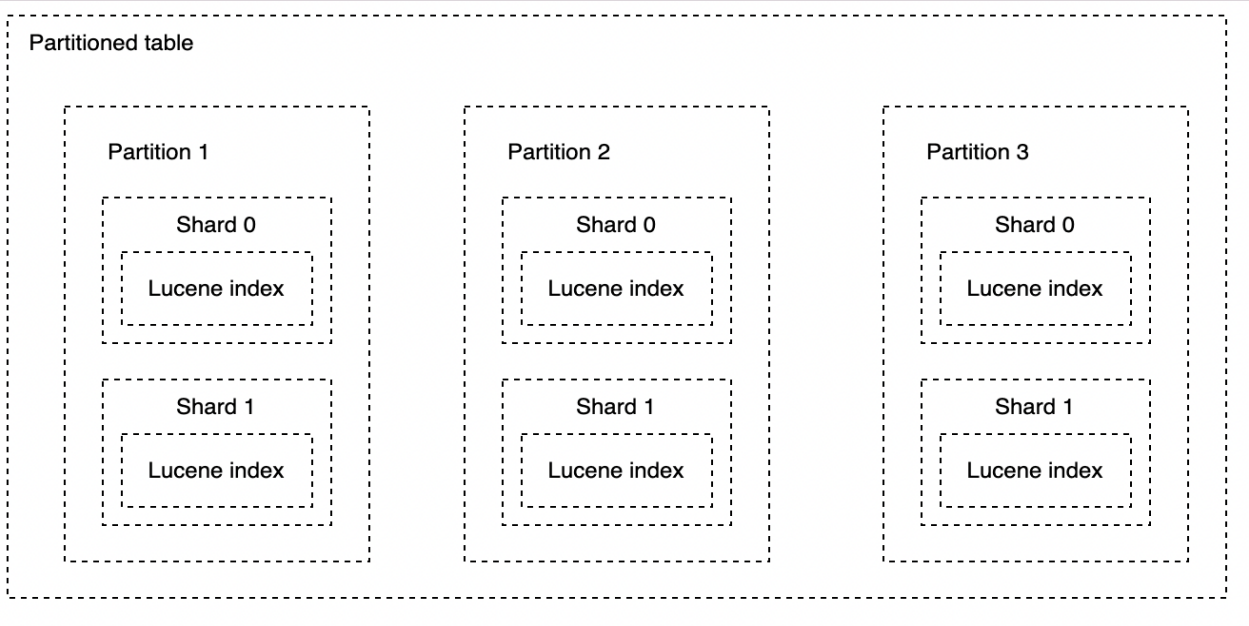


Table of Contents

Methodology I (Data Ingestion)

Use of TSBS in order to load three datasets :

- Small Dataset : 7776000 rows (5 days)
- Medium Dataset : 48211200 rows (1 month)
- Big Dataset : 93312000 rows (2 months)

Comparisons :

1. CrateDB (5 shards/without partition) vs QuestDB
2. CrateDB(3, 6, 9 shards with/without partition)

Steps:

- Generation of Data

```
./tsbs_generate_data --use-case="devops" --seed=666\
--scale=20 \
--timestamp-start="2016-01-01T00:00:00Z" \
--timestamp-end="2016-03-01T00:00:00Z" \
--log-interval="10s" --format="questdb" \
| gzip > /tmp/questdbBIG-data.gz
```

Methodology I (Data Ingestion)

Use of TSBS in order to load three datasets :

- Small Dataset : 7776000 rows (5 days)
- Medium Dataset : 48211200 rows (1 month)
- Big Dataset : 93312000 rows (2 months)

Comparisons :

1. CrateDB (5 shards/without partition) vs QuestDB
2. CrateDB(3, 6, 9 shards with/without partition)

Steps:

- Generation of Data
- Loading of Data

```
cat /tmp/questdbBIG-data.gz | gunzip | \
./tsbs_load_questdb \ --workers=1
```

Methodology II (Query Execution)

Stages in query execution study :

- Four different batches of 100 similar TSBS queries

```
./tsbs_generate_queries --use-case="devops" \
--seed=666 --scale=20 \
--timestamp-start="2016-01-01T00:00:00Z"\
--timestamp-end="2016-03-01T00:00:01Z" \
--queries=100 --query-type="double-groupby-5" \
--format="cratedb" \
|gzip > /tmp/cratedb100-queries-double-groupby-5.gz
```

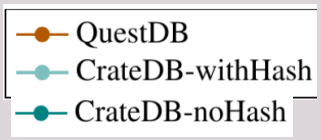
```
cat /tmp/cratedb100-queries-double-groupby-5.gz | \
> gunzip | ./tsbs_run_queries_cratedb --workers=12
```

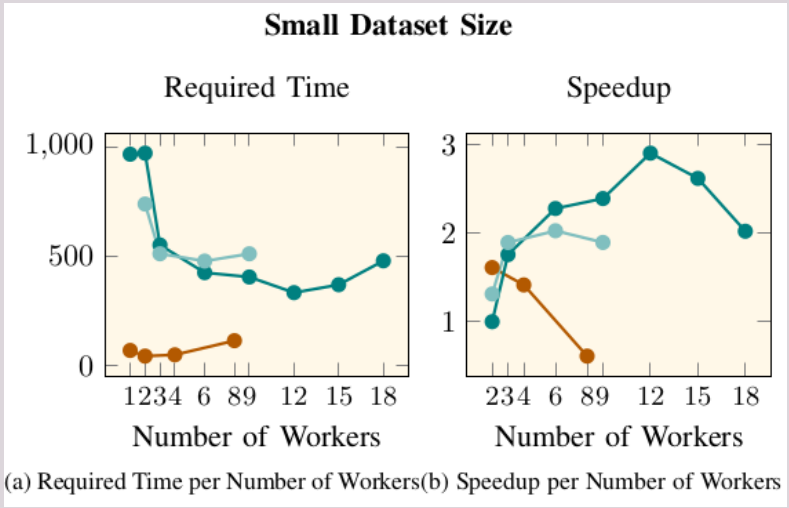
- Single custom queries - Query scenarios

Comparisons (using multiple TSBS workers, i.e. parallel requests for query execution):

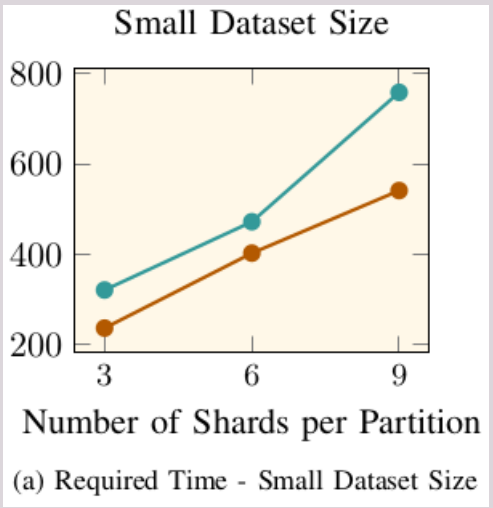
- QuestDB vs CrateDB(3, 6, 9 shards with/without partition)
- QuestDB vs Best case of CrateDB

Data Ingestion - Small Dataset

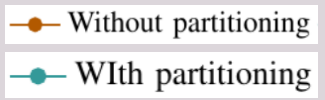




(a) Required Time per Number of Workers(b) Speedup per Number of Workers

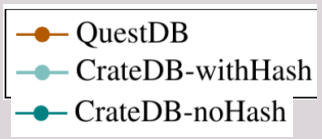


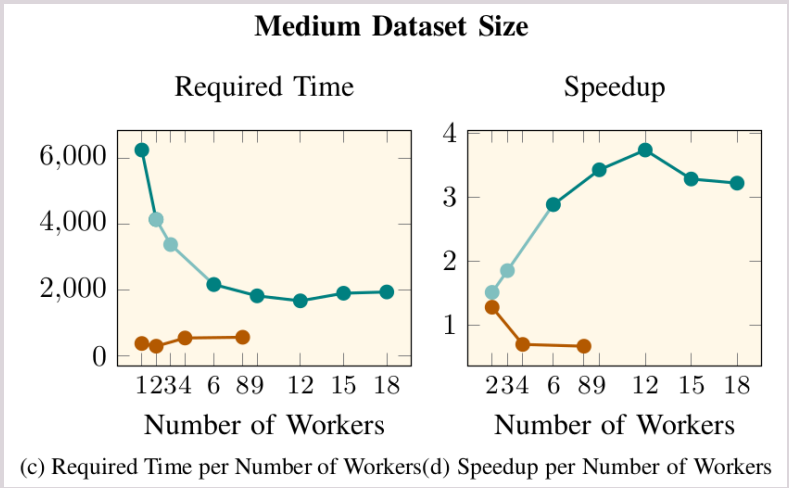
(a) Required Time - Small Dataset Size



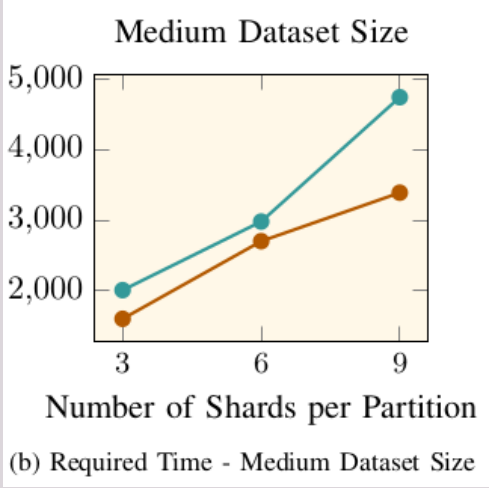
- Hash-workers good for up to 3 workers
- QuestDB: bad scaling, better performance
- CrateDB: good scaling, worse performance
- Higher ingestion time with partitioning enabled
- Ingestion time proportional to dataset size
- Ingestion time not proportional to number of shards

Data Ingestion - Medium Dataset

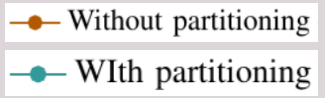




(c) Required Time per Number of Workers (d) Speedup per Number of Workers

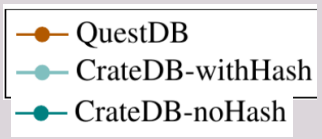


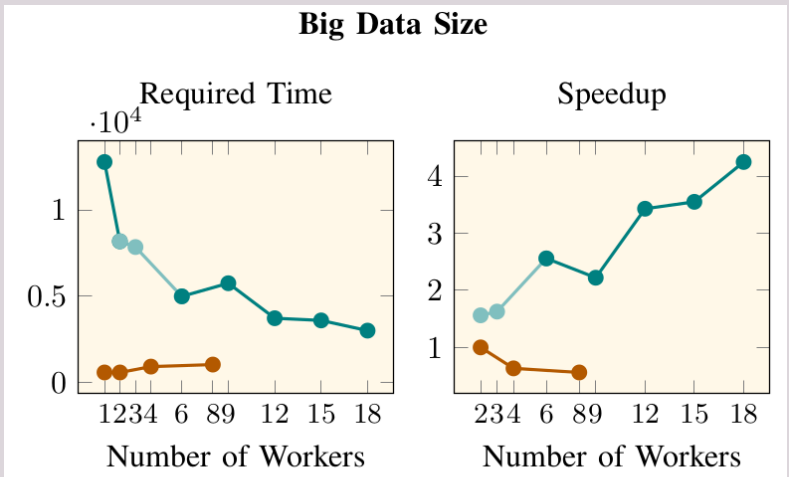
(b) Required Time - Medium Dataset Size



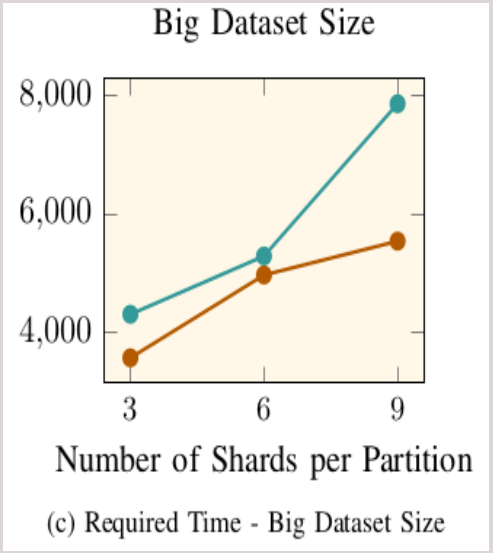
- Hash-workers good for up to 3 workers
- QuestDB: bad scaling, better performance
- CrateDB: good scaling, worse performance
- Higher ingestion time with partitioning enabled
- Ingestion time proportional to dataset size
- Ingestion time not proportional to number of shards

Data Ingestion - Big Dataset





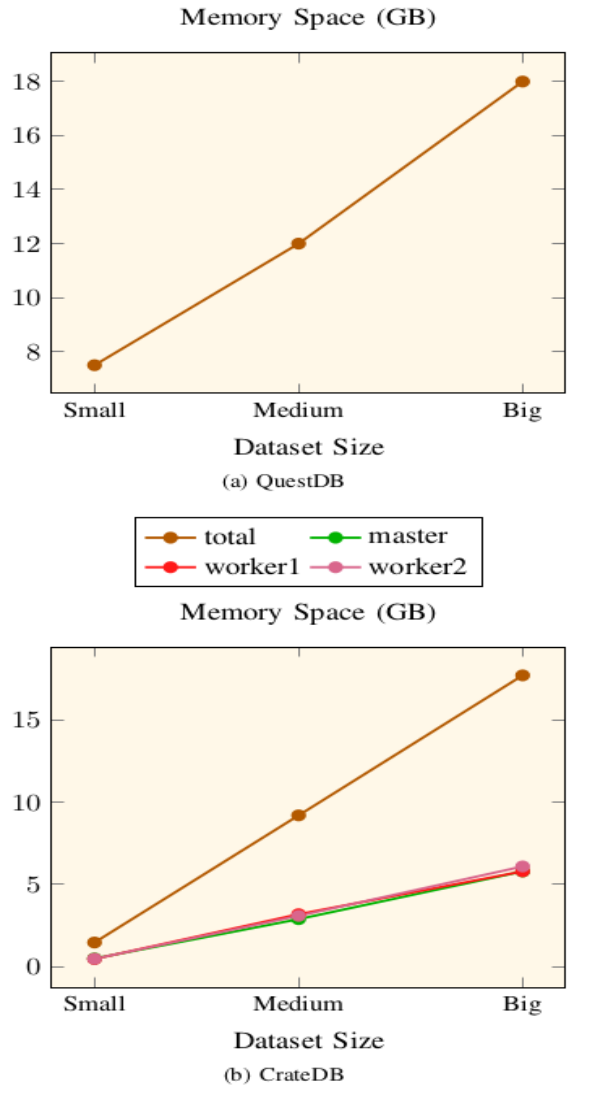
(e) Required Time per Number of Workers (f) Speedup per Number of Workers



(c) Required Time - Big Dataset Size

- Hash-workers good for up to 3 workers
- QuestDB: bad scaling, better performance
- CrateDB: good scaling, worse performance
- Higher ingestion time with partitioning enabled
- Ingestion time proportional to dataset size
- Ingestion time not proportional to number of shards

Memory Space Consumption



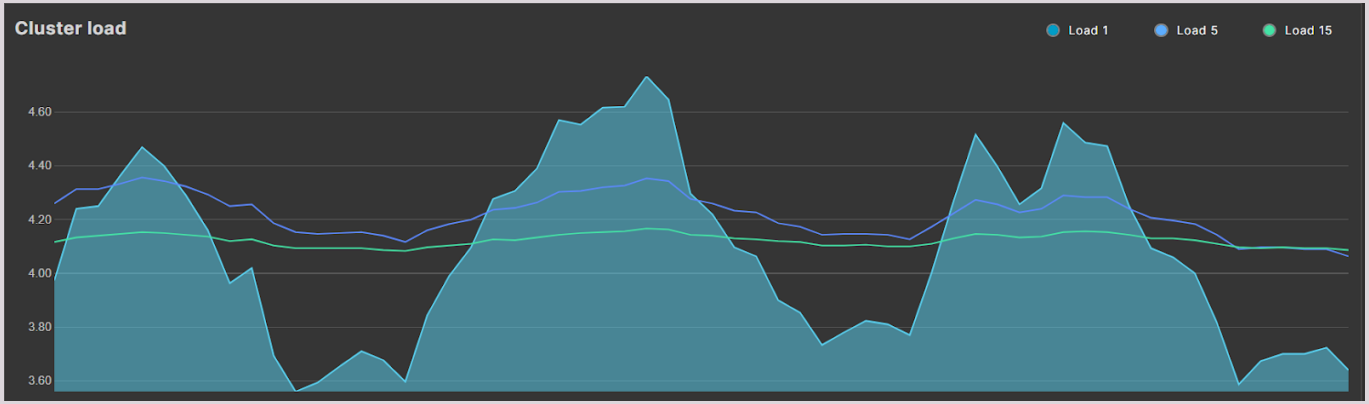
QuestDB :

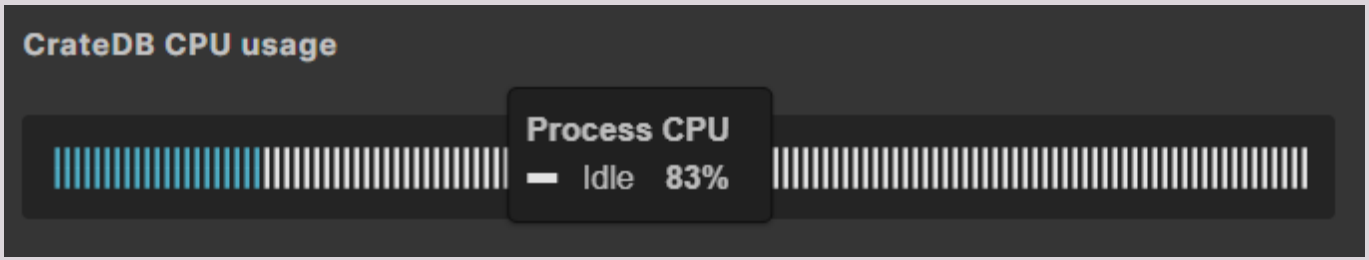
- Compression enabled only for bigger datasets

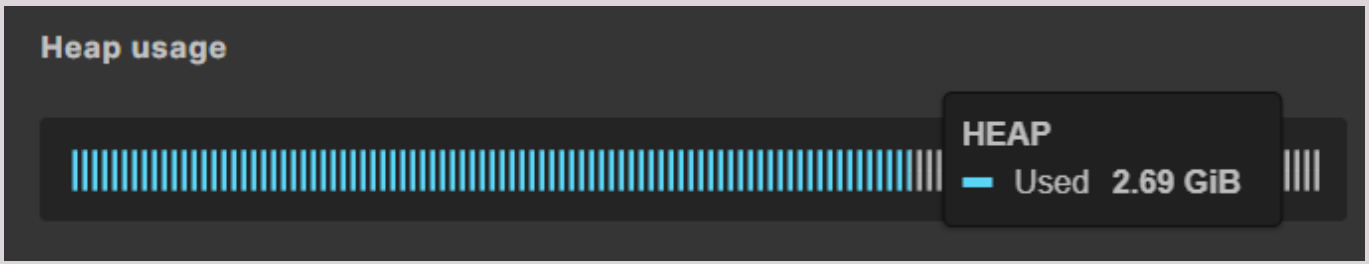
CrateDB :

- Memory consumption proportional to number of rows
- Compression independent of number of rows
- Load balanced between CrateDB nodes

CrateDB : System Load

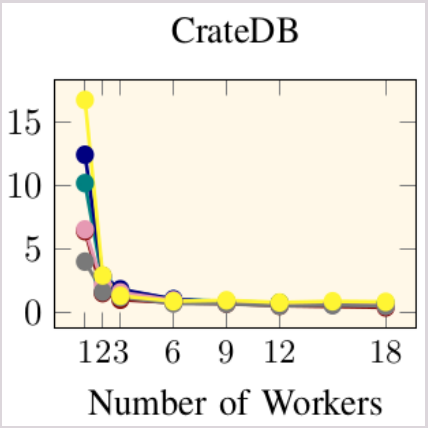


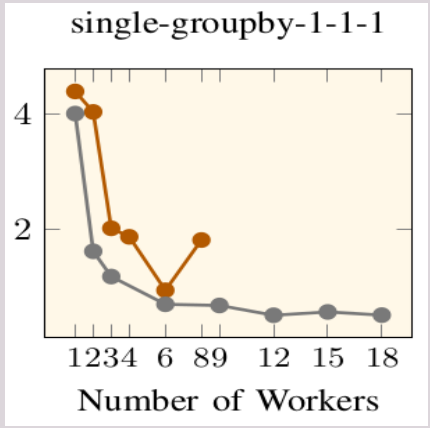


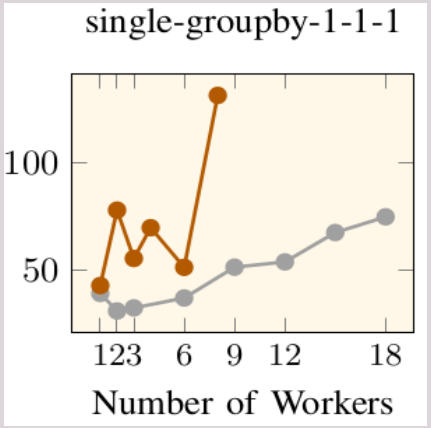


Query Batches Execution

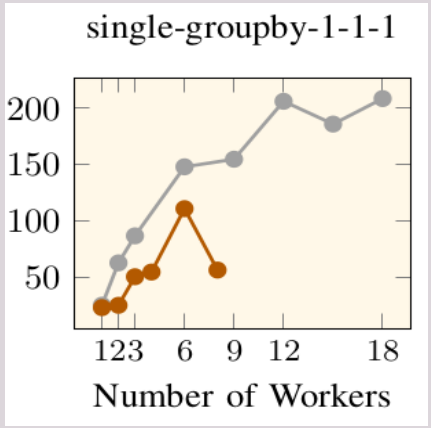
single-groupby-1-1-1



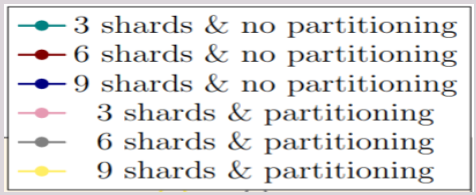




Query Latency



Throughput



Best Case of Crate DB vs QuestDB

CrateDB Best Case : 6 shards with partitioning

- CrateDB

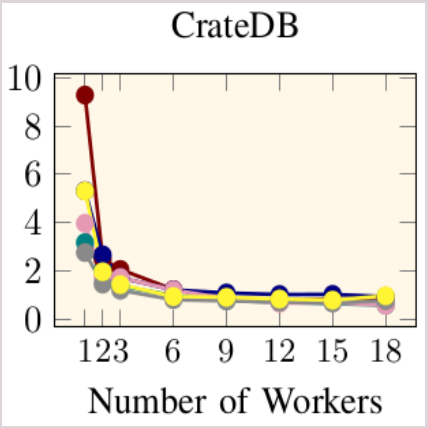
```
SELECT
    date_trunc('minute', ts) as minute,
    max(usage_user) AS max_usage_user
FROM
    cpu
WHERE
     tags['hostname'] IN ('host_1')
    AND ts >= 1453521384428
    AND ts < 1453524984428
GROUP BY minute
ORDER BY minute ASC
```

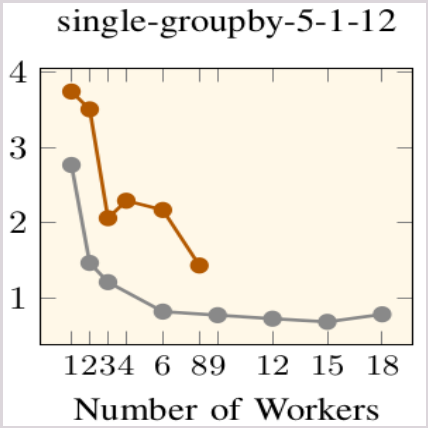
- QuestDB

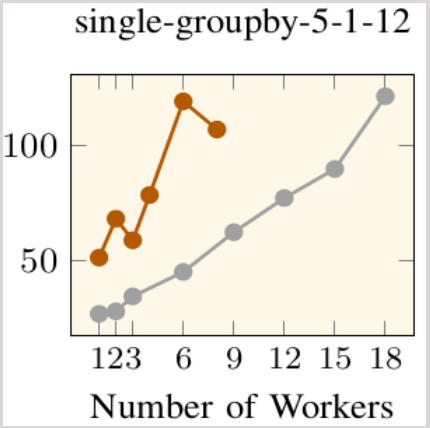
```
SELECT
    timestamp,
    max(usage_user) AS max_usage_user
FROM
    cpu
WHERE
    hostname IN ('host_1')
    AND timestamp >= '2016-01-23T03:56:24Z'
    AND timestamp < '2016-01-23T04:56:24Z'
SAMPLE BY 1m
```

Query Batches Execution

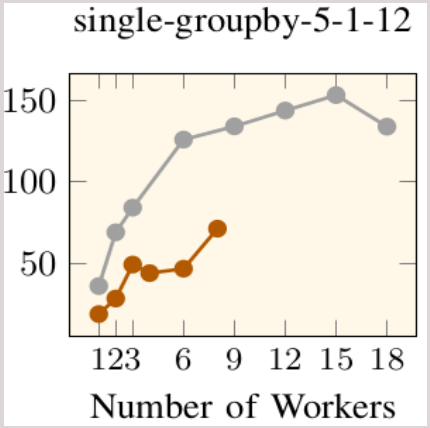
single-groupby-5-1-12







Query Latency



Throughput

Best Case of Crate DB vs QuestDB

CrateDB Best Case : 6 shards with partitioning

- CrateDB

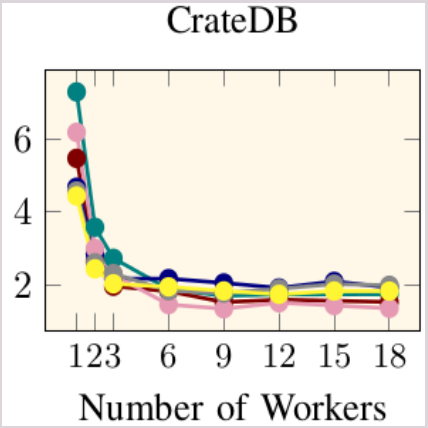
```
SELECT
    date_trunc('minute', ts) as minute,
    max(usage_user) AS max_usage_user,
    max(usage_system) AS max_usage_system,
    max(usage_idle) AS max_usage_idle,
    max(usage_nice) AS max_usage_nice,
    max(usage_iowait) AS max_usage_iowait
FROM
    cpu
WHERE
    tags['hostname'] IN ('host_1')
    AND ts >= 1453028181428
    AND ts < 1453071381428
GROUP BY minute
ORDER BY minute ASC
```

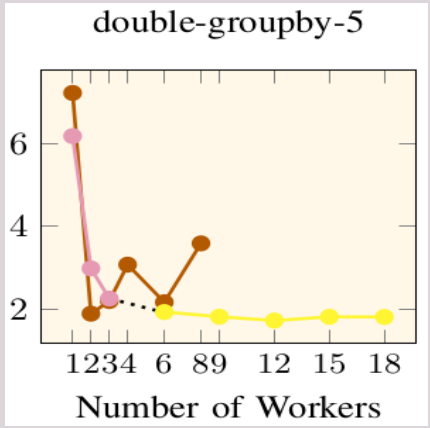
- QuestDB

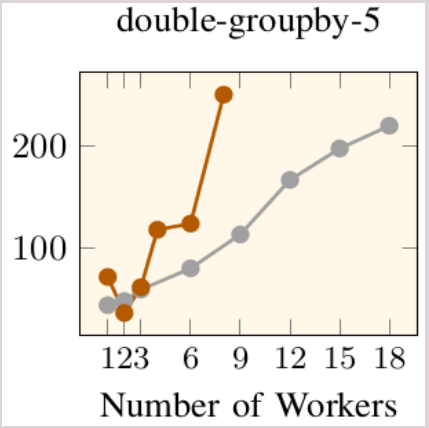
```
SELECT
    timestamp,
    max(usage_user) AS max_usage_user,
    max(usage_system) AS max_usage_system,
    max(usage_idle) AS max_usage_idle,
    max(usage_nice) AS max_usage_nice,
    max(usage_iowait) AS max_usage_iowait
FROM
    cpu
WHERE
    hostname IN ('host_1')
    AND timestamp >= '2016-01-17T10:56:21Z'
    AND timestamp < '2016-01-17T22:56:21Z'
SAMPLE BY 1m
```

Query Batches Execution

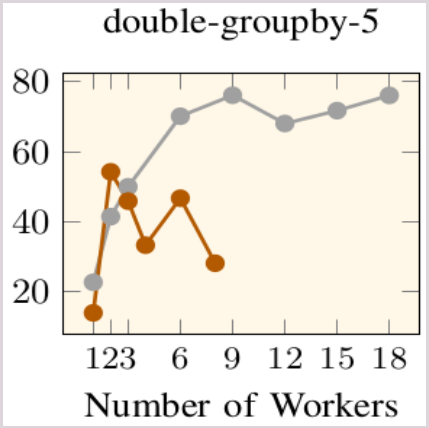
double-groupby-5



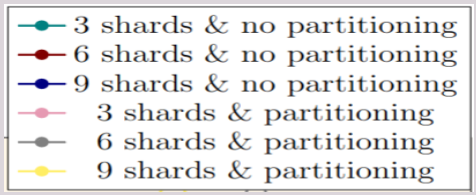




Query Latency



Throughput



Best Case of Crate DB vs QuestDB

CrateDB Best Case : 9 shards with partitioning (up to 3 workers)
3 shards with partitioning ($>$ 3 workers)

- CrateDB

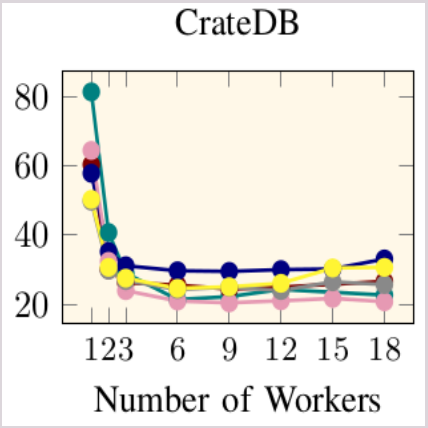
```
SELECT
    date_trunc('hour', ts) AS hour,
    mean(usage_user) AS mean_usage_user,
    mean(usage_system) AS mean_usage_system
        ,
    mean(usage_idle) AS mean_usage_idle,
    mean(usage_nice) AS mean_usage_nice,
    mean(usage_iowait) AS mean_usage_iowait
FROM
    cpu
WHERE
    ts >= 1455759268856
    AND ts < 1455802468856
GROUP BY hour, tags['hostname']
ORDER BY hour
```

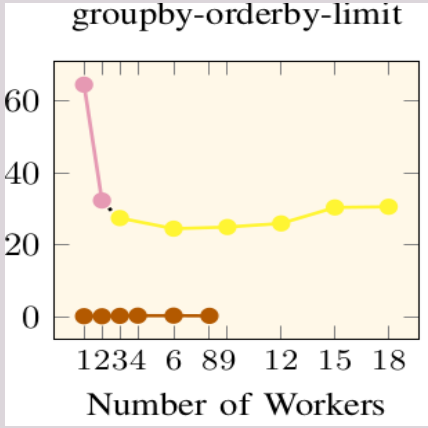
- QuestDB

```
SELECT
    timestamp,
    hostname,
    avg(usage_user) AS avg_usage_user,
    avg(usage_system) AS avg_usage_system,
     avg(usage_idle) AS avg_usage_idle,
     avg(usage_nice) AS avg_usage_nice,
     avg(usage_iowait) AS avg_usage_iowait
FROM
    cpu
WHERE
    timestamp >= '2016-02-18T01:34:28Z'
    AND timestamp < '2016-02-18T13:34:28Z'
SAMPLE BY 1h
GROUP BY timestamp, hostname
```

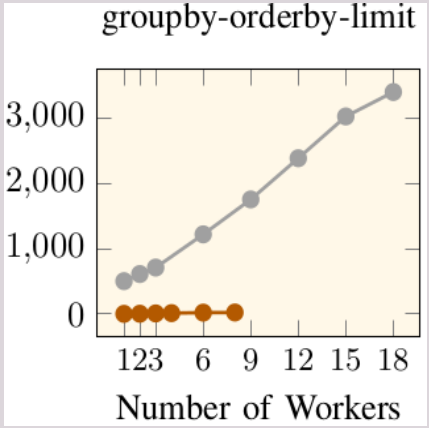
Query Batches Execution

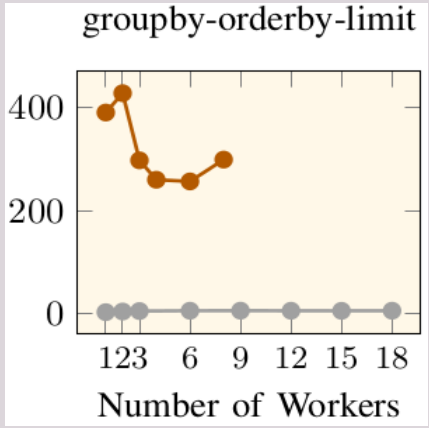
groupby-orderby-limit





Query Latency





Throughput

Best Case of Crate DB vs QuestDB

CrateDB Best Case : 9 shards with partitioning (up to 2 workers)
3 shards with partitioning ($>$ 2 workers)

- CrateDB

```
SELECT
    date_trunc('minute', ts) as minute,
    max(usage_user)
FROM
    cpu
WHERE ts < 1453635272856
GROUP BY minute
ORDER BY minute DESC
LIMIT 5
```

- QuestDB

```
SELECT
    timestamp AS minute,
    max(usage_user)
FROM cpu
WHERE timestamp < '2016-01-24T11:34:32Z'
SAMPLE BY 1m
LIMIT 5
```

Observations and Comments I

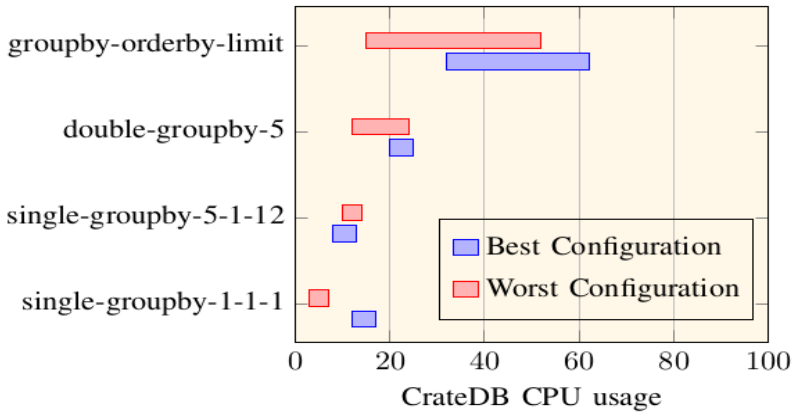
1. single-groupby-1-1-1 & single-groupby-5-1-12 :
   - QuestDB worse than CrateDB
   - CrateDB partitioning is beneficial
   - For one worker (single requests), CrateDB is worse due to distributed system's overheads (i.e. network latencies)
2. double-groupby-5 :
   - QuestDB worse than CrateDB
   - For small number of workers, CrateDB is more efficient when having more shards.
   - For large number of workers, CrateDB is more efficient when having less shards.

Observations and Comments II

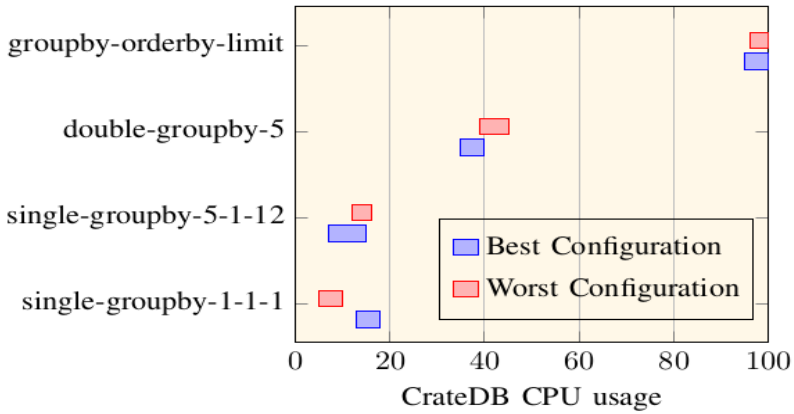
3. groupby-orderby-limit :
   - QuestDB is much better than CrateDB
   - Positive effect of QuestDB caching (overlapping time intervals between different requests)
   - Positive effect of SAMPLE BY (SQL extension)

Finally: CrateDB has better performance while executing (in parallel) batches of queries. QuestDB is less capable dealing with concurrent requests, but it is just a little worse than CrateDB when executing the queries above atomically.

CrateDB CPU usage



Εικόνα 24: CrateDB CPU usage intervals (%) for Master and Worker1 nodes and for the best and worst configuartion scenarios (as far as the execution time concerned)

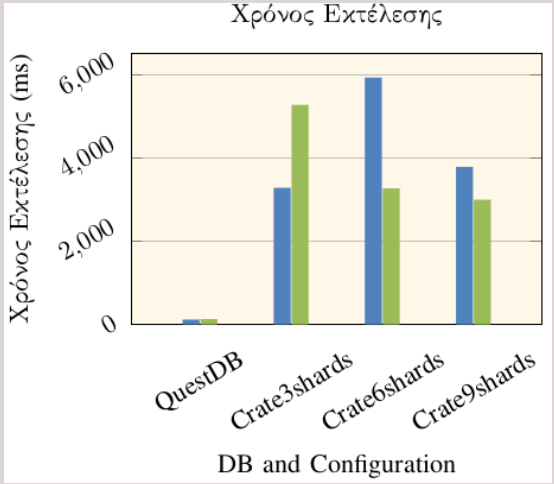


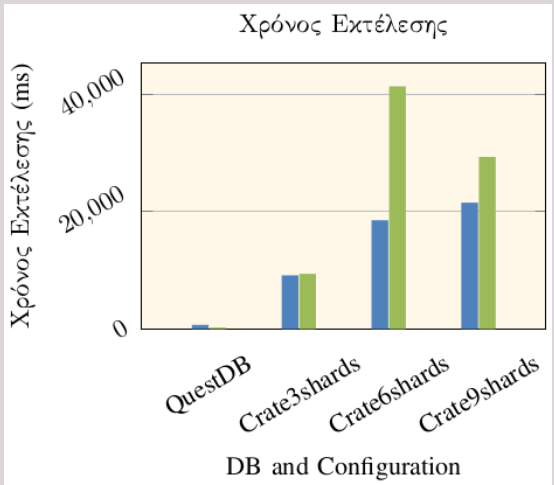
Εικόνα 25: CrateDB CPU usage intervals (%) for Worker2 node and for the best and worst configuartion scenarios (as far as the execution time concerned)

- In most cases, the configurations related to less execution time (best ones) lead to higher CPU usage. Usually, worst configurations lead to CPU underutilization.
- The query groupby-orderby-limit creates a bottleneck in second worker's CPU.

Single Custom Queries

1st and 2nd Query



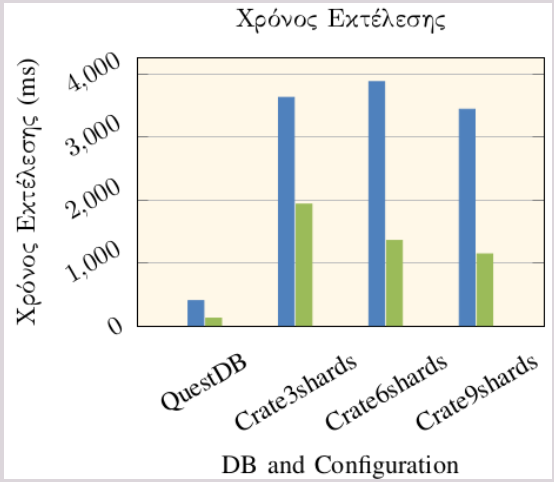



```
//QuestDB:
SELECT * FROM nginx LIMIT 100000

//CrateDB:
SELECT * FROM benchmark.nginx
LIMIT 100000
```

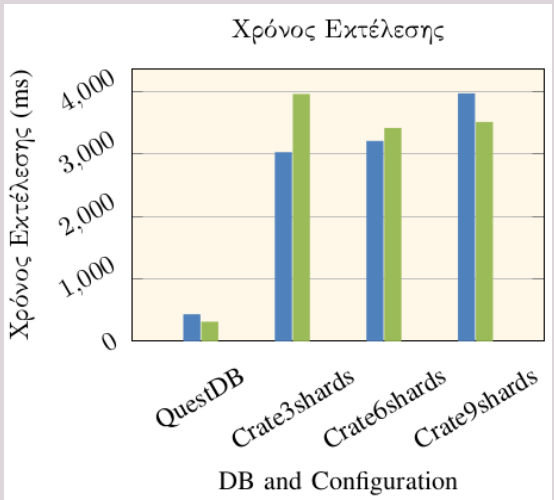
```
//QuestDB:
SELECT
    c.usage_user usage_user,
    d.used_percent
FROM
    cpu as c
ASOF JOIN disk as d;

//CrateDB:
SELECT
    c.usage_user usage_user,
    a.used_percent
FROM
    (SELECT * FROM benchmark.cpu LIMIT 10000)
        as c
JOIN (
    SELECT
        ts,
        tags['hostname'],
        used_percent
    FROM
    (SELECT * FROM benchmark.disk LIMIT 10000)
        as d
) a
ON
    c.ts >= a.ts
    and
    c.tags['hostname']=a.tags['hostname']
ORDER BY
    c.ts, a.ts DESC;
```

Single Custom Queries

3rd and 4th Query



Without Partitioning With Partitioning

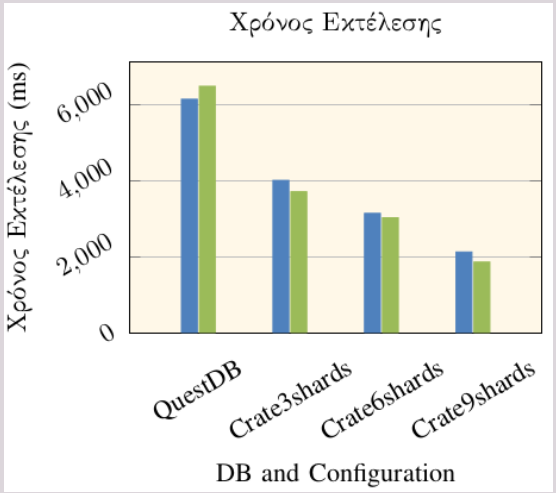



```
//QuestDB:
SELECT
    hostname,
    timestamp,
    usage_user
FROM
    cpu
WHERE
    timestamp in '2016-02'

//CrateDB:
SELECT
    tags['hostname'],
    ts,
    usage_user
FROM
    benchmark.cpu
WHERE
    ts<'2016-03-01T00:00:00.000000Z'
    and
    ts>='2016-02-01T00:00:00.000000Z'
limit 100000;
```

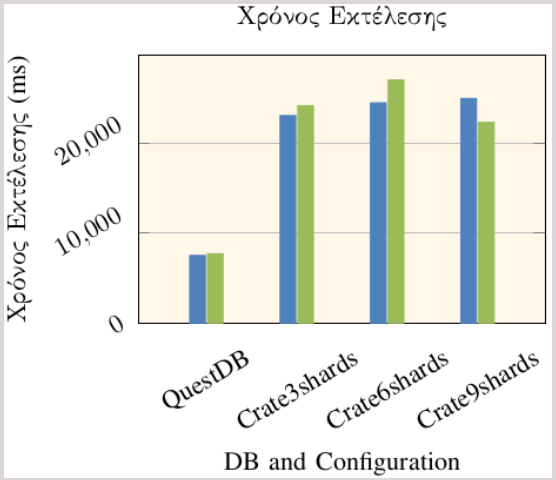
```
//QuestDB:
SELECT
    hostname,
    timestamp,
    usage_user
FROM
    cpu
WHERE
    timestamp<'2016-02-15T00:00:00.000000Z'
    and
    timestamp>='2016-01-15T00:00:00.000000Z'

//CrateDB:
SELECT
    tags['hostname'],
    ts,
    usage_user
FROM
    benchmark.cpu
WHERE
    ts<'2016-02-15T00:00:00.000000Z'
    and
    ts>='2016-01-15T00:00:00.000000Z'
limit 100000;
```

Single Custom Queries

5th and 6th Query



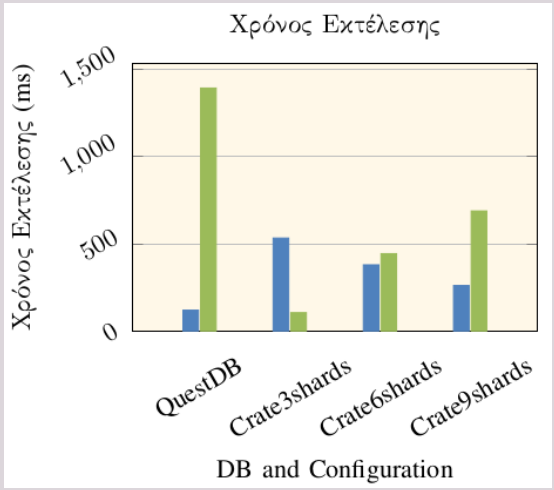
Description : Find the hosts having stable memory usage ($maxValue < 110\% \times minValue$) during a time interval. For this time interval, find the total amount of delivered packets and print them in descending ordering by the maximum memory usage. This query consists of: JOIN, ORDER BY, GROUP BY and time condition in WHERE clause.



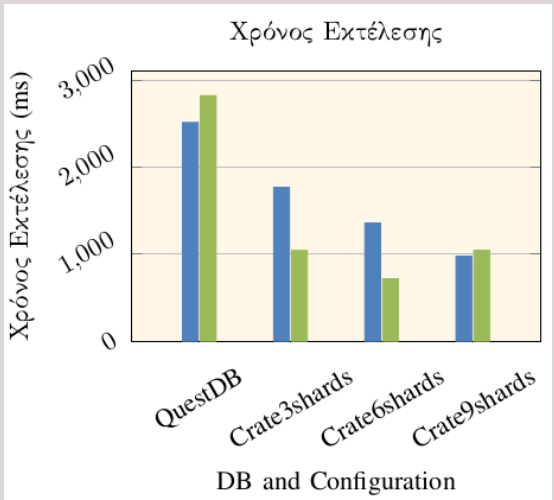
Description : Find the time intervals during which, the memory used_percent of some hosts (that need to be specified as well) was at least ten times greater than the (non zero) cpu usage_user. For these intervals, compute the total amount of context switches that had been encountered until these moments. This query consists of a JOIN operation accross three tables.

Single Custom Queries

7th and 8th Query



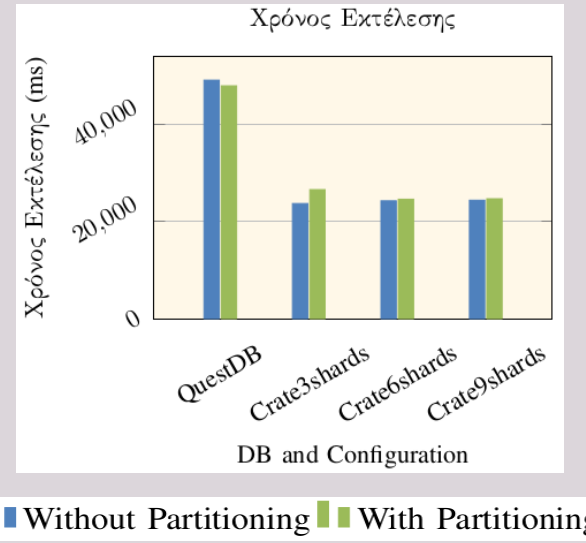

```
//QuestDB:
SELECT hostname, requests
FROM nginx
WHERE timestamp='2016-01-18T17:42:00.000000Z';

//CrateDB:
SELECT tags['hostname'], requests
FROM benchmark.nginx
WHERE ts='2016-01-18T17:42:00.000000Z'
```




```
//QuestDB:
SELECT hostname, max(used_percent) as maxUsedPer
FROM disk
GROUP BY hostname;

//CrateDB:
SELECT tags['hostname'], max(used_percent) as
    maxUsedPer
FROM benchmark.disk
GROUP BY tags['hostname'];
```

Single Custom Queries

9th Query




```
//QuestDB:
SELECT a.hostname, a.used_percent,timestamp,a.
    ranking FROM (SELECT hostname as hostname,
    used_percent,timestamp,ROW_NUMBER() OVER (
    PARTITION BY hostname ORDER BY used_percent DESC
    ) AS ranking FROM mem)AS a WHERE a.ranking<4
    ORDER BY a.hostname,ranking ASC

//CrateDB:
SELECT a.hostname, a.used_percent,ts,a.ranking FROM
    (SELECT tags['hostname'] as hostname,
    used_percent,ts,ROW_NUMBER() OVER (PARTITION BY
    tags['hostname'] ORDER BY used_percent DESC) AS
    ranking FROM benchmark.mem)As a WHERE a.ranking
    <4 ORDER BY a.hostname,ranking ASC
```

Observations and Comments I

1. Query 1:
   - QuestDB better than CrateDB
   - CrateDB benefits from more shards
   - Partitioning does not affect the performance
2. Query 2:
   - QuestDB better than CrateDB due to ASOF JOIN
   - QuestDB benefits from partitioning while CrateDB does not due to its complex implementation of the query.
3. Query 3:
   - QuestDB better than CrateDB thanks to interval scan
   - Both DBs benefit from paritioning since the query deals with data of a particular month (partition).

Observations and Comments II

4. Query 4:
   - QuestDB better than CrateDB thanks to interval scan
   - Partitioning does not affect the performance (data retrieved from different months-partitions)
5. Query 5:
   - CrateDB better than QuestDB due to the existence of highly parallelizable SQL statements (groupings-aggregations)
   - Partitioning does not affect the performance
6. Query 6:
   - QuestDB better than CrateDB due to the nature of JOIN statements (communication between CrateDB nodes)

Observations and Comments III

7. Query 7:
   - CrateDB better that QuestDB due to the support of indexing in every column.
8. Query 8:
   - CrateDB better than QuestDB due to the innate high parallelism of groupings and aggregations
   - Partitioning leads to better performance due to the resulting creation of more shards which increases the parallelism in execution.
9. Query 9:
   - CrateDB better than QuestDB due to the nature of window functions (they contain groupings)

Conclusion

	CRATEDB	QUESTDB
DATA INGESTION		✓
DATA INGESTION SCALABILITY	✓	
STORAGE CAPABILITIES	✓	
COMPRESSION	✓	✓
QUERY BATCHES	✓	
QUERY CACHING		✓
SINGLE QUERIES (IN GENERAL)		✓
INDEXING	✓	
SQL EXTENSIONS		✓
RANGE QUERIES		✓
POINT QUERIES	✓	
AGGREGATIONS/GROUPINGS	✓	
WINDOW FUNCTIONS	✓	
PROGRAMMABILITY	✓	✓