Elettra
Sincrotrone
Trieste

TANGO

# HDB++ benchmark tests @Elettra

Graziano Scalamera

# HDB++ @ Elettra



## Benchmark goals and metrics

| Goal | Question | Metrics |
|---|---|---|
| Measure insertion throughput | What is the max number of events inserted? Comparison SSD vs HDD? Evolution of pending queue in hdbpp-es? | Nb of rows inserted per second per subscriber, latency |
| Measure extraction throughput | Behaviour when extracting devDouble scalar data, devDouble arrays? Comparison SSD vs HDD? | Time in seconds to retrieve the data |
| Measure the disk usage for each supported Tango Data Type | What is the disk usage for a given data set? | Disk usage (before and after insertion) |

# HDB++ @ Elettra

## First Tests
## Hardware and Software

- A (~5 years old) desktop PC with:
  - CPU: Intel Core i7-5820 3.33GHz, 6 core, (12 thread)
  - RAM: 24GB
  - 2 SSD: Kingston 2.5”, 120 GB, SATA 6 Gb/s, EXT4, in raid 1 for OS
  - HDD: WD Black 3.5” 2 TB, 7200 RPM, SATA 6 Gb/s, XFS, for PostgreSQL data
  - HDD: WD Black 3.5” 2 TB, 7200 RPM, SATA 6 Gb/s, XFS, for MySQL data

- Software:
  - Ubuntu 16.04.3 LTS
  - MySQL 5.7.18
  - PostgreSQL 11.3
  - TimescaleDB 1.3.0

# HDB++ @ Elettra

## First Tests
## 1) DUMP

• FERMI HDB++ DB att_scalar_devdouble_ro (**4,082,099,171 rows**), 2 years of data, dumped twice:

- Dump ordered by att_conf_id, data_time (same ordering as Primary Key)
  SELECT **att_conf_id, data_time, recv_time, value_r, quality, att_error_desc_id** INTO OUTFILE 'att_scalar_devdouble_ro.csv' FROM hdbpp.att_scalar_devdouble_ro;

- Dump ordered by data_time (in order to load data as in the real scenario)
- SELECT **att_conf_id, data_time, recv_time, value_r, quality, att_error_desc_id** INTO OUTFILE 'att_scalar_devdouble_ro_p2018_01_02.csv' FROM hdbpp.att_scalar_devdouble_ro PARTITION (p2018_01_02) **ORDER BY data_time ASC**;
  SELECT **att_conf_id, data_time, recv_time, value_r, quality, att_error_desc_id** INTO OUTFILE 'att_scalar_devdouble_ro_p2018_03_04.csv' FROM hdbpp.att_scalar_devdouble_ro PARTITION (p2018_03_04) **ORDER BY data_time ASC**;
  (note: one partition at time)
  ...

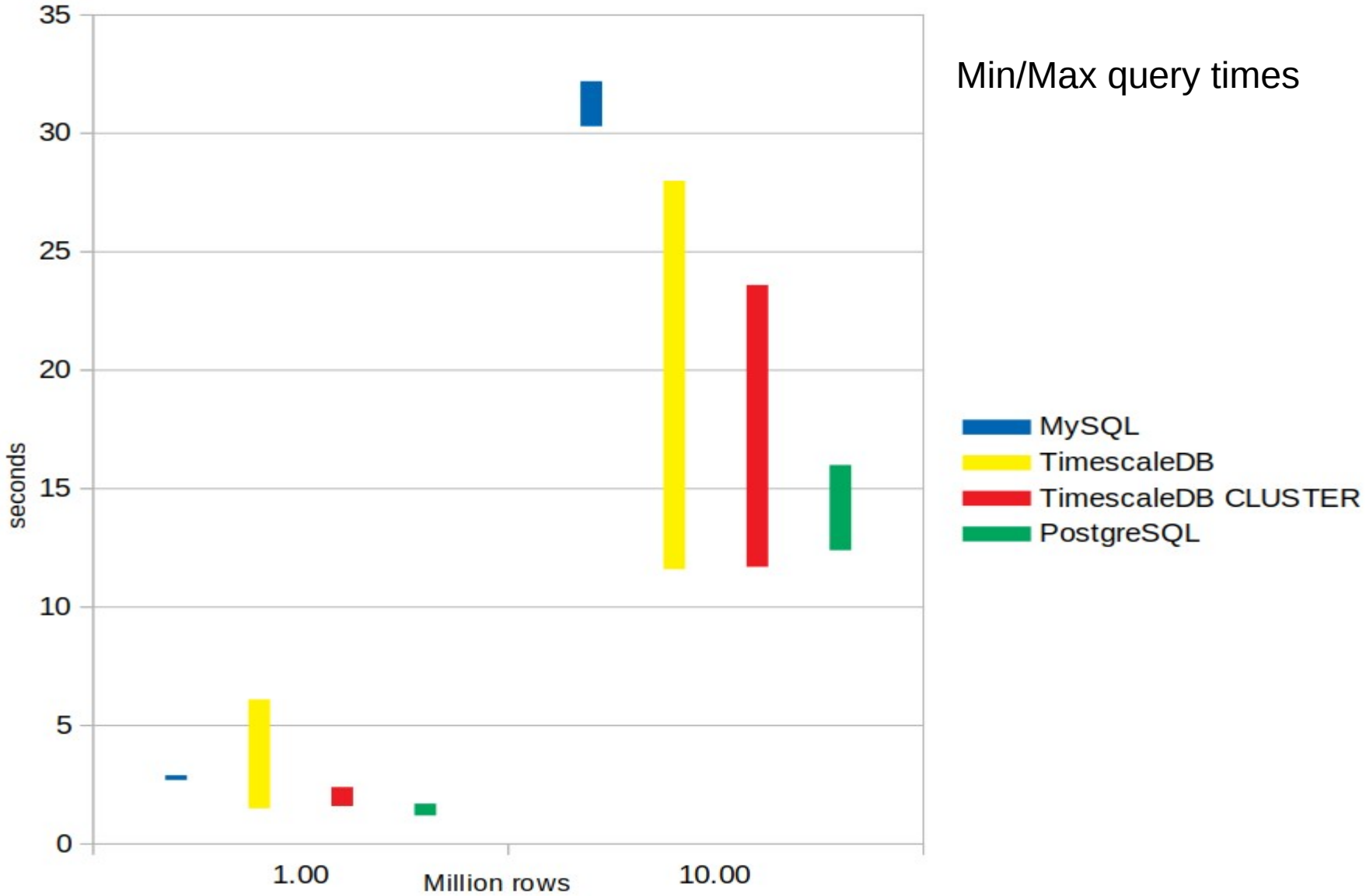
# HDB++ @ Elettra

## First Tests
## 2) QUERY

- Compared results of 2 queries
  - SELECT data_time, value_r FROM att_scalar_devdouble WHERE att_conf_id=3467 AND data_time >= '2017-07-17 18:00:00' AND data_time < '2017-07-31 23:59:00' ORDER BY data_time ASC ---> **1000354 rows** (~1M)
  - SELECT data_time, value_r FROM att_scalar_devdouble WHERE att_conf_id=3467 AND data_time >= '2017-03-07 00:00:00' AND data_time < '2017-07-31 23:59:00' ORDER BY data_time ASC ---> **10023508 rows** (~10M)
- Measured query time 2 times:
  - after service (PostgreSQL / MySQL) restart and cache clearing (echo 3 > /proc/sys/vm/drop_caches)
  - just after first execution
- TimescaleDB tested
  - before and after “CLUSTER att_scalar_devdouble”
  - With just primary key, and 2 additional indexes

# HDB++ @ Elettra

## First Tests
## 3) RESULTS




Min/Max query times

# HDB++ @ Elettra

Elettra Sincrotrone Trieste

TANGO

- MySQL (InnoDB + partitions)

Bulk load with dump ordered by att_conf_id,data_time (same as Primary Key)

| | 1M | 10M |
|---|---|---|
| I (max) | **2.9 s** | **32.2 s** |
| II | **2.7 s** | **30.3 s** |

Bulk load with dump ordered by data_time

| | 1M | 10M |
|---|---|---|
| I (max) | **4.0 s** | **42.3 s** |
| II | **2.8 s** | **29.4 s** |

- TimescaleDB

| | 1M | 10M |
|---|---|---|
| I (max) | **6.1 s** | **28.0 s** |
| II (min) | **1.5 s** | **11.6 s** |

| | 1M | 10M |
|---|---|---|
| I (max) | **118 s** | **1369 s** |
| II (min) | **2.0 s** | **1029 s** |

- TimescaleDB after CLUSTER

| | 1M | 10M |
|---|---|---|
| I (max) | **2.4 s** | **23.6 s** |
| II (min) | **1.6 s** | **11.7 s** |

| | 1M | 10M |
|---|---|---|
| I (max) | **~= s** | **~= s** |
| II (min) | **~= s** | **~= s** |

- PostgreSQL

| | 1M | 10M |
|---|---|---|
| I (max) | **1.7 s** | **16.0 s** |
| II (min) | **1.2 s** | **12.4 s** |

| | 1M | 10M |
|---|---|---|
| I (max) | **130 s** | **1719 s** |
| II (min) | **5.7 s** | **1188 s** |

Elettra Sincrotrone Trieste

# HDB++ @ Elettra

## DB SIZES ON DISK

| **Size of att_scalar_devdouble_ro table** | **Bulk load with dump ordered by att_conf_id,data_time (same as primary key)** | **Bulk load with dump ordered by data_time** |
|---|---|---|
| **MySQL (**InnoDB + partitions**)** | **202** GB (13 hour load time) | **240** GB (15 hour load time) |
| **TimescaleDB (**with 2 indexes**)** | **668** GB (30 hour load time)<br>(**234** GB data + **434** GB indexes) | **790** GB (35 hour load time)<br>(**234** GB data + **556** GB indexes) |
| **TimescaleDB (**with 2 indexes**) after CLUSTER** | **554** GB (12 hour clustering time)<br>(**234** GB data + **320** GB indexes) | **566** GB (72 hour clustering time)<br>(**234** GB data + **332** GB indexes) |
| **TimescaleDB (**just Primary Key**)** | | **474** GB (28 hour load time)<br>(**236** GB data + **238** GB indexes) |
| **TimescaleDB (**just Primary Key**) after CLUSTER** | | **358** GB (10 hour clustering time)<br>(**235** GB data + **123** GB indexes) |
| **PostgreSQL** | **488** GB (12 hour load time)<br>(**272** GB data + **216** GB indexes) | **512** GB (13 hour load time)<br>(**272** GB data + **240** GB indexes) |

# HDB++ @ Elettra



## Results

- PostgreSQL/TimescaleDB generally perform slightly better than MySQL InnoDB in simple queries as long as CLUSTERING is done. Without CLUSTERING during the tests around 55 times more pages need to be read from disk. TimescaleDB is expected to make the difference on more complex, time-based, queries.
- MySQL InnoDB data are stored on disk already clustered.
- CLUSTERING takes a lot of time but it only needs to be performed once on old partitions/chunks. On a 14 days chunck it takes some minutes.
- Secondary indexes have a big impact on disk sizes and load performances, so they should only be added when it is certain they are needed.
- Measuring bulk load time is not the same as measuring insertion throughput, but is easily reproducible and shows the strong relation between time and data size on disk.

# HDB++ @ Elettra



## Considerations and what's next

- Benchmarking DataBases takes a lot of time, so comparing many different engines and configurations (HW and SW) takes much more time.
- In order to share the effort between different institutes standard test procedures and test setups (docker?) should be defined. Benchmarks could be run on AWS.
- As soon as batch insertion feature is implemented, test insertion performances comparing MySQL (InnoDB), TimescaleDB, PostgreSQL, MariaDB, ..
- Go on benchmarking with particular attention to different data types (scalar vs arrays)
- Test more complex queries (to be defined)

# HDB++ @ Elettra



## Questions ?