Elettra
Sincrotrone
Trieste

TANGO

HDB++ TimescaleDB tests @Elettra

Graziano Scalamera

HDB++ @ Elettra

TimescaleDB tests

Hardware and Software

- A (~5 years old) desktop PC with:
  - CPU: Intel Core i7-5820 3.33GHz, esa 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

TimescaleDB tests

1) DUMP

- DUMP of FERMI HDB++ DB (number of rows):

att_scalar_devdouble_ro	**4,082,099,171**
att_scalar_devdouble_rw	**509,271,935**
att_scalar_devlong_ro	**374,746,728**
att_scalar_devlong_rw	**350,896,081**
att_scalar_devfloat_ro	**215,404,541**
att_scalar_devboolean_ro	**100,147,178**
...	**< 100,000,000**

HDB++ @ Elettra

TimescaleDB tests

1) DUMP

- DUMP of FERMI HDB++ DB:
  - Create and fill manually lookup table to match different att_conf_type_id, att_conf_format_id, att_conf_write_id;
  - Dump att_conf
    SELECT **ac.att_conf_id,ac.att_name, acdt_lu.data_type, acdt_lu.data_format, acdt_lu.data_write_type, acdt_lu.table_name, ac.facility,ac.domain,ac.family,ac.member,ac.name, ac.att_ttl** INTO OUTFILE 'att_conf.csv' FROM hdbpp.att_conf ac INNER JOIN att_conf_data_type_lu acdt_lu ON acdt_lu.att_conf_data_type_id = ac.att_conf_data_type_id ORDER BY att_conf_id ASC;
  - Dump att_scalar_devdouble_ro (note: ordered by att_conf_id, data_time)
    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 att_error_desc...

HDB++ @ Elettra

TimescaleDB tests

2) BULK LOAD

Using pt-fifo-split from PERCONA Toolkit to split load in chunks:

- Load att_conf:

```
[...]
/pt-fifo-split --force --lines ${CHUNK_SIZE} ${FLAT_FILE} --fifo ${FIFO_PATH}
while [ -e ${FIFO_PATH} ]
do
  # Write chunk to disk
  cat ${FIFO_PATH} > ${LOAD_FILE}
  # Load chunk into table
  PGPASSWORD=PostgresqlPWD psql postgres -h 127.0.0.1 -d hdb \
  -c "\COPY $2 (att_conf_id, att_name, att_conf_type_id, att_conf_format_id, \
  att_conf_write_id, table_name, cs_name, domain, family,member,name,ttl)
FROM '${LOAD_FILE}' DELIMITER E'\t' NULL '\N';"
done
```

- Load att_error_desc

HDB++ @ Elettra

TimescaleDB tests

2) BULK LOAD

- Load att_scalar_devdouble:

```
[...]
/pt-fifo-split --force --lines ${CHUNK_SIZE} ${FLAT_FILE} --fifo ${FIFO_PATH}
while [ -e ${FIFO_PATH} ]
do
  # Write chunk to disk
  cat ${FIFO_PATH} > ${LOAD_FILE}
  # Load chunk into table
  PGPASSWORD=PostgresqlPWD psql postgres -h 127.0.0.1 -d hdb \
  -c "\COPY $2 (att_conf_id,data_time,value_r,quality,att_error_desc_id) \
  FROM '${LOAD_FILE}' DELIMITER E'\t' NULL '\N';"
done
```

- Time to load att_scalar_devdouble_ro (**4,082,099,171 rows**):

  ~ 30 hours

HDB++ @ Elettra

TimescaleDB tests

2) BULK LOAD

- Tested also with **timescaledb-parallel-copy** instead of **\COPY**:
  PGPASSWORD=Postgresql123 **timescaledb-parallel-copy** --db-name hdb --table $2 --columns "att_conf_id,data_time,value_r,quality,att_error_desc_id" -split "\t" --file ${LOAD_FILE} **--workers 4** --copy-options "NULL 'NULL'"

- Time to load att_scalar_devdouble_ro (**4,082,099,171 rows**):
  ~ 20 hours but increased disk usage (~5 %)

HDB++ @ Elettra

TimescaleDB tests

3) 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' ---> **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' ---> **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

HDB++ @ Elettra

TimescaleDB tests

3) QUERY

- Tested TimescaleDB before and after “CLUSTER att_scalar_devdouble”
  - First execution took more than 10 hours and changed disk size of the table fom 668 GB to 554 GB
  - Second execution on the same table without changes took more than 5 hours, leaving the table with the same disk size
- The same queries were also tested with the “ORDER BY data_time ASC” clause

HDB++ @ Elettra

TimescaleDB tests
4) RESULTS

- MySQL (InnoDB + partitions)

	1M	10M
I (max)	**2.3 s**	**22.7 s**
II (min)	**2.0 s**	**21.0 s**

- TimescaleDB

	1M	10M
I (max)	**5.3 s**	**36.5 s**
II (min)	**1.2 s**	**12.2 s**

- TimescaleDB after CLUSTER

	1M	10M
I (max)	**1.9 s**	**15.0 s**
II (min)	**1.2 s**	**11.9 s**

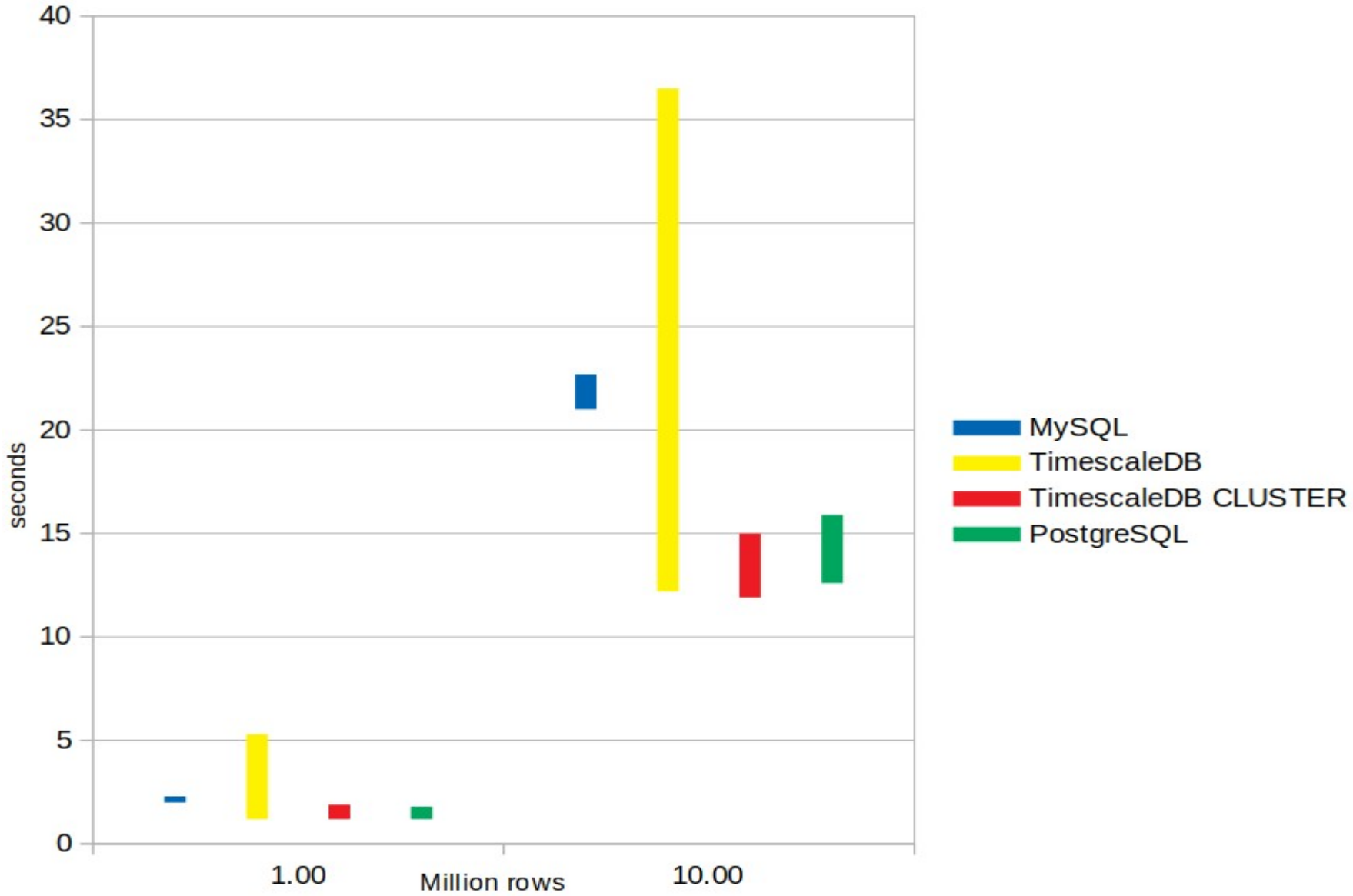
- PostgreSQL

	1M	10M
I (max)	**1.8 s**	**15.9 s**
II (min)	**1.2 s**	**12.6 s**

HDB++ @ Elettra

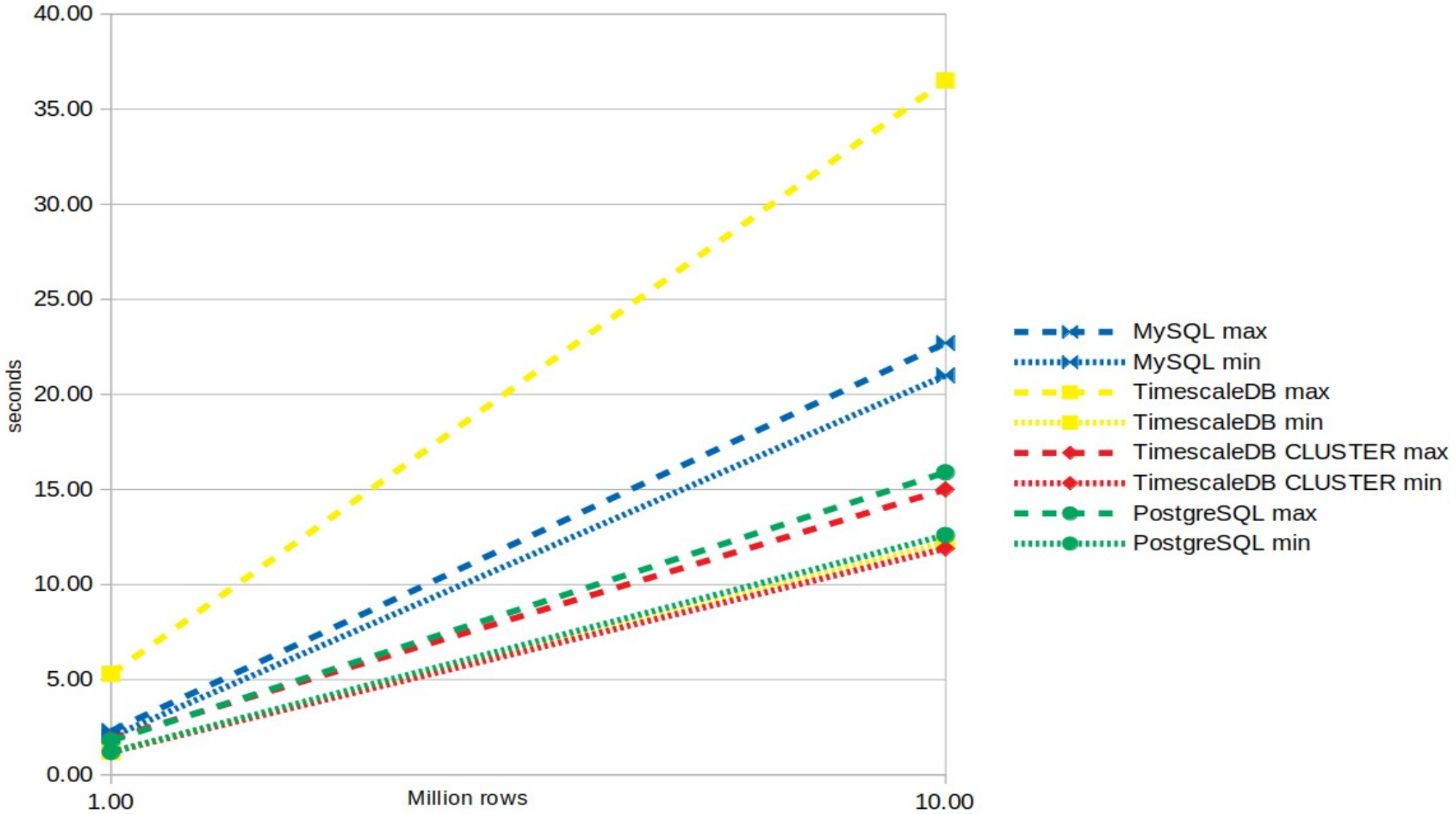
TimescaleDB tests

4) RESULTS



HDB++ @ Elettra

TimescaleDB tests
4) RESULTS



HDB++ @ Elettra

TimescaleDB tests
4) RESULTS

- MySQL (InnoDB + partitions)
  + ORDER BY data_time ASC

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

- TimescaleDB
  + ORDER BY data_time ASC

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

- TimescaleDB after CLUSTER
  + ORDER BY data_time ASC

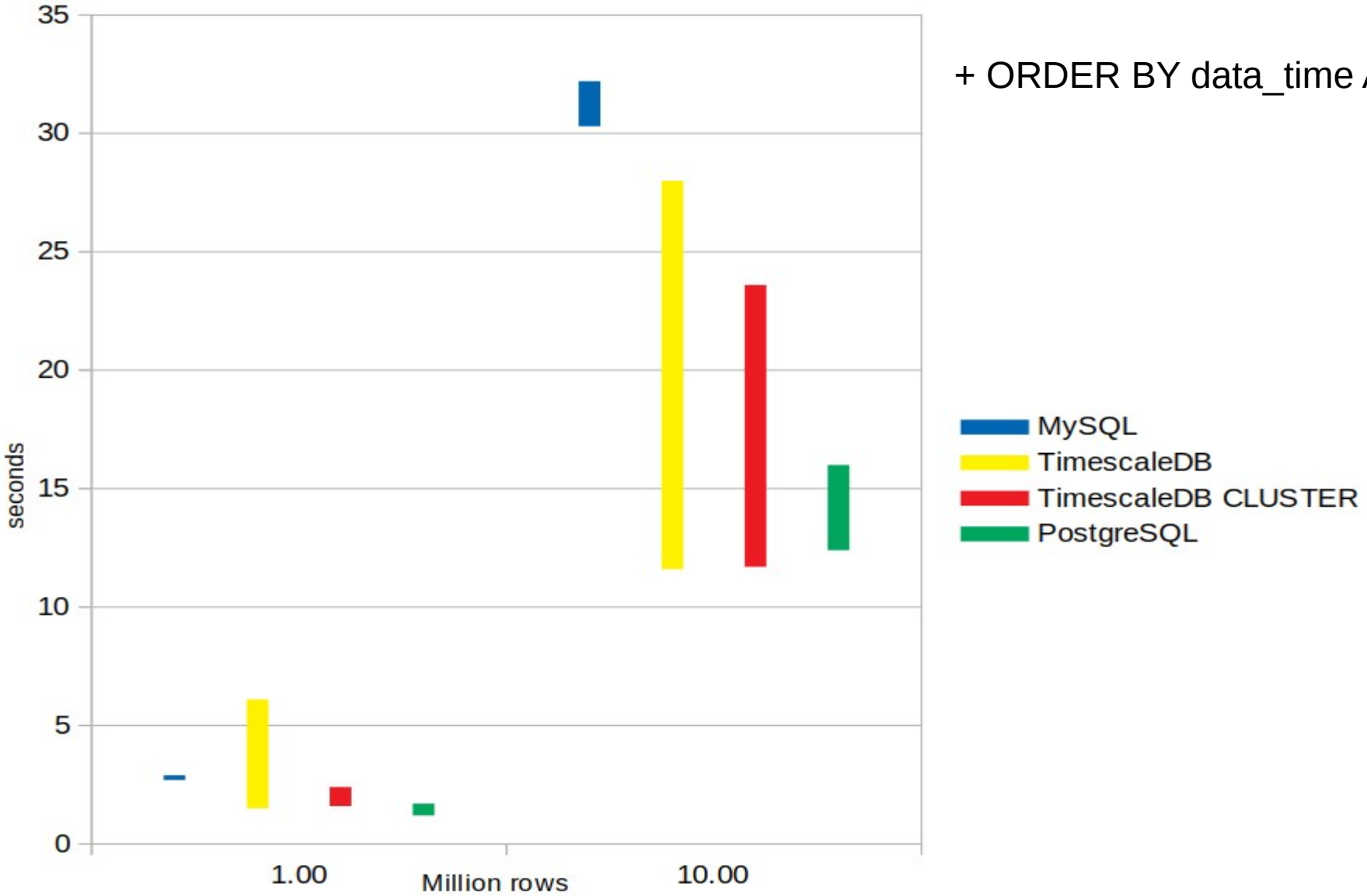
	1M	10M
I (max)	**2.4 s**	**23.6 s**
II (min)	**1.6 s**	**11.7 s**

- PostgreSQL
  + ORDER BY data_time ASC

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

HDB++ @ Elettra

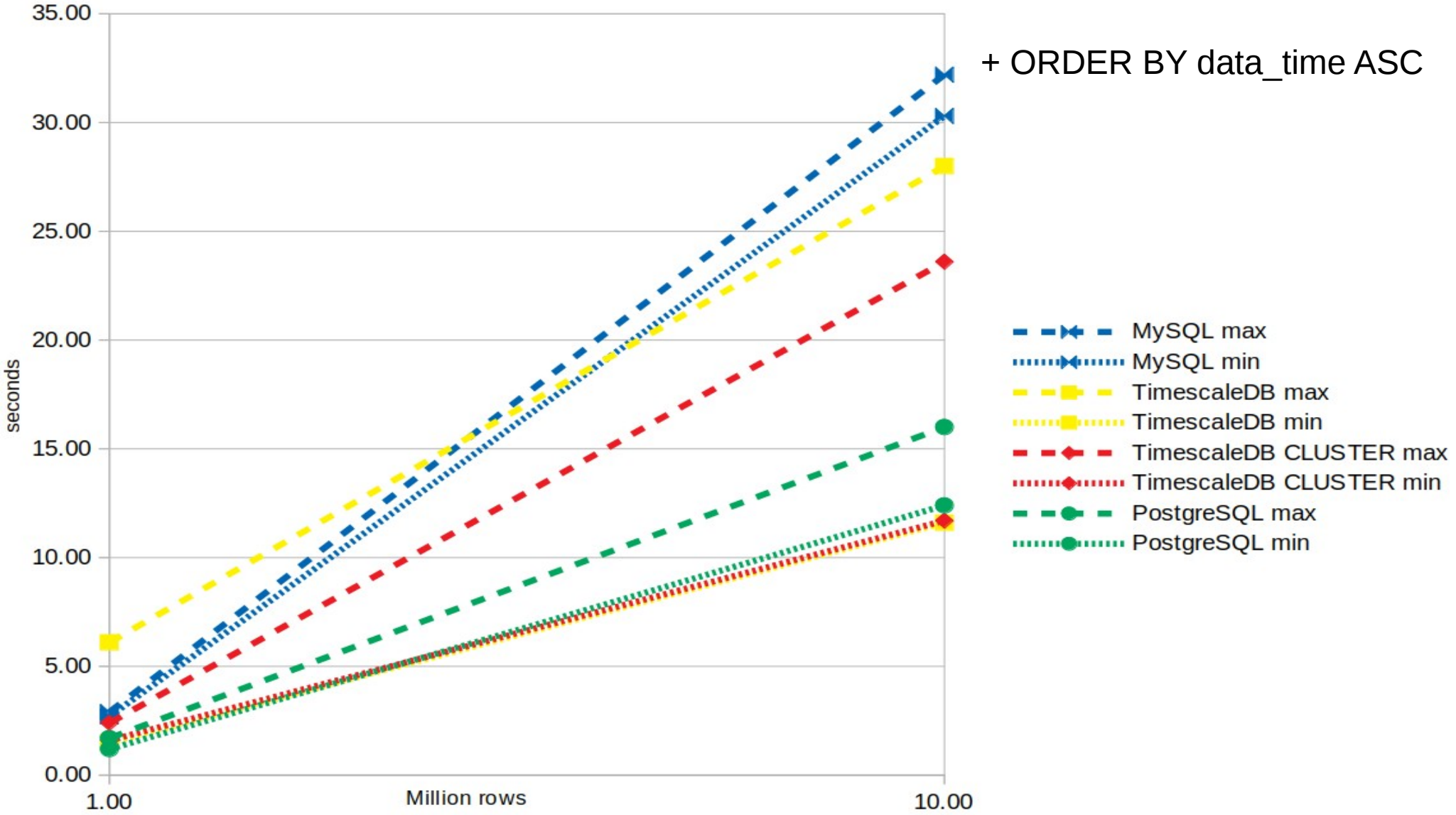
TimescaleDB tests
4) RESULTS



+ ORDER BY data_time ASC

HDB++ @ Elettra

TimescaleDB tests
4) RESULTS



+ ORDER BY data_time ASC

HDB++ @ Elettra

TimescaleDB tests

5) QUESTIONS ON TIMESCALEDB SCHEMA

```
CREATE TABLE IF NOT EXISTS att_scalar_devdouble (
    att_conf_id integer NOT NULL,
    data_time timestamp WITH TIME ZONE NOT NULL,
    value_r double precision,
    value_w double precision,
    quality smallint,
    att_error_desc_id integer,
    details json,
    PRIMARY KEY (att_conf_id, data_time),
    FOREIGN KEY (att_conf_id) REFERENCES att
    FOREIGN KEY (att_error_desc_id) REFE          CES att_error_desc (att_error_desc_id)
);

 COMMENT ON TABLE att_scalar_devdouble IS 'Scalar Double Values Table';
 CREATE INDEX IF NOT EXISTS att_scalar_devdouble_att_conf_id_idx ON att_scalar_devdouble
(att_conf_id);
 CREATE INDEX IF NOT EXISTS att_scalar_devdouble_att_conf_id_data_time_idx ON
att_scalar_devdouble (att_conf_id,data_time DESC);
 SELECT create_hypertable('att_scalar_devdouble', 'data_time', chunk_time_interval => interval '14
day', create_default_indexes => FALSE);
```

Aren't Foreign Keys bad for (INSERT) performances?

Isn't it already contained within the primary key?

HDB++ @ Elettra

TimescaleDB tests

5) CONCLUSION

- PostgreSQL and TimescaleDB performs generally better than MySQL (InnoDB)
- Max query time with TimescaleDB before CLUSTER needs to be investigated: bulk loading the table could have produced a much different result than in the real scenario since data in the dump file are ordered by (att_conf_id, data_time)
- INSERT performances need to be evaluated

HDB++ @ Elettra

TimescaleDB tests

Questions ?