Evaluate MySQL table partitioning for the reliable ingest with the rollback option

1. "Super-transactions" in the Qserv Ingest System

This document captures a study exploring a possibility of using MySQL/MariaDB partitioning as the key technology in the foundation of the "super-transactions". One of the most appealing features of the partitioning mechanism is that allows adding data to tables in well defined increments - the MySQL partitions. Partitioning is a stable and mature technology which is available in MySQL for a few major versions. MySQL provides a rich set of the data management operations (MySQL DML) and tools supporting/manipulating partitions. And there is negligible overhead in terms of space or query performance in using partitions in MySQL. Partitions also map directly to files in both **MyISAM** and **Aria** (a replacement for **MyISAM**) table engines (other table engines were not considered in this study). Each partition-specific file includes an identifier of the corresponding partition. This allows a straightforward implementation of the Replication system to support the partitioned tables in the same way it's done for the monolithic tables.

Here is a possible scenario for implementing the "super-transactions". Let's suppose we have a collection of input data (TSV files) of table **Object** ready to be ingest into Qserv. And this is going to be the very first ingest for the table in some new catalog. And let's assume that we're going to load all this collection of files within some "super-transaction", meaning that if any failure occurs at any stage during the ingest then:

- the loading workflow will known about it (either by receiving some error code from a loading application or by asking a status of the on-going ingest effort from the system)
- and the Ingest/Replication system will always be able to roll a state of the catalog back to a stable and consistent state prior to the beginning of the "super-transaction"

One of the steps which is going to be taken by the Ingest system when the loading workflow will be will be initiating a "super-transaction" would be to allocate a unique identifier of the ingest (the "super-transaction"). This could be as simple as just a number staring with 0 (the very first ingest). The identifier will be also used internally (though, it could also be made known to the catalog loading workflow) by the Ingest/Replication system as a partition identifier of all *chunks* of table **Object** across all worker nodes where the chunks will be residing. When the very first batch of data will be delivered to a worker for (say) chunk number **123** the system will create a partitioned table **Object** for that chunk, and the table will be defined to have a single partition based on the previously allocated identifier. And if all goes well and all input files are successfully loaded into Qserv the "super-transaction" will be marked as completed. Otherwise a client may request the system to perform the *rollback* of the super-transaction. In case if this was the very first attempt to ingest data into the catalog it would mean deleting all tables created during the ingest. For any subsequent "super-transactions" only the last partition would be deleted from the tables should any failure occurred during the load. The Ingest/Replication system will maintain a persistent registry of the "super-transaction" so that any relevant data management/manipulation operations performed by the system would be aware of a state of the latest "super-transaction".

These are just a few benefits of the MySQL partitions in this context:

- it's a mature and reliable technology which is expected to stay in MySQL/MariaDB for)probably) as long as this RDBMS is in use

The only **wish list** for the partitioning technology is that the (MySQL/MariaDB) developers made a further (quite logical) step and used partitions for the parallel execution of queries. In reality, the very same idea is already expressed in one of the MySQL/MariaDB documents.

2. Examples

Create the partitioned table:

```
CREATE DATABASE `partitions`;
USE `partitions`;

CREATE TABLE `t` (
  `id` int(11) NOT NULL AUTO_INCREMENT,
  `p` int(11) NOT NULL,
  `name` varchar(255) NOT NULL,
  PRIMARY KEY (`p`,`id`)
) ENGINE=MyISAM DEFAULT CHARSET=latin1
 PARTITION BY RANGE (`p`)
(PARTITION `p0` VALUES LESS THAN (1) ENGINE = MyISAM,
 PARTITION `p1` VALUES LESS THAN (2) ENGINE = MyISAM,
 PARTITION `p2` VALUES LESS THAN (3) ENGINE = MyISAM,
 PARTITION `p3` VALUES LESS THAN (4) ENGINE = MyISAM,
 PARTITION `p4` VALUES LESS THAN (5) ENGINE = MyISAM);

SELECT PARTITION_NAME, TABLE_ROWS FROM INFORMATION_SCHEMA.PARTITIONS WHERE TABLE_NAME = 't';
+----------------+------------+
| PARTITION_NAME | TABLE_ROWS |
+----------------+------------+
| p0             |          0 |
| p1             |          0 |
| p2             |          0 |
| p3             |          0 |
| p4             |          0 |
+----------------+------------+
```

Insert a few rows:

```
INSERT INTO t VALUES(NULL,0,'one');
INSERT INTO t VALUES(NULL,4,'four');

SELECT PARTITION_NAME, TABLE_ROWS FROM INFORMATION_SCHEMA.PARTITIONS WHERE TABLE_NAME = 't';
+----------------+------------+
| PARTITION_NAME | TABLE_ROWS |

+----------------+------------+
| p0             |          1 |
| p1             |          0 |
| p2             |          0 |
| p3             |          0 |
| p4             |          1 |
+----------------+------------+
```

Add another partition:

```
INSERT INTO t VALUES(NULL,5,'five');
ERROR 1526 (HY000): Table has no partition for value 5

ALTER TABLE t ADD PARTITION (PARTITION `p5` VALUES LESS THAN (6));
INSERT INTO t VALUES(NULL,5,'five');

+----------------+------------+
| PARTITION_NAME | TABLE_ROWS |
+----------------+------------+
| p0             |          1 |
| p1             |          0 |
| p2             |          0 |
| p3             |          0 |
| p4             |          1 |
| p5             |          1 |
+----------------+------------+
```

Watch for the **AUTO_INCREMENT** column:

```
SELECT * FROM t;
+----+---+------+
| id | p | name |
+----+---+------+
|  1 | 0 | one  |
|  1 | 4 | four |
|  1 | 5 | five |
+----+---+------+
```

See the resulting files in the filesystem:

```
/bin/sudo -u qserv ls -al /qserv/replication/mysql/partitions/
total 56
drwx------ 2 qserv qserv  262 Feb  8 21:06 .
drwxr-xr-x 6 qserv qserv 4096 Feb  8 21:06 ..
-rw-rw---- 1 qserv qserv   65 Feb  8 16:12 db.opt
-rw-rw---- 1 qserv qserv 1582 Feb  8 21:06 t.frm
-rw-rw---- 1 qserv qserv   68 Feb  8 21:06 t.par
-rw-rw---- 1 qserv qserv   20 Feb  8 21:04 t#P#p0.MYD
-rw-rw---- 1 qserv qserv 2048 Feb  8 21:06 t#P#p0.MYI
-rw-rw---- 1 qserv qserv    0 Feb  8 20:23 t#P#p1.MYD
-rw-rw---- 1 qserv qserv 1024 Feb  8 20:23 t#P#p1.MYI
-rw-rw---- 1 qserv qserv    0 Feb  8 20:23 t#P#p2.MYD
-rw-rw---- 1 qserv qserv 1024 Feb  8 20:23 t#P#p2.MYI
-rw-rw---- 1 qserv qserv    0 Feb  8 20:23 t#P#p3.MYD
-rw-rw---- 1 qserv qserv 1024 Feb  8 20:23 t#P#p3.MYI
-rw-rw---- 1 qserv qserv   20 Feb  8 21:05 t#P#p4.MYD
-rw-rw---- 1 qserv qserv 2048 Feb  8 21:06 t#P#p4.MYI
-rw-rw---- 1 qserv qserv   20 Feb  8 21:07 t#P#p5.MYD
-rw-rw---- 1 qserv qserv 2048 Feb  8 21:07 t#P#p5.MYI
```

2.1. Planning more tests

- truncating partitions (removing all data from a partition)
- deleting partitions
- translating to the non-partitioned format
- large-scale performance tests
    - LOAD DATA INFILE ... PARTITION `p1`
    - table format translation (merging partitions into a simple table)
    - query performance for regular vs partitioned tables (requires a large number of data on a scale of many million rows)

3. Large scale tests

3.1. Setting up MariaDB container on PDAC node **qserv-db01**.

Here is the first step for preparing the data directories:

```
ssh qserv-db01
/bin/sudo -u qserv mkdir /qserv/replication/mysql
/bin/sudo -u qserv mkdir /qserv/replication/work
```

Starting the container by a script run from the master node **qserv-master01** (could also be run from any other node):

```
#!/bin/bash

set -e
set -a


# Base directory of the replication system on both master and worker nodes
REPLICATION_DATA_DIR="/qserv/replication"

# Base directory where the Replication system's MariaDB/MySQL service
# of the master node will create its folder 'mysql'
DB_DATA_DIR="${REPLICATION_DATA_DIR}"

# Work directory for the applications. It can be used by applications
# to store core files, as well as various debug information.
WORK_DIR="${REPLICATION_DATA_DIR}/work"


DB_IMAGE_TAG="mariadb:10.2.16"
DB_CONTAINER_NAME="qserv-replica-mariadb"
DB_PORT=23306
DB_ROOT_PASSWORD="CHANGEME"

# User account under which the containers will be run
CONTAINER_UID=1000
CONTAINER_GID=1000

HOST=qserv-db01
ssh -n $HOST docker run \
    --detach \
    --name "${DB_CONTAINER_NAME}" \
    -u ${CONTAINER_UID}:${CONTAINER_GID} \
    -v /etc/passwd:/etc/passwd:ro \
    -v "${DB_DATA_DIR}/mysql:/var/lib/mysql" \
    -v "${DB_DATA_DIR}/log:${DB_DATA_DIR}/log" \
    -v "${WORK_DIR}:${WORK_DIR}:ro" \
    -e "MYSQL_ROOT_PASSWORD=${DB_ROOT_PASSWORD}" \
    -p "${DB_PORT}:${DB_PORT}/tcp" \
    "${DB_IMAGE_TAG}" \
    --port="${DB_PORT}" \
    --max-connections=4096 \
    --query-cache-size=0 \
    --log-error="${DB_DATA_DIR}/log/${DB_CONTAINER_NAME}.error.log" \
    --slow-query-log --slow-query-log-file="${DB_DATA_DIR}/log/${DB_CONTAINER_NAME}.slow-query.log" \
    --log-warnings=2 \
    --pid-file="${DB_DATA_DIR}/log/${DB_CONTAINER_NAME}.pid"
```

Connecting to the service:

```
ssh qserv-db01
source /datasets/gapon/stack/loadLSST.bash
setup -t qserv-dev qserv_distrib
mysql -A --protocol=tcp -hlocalhost -P 23306 -uroot -pCHANGEME
```

3.2. Loading 1 billion rows into a table of 10 partitions

3.2.1. Creating the table

```
CREATE TABLE `t` (
  `id` int(11) NOT NULL AUTO_INCREMENT,
  `qserv_load_id` int(11) NOT NULL,
  `name` varchar(255) NOT NULL,
  `c000` double DEFAULT NULL,
  `c001` double DEFAULT NULL,
  `c002` double DEFAULT NULL,
  `c003` double DEFAULT NULL,
  `c004` double DEFAULT NULL,
  `c005` double DEFAULT NULL,
  `c006` double DEFAULT NULL,
  `c007` double DEFAULT NULL,
  `c008` double DEFAULT NULL,
  `c009` double DEFAULT NULL,
  PRIMARY KEY (`id`,`qserv_load_id`)
) ENGINE=MyISAM AUTO_INCREMENT=139648676 DEFAULT CHARSET=latin1
 PARTITION BY LIST (`qserv_load_id`)
(PARTITION `p0` VALUES IN (0) ENGINE = MyISAM,
 PARTITION `p1` VALUES IN (1) ENGINE = MyISAM,
 PARTITION `p2` VALUES IN (2) ENGINE = MyISAM,
 PARTITION `p3` VALUES IN (3) ENGINE = MyISAM,
 PARTITION `p4` VALUES IN (4) ENGINE = MyISAM,
 PARTITION `p5` VALUES IN (5) ENGINE = MyISAM,
 PARTITION `p6` VALUES IN (6) ENGINE = MyISAM,
 PARTITION `p7` VALUES IN (7) ENGINE = MyISAM,
 PARTITION `p8` VALUES IN (8) ENGINE = MyISAM,
 PARTITION `p9` VALUES IN (9) ENGINE = MyISAM)
```

3.2.2. Generating data

Here is the trivial Python generator for partition **p7**:

```
% cat /qserv/replication/work/generate_t_7.py

for i in range(0,100000000):
    print "\N\t7\t'str 0'\t0\t1\t2\t3\t4\t5\t6\t7\t8\t9"
```

Others look similarly (though it would be better to pass a value of column **qserv_load_id** as a script parameter).

3.2.3. Loading and measuring the performance for the first batch of data

Loading 100 million rows into partition **0**:

```
SELECT COUNT(*) FROM t;
+----------+
| COUNT(*) |
+----------+
|        0 |
+----------+

SELECT NOW();
+---------------------+
| NOW()               |
+---------------------+
| 2019-02-11 23:29:19 |
+---------------------+

LOAD DATA INFILE '/qserv/replication/work/data_t_0.tsv' INTO TABLE t;
Query OK, 100000000 rows affected (7 min 53.40 sec)
Records: 100000000  Deleted: 0  Skipped: 0  Warnings: 0

SELECT NOW();
+---------------------+
| NOW()               |
+---------------------+
| 2019-02-11 23:37:13 |
+---------------------+

SELECT COUNT(*) FROM t;
+-----------+
| COUNT(*)  |
+-----------+
| 100000000 |
+-----------+

SELECT * FROM t LIMIT 2;
+----+---------------+---------+------+------+------+------+------+------+------+------+------+------+
| id | qserv_load_id | name    | c000 | c001 | c002 | c003 | c004 | c005 | c006 | c007 | c008 | c009 |
+----+---------------+---------+------+------+------+------+------+------+------+------+------+------+
|  1 |             0 | 'str 0' |    0 |    1 |    2 |    3 |    4 |    5 |    6 |    7 |    8 |    9 |
|  2 |             0 | 'str 0' |    0 |    1 |    2 |    3 |    4 |    5 |    6 |    7 |    8 |    9 |
+----+---------------+---------+------+------+------+------+------+------+------+------+------+------+
```

The filesystem view of the table:

```
[gapon@lsst-qserv-db01 tests]$ /bin/sudo -u qserv ls -alh /qserv/replication/mysql/partitions/
total 10G
drwx------ 2 qserv qserv 4.0K Feb 11 17:28 .
drwxr-xr-x 5 qserv qserv 4.0K Feb 11 15:36 ..
-rw-rw---- 1 qserv qserv   65 Feb 11 15:36 db.opt
-rw-rw---- 1 qserv qserv 2.1K Feb 11 17:28 t.frm
-rw-rw---- 1 qserv qserv  100 Feb 11 17:28 t.par
-rw-rw---- 1 qserv qserv 8.6G Feb 11 17:37 t#P#p0.MYD
-rw-rw---- 1 qserv qserv 1.4G Feb 11 17:37 t#P#p0.MYI
-rw-rw---- 1 qserv qserv    0 Feb 11 17:28 t#P#p1.MYD
-rw-rw---- 1 qserv qserv 1.0K Feb 11 17:28 t#P#p1.MYI
-rw-rw---- 1 qserv qserv    0 Feb 11 17:28 t#P#p2.MYD
-rw-rw---- 1 qserv qserv 1.0K Feb 11 17:28 t#P#p2.MYI
-rw-rw---- 1 qserv qserv    0 Feb 11 17:28 t#P#p3.MYD
-rw-rw---- 1 qserv qserv 1.0K Feb 11 17:28 t#P#p3.MYI
-rw-rw---- 1 qserv qserv    0 Feb 11 17:28 t#P#p4.MYD
-rw-rw---- 1 qserv qserv 1.0K Feb 11 17:28 t#P#p4.MYI
-rw-rw---- 1 qserv qserv    0 Feb 11 17:28 t#P#p5.MYD
-rw-rw---- 1 qserv qserv 1.0K Feb 11 17:28 t#P#p5.MYI
-rw-rw---- 1 qserv qserv    0 Feb 11 17:28 t#P#p6.MYD
-rw-rw---- 1 qserv qserv 1.0K Feb 11 17:28 t#P#p6.MYI
-rw-rw---- 1 qserv qserv    0 Feb 11 17:28 t#P#p7.MYD
-rw-rw---- 1 qserv qserv 1.0K Feb 11 17:28 t#P#p7.MYI
-rw-rw---- 1 qserv qserv    0 Feb 11 17:28 t#P#p8.MYD
-rw-rw---- 1 qserv qserv 1.0K Feb 11 17:28 t#P#p8.MYI
-rw-rw---- 1 qserv qserv    0 Feb 11 17:28 t#P#p9.MYD
-rw-rw---- 1 qserv qserv 1.0K Feb 11 17:28 t#P#p9.MYI
```

CONCLUSION: the overall performance of the loading was **20 MB/s** for **8.6 GB** of data and **1.4 GB** of indexes. This is no different from what was observed earlier.

3.2.4. Loading 8 batches in parallel

```
mysql_exec="mysql -A --protocol=tcp -hlocalhost -P 23306 -uroot -pCHANGEME partitions"


$mysql_exec -e 'SELECT NOW(); LOAD DATA INFILE "/qserv/replication/work/data_t_1.tsv" INTO TABLE t; SELECT
NOW();' >& p1.log&
$mysql_exec -e 'SELECT NOW(); LOAD DATA INFILE "/qserv/replication/work/data_t_2.tsv" INTO TABLE t; SELECT
NOW();' >& p2.log&
$mysql_exec -e 'SELECT NOW(); LOAD DATA INFILE "/qserv/replication/work/data_t_3.tsv" INTO TABLE t; SELECT
NOW();' >& p3.log&
$mysql_exec -e 'SELECT NOW(); LOAD DATA INFILE "/qserv/replication/work/data_t_4.tsv" INTO TABLE t; SELECT
NOW();' >& p4.log&
$mysql_exec -e 'SELECT NOW(); LOAD DATA INFILE "/qserv/replication/work/data_t_5.tsv" INTO TABLE t; SELECT
NOW();' >& p5.log&
$mysql_exec -e 'SELECT NOW(); LOAD DATA INFILE "/qserv/replication/work/data_t_6.tsv" INTO TABLE t; SELECT
NOW();' >& p6.log&
$mysql_exec -e 'SELECT NOW(); LOAD DATA INFILE "/qserv/replication/work/data_t_7.tsv" INTO TABLE t; SELECT
NOW();' >& p7.log&
$mysql_exec -e 'SELECT NOW(); LOAD DATA INFILE "/qserv/replication/work/data_t_8.tsv" INTO TABLE t; SELECT
NOW();' >& p8.log&
```

Okay, this naive approach didn't quite work because of the global table lock:

```
SHOW PROCESSLIST;
+----+-------------+------------------+------------+---------+------+-----------------------------
+-----------------------------------------------------------------------+----------+
| Id | User        | Host             | db         | Command | Time | State                       |
Info                                                                    | Progress |
+----+-------------+------------------+------------+---------+------+-----------------------------
+-----------------------------------------------------------------------+----------+
| 31 | root        | 172.17.0.1:45660 | partitions | Query   |   68 | reading file                | LOAD DATA
INFILE "/qserv/replication/work/data_t_1.tsv" INTO TABLE t |    7.555 |
| 32 | root        | 172.17.0.1:45664 | partitions | Query   |   68 | Waiting for table level lock | LOAD DATA
INFILE "/qserv/replication/work/data_t_2.tsv" INTO TABLE t |    0.000 |
| 33 | root        | 172.17.0.1:45668 | partitions | Query   |   68 | Waiting for table level lock | LOAD DATA
INFILE "/qserv/replication/work/data_t_3.tsv" INTO TABLE t |    0.000 |
| 34 | root        | 172.17.0.1:45672 | partitions | Query   |   68 | Waiting for table level lock | LOAD DATA
INFILE "/qserv/replication/work/data_t_4.tsv" INTO TABLE t |    0.000 |
| 35 | root        | 172.17.0.1:45676 | partitions | Query   |   68 | Waiting for table level lock | LOAD DATA
INFILE "/qserv/replication/work/data_t_5.tsv" INTO TABLE t |    0.000 |
| 36 | root        | 172.17.0.1:45680 | partitions | Query   |   68 | Waiting for table level lock | LOAD DATA
INFILE "/qserv/replication/work/data_t_6.tsv" INTO TABLE t |    0.000 |
| 37 | root        | 172.17.0.1:45684 | partitions | Query   |   68 | Waiting for table level lock | LOAD DATA
INFILE "/qserv/replication/work/data_t_7.tsv" INTO TABLE t |    0.000 |
| 38 | root        | 172.17.0.1:45688 | partitions | Query   |   68 | Waiting for table level lock | LOAD DATA
INFILE "/qserv/replication/work/data_t_8.tsv" INTO TABLE t |    0.000 |
| 39 | root        | 172.17.0.1:45710 | partitions | Query   |    0 | init                        | SHOW
PROCESSLIST                                                |    0.000 |
+----+-------------+------------------+------------+---------+------+-----------------------------
+-----------------------------------------------------------------------+----------+
```

Let's cancel it and clear partitions:

```
SELECT qserv_load_id,COUNT(*) FROM t GROUP BY qserv_load_id;
+---------------+-----------+
| qserv_load_id | COUNT(*)  |
+---------------+-----------+
|             0 | 100000000 |
|             1 |  53776484 |
|             2 |   1418261 |
|             3 |    471881 |
|             4 |    366968 |
|             5 |    411696 |
|             6 |    462091 |
|             7 |    482947 |
|             8 |    948747 |
+---------------+-----------+
9 rows in set (1 min 31.29 sec)
```

```
ALTER TABLE t TRUNCATE PARTITION p1;
Query OK, 0 rows affected (2.03 sec)

ALTER TABLE t TRUNCATE PARTITION p2;
Query OK, 0 rows affected (0.05 sec)

ALTER TABLE t TRUNCATE PARTITION p3;
Query OK, 0 rows affected (0.02 sec)

ALTER TABLE t TRUNCATE PARTITION p4;
Query OK, 0 rows affected (0.01 sec)

ALTER TABLE t TRUNCATE PARTITION p5;
Query OK, 0 rows affected (0.02 sec)

ALTER TABLE t TRUNCATE PARTITION p6;
Query OK, 0 rows affected (0.02 sec)

ALTER TABLE t TRUNCATE PARTITION p7;
Query OK, 0 rows affected (0.03 sec)

ALTER TABLE t TRUNCATE PARTITION p8;
Query OK, 0 rows affected (0.03 sec)

ALTER TABLE t TRUNCATE PARTITION p9;
Query OK, 0 rows affected (0.00 sec)


SELECT qserv_load_id,COUNT(*) FROM t GROUP BY qserv_load_id;
+---------------+-----------+
| qserv_load_id | COUNT(*)  |
+---------------+-----------+
|             0 | 100000000 |
+---------------+-----------+
1 row in set (53.47 sec)
```

Do the targeting loading for each partition:

```
$mysql_exec -e 'SELECT NOW(); LOAD DATA INFILE "/qserv/replication/work/data_t_1.tsv" INTO TABLE t PARTITION
(p1); SELECT NOW();' >& p1.log&
$mysql_exec -e 'SELECT NOW(); LOAD DATA INFILE "/qserv/replication/work/data_t_2.tsv" INTO TABLE t PARTITION
(p2); SELECT NOW();' >& p2.log&
$mysql_exec -e 'SELECT NOW(); LOAD DATA INFILE "/qserv/replication/work/data_t_3.tsv" INTO TABLE t PARTITION
(p3); SELECT NOW();' >& p3.log&
$mysql_exec -e 'SELECT NOW(); LOAD DATA INFILE "/qserv/replication/work/data_t_4.tsv" INTO TABLE t PARTITION
(p4); SELECT NOW();' >& p4.log&
$mysql_exec -e 'SELECT NOW(); LOAD DATA INFILE "/qserv/replication/work/data_t_5.tsv" INTO TABLE t PARTITION
(p5); SELECT NOW();' >& p5.log&
$mysql_exec -e 'SELECT NOW(); LOAD DATA INFILE "/qserv/replication/work/data_t_6.tsv" INTO TABLE t PARTITION
(p6); SELECT NOW();' >& p6.log&
$mysql_exec -e 'SELECT NOW(); LOAD DATA INFILE "/qserv/replication/work/data_t_7.tsv" INTO TABLE t PARTITION
(p7); SELECT NOW();' >& p7.log&
$mysql_exec -e 'SELECT NOW(); LOAD DATA INFILE "/qserv/replication/work/data_t_8.tsv" INTO TABLE t PARTITION
(p8); SELECT NOW();' >& p8.log&
```

Okay, that seems to work. The CPU utilization went up by a factor of almost 8:

```
% top
top - 19:15:25 up 25 days,  8:39,  1 user,  load average: 0.77, 0.44, 0.40
Tasks: 292 total,   1 running, 291 sleeping,   0 stopped,   0 zombie
%Cpu(s): 29.4 us, 37.1 sy,  0.0 ni, 33.5 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
KiB Mem : 13114742+total,  6717236 free, 36780648 used, 87649544 buff/cache
KiB Swap: 26424115+total, 26412160+free,   119552 used. 93441992 avail Mem


    PID USER      PR  NI    VIRT    RES    SHR S  %CPU %MEM     TIME+ COMMAND
 358814 qserv     20   0 2525456 231852  11304 S 750.0  0.2  16:06.28 mysqld
```

Also:

```
SHOW PROCESSLIST;

+----+------------+------------------+------------+---------+------+--------------------------
+-----------------------------------------------------------------------------------------+----------+
| Id | User        | Host             | db         | Command | Time | State                    |
Info                                                                                    | Progress |
+----+-------------+------------------+------------+---------+------+--------------------------
+-----------------------------------------------------------------------------------------+----------+
| 53 | root        | 172.17.0.1:46140 | partitions | Query   |  286 | reading file             | LOAD DATA
INFILE "/qserv/replication/work/data_t_1.tsv" INTO TABLE t PARTITION (p1) |    1.686 |
| 54 | root        | 172.17.0.1:46144 | partitions | Query   |  286 | reading file             | LOAD DATA
INFILE "/qserv/replication/work/data_t_2.tsv" INTO TABLE t PARTITION (p2) |    1.429 |
| 55 | root        | 172.17.0.1:46148 | partitions | Query   |  286 | reading file             | LOAD DATA
INFILE "/qserv/replication/work/data_t_3.tsv" INTO TABLE t PARTITION (p3) |    1.837 |
| 56 | root        | 172.17.0.1:46152 | partitions | Query   |  286 | reading file             | LOAD DATA
INFILE "/qserv/replication/work/data_t_4.tsv" INTO TABLE t PARTITION (p4) |    1.830 |
| 57 | root        | 172.17.0.1:46156 | partitions | Query   |  286 | reading file             | LOAD DATA
INFILE "/qserv/replication/work/data_t_5.tsv" INTO TABLE t PARTITION (p5) |    1.751 |
| 58 | root        | 172.17.0.1:46160 | partitions | Query   |  286 | reading file             | LOAD DATA
INFILE "/qserv/replication/work/data_t_6.tsv" INTO TABLE t PARTITION (p6) |    1.300 |
| 59 | root        | 172.17.0.1:46164 | partitions | Query   |  286 | reading file             | LOAD DATA
INFILE "/qserv/replication/work/data_t_7.tsv" INTO TABLE t PARTITION (p7) |    1.802 |
| 60 | root        | 172.17.0.1:46168 | partitions | Query   |  285 | reading file             | LOAD DATA
INFILE "/qserv/replication/work/data_t_8.tsv" INTO TABLE t PARTITION (p8) |    1.740 |
| 61 | root        | 172.17.0.1:46258 | partitions | Query   |    0 | init                     | SHOW
PROCESSLIST                                                                             |    0.000 |
+----+-------------+------------------+------------+---------+------+--------------------------
+-----------------------------------------------------------------------------------------+----------+
```

When the loading finished:

```
SELECT COUNT(*) FROM t;
+-----------+
| COUNT(*)  |
+-----------+
| 900000000 |
+-----------+
1 row in set (0.00 sec)

SELECT qserv_load_id,COUNT(*) FROM t GROUP BY qserv_load_id;
+---------------+-----------+
| qserv_load_id | COUNT(*)  |
+---------------+-----------+
|             0 | 100000000 |
|             1 | 100000000 |
|             2 | 100000000 |
|             3 | 100000000 |
|             4 | 100000000 |
|             5 | 100000000 |
|             6 | 100000000 |
|             7 | 100000000 |
|             8 | 100000000 |
+---------------+-----------+
9 rows in set (10 min 35.08 sec)
```

And the file system view:

```
ls -alh /qserv/replication/mysql/partitions/
total 93G
drwx------ 2 qserv qserv 4.0K Feb 11 17:28 .
drwxr-xr-x 5 qserv qserv 4.0K Feb 11 15:36 ..
-rw-rw---- 1 qserv qserv   65 Feb 11 15:36 db.opt
-rw-rw---- 1 qserv qserv 2.1K Feb 11 17:28 t.frm
-rw-rw---- 1 qserv qserv  100 Feb 11 17:28 t.par
-rw-rw---- 1 qserv qserv 8.6G Feb 11 17:37 t#P#p0.MYD
-rw-rw---- 1 qserv qserv 1.4G Feb 11 19:09 t#P#p0.MYI
-rw-rw---- 1 qserv qserv 9.0G Feb 11 22:24 t#P#p1.MYD
-rw-rw---- 1 qserv qserv 1.4G Feb 11 22:24 t#P#p1.MYI
-rw-rw---- 1 qserv qserv 9.0G Feb 11 22:25 t#P#p2.MYD
-rw-rw---- 1 qserv qserv 1.4G Feb 11 22:25 t#P#p2.MYI
-rw-rw---- 1 qserv qserv 9.0G Feb 11 22:19 t#P#p3.MYD
-rw-rw---- 1 qserv qserv 1.4G Feb 11 22:19 t#P#p3.MYI
-rw-rw---- 1 qserv qserv 9.0G Feb 11 22:23 t#P#p4.MYD
-rw-rw---- 1 qserv qserv 1.4G Feb 11 22:23 t#P#p4.MYI
-rw-rw---- 1 qserv qserv 9.0G Feb 11 22:24 t#P#p5.MYD
-rw-rw---- 1 qserv qserv 1.4G Feb 11 22:24 t#P#p5.MYI
-rw-rw---- 1 qserv qserv 9.0G Feb 11 22:25 t#P#p6.MYD
-rw-rw---- 1 qserv qserv 1.4G Feb 11 22:25 t#P#p6.MYI
-rw-rw---- 1 qserv qserv 9.0G Feb 11 22:25 t#P#p7.MYD
-rw-rw---- 1 qserv qserv 1.4G Feb 11 22:25 t#P#p7.MYI
-rw-rw---- 1 qserv qserv 9.0G Feb 11 22:25 t#P#p8.MYD
-rw-rw---- 1 qserv qserv 1.4G Feb 11 22:25 t#P#p8.MYI
-rw-rw---- 1 qserv qserv    0 Feb 11 17:28 t#P#p9.MYD
-rw-rw---- 1 qserv qserv 1.0K Feb 11 19:09 t#P#p9.MYI
```

3.3. Removing partitioning from a table

This operation will turn the partitioned table into the monolithic one:

```
ALTER TABLE `t` REMOVE PARTITIONING;
```

The progress:

```
SHOW PROCESSLIST;
+----+-------------+------------------+------------+---------+------+-------------------------
+-------------------------------------+----------+
| Id | User        | Host             | db         | Command | Time | State                   |
Info                                 | Progress |
+----+-------------+------------------+------------+---------+------+-------------------------
+-------------------------------------+----------+
| 64 | root        | 172.17.0.1:49082 | partitions | Query   |  258 | copy to tmp table       | ALTER TABLE
`t` REMOVE PARTITIONING |    3.657 |
+----+-------------+------------------+------------+---------+------+-------------------------
+-------------------------------------+----------+
```

Watching the performance of the operation:

```
% top
    PID USER      PR  NI    VIRT    RES    SHR S  %CPU %MEM     TIME+ COMMAND
 358814 qserv     20   0 2525456 230584  11252 S 100.0  0.2   1475:44 mysqld


% iostat -m 1 | grep sdb
Device:            tps    MB_read/s    MB_wrtn/s    MB_read    MB_wrtn

sdb             160.00        20.00         0.00         20          0
sdb             178.00        22.00         0.34         22          0
sdb             176.00        22.00         0.00         22          0
sdb             176.00        22.00         0.00         22          0
sdb             176.00        22.00         0.00         22          0
sdb             192.00        24.00         0.00         24          0
sdb             192.00        24.00         0.00         24          0
sdb             192.00        24.00         0.00         24          0
sdb             192.00        24.00         0.00         24          0
sdb             192.00        24.00         0.00         24          0
sdb             192.00        24.00         0.00         24          0
sdb             208.00        26.00         0.00         26          0
sdb             208.00        26.00         0.00         26          0
sdb             191.00        24.00         0.00         24          0
sdb             208.00        26.00         0.00         26          0
```

File system view:

```
ls -alh /qserv/replication/mysql/partitions/
total 102G
drwx------ 2 qserv qserv 4.0K Feb 14 13:50 .
drwxr-xr-x 5 qserv qserv 4.0K Feb 11 15:36 ..
-rw-rw---- 1 qserv qserv   65 Feb 11 15:36 db.opt
-rw-rw---- 1 qserv qserv 1.5K Feb 14 13:50 #sql-1_40.frm
-rw-rw---- 1 qserv qserv 7.8G Feb 14 13:56 #sql-1_40.MYD
-rw-rw---- 1 qserv qserv 1.2G Feb 14 13:56 #sql-1_40.MYI
-rw-rw---- 1 qserv qserv 2.1K Feb 11 17:28 t.frm
-rw-rw---- 1 qserv qserv  100 Feb 11 17:28 t.par
-rw-rw---- 1 qserv qserv 8.6G Feb 11 17:37 t#P#p0.MYD
-rw-rw---- 1 qserv qserv 1.4G Feb 11 19:09 t#P#p0.MYI
-rw-rw---- 1 qserv qserv 9.0G Feb 11 22:24 t#P#p1.MYD
-rw-rw---- 1 qserv qserv 1.4G Feb 11 22:24 t#P#p1.MYI
-rw-rw---- 1 qserv qserv 9.0G Feb 11 22:25 t#P#p2.MYD
-rw-rw---- 1 qserv qserv 1.4G Feb 11 22:25 t#P#p2.MYI
-rw-rw---- 1 qserv qserv 9.0G Feb 11 22:19 t#P#p3.MYD
-rw-rw---- 1 qserv qserv 1.4G Feb 11 22:19 t#P#p3.MYI
-rw-rw---- 1 qserv qserv 9.0G Feb 11 22:23 t#P#p4.MYD
-rw-rw---- 1 qserv qserv 1.4G Feb 11 22:23 t#P#p4.MYI
-rw-rw---- 1 qserv qserv 9.0G Feb 11 22:24 t#P#p5.MYD
-rw-rw---- 1 qserv qserv 1.4G Feb 11 22:24 t#P#p5.MYI
-rw-rw---- 1 qserv qserv 9.0G Feb 11 22:25 t#P#p6.MYD
-rw-rw---- 1 qserv qserv 1.4G Feb 11 22:25 t#P#p6.MYI
-rw-rw---- 1 qserv qserv 9.0G Feb 11 22:25 t#P#p7.MYD
-rw-rw---- 1 qserv qserv 1.4G Feb 11 22:25 t#P#p7.MYI
-rw-rw---- 1 qserv qserv 9.0G Feb 11 22:25 t#P#p8.MYD
-rw-rw---- 1 qserv qserv 1.4G Feb 11 22:25 t#P#p8.MYI
-rw-rw---- 1 qserv qserv    0 Feb 11 17:28 t#P#p9.MYD
-rw-rw---- 1 qserv qserv 1.0K Feb 11 19:09 t#P#p9.MYI
```

OBSERVATION: apparently this is is a **SEQUENTIAL** operation performing at **20 MB/s** (which is the same speed as loading data into tables using *LOAD DATA INFILE*.

Timing report:

```
SELECT NOW();
2019-02-14 19:50:41

SELECT NOW();
2019-02-14 21:07:07
```

File system view after completion of the transformation:

```
ls -alh /qserv/replication/mysql/partitions/
total 95G
drwx------ 2 qserv qserv   75 Feb 14 15:07 .
drwxr-xr-x 5 qserv qserv 4.0K Feb 11 15:36 ..
-rw-rw---- 1 qserv qserv   65 Feb 11 15:36 db.opt
-rw-rw---- 1 qserv qserv 1.5K Feb 14 13:50 t.frm
-rw-rw---- 1 qserv qserv  81G Feb 14 15:05 t.MYD
-rw-rw---- 1 qserv qserv  15G Feb 14 15:05 t.MYI
```

3.3.1. There is some small performance penalty for partitioning tables

This could be seen in this *GROUP BY* query run after removing the partitions:

```
SELECT COUNT(*) FROM t;
+-----------+

| COUNT(*)  |

+-----------+
| 900000000 |
+-----------+
1 row in set (0.03 sec)

SELECT qserv_load_id,COUNT(*) FROM t GROUP BY qserv_load_id;
+---------------+-----------+
| qserv_load_id | COUNT(*)  |
+---------------+-----------+
|             0 | 100000000 |
|             1 | 100000000 |
|             2 | 100000000 |
|             3 | 100000000 |
|             4 | 100000000 |
|             5 | 100000000 |
|             6 | 100000000 |
|             7 | 100000000 |
|             8 | 100000000 |
+---------------+-----------+
9 rows in set (7 min 16.41 sec)
```

The speed up for this particular query is roughly **30 %**. Note, there is no penalty in terms of the disk space utilization for partitioning tables.

3.3.2. Improved performance for parallel loading into multiple partitions

Replacing the combined *PRIMRY KEY*:

```
PRIMARY KEY (`id`,`qserv_load_id`)
```

with a simple non-unique *KEY*:

```
KEY (`qserv_load_id`)
```

has improved the overall performance for the loading into 8 partitions simultaneously:

```
# load time for each partition
8 minutes * 60 seconds + 14 seconds = 494 seconds


# Aggregate I/O rate
(8 partitions * 9000 MB) / 494 seconds =  146 MB/s
```

NOTE: the CPU utilization during the loading was close **800 %**.

4. References

https://dev.mysql.com/doc/refman/8.0/en/partitioning.html

https://dev.mysql.com/doc/refman/8.0/en/alter-table-partition-operations.html