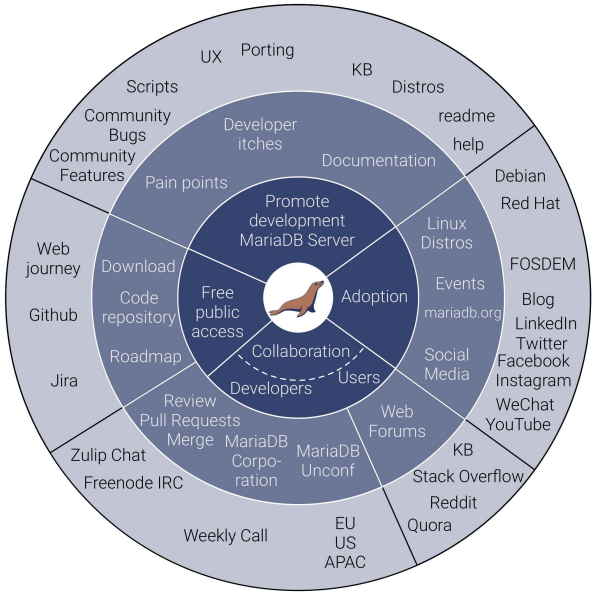


MariaDB Foundation

ColumnStore - First class citizen in MariaDB

Vicențiu Ciorbaru
Software Developer Team Lead
@ MariaDB Foundation

Percona Live 2021
2021 May - MariaDB Community Track

Conclusion

	InnoDB	ColumnStore
Insert 100 mil rows	3min 12s	10s
select count(*)	16.902s	0.45s
select max(unindexed_col)	20.435s	1.201s
select avg(unindexed_col)	25.101s	1.487s

Row based storage

Clients					
ID	Name	Enrollment Date	Country	City	Age
1	John	2021-04-01	DE	Berlin	24
2	Bill	2020-03-17	GB	London	46
3	Bob	2021-01-07	US	New York	37
4	Liu	2021-02-06	CN	Beijing	51

One file per table

Row based storage

Clients					
ID	Name	Enrollment Date	Country	City	Age
1	John	2021-04-01	DE	Berlin	24
2	Bill	2020-03-17	GB	London	46
3	Bob	2021-01-07	US	New York	37
4	Liu	2021-02-06	CN	Beijing	51

To select one column, requires accessing all table rows.

Row based storage

Clients					
ID	Name	Enrollment Date	Country	City	Age
1	John	2021-04-01	DE	Berlin	24
2	Bill	2020-03-17	GB	London	46
3	Bob	2021-01-07	US	New York	37
4	Liu	2021-02-06	CN	Beijing	51

Modifying all values in a column, updates all rows.

Row based storage

Clients					
ID	Name	Enrollment Date	Country	City	Age
1	John	2021-04-01	DE	Berlin	24
2	Bill	2020-03-17	GB	London	46
3	Bob	2021-01-07	US	New York	37
4	Liu	2021-02-06	CN	Beijing	51

Deleting a row, single seek and delete

Row based storage

Clients					
ID	Name	Enrollment Date	Country	City	Age
1	John	2021-04-01	DE	Berlin	24
2	Bill	2020-03-17	GB	London	46
3	Bob	2021-01-07	US	New York	37
4	Liu	2021-02-06	CN	Beijing	51
5	Jane	2021-04-01	IE	Dublin	31

Inserting a row just appends

Column based storage

Clients				
ID	Name	Enrollment Date	Country	Age
1	John	2021-04-01	DE	24
2	Bill	2020-03-17	GB	46
3	Bob	2021-01-07	US	37
4	Liu	2021-02-06	CN	51

One file per **column**

Column based storage

Clients				
ID	Name	Enrollment Date	Country	Age
1	John	2021-04-01	DE	24
2	Bill	2020-03-17	GB	46
3	Bob	2021-01-07	US	37
4	Liu	2021-02-06	CN	51

Reading a column accesses only that column

Column based storage

Clients				
ID	Name	Enrollment Date	Country	Age
1	John	2021-04-01	DE	24
2	Bill	2020-03-17	GB	46
3	Bob	2021-01-07	US	37
4	Liu	2021-02-06	CN	51

Modifying a column accesses only that column

Column based storage

Clients				
ID	Name	Enrollment Date	Country	Age
1	John	2021-04-01	DE	24
2	Bill	2020-03-17	GB	46
3	Bob	2021-01-07	US	37
4	Liu	2021-02-06	CN	51

Deleting a row accesses **all** files

Column based storage

Clients				
ID	Name	Enrollment Date	Country	Age
1	John	2021-04-01	DE	24
2	Bill	2020-03-17	GB	46
3	Bob	2021-01-07	US	37
4	Liu	2021-02-06	CN	51
5	Jane	2021-04-01	IE	31

Inserting a row accesses **all** files

Differences between OLTP and OLAP

OLTP

- Short transactions
- Many concurrent connections.
- Data inserted in small increments

OLAP

- Fewer, but complex queries.
- Few concurrent connections
- Bulk loading of data

Architectures

MariaDB & InnoDB

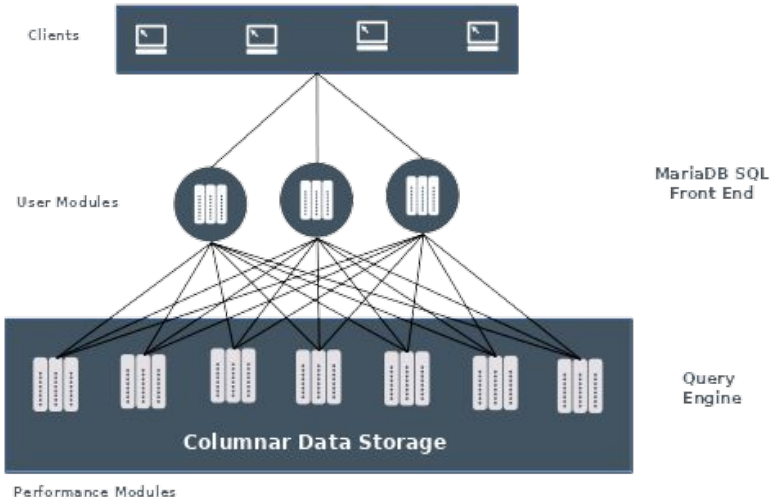
- Threadpool to manage incoming connections.
- Optimized for parallel transactions, each query runs in a single thread.

MariaDB & ColumnStore

- Query is passed down to ColumnStore engine
- ColumnStore executes queries in parallel

Architectures

MariaDB ColumnStore



Installing MariaDB ColumnStore

- Setup a MariaDB repository
  https://mariadb.org/download/#mariadb-repositories

Installing MariaDB ColumnStore

- `# apt-get install mariadb-plugin-columnstore`

Installing MariaDB ColumnStore

- `# apt-get install mariadb-plugin-columnstore`
- `# systemctl start mariadb-columnstore`

Installing MariaDB ColumnStore

- `# apt-get install mariadb-plugin-columnstore`
- `# systemctl start mariadb-columnstore`

DONE!

Inserting data into MariaDB ColumnStore

1. Insert via regular INSERT statements:

```
MariaDB [statistics]>
    -> INSERT into sat_scores (id, score) values (100, 1060);
Query OK, 1 row affected (0.240 sec)
```

- For 100mil rows -> 277 Days!

Inserting data into MariaDB ColumnStore

2. Insert via INSERT ... SELECT

```
MariaDB [statistics]>
  insert into sat_scores_cs (id, score)
    (select * from sat_scores);
Query OK, 100000000 rows affected (2 min 2.314 sec)
```

- As fast as it takes InnoDB to scan the whole table
  - Single threaded because of MariaDB's single thread select.

Inserting data into MariaDB ColumnStore

3. Convert InnoDB table to ColumnStore

```
MariaDB [statistics]> alter table sat_scores engine=ColumnStore;
ERROR 1069 (42000): Too many keys specified; max 0 keys allowed
```

- ColumnStore doesn't support keys

- Performance is as slow as single insert

Inserting data into MariaDB ColumnStore

4. LOAD DATA INFILE

```
MariaDB [statistics]>
    load data infile './data.csv' into table sat_scores_cs
         fields terminated by ',' optionally enclosed by '\'';
Query OK, 100000000 rows affected (1 min 13.297 sec) done
Records: 100000000  Deleted: 0  Skipped: 0  Warnings: 0
```

- Uses ColumnStore's **cpimport** tool behind the scenes.

Inserting data into MariaDB ColumnStore

5. Use `cpimport` directly

```
sudo cpimport statistics sat_scores_cs -s ,  < data.csv

...
2021-04-30 20:35:31 (29561) INFO : For table statistics.sat_scores_cs: 100000000 rows
processed and 100000000 rows inserted.
2021-04-30 20:35:32 (29561) INFO : Bulk load completed, total run time : 10.0944 seconds
```

- Best tool to use for fast inserts!
- Configurable number of worker threads.

Conclusion

	InnoDB	ColumnStore
Insert 100 mil rows	3min 12s	10s
select count(*)	16.902s	0.82s
select max(unindexed_col)	20.435s	1.201s
select avg(unindexed_col)	25.101s	1.487s

Selecting data from MariaDB ColumnStore

- Uses regular SELECT interface

```
MariaDB [statistics]> select count(*) from sat_scores_cs where id > 2000000;
+----------+
| count(*) |
+----------+
| 97999999 |
+----------+
1 row in set (0.806 sec)

MariaDB [statistics]> explain select count(*) from sat_scores_cs where id > 2000000;
+------+---------------+-------+...
| id   | select_type   | table |...
+------+---------------+-------+...
|    1 | PUSHED SELECT | NULL  |...
+------+---------------+-------+...
```

Selecting data from MariaDB ColumnStore

- Uses regular SELECT interface

```
MariaDB [statistics]> select count(*) from sat_scores_cs where id > 2000000;
+----------+
| count(*) |
+----------+
| 97999999 |
+----------+
1 row in set (0.806 sec)

MariaDB [statistics]> explain select count(*) from sat_scores_cs where id > 2000000;
+------+---------------+-------+...
| id   | select_type   | table |...
+------+---------------+-------+...
|    1 | PUSHED SELECT | NULL  |...
+------+---------------+-------+...
```

Query Plan not that useful

Selecting data from MariaDB ColumnStore

- ColumnStore QueryPlan has a different format

```
MariaDB [statistics]> select calSetTrace(1);
```

Selecting data from MariaDB ColumnStore

- ColumnStore QueryPlan has a different format

```
MariaDB [statistics]> select calSetTrace(1);

MariaDB [statistics]> select row_number() over (),
                             max(t1.score)
                      from sat_scores_cs t1
                      group by id mod 100000;
...
100000 rows in set, 1 warning (7.569 sec)
```

Selecting data from MariaDB ColumnStore

- ColumnStore QueryPlan has a different format

```
MariaDB [statistics]> select calSetTrace(1);

MariaDB [statistics]> select row_number() over (),
                             max(t1.score)
                      from sat_scores_cs t1
                      group by id mod 100000;
...
100000 rows in set, 1 warning (7.569 sec)

MariaDB [statistics]> select calGetTrace()\G
*************************** 1. row ***************************
calGetTrace():
Desc Mode Table TableOID ReferencedColumns PIO   LIO    PBE Elapsed Rows
BPS  PM   t1    3002     (id,score)        97660 695329 0   7.440   19200000
TAS  UM   -     -        -                 -     -      -   7.385   100000
WFS  UM   -     -        -                 -     -      -   0.018   100000
TNS  UM   -     -        -                 -     -      -   0.004   100000
```

Selecting data from MariaDB ColumnStore

- ColumnStore QueryPlan has a different format

```
MariaDB [statistics]> select calSetTrace(1);

MariaDB [statistics]> select row_number() over (),
                             max(t1.score)
                      from sat_scores_cs t1
                      group by id mod 100000;
...
100000 rows in set, 1 warning (7.569 sec)

                  ics]> select calGetTrace()\G
                  *********** 1. row ***************************
Desc Mode Table TableOID ReferencedColumns PIO   LIO    PBE Elapsed Rows
BPS  PM   t1    3002     (id,score)        97660 695329 0   7.440   19200000
TAS  UM   -     -        -                 -     -      -   7.385   100000
WFS  UM   -     -        -                 -     -      -   0.018   100000
TNS  UM   -     -        -                 -     -      -   0.004   100000
```

Execution Steps

Selecting data from MariaDB ColumnStore

- ColumnStore QueryPlan has a different format

```
MariaDB [statistics]> select calSetTrace(1);

MariaDB [statistics]> select row_number() over (),
                             max(t1.score)
                      from sat_scores_cs t1
                      group by id mod 100000;
...
100000 rows in set, 1 warning (7.569 sec)

MariaDB [statistics]> select calGetTrace()\G
*************************** 1. row ***************************
calGetTrace():
Desc Mode Table TableOID ReferencedColumns PIO   LIO    PBE Elapsed Rows
BPS  PM   t1    3002     (id,score)        97660 695329 0   7.440   19200000
TAS  UM   -     -        -                 -     -      -   7.385   100000
WFS  UM   -     -        -                 -     -      -   0.018   100000
TNS  UM   -     -        -                 -     -      -   0.004   100000
```

Time spent at each step

Selecting data from MariaDB ColumnStore

- ColumnStore QueryPlan has a different format

```
MariaDB [statistics]> select calSetTrace(1);

MariaDB [statistics]> select row_number() over (),
                             max(t1.score)
                      from sat_scores_cs t1
                      group by id mod 100000;
...
100000 rows in set, 1 warning (7.569 sec)

MariaDB [statistics]> select calGetTrace()\G
*************************** 1. row ***************************
calGetTrace():
Desc Mode Table TableOID ReferencedColumns PIO   LIO    PBE Elapsed Rows
BPS  PM   t1    3002     (id,score)        97660 695329 0   7.440   19200000
TAS  UM   -     -        -                 -     -      -   7.385   100000
WFS  UM   -     -        -                 -     -      -   0.018   100000
TNS  UM   -     -        -                 -     -      -   0.004   100000
```

Time spent at each step

Selecting data from MariaDB ColumnStore

```
MariaDB [statistics]> select calGetTrace()\G
*************************** 1. row ***************************
calGetTrace():
Desc Mode Table TableOID ReferencedColumns PIO   LIO    PBE Elapsed Rows
BPS  PM   t1    3002     (id,score)        97660 695329 0   7.440   19200000
TAS  UM   -     -        -                 -     -      -   7.385   100000
WFS  UM   -     -        -                 -     -      -   0.018   100000
TNS  UM   -     -        -                 -     -      -   0.004   100000
```

- Documentation available at: https://mariadb.com/kb/en/analyzing-queries-in-columnstore/#viewing-the-columnstore-query-plan

CrossEngine Joins

- Requires ColumnStore to connect to MariaDB to fetch external engines data.

- Default values:

```
CrossEngineSupport.Host     = 127.0.0.1
CrossEngineSupport.Port     = 3306
CrossEngineSupport.User     = root
CrossEngineSupport.Password =
CrossEngineSupport.TLSCA          =
CrossEngineSupport.TLSClientCert =
CrossEngineSupport.TLSClientKey  =
```

- `mcsGetConfig` and `mcsSetConfig` to update the configuration

Conclusions

- ColumnStore is available as a separate Storage Engine in MariaDB since **10.5.4**
- Compared to traditional MariaDB Storage Engines, it has more moving parts.
- Much faster than InnoDB for Analytical Queries
- Inserts must be done in bulk
- Specific configuration steps, separate from mariadb.cnf
- Easy to start up with systemd, different flow to examine queries.

Thank you!

Contact details:

vicentiu@mariadb.org

About:

https://mariadb.org/vicentiu