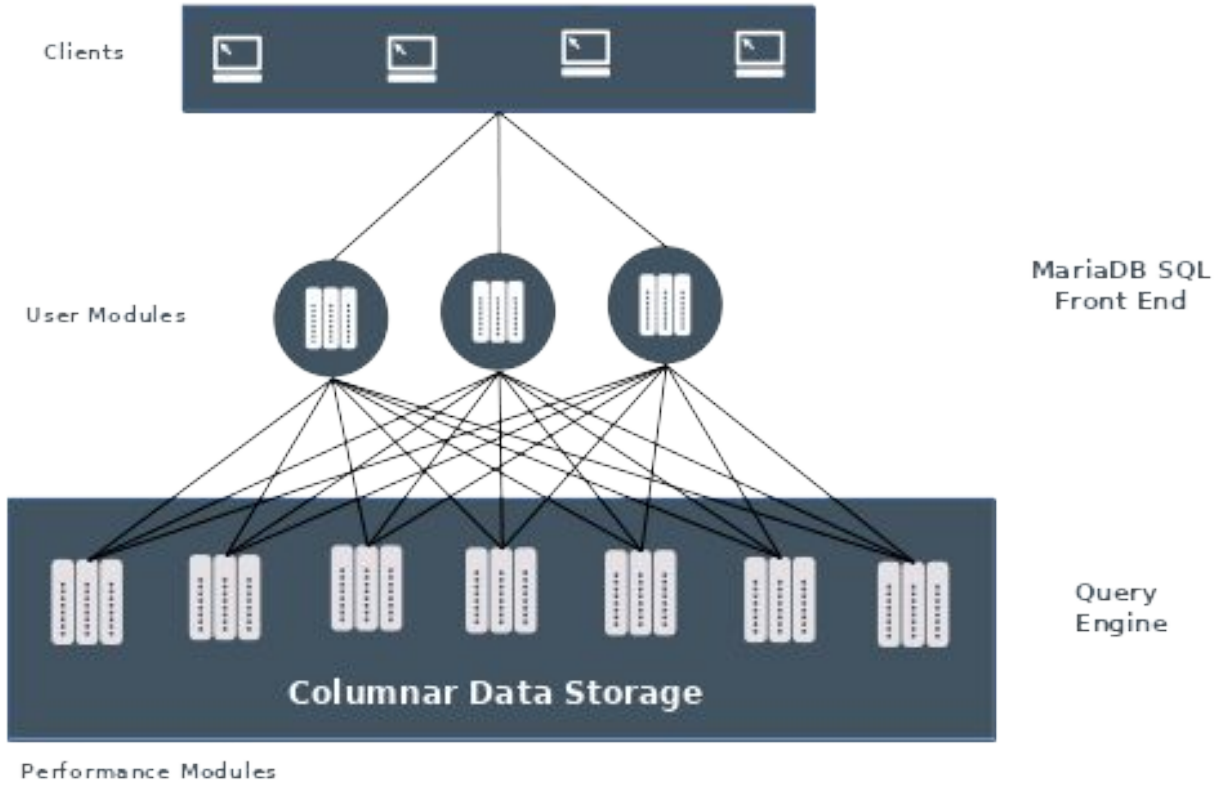
What is new in analytics for MariaDB

Roman Nozdrin
MariaDB Corporation

What is MariaDB Columnstore?

- OLAP engine
  - columnar-oriented
  - Massive Parallel Processing
  - 2-Tier distributed storage
    - PM-s
    - dbroots



MariaDB

ColumnStore to MariaDB version mapping

- A.B.P -> Y.M.P
  - A, I, P are major, minor, patch numbers, e.g. MCS 6.4.7
  - Y - year, M - month when engine release is first published, e.g. MCS 23.02.01

- IMHO The most obscure topic in this speech is engine version to Community Server version mapping
  - MDB 10.5 -> MCS 1.5.P
  - MDB 10.6 - 10.10 -> MCS 6.I.P
  - next MDB 10.11.P -> MCS 23.02.P (scheduled for May-April 2023)

Current stable features: MariaDB Columnstore 6.4.7

- filtering vectorization for x86_64
  - SIMD processing is not the ultimate answer about life universe and everything
  - Microbenchmarks demonstrated 10x speedup but full pipeline gets 30-40% speedup on filtering intensive queries. (MCOL-4809)
- external GROUP BY
  - It hash two phases
    - 1st phase renders partial aggregates and stores them on disk
    - 2nd phase merges partial aggregates
      - needs 2x memory(comparing with the 1st phase) in worst-case
  - Disabled by default. To enable change /etc/columnstore/Columnstore.xml

```
<RowAggregation>

    <AllowDiskBasedAggregation>Y</AllowDiskBasedAggregation>

</RowAggregation>
```

MariaDB

Current stable features: MariaDB Columnstore 6.4.7

- LZ4 compression for data files
  - Snappy still delivers better compression(about 5%) so there is a tradeoff between space/decompression speed
  - Disabled by default in current stable 6.4.7
  - To enable
    - SET columnstore_compression_type=LZ4;
  - Will become a new default since 23.02

Next stable features: MariaDB Columnstore 23.02

- ARM 64 bit support.
  - According with our measurements ARM builds work faster comparing with x86_64 running perf tests.
  - DML-heavy workload consumes 5% more RAM comparing with x86_64 builds though.

- mcsRebuildEM tool
  - There are two parts for a data stored in Columnstore:
    - data itself
    - metadata to search for a data unit in a cluster
  - Previously once one loses his meta there were no way to access data
  - Now mcsRebuildEM can be used to restore meta from the data itself
  - Data files must be created MCS >= 6.4.4

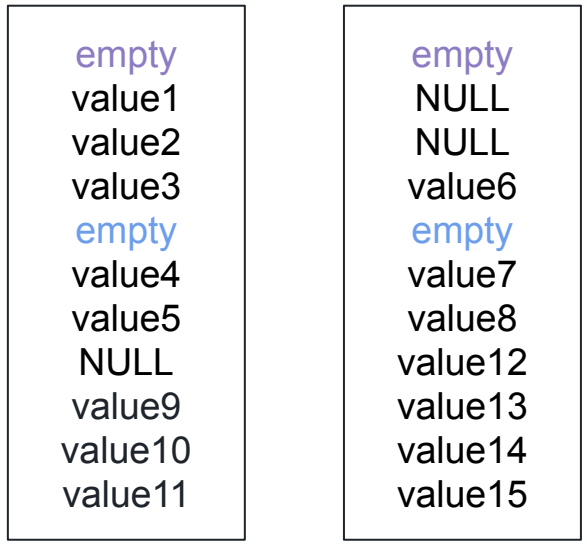
- distributed JSON functions support AKA MCOL-785
- There are two functions unimplemented yet: JSON_OBJECTAGG and JSON_TABLE

Next stable features: MariaDB Columnstore 23.02

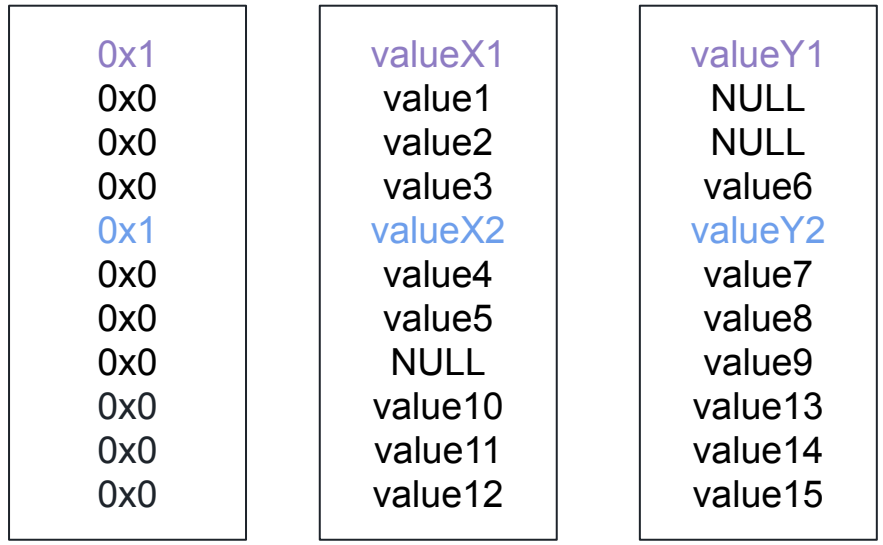
- Auxiliary column AKA MCOL-5021. This speeds up DELETE from 3x up to 50x(depending on SQL schema)
- Has an additional speed-up config option

```
<WriteEngine>
    <FastDelete>y</FastDelete>
</WriteEngine>
```

No fast DELETE column files layout



Fast DELETE column files layout



MariaDB

Next stable features: MariaDB Columnstore 23.02

- Extent Map scalability improvement
  - EM does:
    - block number(cluster unique) to (oid, node, partition, segment) tuple mapping
    - (oid, node, partition, segment) tuple mapping to block number mapping
  - EM was an array with O(n) lookup complexity
  - Array is replaced with RBTree
    - Block to tuple mapping now has O(logN) complexity
  - EM Index is a new 2 hash map layers + 1 vector “burger” to map tuple to block
    - Tuple(or partial tuple) to block mapping now has amortized O(1) complexity

---

Before

2023-01-15 08:06:21 (2680877) INFO : PreProcessing check completed

Extent Map size is roughly 300 MB

2023-01-15 08:06:21 (2680877) INFO : preProcess completed, run time for this step : 28.0307622 seconds

After

2023-01-15 08:15:01 (2680533) INFO : PreProcessing check completed

Extent Map size is 325 MB
EM Index size is 250 MB

2023-01-15 08:15:01 (2680533) INFO : preProcess completed, run time for this step : 4.0503215 seconds

MariaDB

Next stable features: MariaDB Columnstore 23.02

- PrimProc and ExeMgr processes merge
- Columnstore is a bunch of linux processes exchanging messages over TCP
- Every SELECT is processed by a query coordinator(ExeMgr) and group of workers(PrimProc-s)

- Before PP and EM merge
  - Same node communication goes over loopback
  - No compression used for traffic goes over loopback
- After the merge
  - Same node communication leverages in-memory messaging queues
    - No serialization needed
- 4-7% overall speedup for same host communication

Next stable features: MariaDB Columnstore 23.02

- UNION pushdown
- MDB has variety of pushdown types to hand a query(or its part) over to MCS
- Pushdown types are handled by... handlers: Select Handler, Derived Handler
- UNION processing path differs from SELECT processing path

- Before
  - SELECT * FROM (SELECT c1 FROM t1 UNION ALL SELECT c2 FROM t2) s
- After
  - SELECT c1 FROM t1 UNION ALL SELECT c2 FROM t2

Next stable features: MariaDB Columnstore 23.02

- TPC-H full support
- Scalar correlated subquery with an aggregate (TPC-H q2, q17)
  - `SELECT * FROM t1, t2 WHERE t1.a = t2.a AND t2.b > (SELECT avg(b) FROM t2 WHERE t1.a = t2.a GROUP BY a);`
  - Translated into JOIN (JOIN(aggregate(t2), t1), t2) (t1 is a small side in this case)

- Common conjunction extraction re-write (TPC-H q19)
- Before
  - `SELECT a.x, a.p, b.q FROM a, b WHERE (a.x = b.x and a.p = 1) OR (a.x = b.x and b.q = 3);`
- After
  - `SELECT a.x, a.p, b.q FROM a, b WHERE a.x = b.x AND (a.p = 1 OR b.q = 3);`

MariaDB

Next stable features: MariaDB Columnstore 23.02

- External DISTINCT
- A small detour.
  - Query text -> tree-ish SELECT_LEX -> tree-ish CSEP -> query execution program JobList
  - JobList step = TupleJoinStep | TupleAggregateStep | TupleaNexStep ...
  - TupleaNexStep : Maybe ORDER BY -> Maybe DISTINCT -> Maybe HAVING
    - Sorting and Distinct were tightly coupled together
    - DISTINCT was based on a hash-map lookup

- ORDER BY and DISTINCT are decoupled now.
- Hash map-based DISTINCT is replaced by GROUP BY facility
  - GROUP BY supports external operation(since MCS 6.x) and it is also parallelized.

Next stable features: MariaDB Columnstore 23.02

- In-memory ORDER BY changes
- ORDER BY code was based on Priority Queue that is not so bad for TOP-k queries.
  - SELECT c1, c2...cN FROM t1 ORDER BY c1 LIMIT 10
- PQ timings are terrifying when ORDER BY has either no LIMIT or LIMIT number is big

---

- New algo has 2.5 phases comparing with 2 phases previously:
  - 1st phase to produce a sorted runs in parallel
  - 1.5th calculates nonoverlapping permutation ranges for the 2nd phase
  - 2nd phase efficiently merges non-overlapping previous phase results in parallel
- 1st phase threads and all but one 2nd phase threads use modified pdqsort code.
  - The first thread from the 2nd phase uses merge sort that reduces time-to-first-results value.
- TNS uses PQ-based for TOP-k and the new algo for other ORDER BY use cases.

Next stable features: MariaDB Columnstore 23.02

- In-memory ORDER BY comparison
- TPC-DS scale factor 10:
  - 500 000 in customers (char columns)
  - 14 000 000 roughly in catalog_sales (integer columns)
- Intel(R) Core(TM) i7-10750H CPU @ 2.60GHz, 16 GB RAM
- Integer sorting key columns
  - SELECT * FROM (SELECT * FROM catalog_sales ORDER BY cs_quantity, cs_item_sk) s LIMIT 100000;
    - Before : 16,673 sec
    - After : 3,784 sec
- Char sorting key columns
  - select * from (select * from customer order by c_first_name, c_last_name) s limit 100000;
    - Before : 2,029 sec
    - After : 0,902 sec

MariaDB

Links

- https://github.com/mariadb-corporation/mariadb-columnstore-engine
- https://jira.mariadb.org/projects/MCOL/issues

Thank you