What is MariaDB 5.5?

Daniel Bartholomew
Monty Program, Ab

Oct 2012

Abstract

MariaDB 5.5 is the current stable, generally available (GA) release of MariaDB. It builds upon — and includes several major new features and changes on top of — MySQL®. It is a compelling and attractive upgrade for all MySQL users, large and small.

This whitepaper describes several features of MariaDB 5.5 and how they improve upon the features in MySQL 5.5 (the current GA version of MySQL) and the upcoming MySQL 5.6 (the current development version of MySQL).

Contents

New Functionality in MariaDB

Performance

Subqueries and the Query Optimizer

One of the biggest recent improvements in MariaDB revolves around radically improving the performance of subqueries, joins, and single-table queries over large data sets.

Subqueries in MariaDB are finally usable in practice. Ever since subqueries were introduced into MySQL it has been necessary to rewrite them manually into joins or into separate queries if you cared at all about performance. MariaDB ends this with its efficient handling for all kinds of subqueries.[1]

Subqueries are not the only things improved in the MariaDB query optimizer. Other improvements include things like optimizations for derived tables and views, disk access, joins, and others.[2]

An optimizer feature comparison matrix for MariaDB and MySQL is available in *Appendix A: Optimizer Feature Comparison Matrix*.

Thread pool

The task of scalable server software (and a DBMS like MariaDB is an example of such software) is to maintain top performance with an increasing number of clients. MySQL traditionally has assigned a thread for every client connection, and as the number of concurrent users grows this model shows performance drops. Many active threads are a performance killer. An ideal solution is to maintain a lower number of threads than the number of clients and at least a single active thread for each CPU on the machine.

MariaDB includes a dynamic and adaptive pool that itself takes care of creating new threads in times of high demand and retiring threads if they have nothing to do. This is a complete reimplementation of the legacy pool-of-threads scheduler, with the following goals:

- Make the pool dynamic (grow and shrink whenever required)
- Minimize the overhead required to maintain threadpool itself
- Make the best use of underlying OS capabilities (use the native OS threadpool if provided, and use the best I/O multiplexing method available otherwise)
- Limit the resources uses by threads

There are currently two different low-level threadpool implementations in MariaDB: one for Windows which utilizes Windows-native threadpool, and one for Unix-like systems.

This open-source feature is comparable in functionality to the closed-source thread pool found in MySQL Enterprise Edition, but the MariaDB threadpool has a more efficient implementation and better performance.[3]

Group Commit for the Binary Log

MariaDB (and MySQL) does an fsync() call on the transaction log file / binary log file during a transaction COMMIT in order to ensure that data is stored durably on the disk. An fsync() is a time-consuming operation, and can easily limit throughput in terms of the number of commits per second which can be sustained. The idea with group commit is to amortize the costs of each fsync() over multiple commits from multiple parallel transactions.

If there are say 10 transactions in parallel trying to commit, MariaDB will force all of them to disk at once with a single fsync(), rather than do one fsync() for each. This can greatly reduce the need for fsync() calls, and consequently greatly improve the commits-per-second throughput.

This feature works automatically; there are no options needed to enable it, and there are no negative consequences: performance, reliability, or otherwise. Not everyone will see a benefit from it, but for those who have a workload with lots of parallel transactions per second wanting to commit, the benefits can be significant.[4]

Features

Extended Keys

The Extended Keys feature in MariaDB makes use of existing components of InnoDB/XtraDB keys to generate more efficient execution plans. Using these components in many cases allows the server to generate and use execution plans which employ index-only look-ups, which results in better performance.[5]

NoSQL

MariaDB includes some features for users who have workloads which are better suited to NoSQL-type solutions. Two of these are HandlerSocket and Dynamic Columns.

HandlerSocket is a NoSQL plugin for MariaDB which gives you direct access to InnoDB/XtraDB tables. It works as a daemon inside the mysqld process, accepting TCP connections, and executing requests from clients. HandlerSocket does not support SQL queries. Instead, it supports simple CRUD operations on tables.

HandlerSocket can be much faster than mysqld/libmysql in some cases because it has lower CPU, disk, and network overhead.[6]

Another NoSQL feature in MariaDB is Dynamic Columns. This features allows you to have a different set of “virtual columns” for each row in a given table with the ability to add or remove columns from each row at any time.

This allows you to to solve problems like a store application, where you have a lot of different things — such as t-shirts and phones — and you want to store different attributes for each item. This is something that traditionally is very hard to do in a relational database.

Dynamic columns works by storing the extra columns in a blob and providing a small set of functions to manipulate them.[7]

In MariaDB 10.0 there is a COLUMN_JSON function which converts all dynamic column record content to a JSON array of objects.[8]

Microsecond Support

In today's world, it sometimes isn't enough to track or log certain data to the second. Sometimes more precision is needed, so the developers have added microsecond support to MariaDB.

The amount of precision is configurable and can be specified for any TIME, DATETIME, or TIMESTAMP column, in parentheses, after the type name. The datetime precision specifies number of digits after the decimal dot and can be any integer number from 0 to 6. If no precision is specified it is assumed to be 0, for backward compatibility reasons. Temporal functions such as NOW() and CURTIME() also support microseconds.[9]

Non-blocking Client Library

MariaDB supports non-blocking operation in the client-library. This allows an application to start a query or other operation against the database, and then continue to do other work (in the same thread) while the request is sent over the network, the query is processed in the server, and the result travels back. As parts of the result become ready, the application can — at its leisure — call back into the library to continue processing, repeating this until the operation is completed.

Non-blocking operation is implemented entirely within the client library. This means no special server support is necessary and non-blocking operation works with any version of the MariaDB or MySQL server, the same as the normal blocking API. It also means that it is not possible to have two queries running at the same time on the same connection (this is a protocol limitation). But a single thread can have any number of non-blocking queries running at the same time, each using its own MYSQL connection object.[10,11]

What does it mean that MariaDB is based on MySQL?

MySQL Community Edition is an open-source database. This means that the source code is freely available under a license which allows reuse and modification. MySQL Enterprise Edition is the Community Edition plus some proprietary, closed-source additions.

MariaDB started as a branch of MySQL Community Edition. What this means is that the MariaDB developers started with a copy of the MySQL source code. They then proceeded to make the modifications, additions, and improvements mentioned in the previous section. As new versions of MySQL are released those changes are added to MariaDB as well.

All code added to MariaDB, new code from MariaDB contributors and code imported from MySQL, is audited for quality. If necessary, the code is re-engineered and enriched. On occasion, entire sections of imported code are thrown away and rewritten from scratch (i.e. sometimes the idea may be good, but the implementation is not). Only code that meets the MariaDB developers' rigorous standards is committed into the MariaDB codebase.

By starting with MySQL, MariaDB is 100% compatible. MariaDB can read data, configuration, and all other files created or used by MySQL. Also, applications and clients which support MySQL automatically work with MariaDB without any changes.

The MariaDB developers are careful to maintain this compatibility. As mentioned previously, they regularly import changes made by the MySQL developers (checking first that the changes meet the stricter MariaDB standards) and are very careful when implementing new features to maintain compatibility with MySQL.

For example, two recent changes in MySQL include a new version of the InnoDB storage engine and improved DDL (data definition language) Locking. These greatly improved the scalability and performance of MySQL. As with all other open-source MySQL features, both of these are included in MariaDB.

Conclusion

So what is MariaDB? A great open-source database? An enhanced, drop-in replacement and upgrade for MySQL®? A logical choice for companies and individuals looking for a robust, scalable, and reliable relational database server? **All of the above.**[12]

Appendix A: Optimizer Feature Comparison Matrix

The “Optimizer” is the part of MariaDB and MySQL which takes the SQL (Structured Query Language) commands entered by users and applications and then determine the most efficient way to execute them. Having a large number of available optimization strategies helps the server quickly and efficiently execute a wide range of queries.

The table below compares MariaDB 5.5 to MySQL 5.5 and MySQL 5.6.

Feature	MariaDB 5.5	MySQL 5.5	MySQL 5.6 (dev)
Disk access optimizations			
Index Condition Pushdown (ICP)	YES	no	YES
Disk-sweep Multi-range read (DS-MRR)	YES	no	YES
DS-MRR with Key-ordered retrieval	YES	no	no
Index_merge / Sort_intersection	YES	no	no
Cost-based choice of range vs. index_merge	YES	no	no
ORDER BY ... LIMIT <small_limit>	no	no	YES
Use extended (hidden) primary keys for innodb/xtradb	YES	no	no
Join optimizations			
Batched key access (BKA)	YES	no	YES
Block hash join	YES	no	no
User-set memory limits on all join buffers	YES	no	no
Apply early outer table ON conditions	YES	no	no
Null-rejecting conditions tested early for NULLs	YES	no	no
Subquery optimizations			
In-to-exists	YES	YES	YES
Semi-join	YES	no	YES
Materialization	YES	no	YES
NULL-aware Materialization	YES	no	no
Cost choice of materialization vs. in-to-exists	YES	no	no
Subquery cache	YES	no	no
Fast explain with subqueries	YES	no	no
Optimization for derived tables / views			
Delayed materialization of derived tables / materialized views	YES	no	YES
Instant EXPLAIN for derived tables	YES	no	YES
Derived Table with Keys optimization	YES	no	YES
Fields of merge-able views and derived tables	YES	no	no
Execution control			
LIMIT ROWS EXAMINED rows_limit	YES	no	no
Optimizer control (optimizer switch)			
Systematic control of all optimizer strategies	YES	no	partial
EXPLAIN improvements			
Explain for DELETE, INSERT, REPLACE, and UPDATE	no	no	YES
EXPLAIN in JSON format	no	no	YES
More detailed and consistent EXPLAIN for subqueries	YES	no	no

Notes

[1] *Subquery Optimizations* documentation: `http://kb.askmonty.org/en/subquery-optimizations`
[2] *Query Optimizer* documentation: `http://kb.askmonty.org/en/query-optimizer`
[3] *Threadpool* documentation: `http://kb.askmonty.org/en/thread-pool-in-mariadb-55`
[4] *Binary Log Group Commit* documentation: `http://kb.askmonty.org/en/group-commit-for-the-binary-log`
[5] *Extended Keys* documentation: `http://kb.askmonty.org/en/extended-keys`
[6] *HandlerSocket* documentation: `http://kb.askmonty.org/en/handlersocket`
[7] *Dynamic Columns* documentation: `http://kb.askmonty.org/en/dynamic-columns`
[8] *Dynamic Columns in MariaDB 10.0:* `http://kb.askmonty.org/en/dynamic-columns-in-mariadb-10`
[9] *Microsecond* documentation: `http://kb.askmonty.org/en/microseconds-in-mariadb`
[10] *Non-blocking Client Library* documentation: `http://kb.askmonty.org/en/non-blocking-client-library`
[11] The MariaDB non-blocking client API and node.js: `http://blog.mariadb.org/mariadb-non-blocking-client-api-and-node-js`
[12] More information on MariaDB: `http://kb.askmonty.org/en/what-is-in-the-different-mariadb-releases`