MariaDB Tools Documentation

Release 6.0.0rc

Nov 04, 2021

CONTENTS

I Getting MariaDB Tools 3

II Tools 7

III Configuration 163

IV Miscellaneous 165

MariaDB Tools are a collection of advanced command-line tools used by MariaDB) support staff to perform a variety of MariaDB database and system tasks that are too difficult or complex to perform manually.

These tools are ideal alternatives to private or "one-off" scripts, because they are professionally developed, formally tested, and fully documented. They are also fully self-contained, so installation is quick and easy, and no libraries are installed.

MariaDB Tools were derived from MariaDB scripts, Percona Toolkit, Maatkit, PalaminoDB Tools and Aspersa, some of the best-known toolkits for MariaDB server administration. It is developed and supported by MariaDB.

Part I

Getting MariaDB Tools

CHAPTER

ONE

INSTALLING MARIADB TOOLS

MariaDB provides packages for most popular 64-bit Linux distributions:

- Debian 7 (“wheezy”)
- Debian 8 (“jessie”)
- Ubuntu 14.04 LTS (Trusty Tahr)
- Ubuntu 16.04 LTS (Xenial Xerus)
- Ubuntu 16.10 (Yakkety Yak)
- Ubuntu 17.04 (Zesty Zapus)
- Red Hat Enterprise Linux or CentOS 6 (Santiago)
- Red Hat Enterprise Linux or CentOS 7 (Maipo)

Note: MariaDB Tools should work on other DEB-based and RPM-based systems (for example, Oracle Linux and Amazon Linux AMI), but it is tested only on those listed above.

It is recommended to install MariaDB software from official repositories:

1. Configure repositories as described in MariaDB Enterprise Documentation.
2. Install MariaDB Tools using the corresponding package manager:
   - For Debian or Ubuntu:
     ```
     sudo apt-get install MariaDB-Tools
     ```
   - For RHEL or CentOS:
     ```
     sudo yum install MariaDB-Tools
     ```

1.1 Alternative Install Methods

You can also download the packages from the MariaDB Customer Portal and install it using tools like `dpkg` and `rpm`, depending on your system.

If you want to download a specific tool, use the following address: http://tools.mariadb.com/get

For example, to download the `mariadb-summary` tool, run:

```
wget tools.mariadb.com/get/mariadb-summary
```

Part II

Tools

CHAPTER TWO

MARIADB-ALIGN-OUTPUT

2.1 NAME

mariadb-align-output - Align output from other tools to columns.

2.2 SYNOPSIS

2.2.1 Usage

```
mariadb-align-output [FILES]
```

mariadb-align-output aligns output from other tools to columns. If no FILES are specified, STDIN is read.

If a tool prints the following output,

```
DATABASE TABLE   ROWS
foo      bar      100
long_db_name table  1
another  long_name 500
```

then **mariadb-align-output** reprints the output as,

```
DATABASE     TABLE     ROWS
foo          bar        100
long_db_name table        1
another      long_name  500
```

2.3 RISKS

mariadb-align-output is mature, proven in the real world, and well tested, but all database tools can pose a risk to the system and the database server. Before using this tool, please:

- Read the tool's documentation
- Review the tool's known "BUGS"
- Test the tool on a non-production server
- Backup your production server and verify the backups

2.4 DESCRIPTION

`mariadb-align-output` reads lines and splits them into words. It counts how many words each line has, and if there is one number that predominates, it assumes this is the number of words in each line. Then it discards all lines that don't have that many words, and looks at the 2nd line that does. It assumes this is the first non-header line. Based on whether each word looks numeric or not, it decides on column alignment. Finally, it goes through and decides how wide each column should be, and then prints them out.

This is useful for things like aligning the output of vmstat or iostat so it is easier to read.

2.5 OPTIONS

This tool accepts additional command-line arguments. Refer to the "SYNOPSIS" and usage information for details.

`--help`
Show help and exit.

`--version`
Show version and exit.

2.6 ENVIRONMENT

This tool does not use any environment variables.

2.7 SYSTEM REQUIREMENTS

You need Perl, and some core packages that ought to be installed in any reasonably new version of Perl.

2.8 AUTHORS

Cole Busby,Baron Schwartz, Brian Fraser, and Daniel Nichter

2.9 ABOUT THIS MARIADB TOOL

This tool is part of MariaDB client tools. This MariaDB Tool was forked from Percona Toolkit's pt-align in August, 2019. Percona Toolkit was forked from two projects in June, 2011: Maatkit and Aspersa. Those projects were created by Baron Schwartz and primarily developed by him and Daniel Nichter.

2.10 COPYRIGHT, LICENSE, AND WARRANTY

2.11 VERSION

`mariadb-align-output` 6.0.0rc

CHAPTER

THREE

MARIADB-ARCHIVER

3.1 NAME

mariadb-archiver - Archive rows from a MariaDB table into another table or a file.

3.2 SYNOPSIS

3.2.1 Usage

```
mariadb-archiver [OPTIONS] --source DSN --where WHERE
```

mariadb-archiver nibbles records from a MariaDB table. The –source and –dest arguments use DSN syntax; if COPY is yes, –dest defaults to the key's value from –source.

3.2.2 Examples

Archive all rows from oltp_server to olap_server and to a file:

```
mariadb-archiver --source h=oltp_server,D=test,t=tbl --dest h=olap_server \
  --file '/var/log/archive/%Y-%m-%d-%D.%t'                            \
  --where "1=1" --limit 1000 --commit-each
```

Purge (delete) orphan rows from child table:

```
mariadb-archiver --source h=host,D=db,t=child --purge \
  --where 'NOT EXISTS(SELECT * FROM parent WHERE col=child.col)'
```

3.3 RISKS

mariadb-archiver is mature, proven in the real world, and well tested, but all database tools can pose a risk to the system and the database server. Before using this tool, please:

- Read the tool's documentation
- Review the tool's known "BUGS"
- Test the tool on a non-production server
- Backup your production server and verify the backups

3.4 DESCRIPTION

mariadb-archiver is the tool I use to archive tables as described in http://tinyurl.com/mysql-archiving. The goal is a low-impact, forward-only job to nibble old data out of the table without impacting OLTP queries much. You can insert the data into another table, which need not be on the same server. You can also write it to a file in a format suitable for LOAD DATA INFILE. Or you can do neither, in which case it's just an incremental DELETE.

mariadb-archiver is extensible via a plugin mechanism. You can inject your own code to add advanced archiving logic that could be useful for archiving dependent data, applying complex business rules, or building a data warehouse during the archiving process.

You need to choose values carefully for some options. The most important are *--limit*, *--retries*, and *--txn-size*.

The strategy is to find the first row(s), then scan some index forward-only to find more rows efficiently. Each subsequent query should not scan the entire table; it should seek into the index, then scan until it finds more archivable rows. Specifying the index with the 'i' part of the *--source* argument can be crucial for this; use *--dry-run* to examine the generated queries and be sure to EXPLAIN them to see if they are efficient (most of the time you probably want to scan the PRIMARY key, which is the default). Even better, examine the difference in the Handler status counters before and after running the query, and make sure it is not scanning the whole table every query.

You can disable the seek-then-scan optimizations partially or wholly with *--no-ascend* and *--ascend-first*. Sometimes this may be more efficient for multi-column keys. Be aware that **mariadb-archiver** is built to start at the beginning of the index it chooses and scan it forward-only. This might result in long table scans if you're trying to nibble from the end of the table by an index other than the one it prefers. See *--source* and read the documentation on the `i` part if this applies to you.

3.5 MariaDB Galera Cluster

mariadb-archiver works with MariaDB Galera Cluster 10.1 and newer, but there are three limitations you should consider before archiving on a cluster:

Error on commit

> **mariadb-archiver** does not check for error when it commits transactions. Commits on Galera can fail, but the tool does not yet check for or retry the transaction when this happens. If it happens, the tool will die.

MyISAM tables

> Archiving MyISAM tables works, but MyISAM support in Galera is still experimental at the time of this release. There are several known bugs with Galera, MyISAM tables, and `AUTO_INCREMENT` columns. Therefore, you must ensure that archiving will not directly or indirectly result in the use of default `AUTO_INCREMENT` values for a MyISAM table. For example, this happens with *--dest* if *--columns* is used and the `AUTO_INCREMENT` column is not included. The tool does not check for this!

Non-cluster options

> Certain options may or may not work. For example, if a cluster node is not also a slave, then *--check-slave-lag* does not work. And since Galera tables are usually InnoDB, but InnoDB doesn't support `INSERT DELAYED`, then *--delayed-insert* does not work. Other options may also not work, but the tool does not check them, therefore you should test archiving on a test cluster before archiving on your real cluster.

3.6 OUTPUT

If you specify *--progress*, the output is a header row, plus status output at intervals. Each row in the status output lists the current date and time, how many seconds **mariadb-archiver** has been running, and how many rows it has archived.

If you specify *--statistics*, **mariadb-archiver** outputs timing and other information to help you identify which part of your archiving process takes the most time.

3.7 ERROR-HANDLING

mariadb-archiver tries to catch signals and exit gracefully; for example, if you send it SIGTERM (Ctrl-C on UNIX-ish systems), it will catch the signal, print a message about the signal, and exit fairly normally. It will not execute *--analyze* or *--optimize*, because these may take a long time to finish. It will run all other code normally, including calling after_finish() on any plugins (see "EXTENDING").

In other words, a signal, if caught, will break out of the main archiving loop and skip optimize/analyze.

3.8 OPTIONS

Specify at least one of *--dest*, *--file*, or *--purge*.

--ignore and *--replace* are mutually exclusive.

--txn-size and *--commit-each* are mutually exclusive.

--low-priority-insert and *--delayed-insert* are mutually exclusive.

--share-lock and *--for-update* are mutually exclusive.

--analyze and *--optimize* are mutually exclusive.

--no-ascend and *--no-delete* are mutually exclusive.

DSN values in *--dest* default to values from *--source* if COPY is yes.

--analyze

type: string

Run ANALYZE TABLE afterwards on *--source* and/or *--dest*.

Runs ANALYZE TABLE after finishing. The argument is an arbitrary string. If it contains the letter 's', the source will be analyzed. If it contains 'd', the destination will be analyzed. You can specify either or both. For example, the following will analyze both:

```
--analyze=ds
```

See https://mariadb.com/kb/en/library/analyze-table/ for details on ANALYZE TABLE.

--ascend-first

Ascend only first column of index.

If you do want to use the ascending index optimization (see *--no-ascend*), but do not want to incur the overhead of ascending a large multi-column index, you can use this option to tell **mariadb-archiver** to ascend only the leftmost column of the index. This can provide a significant performance boost over not ascending the index at all, while avoiding the cost of ascending the whole index.

See "EXTENDING" for a discussion of how this interacts with plugins.

--ask-pass
Prompt for a password when connecting to MariaDB.

--buffer
Buffer output to *--file* and flush at commit.

Disables autoflushing to *--file* and flushes *--file* to disk only when a transaction commits. This typically means the file is block-flushed by the operating system, so there may be some implicit flushes to disk between commits as well. The default is to flush *--file* to disk after every row.

The danger is that a crash might cause lost data.

The performance increase I have seen from using *--buffer* is around 5 to 15 percent. Your mileage may vary.

--bulk-delete
Delete each chunk with a single statement (implies *--commit-each*).

Delete each chunk of rows in bulk with a single `DELETE` statement. The statement deletes every row between the first and last row of the chunk, inclusive. It implies *--commit-each*, since it would be a bad idea to `INSERT` rows one at a time and commit them before the bulk `DELETE`.

The normal method is to delete every row by its primary key. Bulk deletes might be a lot faster. **They also might not be faster** if you have a complex `WHERE` clause.

This option completely defers all `DELETE` processing until the chunk of rows is finished. If you have a plugin on the source, its `before_delete` method will not be called. Instead, its `before_bulk_delete` method is called later.

WARNING: if you have a plugin on the source that sometimes doesn't return true from `is_archivable()`, you should use this option only if you understand what it does. If the plugin instructs **mariadb-archiver** not to archive a row, it will still be deleted by the bulk delete!

--[no]bulk-delete-limit
default: yes

Add *--limit* to *--bulk-delete* statement.

This is an advanced option and you should not disable it unless you know what you are doing and why! By default, *--bulk-delete* appends a *--limit* clause to the bulk delete SQL statement. In certain cases, this clause can be omitted by specifying `--no-bulk-delete-limit`. *--limit* must still be specified.

--bulk-insert
Insert each chunk with LOAD DATA INFILE (implies *--bulk-delete --commit-each*).

Insert each chunk of rows with `LOAD DATA LOCAL INFILE`. This may be much faster than inserting a row at a time with `INSERT` statements. It is implemented by creating a temporary file for each chunk of rows, and writing the rows to this file instead of inserting them. When the chunk is finished, it uploads the rows.

To protect the safety of your data, this option forces bulk deletes to be used. It would be unsafe to delete each row as it is found, before inserting the rows into the destination first. Forcing bulk deletes guarantees that the deletion waits until the insertion is successful.

The *--low-priority-insert*, *--replace*, and *--ignore* options work with this option, but *--delayed-insert* does not.

If `LOAD DATA LOCAL INFILE` throws an error in the lines of `The used command is not allowed with this MariaDB version`, refer to the documentation for the `L` DSN option.

--channel
type: string

Channel name used when connected to a server using replication channels. Suppose you have two masters, master_a at port 12345, master_b at port 1236 and a slave connected to both masters using channels chan_master_a and chan_master_b. If you want to run **mariadb-archiver** to syncronize the slave against master_a, **mariadb-archiver** won't be able to determine what's the correct master since SHOW SLAVE STATUS will return 2 rows. In this case, you can use –channel=chan_master_a to specify the channel name to use in the SHOW SLAVE STATUS command.

--charset
short form: -A; type: string

Default character set. If the value is utf8, sets Perl's binmode on STDOUT to utf8, passes the mysql_enable_utf8 option to DBD::mysql, and runs SET NAMES UTF8 after connecting to MariaDB. Any other value sets binmode on STDOUT without the utf8 layer, and runs SET NAMES after connecting to MariaDB.

Note that only charsets as known by MariaDB are recognized; So for example, "UTF8" will work, but "UTF-8" will not.

See also *--[no]check-charset*.

--[no]check-charset
default: yes

Ensure connection and table character sets are the same. Disabling this check may cause text to be erroneously converted from one character set to another (usually from utf8 to latin1) which may cause data loss or mojibake. Disabling this check may be useful or necessary when character set conversions are intended.

--[no]check-columns
default: yes

Ensure *--source* and *--dest* have same columns.

Enabled by default; causes **mariadb-archiver** to check that the source and destination tables have the same columns. It does not check column order, data type, etc. It just checks that all columns in the source exist in the destination and vice versa. If there are any differences, **mariadb-archiver** will exit with an error.

To disable this check, specify –no-check-columns.

--check-interval
type: time; default: 1s

If *--check-slave-lag* is given, this defines how long the tool pauses each time it discovers that a slave is lagging. This check is performed every 100 rows.

--check-slave-lag
type: string; repeatable: yes

Pause archiving until the specified DSN's slave lag is less than *--max-lag*. This option can be specified multiple times for checking more than one slave.

--columns
short form: -c; type: array

Comma-separated list of columns to archive.

Specify a comma-separated list of columns to fetch, write to the file, and insert into the destination table. If specified, **mariadb-archiver** ignores other columns unless it needs to add them to the `SELECT` statement for ascending an index or deleting rows. It fetches and uses these extra columns internally, but does not write them to the file or to the destination table. It *does* pass them to plugins.

See also *--primary-key-only*.

--commit-each
Commit each set of fetched and archived rows (disables *--txn-size*).

Commits transactions and flushes *--file* after each set of rows has been archived, before fetching the next set of rows, and before sleeping if *--sleep* is specified. Disables *--txn-size*; use *--limit* to control the transaction size with *--commit-each*.

This option is useful as a shortcut to make *--limit* and *--txn-size* the same value, but more importantly it avoids transactions being held open while searching for more rows. For example, imagine you are archiving old rows from the beginning of a very large table, with *--limit* 1000 and *--txn-size* 1000. After some period of finding and archiving 1000 rows at a time, **mariadb-archiver** finds the last 999 rows and archives them, then executes the next SELECT to find more rows. This scans the rest of the table, but never finds any more rows. It has held open a transaction for a very long time, only to determine it is finished anyway. You can use *--commit-each* to avoid this.

--config
type: Array

Read this comma-separated list of config files; if specified, this must be the first option on the command line.

--database
short form: -D; type: string

Connect to this database.

--delayed-insert
Add the DELAYED modifier to INSERT statements.

Adds the DELAYED modifier to INSERT or REPLACE statements. See https://mariadb.com/kb/en/library/insert/ for details.

--dest
type: DSN

DSN specifying the table to archive to.

This item specifies a table into which **mariadb-archiver** will insert rows archived from *--source*. It uses the same key=val argument format as *--source*. Most missing values default to the same values as *--source*, so you don't have to repeat options that are the same in *--source* and *--dest*. Use the *--help* option to see which values are copied from *--source*.

WARNING: Using a default options file (F) DSN option that defines a socket for *--source* causes **mariadb-archiver** to connect to *--dest* using that socket unless another socket for *--dest* is specified. This means that **mariadb-archiver** may incorrectly connect to *--source* when it connects to *--dest*. For example:

```
--source F=host1.cnf,D=db,t=tbl --dest h=host2
```

When **mariadb-archiver** connects to *--dest*, host2, it will connect via the *--source*, host1, socket defined in host1.cnf.

--dry-run
Print queries and exit without doing anything.

Causes **mariadb-archiver** to exit after printing the filename and SQL statements it will use.

--file
type: string

File to archive to, with DATE_FORMAT()-like formatting.

Filename to write archived rows to. A subset of MariaDB's DATE_FORMAT() formatting codes are allowed in the filename, as follows:

```
%d    Day of the month, numeric (01..31)
%H    Hour (00..23)
%i    Minutes, numeric (00..59)
%m    Month, numeric (01..12)
%s    Seconds (00..59)
%Y    Year, numeric, four digits
```

You can use the following extra format codes too:

```
%D    Database name
%t    Table name
```

Example:

```
--file '/var/log/archive/%Y-%m-%d-%D.%t'
```

The file's contents are in the same format used by SELECT INTO OUTFILE, as documented in the MariaDB manual: rows terminated by newlines, columns terminated by tabs, NULL characters are represented by N, and special characters are escaped by . This lets you reload a file with LOAD DATA INFILE's default settings.

If you want a column header at the top of the file, see *--header*. The file is auto-flushed by default; see *--buffer*.

--for-update

Adds the FOR UPDATE modifier to SELECT statements.

For details, see http://dev.mysql.com/doc/en/innodb-locking-reads.html.

--header

Print column header at top of *--file*.

Writes column names as the first line in the file given by *--file*. If the file exists, does not write headers; this keeps the file loadable with LOAD DATA INFILE in case you append more output to it.

--help

Show help and exit.

--high-priority-select

Adds the HIGH_PRIORITY modifier to SELECT statements.

See https://mariadb.com/kb/en/library/select/ for details.

--host

short form: -h; type: string

Connect to host.

--ignore

Use IGNORE for INSERT statements.

Causes INSERTs into *--dest* to be INSERT IGNORE.

--limit

type: int; default: 1

Number of rows to fetch and archive per statement.

Limits the number of rows returned by the SELECT statements that retrieve rows to archive. Default is one row. It may be more efficient to increase the limit, but be careful if you are archiving sparsely, skipping over many rows; this can potentially cause more contention with other queries, depending on the storage engine, transaction isolation level, and options such as *--for-update*.

--local

Do not write OPTIMIZE or ANALYZE queries to binlog.

Adds the NO_WRITE_TO_BINLOG modifier to ANALYZE and OPTIMIZE queries. See *--analyze* for details.

--low-priority-delete

Adds the LOW_PRIORITY modifier to DELETE statements.

See https://mariadb.com/kb/en/library/delete/ for details.

--low-priority-insert

Adds the LOW_PRIORITY modifier to INSERT or REPLACE statements.

See https://mariadb.com/kb/en/library/insert/ for details.

--max-flow-ctl

type: float

Somewhat similar to –max-lag but for Galera clusters. Check average time cluster spent pausing for Flow Control and make tool pause if it goes over the percentage indicated in the option. Default is no Flow Control checking. This option is available for Galera versions 5.6 or higher.

--max-lag

type: time; default: 1s

Pause archiving if the slave given by *--check-slave-lag* lags.

This option causes **mariadb-archiver** to look at the slave every time it's about to fetch another row. If the slave's lag is greater than the option's value, or if the slave isn't running (so its lag is NULL), pt-table-checksum sleeps for *--check-interval* seconds and then looks at the lag again. It repeats until the slave is caught up, then proceeds to fetch and archive the row.

This option may eliminate the need for *--sleep* or *--sleep-coef*.

--no-ascend

Do not use ascending index optimization.

The default ascending-index optimization causes **mariadb-archiver** to optimize repeated `SELECT` queries so they seek into the index where the previous query ended, then scan along it, rather than scanning from the beginning of the table every time. This is enabled by default because it is generally a good strategy for repeated accesses.

Large, multiple-column indexes may cause the WHERE clause to be complex enough that this could actually be less efficient. Consider for example a four-column PRIMARY KEY on (a, b, c, d). The WHERE clause to start where the last query ended is as follows:

```
WHERE (a > ?)
   OR (a = ? AND b > ?)
   OR (a = ? AND b = ? AND c > ?)
   OR (a = ? AND b = ? AND c = ? AND d >= ?)
```

Populating the placeholders with values uses memory and CPU, adds network traffic and parsing overhead, and may make the query harder for MariaDB to optimize. A four-column key isn't a big deal, but a ten-column key in which every column allows `NULL` might be.

Ascending the index might not be necessary if you know you are simply removing rows from the beginning of the table in chunks, but not leaving any holes, so starting at the beginning of the table is actually the most efficient thing to do.

See also *--ascend-first*. See "EXTENDING" for a discussion of how this interacts with plugins.

--no-delete

Do not delete archived rows.

Causes **mariadb-archiver** not to delete rows after processing them. This disallows *--no-ascend*, because enabling them both would cause an infinite loop.

If there is a plugin on the source DSN, its `before_delete` method is called anyway, even though **mariadb-archiver** will not execute the delete. See "EXTENDING" for more on plugins.

--optimize

type: string

Run OPTIMIZE TABLE afterwards on *--source* and/or *--dest*.

Runs OPTIMIZE TABLE after finishing. See *--analyze* for the option syntax and https://mariadb.com/kb/en/library/optimize-table/ for details on OPTIMIZE TABLE.

--output-format

type: string

Used with *--file* to specify the output format.

Valid formats are: dump: MariaDB dump format using tabs as field separator (default) csv : Dump rows using ',' as separator and optionally enclosing fields by '"'.

This format is equivalent to FIELDS TERMINATED BY ',' OPTIONALLY ENCLOSED BY '"'.

--password

short form: -p; type: string

Password to use when connecting. If password contains commas they must be escaped with a backslash: "exam,ple"

--pid

type: string

Create the given PID file. The tool won't start if the PID file already exists and the PID it contains is different than the current PID. However, if the PID file exists and the PID it contains is no longer running, the tool will overwrite the PID file with the current PID. The PID file is removed automatically when the tool exits.

--plugin

type: string

Perl module name to use as a generic plugin.

Specify the Perl module name of a general-purpose plugin. It is currently used only for statistics (see *--statistics*) and must have `new()` and a `statistics()` method.

The `new( src` = $src, dst => $dst, opts => $o)> method gets the source and destination DSNs, and their database connections, just like the connection-specific plugins do. It also gets an OptionParser object (`$o`) for accessing command-line options (example: ``$o-``get('purge');>).

The `statistics(\%stats, $time)` method gets a hashref of the statistics collected by the archiving job, and the time the whole job started.

--port

short form: -P; type: int

Port number to use for connection.

--primary-key-only

Primary key columns only.

A shortcut for specifying *--columns* with the primary key columns. This is an efficiency if you just want to purge rows; it avoids fetching the entire row, when only the primary key columns are needed for `DELETE` statements. See also *--purge*.

--progress
type: int

Print progress information every X rows.

Prints current time, elapsed time, and rows archived every X rows.

--purge
Purge instead of archiving; allows omitting *--file* and *--dest*.

Allows archiving without a *--file* or *--dest* argument, which is effectively a purge since the rows are just deleted.

If you just want to purge rows, consider specifying the table's primary key columns with *--primary-key-only*. This will prevent fetching all columns from the server for no reason.

--quick-delete
Adds the QUICK modifier to DELETE statements.

See https://mariadb.com/kb/en/library/delete/ for details. As stated in the documentation, in some cases it may be faster to use DELETE QUICK followed by OPTIMIZE TABLE. You can use *--optimize* for this.

--quiet
short form: -q

Do not print any output, such as for *--statistics*.

Suppresses normal output, including the output of *--statistics*, but doesn't suppress the output from *--why-quit*.

--replace
Causes INSERTs into *--dest* to be written as REPLACE.

--retries
type: int; default: 1

Number of retries per timeout or deadlock.

Specifies the number of times **mariadb-archiver** should retry when there is an InnoDB lock wait timeout or deadlock. When retries are exhausted, **mariadb-archiver** will exit with an error.

Consider carefully what you want to happen when you are archiving between a mixture of transactional and non-transactional storage engines. The INSERT to *--dest* and DELETE from *--source* are on separate connections, so they do not actually participate in the same transaction even if they're on the same server. However, **mariadb-archiver** implements simple distributed transactions in code, so commits and rollbacks should happen as desired across the two connections.

At this time I have not written any code to handle errors with transactional storage engines other than InnoDB. Request that feature if you need it.

--run-time
type: time

Time to run before exiting.

Optional suffix s=seconds, m=minutes, h=hours, d=days; if no suffix, s is used.

--[no]safe-auto-increment
default: yes

Do not archive row with max AUTO_INCREMENT.

Adds an extra WHERE clause to prevent **mariadb-archiver** from removing the newest row when ascending a single-column AUTO_INCREMENT key. This guards against re-using AUTO_INCREMENT values if the server restarts, and is enabled by default.

The extra WHERE clause contains the maximum value of the auto-increment column as of the beginning of the archive or purge job. If new rows are inserted while **mariadb-archiver** is running, it will not see them.

--sentinel
type: string; default: /tmp/mariadb-archiver-sentinel

Exit if this file exists.

The presence of the file specified by *--sentinel* will cause **mariadb-archiver** to stop archiving and exit. The default is /tmp/mariadb-archiver-sentinel. You might find this handy to stop cron jobs gracefully if necessary. See also *--stop*.

--slave-user
type: string

Sets the user to be used to connect to the slaves. This parameter allows you to have a different user with less privileges on the slaves but that user must exist on all slaves.

--slave-password
type: string

Sets the password to be used to connect to the slaves. It can be used with –slave-user and the password for the user must be the same on all slaves.

--set-vars
type: Array

Set the MariaDB variables in this comma-separated list of `variable=value` pairs.

By default, the tool sets:

```
wait_timeout=10000
```

Variables specified on the command line override these defaults. For example, specifying `--set-vars wait_timeout=500` overrides the default value of `10000`.

The tool prints a warning and continues if a variable cannot be set.

--share-lock
Adds the LOCK IN SHARE MODE modifier to SELECT statements.

See http://dev.mysql.com/doc/en/innodb-locking-reads.html.

--skip-foreign-key-checks
Disables foreign key checks with SET FOREIGN_KEY_CHECKS=0.

--sleep
type: int

Sleep time between fetches.

Specifies how long to sleep between SELECT statements. Default is not to sleep at all. Transactions are NOT committed, and the *--file* file is NOT flushed, before sleeping. See *--txn-size* to control that.

If *--commit-each* is specified, committing and flushing happens before sleeping.

--sleep-coef
type: float

Calculate *--sleep* as a multiple of the last SELECT time.

If this option is specified, **mariadb-archiver** will sleep for the query time of the last SELECT multiplied by the specified coefficient.

This is a slightly more sophisticated way to throttle the SELECTs: sleep a varying amount of time between each SELECT, depending on how long the SELECTs are taking.

--socket
short form: -S; type: string

Socket file to use for connection.

--source
type: DSN

DSN specifying the table to archive from (required). This argument is a DSN. See DSN OPTIONS for the syntax. Most options control how **mariadb-archiver** connects to MariaDB, but there are some extended DSN options in this tool's syntax. The D, t, and i options select a table to archive:

```
--source h=my_server,D=my_database,t=my_tbl
```

The a option specifies the database to set as the connection's default with USE. If the b option is true, it disables binary logging with SQL_LOG_BIN. The m option specifies pluggable actions, which an external Perl module can provide. The only required part is the table; other parts may be read from various places in the environment (such as options files).

The 'i' part deserves special mention. This tells **mariadb-archiver** which index it should scan to archive. This appears in a FORCE INDEX or USE INDEX hint in the SELECT statements used to fetch archivable rows. If you don't specify anything, **mariadb-archiver** will auto-discover a good index, preferring a `PRIMARY KEY` if one exists. In my experience this usually works well, so most of the time you can probably just omit the 'i' part.

The index is used to optimize repeated accesses to the table; **mariadb-archiver** remembers the last row it retrieves from each SELECT statement, and uses it to construct a WHERE clause, using the columns in the specified index, that should allow MariaDB to start the next SELECT where the last one ended, rather than potentially scanning from the beginning of the table with each successive SELECT. If you are using external plugins, please see "EXTENDING" for a discussion of how they interact with ascending indexes.

The 'a' and 'b' options allow you to control how statements flow through the binary log. If you specify the 'b' option, binary logging will be disabled on the specified connection. If you specify the 'a' option, the connection will `USE` the specified database, which you can use to prevent slaves from executing the binary log events with `--replicate-ignore-db` options. These two options can be used as different methods to achieve the same goal: archive data off the master, but leave it on the slave. For example, you can run a purge job on the master and prevent it from happening on the slave using your method of choice.

WARNING: Using a default options file (F) DSN option that defines a socket for *--source* causes **mariadb-archiver** to connect to *--dest* using that socket unless another socket for *--dest* is specified. This means that **mariadb-archiver** may incorrectly connect to *--source* when it is meant to connect to *--dest*. For example:

```
--source F=host1.cnf,D=db,t=tbl --dest h=host2
```

When **mariadb-archiver** connects to *--dest*, host2, it will connect via the *--source*, host1, socket defined in host1.cnf.

--statistics
Collect and print timing statistics.

Causes **mariadb-archiver** to collect timing statistics about what it does. These statistics are available to the plugin specified by *--plugin*

Unless you specify *--quiet*, **mariadb-archiver** prints the statistics when it exits. The statistics look like this:

```
Started at 2008-07-18T07:18:53, ended at 2008-07-18T07:18:53
Source: D=db,t=table
SELECT 4
INSERT 4
DELETE 4
Action         Count       Time        Pct
commit            10     0.1079      88.27
select             5     0.0047       3.87
deleting           4     0.0028       2.29
inserting          4     0.0028       2.28
other              0     0.0040       3.29
```

The first two (or three) lines show times and the source and destination tables. The next three lines show how many rows were fetched, inserted, and deleted.

The remaining lines show counts and timing. The columns are the action, the total number of times that action was timed, the total time it took, and the percent of the program's total runtime. The rows are sorted in order of descending total time. The last row is the rest of the time not explicitly attributed to anything. Actions will vary depending on command-line options.

If *--why-quit* is given, its behavior is changed slightly. This option causes it to print the reason for exiting even when it's just because there are no more rows.

This option requires the standard Time::HiRes module, which is part of core Perl on reasonably new Perl releases.

--stop

Stop running instances by creating the sentinel file.

Causes **mariadb-archiver** to create the sentinel file specified by *--sentinel* and exit. This should have the effect of stopping all running instances which are watching the same sentinel file.

--txn-size

type: int; default: 1

Number of rows per transaction.

Specifies the size, in number of rows, of each transaction. Zero disables transactions altogether. After **mariadb-archiver** processes this many rows, it commits both the *--source* and the *--dest* if given, and flushes the file given by *--file*.

This parameter is critical to performance. If you are archiving from a live server, which for example is doing heavy OLTP work, you need to choose a good balance between transaction size and commit overhead. Larger transactions create the possibility of more lock contention and deadlocks, but smaller transactions cause more frequent commit overhead, which can be significant. To give an idea, on a small test set I worked with while writing **mariadb-archiver**, a value of 500 caused archiving to take about 2 seconds per 1000 rows on an otherwise quiet MariaDB instance on my desktop machine, archiving to disk and to another table. Disabling transactions with a value of zero, which turns on autocommit, dropped performance to 38 seconds per thousand rows.

If you are not archiving from or to a transactional storage engine, you may want to disable transactions so **mariadb-archiver** doesn't try to commit.

--user

short form: -u; type: string

User for login if not current user.

--version

Show version and exit.

--where

type: string

WHERE clause to limit which rows to archive (required).

Specifies a WHERE clause to limit which rows are archived. Do not include the word WHERE. You may need to quote the argument to prevent your shell from interpreting it. For example:

```
--where 'ts < current_date - interval 90 day'
```

For safety, *--where* is required. If you do not require a WHERE clause, use *--where* 1=1.

--why-quit

Print reason for exiting unless rows exhausted.

Causes **mariadb-archiver** to print a message if it exits for any reason other than running out of rows to archive. This can be useful if you have a cron job with *--run-time* specified, for example, and you want to be sure **mariadb-archiver** is finishing before running out of time.

If *--statistics* is given, the behavior is changed slightly. It will print the reason for exiting even when it's just because there are no more rows.

This output prints even if *--quiet* is given. That's so you can put **mariadb-archiver** in a `cron` job and get an email if there's an abnormal exit.

3.9 DSN OPTIONS

These DSN options are used to create a DSN. Each option is given like `option=value`. The options are case-sensitive, so P and p are not the same option. There cannot be whitespace before or after the = and if the value contains whitespace it must be quoted. DSN options are comma-separated. See the mariadb-tools manpage for full details.

- a

  copy: no

  Database to USE when executing queries.

- A

  dsn: charset; copy: yes

  Default character set.

- b

  copy: no

  If true, disable binlog with SQL_LOG_BIN.

- D

  dsn: database; copy: yes

  Database that contains the table.

- F

  dsn: mysql_read_default_file; copy: yes

  Only read default options from the given file

- h

  dsn: host; copy: yes

  Connect to host.

- i

  copy: yes

  Index to use.

- L

  copy: yes

  Explicitly enable LOAD DATA LOCAL INFILE.

  For some reason, some vendors compile libmysql without the –enable-local-infile option, which disables the statement. This can lead to weird situations, like the server allowing LOCAL INFILE, but the client throwing exceptions if it's used.

  However, as long as the server allows LOAD DATA, clients can easily re-enable it; See https://mariadb.com/kb/en/library/load-data-infile/ and http://search.cpan.org/~capttofu/DBD-mysql/lib/DBD/mysql.pm. This option does exactly that.

  Although we've not found a case where turning this option leads to errors or differing behavior, to be on the safe side, this option is not on by default.

- m

  copy: no

  Plugin module name.

- p

  dsn: password; copy: yes

  Password to use when connecting. If password contains commas they must be escaped with a backslash: "exam,ple"

- P

  dsn: port; copy: yes

  Port number to use for connection.

- S

  dsn: mysql_socket; copy: yes

  Socket file to use for connection.

- t

  copy: yes

  Table to archive from/to.

- u

  dsn: user; copy: yes

  User for login if not current user.

3.10 EXTENDING

mariadb-archiver is extensible by plugging in external Perl modules to handle some logic and/or actions. You can specify a module for both the *--source* and the *--dest*, with the 'm' part of the specification. For example:

```
--source D=test,t=test1,m=My::Module1 --dest m=My::Module2,t=test2
```

This will cause **mariadb-archiver** to load the My::Module1 and My::Module2 packages, create instances of them, and then make calls to them during the archiving process.

You can also specify a plugin with *--plugin*.

The module must provide this interface:

new(dbh => $dbh, db => $db_name, tbl => $tbl_name)

The plugin's constructor is passed a reference to the database handle, the database name, and table name. The plugin is created just after **mariadb-archiver** opens the connection, and before it examines the table given in the arguments. This gives the plugin a chance to create and populate temporary tables, or do other setup work.

before_begin(cols => @cols, allcols => @allcols)

This method is called just before **mariadb-archiver** begins iterating through rows and archiving them, but after it does all other setup work (examining table structures, designing SQL queries, and so on). This is the only time **mariadb-archiver** tells the plugin column names for the rows it will pass the plugin while archiving.

The `cols` argument is the column names the user requested to be archived, either by default or by the *--columns* option. The `allcols` argument is the list of column names for every row **mariadb-archiver** will fetch from the source table. It may fetch more columns than the user requested, because it needs some columns for its own use. When subsequent plugin functions receive a row, it is the full row containing all the extra columns, if any, added to the end.

is_archivable(row => @row)

This method is called for each row to determine whether it is archivable. This applies only to *--source*. The argument is the row itself, as an arrayref. If the method returns true, the row will be archived; otherwise it will be skipped.

Skipping a row adds complications for non-unique indexes. Normally **mariadb-archiver** uses a WHERE clause designed to target the last processed row as the place to start the scan for the next SELECT statement. If you have skipped the row by returning false from is_archivable(), **mariadb-archiver** could get into an infinite loop because the row still exists. Therefore, when you specify a plugin for the *--source* argument, **mariadb-archiver** will change its WHERE clause slightly. Instead of starting at "greater than or equal to" the last processed row, it will start "strictly greater than." This will work fine on unique indexes such as primary keys, but it may skip rows (leave holes) on non-unique indexes or when ascending only the first column of an index.

mariadb-archiver will change the clause in the same way if you specify *--no-delete*, because again an infinite loop is possible.

If you specify the *--bulk-delete* option and return false from this method, **mariadb-archiver** may not do what you want. The row won't be archived, but it will be deleted, since bulk deletes operate on ranges of rows and don't know which rows the plugin selected to keep.

If you specify the *--bulk-insert* option, this method's return value will influence whether the row is written to the temporary file for the bulk insert, so bulk inserts will work as expected. However, bulk inserts require bulk deletes.

before_delete(row => @row)

This method is called for each row just before it is deleted. This applies only to *--source*. This is a good place for you to handle dependencies, such as deleting things that are foreign-keyed to the row you are about to delete. You could also use this to recursively archive all dependent tables.

This plugin method is called even if *--no-delete* is given, but not if *--bulk-delete* is given.

before_bulk_delete(first_row => @row, last_row => @row)

This method is called just before a bulk delete is executed. It is similar to the `before_delete` method, except its arguments are the first and last row of the range to be deleted. It is called even if *--no-delete* is given.

before_insert(row => @row)

This method is called for each row just before it is inserted. This applies only to *--dest*. You could use this to insert the row into multiple tables, perhaps with an ON DUPLICATE KEY UPDATE clause to build summary tables in a data warehouse.

This method is not called if *--bulk-insert* is given.

before_bulk_insert(first_row => @row, last_row => @row, filename => bulk_insert_filename)

This method is called just before a bulk insert is executed. It is similar to the `before_insert` method, except its arguments are the first and last row of the range to be deleted.

custom_sth(row => @row, sql => $sql)

This method is called just before inserting the row, but after "before_insert()". It allows the plugin to specify different `INSERT` statement if desired. The return value (if any) should be a DBI statement handle. The `sql` parameter is the SQL text used to prepare the default `INSERT` statement. This method is not called if you specify *--bulk-insert*.

If no value is returned, the default `INSERT` statement handle is used.

This method applies only to the plugin specified for *--dest*, so if your plugin isn't doing what you expect, check that you've specified it for the destination and not the source.

custom_sth_bulk(first_row => @row, last_row => @row, sql => $sql, filename => $bulk_insert_filename)

If you've specified *--bulk-insert*, this method is called just before the bulk insert, but after "before_bulk_insert()", and the arguments are different.

This method's return value etc is similar to the "custom_sth()" method.

after_finish()

This method is called after **mariadb-archiver** exits the archiving loop, commits all database handles, closes *--file*, and prints the final statistics, but before **mariadb-archiver** runs ANALYZE or OPTIMIZE (see *--analyze* and *--optimize*).

If you specify a plugin for both *--source* and *--dest*, **mariadb-archiver** constructs, calls before_begin(), and calls after_finish() on the two plugins in the order *--source*, *--dest*.

mariadb-archiver assumes it controls transactions, and that the plugin will NOT commit or roll back the database handle. The database handle passed to the plugin's constructor is the same handle **mariadb-archiver** uses itself. Remember that *--source* and *--dest* are separate handles.

A sample module might look like this:

```
package My::Module;

sub new {
   my ( $class, %args ) = @_;
   return bless(\%args, $class);
```

(continues on next page)

(continued from previous page)

```
}

sub before_begin {
   my ( $self, %args ) = @_;
   # Save column names for later
   $self->{cols} = $args{cols};
}

sub is_archivable {
   my ( $self, %args ) = @_;
   # Do some advanced logic with $args{row}
   return 1;
}

sub before_delete {} # Take no action
sub before_insert {} # Take no action
sub custom_sth    {} # Take no action
sub after_finish  {} # Take no action

1;
```

3.11 ENVIRONMENT

The environment variable `PTDEBUG` enables verbose debugging output to STDERR. To enable debugging and capture all output to a file, run the tool like:

```
PTDEBUG=1 mariadb-archiver ... > FILE 2>&1
```

Be careful: debugging output is voluminous and can generate several megabytes of output.

3.12 SYSTEM REQUIREMENTS

You need Perl, DBI, DBD::mysql, and some core packages that ought to be installed in any reasonably new version of Perl.

3.13 AUTHORS

Cole Busby, Baron Schwartz

3.14 ACKNOWLEDGMENTS

Andrew O'Brien

3.15 ABOUT THIS MARIADB TOOL

This tool is part of MariaDB client tools. This MariaDB Tool was forked from Percona Toolkit's pt-archiver in August, 2019. Percona Toolkit was forked from two projects in June, 2011: Maatkit and Aspersa. Those projects were created by Baron Schwartz and primarily developed by him and Daniel Nichter.

3.16 COPYRIGHT, LICENSE, AND WARRANTY

3.17 VERSION

`mariadb-archiver` 6.0.0rc

CHAPTER

FOUR

MARIADB-BACKUP-MANAGER

CHAPTER

FIVE

MARIADB-DATABASE-SUMMARY

5.1 NAME

mariadb-database-summary - Summarize MariaDB information nicely.

5.2 SYNOPSIS

5.2.1 Usage

```
mariadb-database-summary [OPTIONS]
```

mariadb-database-summary conveniently summarizes the status and configuration of a MariaDB database server so that you can learn about it at a glance. It is not a tuning tool or diagnosis tool. It produces a report that is easy to diff and can be pasted into emails without losing the formatting. It should work well on any modern UNIX systems.

5.3 RISKS

mariadb-database-summary is mature, proven in the real world, and well tested, but all database tools can pose a risk to the system and the database server. Before using this tool, please:

- Read the tool's documentation
- Review the tool's known "BUGS"
- Test the tool on a non-production server
- Backup your production server and verify the backups

5.4 DESCRIPTION

mariadb-database-summary works by connecting to a MariaDB database server and querying it for status and configuration information. It saves these bits of data into files in a temporary directory, and then formats them neatly with awk and other scripting languages.

To use, simply execute it. Optionally add a double dash and then the same command-line options you would use to connect to MariaDB, such as the following:

```
mariadb-database-summary --user=root
```

The tool interacts minimally with the server upon which it runs. It assumes that you'll run it on the same server you're inspecting, and therefore it assumes that it will be able to find the my.cnf configuration file, for example. However, it should degrade gracefully if this is not the case. Note, however, that its output does not indicate which information comes from the MariaDB database and which comes from the host operating system, so it is possible for confusing output to be generated if you run the tool on one server and connect to a MariaDB database server running on another server.

5.5 OUTPUT

Many of the outputs from this tool are deliberately rounded to show their magnitude but not the exact detail. This is called fuzzy-rounding. The idea is that it does not matter whether a server is running 918 queries per second or 921 queries per second; such a small variation is insignificant, and only makes the output hard to compare to other servers. Fuzzy-rounding rounds in larger increments as the input grows. It begins by rounding to the nearest 5, then the nearest 10, nearest 25, and then repeats by a factor of 10 larger (50, 100, 250), and so on, as the input grows.

The following is a sample of the report that the tool produces:

```
# MariaDB Server Summary Report ##############################
              System time | 2012-03-30 18:46:05 UTC
                            (local TZ: EDT -0400)
# Instances ##################################################
  Port  Data Directory             Nice OOM Socket
  ===== ========================== ==== === ======
  12345 /tmp/12345/data            0    0   /tmp/12345.sock
  12346 /tmp/12346/data            0    0   /tmp/12346.sock
  12347 /tmp/12347/data            0    0   /tmp/12347.sock
```

The first two sections show which server the report was generated on and which MariaDB instances are running on the server. This is detected from the output of `ps` and does not always detect all instances and parameters, but often works well. From this point forward, the report will be focused on a single MariaDB instance, although several instances may appear in the above paragraph.

```
# Report On Port 12345 #######################################
                     User | msandbox@%
                     Time | 2012-03-30 14:46:05 (EDT)
                 Hostname | localhost.localdomain
                  Version | 10.4.7-MariaDB-1:10.4.7+maria~bionic
                 Built On | linux2.6 i686
                  Started | 2012-03-28 23:33 (up 1+15:12:09)
                Databases | 4
                  Datadir | /tmp/12345/data/
                Processes | 2 connected, 2 running
              Replication | Is not a slave, has 1 slaves connected
                  Pidfile | /tmp/12345/data/12345.pid (exists)
```

This section is a quick summary of the MariaDB instance: version, uptime, and other very basic parameters. The Time output is generated from the MariaDB server, unlike the system date and time printed earlier, so you can see whether the database and operating system times match.

```
# Processlist ################################################

  Command                        COUNT(*) Working SUM(Time) MAX(Time)
  ------------------------------ -------- ------- --------- ---------
  Binlog Dump                           1       1    150000    150000
  Query                                 1       1         0         0

  User                           COUNT(*) Working SUM(Time) MAX(Time)
  ------------------------------ -------- ------- --------- ---------
  msandbox                              2       2    150000    150000

  Host                           COUNT(*) Working SUM(Time) MAX(Time)
  ------------------------------ -------- ------- --------- ---------
  localhost                             2       2    150000    150000

  db                             COUNT(*) Working SUM(Time) MAX(Time)
  ------------------------------ -------- ------- --------- ---------
  NULL                                  2       2    150000    150000

  State                          COUNT(*) Working SUM(Time) MAX(Time)
  ------------------------------ -------- ------- --------- ---------
  Master has sent all binlog to         1       1    150000    150000
  NULL                                  1       1         0         0
```

This section is a summary of the output from SHOW PROCESSLIST. Each sub-section is aggregated by a different item, which is shown as the first column heading. When summarized by Command, every row in SHOW PROCESSLIST is included, but otherwise, rows whose Command is Sleep are excluded from the SUM and MAX columns, so they do not skew the numbers too much. In the example shown, the server is idle except for this tool itself, and one connected replica, which is executing Binlog Dump.

The columns are the number of rows included, the number that are not in Sleep status, the sum of the Time column, and the maximum Time column. The numbers are fuzzy-rounded.

```
# Status Counters (Wait 10 Seconds) ##########################
Variable                            Per day  Per second     10 secs
Binlog_cache_disk_use                     4
Binlog_cache_use                         80
Bytes_received                     15000000         175         200
Bytes_sent                         15000000         175        2000
Com_admin_commands                        1
...................(many lines omitted)............................
Threads_created                          40                       1
Uptime                                90000           1           1
```

This section shows selected counters from two snapshots of SHOW GLOBAL STATUS, gathered approximately 10 seconds apart and fuzzy-rounded. It includes only items that are incrementing counters; it does not include absolute numbers such as the Threads_running status variable, which represents a current value, rather than an accumulated number over time.

The first column is the variable name, and the second column is the counter from the first snapshot divided by 86400 (the number of seconds in a day), so you can see the magnitude of the counter's change per day. 86400 fuzzy-rounds to 90000, so the Uptime counter should always be about 90000.

The third column is the value from the first snapshot, divided by Uptime and then fuzzy-rounded, so it represents approximately how quickly the counter is growing per-second over the uptime of the server.

The third column is the incremental difference from the first and second snapshot, divided by the difference in uptime and then fuzzy-rounded. Therefore, it shows how quickly the counter is growing per second at the time the report was generated.

```
# Table cache ################################################
                     Size | 400
                    Usage | 15%
```

This section shows the size of the table cache, followed by the percentage of the table cache in use. The usage is fuzzy-rounded.

```
# Key MariaDB Server features ################################
      Table & Index Stats | Not Supported
     Multiple I/O Threads | Enabled
     Corruption Resilient | Not Supported
      Durable Replication | Not Supported
     Import InnoDB Tables | Not Supported
     Fast Server Restarts | Not Supported
         Enhanced Logging | Not Supported
     Replica Perf Logging | Not Supported
      Response Time Hist. | Not Supported
          Smooth Flushing | Not Supported
      HandlerSocket NoSQL | Not Supported
           Fast Hash UDFs | Unknown
```

This section shows features that are available in MariaDB Server and whether they are enabled or not.

```
# Plugins ####################################################
       InnoDB compression | ACTIVE
```

This feature shows specific plugins and whether they are enabled.

```
# Query cache ################################################
         query_cache_type | ON
                     Size | 0.0
                    Usage | 0%
         HitToInsertRatio | 0%
```

This section shows whether the query cache is enabled and its size, followed by the percentage of the cache in use and the hit-to-insert ratio. The latter two are fuzzy-rounded.

```
# Schema #####################################################

  Database           Tables Views SPs Trigs Funcs   FKs Partn
  mysql                  24
  performance_schema     17
  sakila                 16     7   3     6     3    22

  Database           MyISAM CSV PERFORMANCE_SCHEMA InnoDB
  mysql                  22   2
  performance_schema                           17
  sakila                  8                               15

  Database           BTREE FULLTEXT
  mysql                 31
  performance_schema
  sakila                63        1
```

(continues on next page)

(continued from previous page)

```
                       c   t   s   e   l   d   i   t   m   v   s
                       h   i   e   n   o   a   n   i   e   a   m
                       a   m   t   u   n   t   t   n   d   r   a
                       r   e       m   g   e       y   i   c   l
                           s           b   t       i   u   h   l
                           t           l   i       n   m   a   i
                           a           o   m       t   t   r   n
                           m           b   e           e       t
                           p                           x
                                                       t
  Database           === === === === === === === === === === ===
  mysql               61  10   6  78   5   4  26   3   4   5   3
  performance_schema                5          16         33
  sakila               1  15   1   3       4   3  19      42  26
```

If you specify *--databases* or *--all-databases*, the tool will print the above section. This summarizes the number and type of objects in the databases. It is generated by running `mariadb-dump --no-data`, not by querying the INFORMATION_SCHEMA, which can freeze a busy server.

The first sub-report in the section is the count of objects by type in each database: tables, views, and so on. The second one shows how many tables use various storage engines in each database. The third sub-report shows the number of each type of indexes in each database.

The last section shows the number of columns of various data types in each database. For compact display, the column headers are formatted vertically, so you need to read downwards from the top. In this example, the first column is `char` and the second column is `timestamp`. This example is truncated so it does not wrap on a terminal.

All of the numbers in this portion of the output are exact, not fuzzy-rounded.

```
# Noteworthy Technologies ####################################
       Full Text Indexing | Yes
         Geospatial Types | No
             Foreign Keys | Yes
             Partitioning | No
       InnoDB Compression | Yes
                      SSL | No
     Explicit LOCK TABLES | No
           Delayed Insert | No
          XA Transactions | No
      ColumnStore Cluster | No
      Prepared Statements | No
 Prepared statement count | 0
```

This section shows some specific technologies used on this server. Some of them are detected from the schema dump performed for the previous sections; others can be detected by looking at SHOW GLOBAL STATUS.

```
# InnoDB #####################################################
                  Version | 1.1.8
         Buffer Pool Size | 16.0M
         Buffer Pool Fill | 100%
        Buffer Pool Dirty | 0%
           File Per Table | OFF
                Page Size | 16k
            Log File Size | 2 * 5.0M = 10.0M
          Log Buffer Size | 8M
             Flush Method |
      Flush Log At Commit | 1
```

(continues on next page)

(continued from previous page)

```
               XA Support | ON
                Checksums | ON
              Doublewrite | ON
          R/W I/O Threads | 4 4
             I/O Capacity | 200
       Thread Concurrency | 0
      Concurrency Tickets | 500
       Commit Concurrency | 0
      Txn Isolation Level | REPEATABLE-READ
        Adaptive Flushing | ON
      Adaptive Checkpoint |
           Checkpoint Age | 0
             InnoDB Queue | 0 queries inside InnoDB, 0 queries in queue
       Oldest Transaction | 0 Seconds
         History List Len | 209
               Read Views | 1
         Undo Log Entries | 1 transactions, 1 total undo, 1 max undo
        Pending I/O Reads | 0 buf pool reads, 0 normal AIO,
                            0 ibuf AIO, 0 preads
       Pending I/O Writes | 0 buf pool (0 LRU, 0 flush list, 0 page);
                            0 AIO, 0 sync, 0 log IO (0 log, 0 chkp);
                            0 pwrites
      Pending I/O Flushes | 0 buf pool, 0 log
       Transaction States | 1xnot started
```

This section shows important configuration variables for the InnoDB storage engine. The buffer pool fill percent and dirty percent are fuzzy-rounded. The last few lines are derived from the output of SHOW INNODB STATUS. It is likely that this output will change in the future to become more useful.

```
# MyISAM #####################################################
                Key Cache | 16.0M
                 Pct Used | 10%
                Unflushed | 0%
```

This section shows the size of the MyISAM key cache, followed by the percentage of the cache in use and percentage unflushed (fuzzy-rounded).

```
# Aria #######################################################
        Page Cache Buffer | 16.0M
                 Pct Used | 10%
                Unflushed | 0%
```

This section shows the size of the Aria page cache, followed by the percentage of the cache in use and percentage unflushed (fuzzy-rounded).

```
# Security ###################################################
                    Users | 2 users, 0 anon, 0 w/o pw, 0 old pw
            Old Passwords | OFF
```

This section is generated from queries to tables in the mysql system database. It shows how many users exist, and various potential security risks such as old-style passwords and users without passwords.

```
# Binary Logging #############################################
                  Binlogs | 1
               Zero-Sized | 0
               Total Size | 21.8M
```

(continues on next page)

(continued from previous page)

```
            binlog_format | STATEMENT
         expire_logs_days | 0
              sync_binlog | 0
                server_id | 12345
             binlog_do_db |
         binlog_ignore_db |
```

This section shows configuration and status of the binary logs. If there are zero-sized binary logs, then it is possible that the binlog index is out of sync with the binary logs that actually exist on disk.

```
# Noteworthy Variables #########################################
     Auto-Inc Incr/Offset | 1/1
   default_storage_engine | InnoDB
               flush_time | 0
             init_connect |
                init_file |
                 sql_mode |
         join_buffer_size | 128k
         sort_buffer_size | 2M
         read_buffer_size | 128k
     read_rnd_buffer_size | 256k
       bulk_insert_buffer | 0.00
      max_heap_table_size | 16M
           tmp_table_size | 16M
       max_allowed_packet | 1M
             thread_stack | 192k
                      log | OFF
                log_error | /tmp/12345/data/mysqld.log
             log_warnings | 1
         log_slow_queries | ON
log_queries_not_using_indexes | OFF
        log_slave_updates | ON
```

This section shows several noteworthy server configuration variables that might be important to know about when working with this server.

```
# Configuration File #########################################
              Config File | /tmp/12345/my.sandbox.cnf
[client]
user                                = msandbox
password                            = msandbox
port                                = 12345
socket                              = /tmp/12345/mysql_sandbox12345.sock
[mysqld]
port                                = 12345
socket                              = /tmp/12345/mysql_sandbox12345.sock
pid-file                            = /tmp/12345/data/mysql_sandbox12345.pid
basedir                             = /home/baron/5.5.20
datadir                             = /tmp/12345/data
key_buffer_size                     = 16M
innodb_buffer_pool_size             = 16M
innodb_data_home_dir                = /tmp/12345/data
innodb_log_group_home_dir           = /tmp/12345/data
innodb_data_file_path               = ibdata1:10M:autoextend
innodb_log_file_size                = 5M
log-bin                             = mariadb-bin
```

(continues on next page)

(continued from previous page)

```
relay_log                              = mariadb-relay-bin
log_slave_updates
server-id                              = 12345
report-host                            = 127.0.0.1
report-port                            = 12345
log-error                              = mysqld.log
innodb_lock_wait_timeout               = 3
# The End #####################################################
```

This section shows a pretty-printed version of the my.cnf file, with comments removed and with whitespace added to align things for easy reading. The tool tries to detect the my.cnf file by looking at the output of ps, and if it does not find the location of the file there, it tries common locations until it finds a file. Note that this file might not actually correspond with the server from which the report was generated. This can happen when the tool isn't run on the same server it's reporting on, or when detecting the location of the configuration file fails.

5.6 OPTIONS

All options after – are passed to `mariadb`.

`--all-databases`
mariadb-dump and summarize all databases. See *`--databases`*.

`--ask-pass`
Prompt for a password when connecting to MariaDB.

`--config`
type: string

Read this comma-separated list of config files. If specified, this must be the first option on the command line.

`--databases`
type: string

mariadb-dump and summarize this comma-separated list of databases. Specify *`--all-databases`* instead if you want to dump and summary all databases.

`--defaults-file`
short form: -F; type: string

Only read mariadb options from the given file. You must give an absolute pathname.

`--help`
Print help and exit.

`--host`
short form: -h; type: string

Host to connect to.

`--list-encrypted-tables`
default: false

Include a list of the encrypted tables in all databases. This can cause slowdowns since querying Information Schema tables can be slow.

`--password`
short form: -p; type: string

Password to use when connecting. If password contains commas they must be escaped with a backslash: "exam,ple"

--port
short form: -P; type: int

Port number to use for connection.

--read-samples
type: string

Create a report from the files found in this directory.

--save-samples
type: string

Save the data files used to generate the summary in this directory.

--sleep
type: int; default: 10

Seconds to sleep when gathering status counters.

--socket
short form: -S; type: string

Socket file to use for connection.

--user
short form: -u; type: string

User for login if not current user.

--version
Print tool's version and exit.

5.7 ENVIRONMENT

This tool does not use any environment variables.

5.8 SYSTEM REQUIREMENTS

This tool requires Bash v3 or newer, Perl 5.8 or newer, and binutils. These are generally already provided by most distributions. On BSD systems, it may require a mounted procfs.

5.9 AUTHORS

Cole Busby,Baron Schwartz, Brian Fraser, and Daniel Nichter

5.10 ABOUT THIS MARIADB TOOL

This tool is part of MariaDB client tools. This MariaDB Tool was forked from Percona Toolkit's pt-mysql-summary in August, 2019. Percona Toolkit was forked from two projects in June, 2011: Maatkit and Aspersa. Those projects were created by Baron Schwartz and primarily developed by him and Daniel Nichter.

5.11 COPYRIGHT, LICENSE, AND WARRANTY

5.12 VERSION

`mariadb-database-summary` 6.0.0rc

CHAPTER

SIX

MARIADB-INDEX-CHECKER

6.1 NAME

mariadb-index-checker - Find duplicate indexes and foreign keys on MariaDB tables.

6.2 SYNOPSIS

6.2.1 Usage

```
mariadb-index-checker [OPTIONS] [DSN]
```

mariadb-index-checker examines MariaDB tables for duplicate or redundant indexes and foreign keys. Connection options are read from MariaDB option files.

```
mariadb-index-checker --host host1
```

6.3 RISKS

MariaDB Tools is mature, proven in the real world, and well tested, but all database tools can pose a risk to the system and the database server. Before using this tool, please:

- Read the tool's documentation
- Review the tool's known "BUGS"
- Test the tool on a non-production server
- Backup your production server and verify the backups

6.4 DESCRIPTION

This program examines the output of SHOW CREATE TABLE on MariaDB tables, and if it finds indexes that cover the same columns as another index in the same order, or cover an exact leftmost prefix of another index, it prints out the suspicious indexes. By default, indexes must be of the same type, so a BTREE index is not a duplicate of a FULLTEXT index, even if they have the same columns. You can override this.

It also looks for duplicate foreign keys. A duplicate foreign key covers the same columns as another in the same table, and references the same parent table.

The output ends with a short summary that includes an estimate of the total size, in bytes, that the duplicate indexes are using. This is calculated by multiplying the index length by the number of rows in their respective tables.

6.5 OPTIONS

This tool accepts additional command-line arguments. Refer to the "SYNOPSIS" and usage information for details.

`--all-structs`

Compare indexes with different structs (BTREE, HASH, etc).

By default this is disabled, because a BTREE index that covers the same columns as a FULLTEXT index is not really a duplicate, for example.

`--ask-pass`

Prompt for a password when connecting to MariaDB.

`--charset`

short form: -A; type: string

Default character set. If the value is utf8, sets Perl's binmode on STDOUT to utf8, passes the mysql_enable_utf8 option to DBD::mysql, and runs SET NAMES UTF8 after connecting to MariaDB. Any other value sets binmode on STDOUT without the utf8 layer, and runs SET NAMES after connecting to MariaDB.

`--[no]clustered`

default: yes

PK columns appended to secondary key is duplicate.

Detects when a suffix of a secondary key is a leftmost prefix of the primary key, and treats it as a duplicate key. Only detects this condition on storage engines whose primary keys are clustered (currently InnoDB and solidDB).

Clustered storage engines append the primary key columns to the leaf nodes of all secondary keys anyway, so you might consider it redundant to have them appear in the internal nodes as well. Of course, you may also want them in the internal nodes, because just having them at the leaf nodes won't help for some queries. It does help for covering index queries, however.

Here's an example of a key that is considered redundant with this option:

```
PRIMARY KEY  (`a`)
KEY `b` (`b`,`a`)
```

The use of such indexes is rather subtle. For example, suppose you have the following query:

```
SELECT ... WHERE b=1 ORDER BY a;
```

This query will do a filesort if we remove the index on `b,a`. But if we shorten the index on `b,a` to just `b` and also remove the ORDER BY, the query should return the same results.

The tool suggests shortening duplicate clustered keys by dropping the key and re-adding it without the primary key prefix. The shortened clustered key may still duplicate another key, but the tool cannot currently detect when this happens without being ran a second time to re-check the newly shortened clustered keys. Therefore, if you shorten any duplicate clustered keys, you should run the tool again.

`--config`
type: Array

Read this comma-separated list of config files; if specified, this must be the first option on the command line.

`--databases`
short form: -d; type: hash

Check only this comma-separated list of databases.

`--defaults-file`
short form: -F; type: string

Only read mysql options from the given file. You must give an absolute pathname.

`--engines`
short form: -e; type: hash

Check only tables whose storage engine is in this comma-separated list.

`--help`
Show help and exit.

`--host`
short form: -h; type: string

Connect to host.

`--ignore-databases`
type: Hash

Ignore this comma-separated list of databases.

`--ignore-engines`
type: Hash

Ignore this comma-separated list of storage engines.

`--ignore-order`
Ignore index order so KEY(a,b) duplicates KEY(b,a).

`--ignore-tables`
type: Hash

Ignore this comma-separated list of tables. Table names may be qualified with the database name.

`--key-types`
type: string; default: fk

Check for duplicate f=foreign keys, k=keys or fk=both.

`--password`
short form: -p; type: string

Password to use when connecting. If password contains commas they must be escaped with a backslash: “exam,ple”

`--pid`
type: string

Create the given PID file. The tool won't start if the PID file already exists and the PID it contains is different than the current PID. However, if the PID file exists and the PID it contains is no longer running, the tool will overwrite the PID file with the current PID. The PID file is removed automatically when the tool exits.

--port
short form: -P; type: int

Port number to use for connection.

--set-vars
type: Array

Set the MariaDB variables in this comma-separated list of `variable=value` pairs.

By default, the tool sets:

```
wait_timeout=10000
```

Variables specified on the command line override these defaults. For example, specifying `--set-vars wait_timeout=500` overrides the defaultvalue of `10000`.

The tool prints a warning and continues if a variable cannot be set.

--socket
short form: -S; type: string

Socket file to use for connection.

--[no]sql
default: yes

Print DROP KEY statement for each duplicate key. By default an ALTER TABLE DROP KEY statement is printed below each duplicate key so that, if you want to remove the duplicate key, you can copy-paste the statement into MariaDB.

To disable printing these statements, specify `--no-sql`.

--[no]summary
default: yes

Print summary of indexes at end of output.

--tables
short form: -t; type: hash

Check only this comma-separated list of tables.

Table names may be qualified with the database name.

--user
short form: -u; type: string

User for login if not current user.

--verbose
short form: -v

Output all keys and/or foreign keys found, not just redundant ones.

--version
Show version and exit.

6.6 DSN OPTIONS

These DSN options are used to create a DSN. Each option is given like `option=value`. The options are case-sensitive, so P and p are not the same option. There cannot be whitespace before or after the = and if the value contains whitespace it must be quoted. DSN options are comma-separated. See the mariadb-tools manpage for full details.

- A

  dsn: charset; copy: yes

  Default character set.

- D

  dsn: database; copy: yes

  Default database.

- F

  dsn: mysql_read_default_file; copy: yes

  Only read default options from the given file

- h

  dsn: host; copy: yes

  Connect to host.

- p

  dsn: password; copy: yes

  Password to use when connecting. If password contains commas they must be escaped with a backslash: "exam,ple"

- P

  dsn: port; copy: yes

  Port number to use for connection.

- S

  dsn: mysql_socket; copy: yes

  Socket file to use for connection.

- u

  dsn: user; copy: yes

  User for login if not current user.

6.7 ENVIRONMENT

This tool does not use any environment variables.

6.8 SYSTEM REQUIREMENTS

You need Perl, and some core packages that ought to be installed in any reasonably new version of Perl.

6.9 AUTHORS

Cole Busby, Baron Schwartz, Brian Fraser, and Daniel Nichter

6.10 ABOUT THIS MARIADB TOOL

This tool is part of MariaDB client tools. This MariaDB Tool was forked from Percona Toolkit's pt-duplicate-key-checker in October, 2021. Percona Toolkit was forked from two projects in June, 2011: Maatkit and Aspersa. Those projects were created by Baron Schwartz and primarily developed by him and Daniel Nichter.

6.11 COPYRIGHT, LICENSE, AND WARRANTY

This program is copyright 2021 MariaDB Corporation and/or its affiliates, 2011-2018 Percona LLC and/or its affiliates, 2010-2011 Baron Schwartz.

THIS PROGRAM IS PROVIDED "AS IS" AND WITHOUT ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, WITHOUT LIMITATION, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE.

This program is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation, version 2; OR the Perl Artistic License. On UNIX and similar systems, you can issue `man perlgpl' or `man perlartistic' to read these licenses.

You should have received a copy of the GNU General Public License along with this program; if not, write to the Free Software Foundation, Inc., 59 Temple Place, Suite 330, Boston, MA 02111-1307 USA.

6.12 VERSION

`mariadb-index-checker` 6.0.0rc

CHAPTER

SEVEN

MARIADB-IOSTAT

7.1 NAME

mariadb-iostat - An interactive I/O monitoring tool for GNU/Linux.

7.2 SYNOPSIS

7.2.1 Usage

```
mariadb-iostat [OPTIONS] [FILES]
```

mariadb-iostat prints disk I/O statistics for GNU/Linux. It is somewhat similar to iostat, but it is interactive and more detailed. It can analyze samples gathered from another machine.

7.3 RISKS

mariadb-iostat is mature, proven in the real world, and well tested, but all database tools can pose a risk to the system and the database server. Before using this tool, please:

- Read the tool's documentation
- Test the tool on a non-production server
- Backup your production server and verify the backups

7.4 DESCRIPTION

The **mariadb-iostat** tool is similar to iostat, but has some advantages. It prints read and write statistics separately, and has more columns. It is menu-driven and interactive, with several different ways to aggregate the data. It integrates well with the mariadb-stat tool. It also does the "right thing" by default, such as hiding disks that are idle. These properties make it very convenient for quickly drilling down into I/O performance and inspecting disk behavior.

This program works in two modes. The default is to collect samples of */proc/diskstats* and print out the formatted statistics at intervals. The other mode is to process a file that contains saved samples of */proc/diskstats*; there is a shell script later in this documentation that shows how to collect such a file.

In both cases, the tool is interactively controlled by keystrokes, so you can redisplay and slice the data flexibly and easily. It loops forever, until you exit with the ‘q’ key. If you press the ‘?’ key, you will bring up the interactive help menu that shows which keys control the program.

When the program is gathering samples of */proc/diskstats* and refreshing its display, it prints information about the newest sample each time it refreshes. When it is operating on a file of saved samples, it redraws the entire file’s contents every time you change an option.

The program doesn’t print information about every block device on the system. It hides devices that it has never observed to have any activity. You can enable and disable this by pressing the ‘i’ key.

7.5 OUTPUT

In the rest of this documentation, we will try to clarify the distinction between block devices (/dev/sda1, for example), which the kernel presents to the application via a filesystem, versus the (usually) physical device underneath the block device, which could be a disk, a RAID controller, and so on. We will sometimes refer to logical I/O operations, which occur at the block device, versus physical I/Os which are performed on the underlying device. When we refer to the queue, we are speaking of the queue associated with the block device, which holds requests until they’re issued to the physical device.

The program’s output looks like the following sample, which is too wide for this manual page, so we have formatted it as several samples with line breaks:

```
#ts device rd_s rd_avkb rd_mb_s rd_mrg rd_cnc   rd_rt
{6} sda      0.9     4.2     0.0     0%    0.0   17.9
{6} sdb      0.4     4.0     0.0     0%    0.0   26.1
{6} dm-0     0.0     4.0     0.0     0%    0.0   13.5
{6} dm-1     0.8     4.0     0.0     0%    0.0   16.0

    ...     wr_s wr_avkb wr_mb_s wr_mrg wr_cnc   wr_rt
    ...     99.7     6.2     0.6    35%    3.7   23.7
    ...     14.5    15.8     0.2    75%    0.5    9.2
    ...      1.0     4.0     0.0     0%    0.0    2.3
    ...    117.7     4.0     0.5     0%    4.1   35.1

    ...             busy in_prg    io_s  qtime stime
    ...               6%      0   100.6   23.3   0.4
    ...               4%      0    14.9    8.6   0.6
    ...               0%      0     1.1    1.5   1.2
    ...               5%      0   118.5   34.5   0.4
```

The columns are as follows:

#ts

This column’s contents vary depending on the tool’s aggregation mode. In the default mode, when each line contains information about a single disk but possibly aggregates across several samples from that disk, this column shows the number of samples that were included into the line of output, in {curly braces}. In the example shown, each line of output aggregates {10} samples of */proc/diskstats*.

In the “all” group-by mode, this column shows timestamp offsets, relative to the time the tool began aggregating or the timestamp of the previous lines printed, depending on the mode. The output can be confusing to explain, but it’s rather intuitive when you see the lines appearing on your screen periodically.

Similarly, in “sample” group-by mode, the number indicates the total time span that is grouped into each sample.

If you specify *--show-timestamps*, this field instead shows the timestamp at which the sample was taken; if multiple timestamps are present in a single line of output, then the first timestamp is used.

device

The device name. If there is more than one device, then instead the number of devices aggregated into the line is shown, in {curly braces}.

rd_s

The average number of reads per second. This is the number of I/O requests that were sent to the underlying device. This usually is a smaller number than the number of logical IO requests made by applications. More requests might have been queued to the block device, but some of them usually are merged before being sent to the disk.

This field is computed from the contents of */proc/diskstats* as follows. See "KERNEL DOCUMENTATION" below for the meaning of the field numbers:

```
delta[field1] / delta[time]
```

rd_avkb

The average size of the reads, in kilobytes. This field is computed as follows:

```
2 * delta[field3] / delta[field1]
```

rd_mb_s

The average number of megabytes read per second. Computed as follows:

```
2 * delta[field3] / delta[time]
```

rd_mrg

The percentage of read requests that were merged together in the queue scheduler before being sent to the physical device. The field is computed as follows:

```
100 * delta[field2] / (delta[field2] + delta[field1])
```

rd_cnc

The average concurrency of the read operations, as computed by Little's Law. This is the end-to-end concurrency on the block device, not the underlying disk's concurrency. It includes time spent in the queue. The field is computed as follows:

```
delta[field4] / delta[time] / 1000 / devices-in-group
```

rd_rt

The average response time of the read operations, in milliseconds. This is the end-to-end response time, including time spent in the queue. It is the response time that the application making I/O requests sees, not the response time of the physical disk underlying the block device. It is computed as follows:

```
delta[field4] / (delta[field1] + delta[field2])
```

wr_s, wr_avkb, wr_mb_s, wr_mrg, wr_cnc, wr_rt

These columns show write activity, and they match the corresponding columns for read activity.

busy

The fraction of wall-clock time that the device had at least one request in progress; this is what iostat calls %util, and indeed it is utilization, depending on how you define utilization, but that is sometimes ambiguous in common parlance. It may also be called the residence time; the time during which at least one request was resident in the system. It is computed as follows:

```
100 * delta[field10] / (1000 * delta[time])
```

This field cannot exceed 100% unless there is a rounding error, but it is a common mistake to think that a device that's busy all the time is saturated. A device such as a RAID volume should support concurrency higher than 1, and solid-state drives can support very high concurrency. Concurrency can grow without bound, and is a more reliable indicator of how loaded the device really is.

in_prg

The number of requests that were in progress. Unlike the read and write concurrencies, which are averages that are generated from reliable numbers, this number is an instantaneous sample, and you can see that it might represent a spike of requests, rather than the true long-term average. If this number is large, it essentially means that the device is heavily loaded. It is computed as follows:

```
field9
```

ios_s

The average throughput of the physical device, in I/O operations per second (IOPS). This column shows the total IOPS the underlying device is handling. It is the sum of rd_s and wr_s.

qtime

The average queue time; that is, time a request spends in the device scheduler queue before being sent to the physical device. This is an average over reads and writes.

It is computed in a slightly complex way: the average response time seen by the application, minus the average service time (see the description of the next column). This is derived from the queueing theory formula for response time, R = W + S: response time = queue time + service time. This is solved for W, of course, to give W = R - S. The computation follows:

```
delta[field11] / (delta[field1, 2, 5, 6] + delta[field9])
   - delta[field10] / delta[field1, 2, 5, 6]
```

See the description for `stime` for more details and cautions.

stime

The average service time; that is, the time elapsed while the physical device processes the request, after the request finishes waiting in the queue. This is an average over reads and writes. It is computed from the queueing theory utilization formula, U = SX, solved for S. This means that utilization divided by throughput gives service time:

```
delta[field10] / (delta[field1, 2, 5, 6])
```

Note, however, that there can be some kernel bugs that cause field 9 in */proc/diskstats* to become negative, and this can cause field 10 to be wrong, thus making the service time computation not wholly trustworthy.

Note that in the above formula we use utilization very specifically. It is a duration, not a percentage.

You can compare the stime and qtime columns to see whether the response time for reads and writes is spent in the queue or on the physical device. However, you cannot see the difference between reads and writes. Changing the block device scheduler algorithm might improve queue time greatly. The default algorithm, cfq, is very bad for servers, and should only be used on laptops and workstations that perform tasks such as working with spreadsheets and surfing the Internet.

If you are used to using iostat, you might wonder where you can find the same information in **mariadb-iostat**. Here are two samples of output from both tools on the same machine at the same time, for */dev/sda*, wrapped to fit:

```
     #ts dev rd_s rd_avkb rd_mb_s rd_mrg rd_cnc   rd_rt
08:50:10 sda  0.0     0.0     0.0     0%    0.0     0.0
08:50:20 sda  0.4     4.0     0.0     0%    0.0    15.5
08:50:30 sda  2.1     4.4     0.0     0%    0.0    21.1
08:50:40 sda  2.4     4.0     0.0     0%    0.0    15.4
08:50:50 sda  0.1     4.0     0.0     0%    0.0    33.0

             wr_s wr_avkb wr_mb_s wr_mrg wr_cnc   wr_rt
              7.7    25.5     0.2    84%    0.0     0.3
             49.6     6.8     0.3    41%    2.4    28.8
            210.1     5.6     1.1    28%    7.4    25.2
            297.1     5.4     1.6    26%   11.4    28.3
             11.9    11.7     0.1    66%    0.2     4.9

                     busy  in_prg    io_s  qtime   stime
                       1%       0     7.7    0.1     0.2
                       6%       0    50.0   28.1     0.7
                      12%       0   212.2   24.8     0.4
                      16%       0   299.5   27.8     0.4
                       1%       0    12.0    4.7     0.3

         Dev rrqm/s   wrqm/s    r/s     w/s  rMB/s   wMB/s
08:50:10 sda   0.00    41.40   0.00    7.70   0.00    0.19
08:50:20 sda   0.00    34.70   0.40   49.60   0.00    0.33
08:50:30 sda   0.00    83.30   2.10  210.10   0.01    1.15
08:50:40 sda   0.00   105.10   2.40  297.90   0.01    1.58
08:50:50 sda   0.00    22.50   0.10   11.10   0.00    0.13

                 avgrq-sz avgqu-sz   await  svctm   %util
                    51.01     0.02    2.04   1.25    0.96
                    13.55     2.44   48.76   1.16    5.79
                    11.15     7.45   35.10   0.55   11.76
                    10.81    11.40   37.96   0.53   15.97
                    24.07     0.17   15.60   0.87    0.97
```

The correspondence between the columns is not one-to-one. In particular:

rrqm/s, wrqm/s

These columns in iostat are replaced by rd_mrg and wr_mrg in **mariadb-iostat**.

avgrq-sz

This column is in sectors in iostat, and is a combination of reads and writes. The **mariadb-iostat** output breaks these out separately and shows them in kB. You can derive it via a weighted average of rd_avkb and wr_avkb in **mariadb-iostat**, and then multiply by 2 to get sectors (each sector is 512 bytes).

avgqu-sz

This column really represents concurrency at the block device scheduler. The **mariadb-iostat** output shows concurrency for reads and writes separately: rd_cnc and wr_cnc.

await

This column is the average response time from the beginning to the end of a request to the block device, including queue time and service time, and is not shown in **mariadb-iostat**. Instead,

mariadb-iostat shows individual response times at the disk level for reads and writes (rd_rt and wr_rt), as well as queue time versus service time for reads and writes in aggregate.

svctm

This column is the average service time at the disk, and is shown as stime in **mariadb-iostat**.

%util

This column is called busy in **mariadb-iostat**. Utilization is usually defined as the portion of time during which there was at least one active request, not as a percentage, which is why we chose to avoid this confusing term.

7.6 COLLECTING DATA

It is straightforward to gather a sample of data for this tool. Files should have this format, with a timestamp line preceding each sample of statistics:

```
TS <timestamp>
<contents of /proc/diskstats>
TS <timestamp>
<contents of /proc/diskstats>
... et cetera
```

You can simply use **mariadb-iostat** with *--save-samples* to collect this data for you. If you wish to capture samples as part of some other tool, and use **mariadb-iostat** to analyze them, you can include a snippet of shell script such as the following:

```
INTERVAL=1
while true; do
   sleep=$(date +%s.%N | awk "{print $INTERVAL - (\$1 % $INTERVAL)}")
   sleep $sleep
   date +"TS %s.%N %F %T" >> diskstats-samples.txt
   cat /proc/diskstats >> diskstats-samples.txt
done
```

7.7 KERNEL DOCUMENTATION

This documentation supplements the official documentation on the contents of */proc/diskstats*. That documentation can sometimes be difficult to understand for those who are not familiar with Linux kernel internals. The contents of */proc/diskstats* are generated by the `diskstats_show()` function in the kernel source file *block/genhd.c*.

Here is a sample of */proc/diskstats* on a recent kernel.

```
8 1 sda1 426 243 3386 2056 3 0 18 87 0 2135 2142
```

The fields in this sample are as follows. The first three fields are the major and minor device numbers (8, 1), and the device name (sda1). They are followed by 11 fields of statistics:

1. The number of reads completed. This is the number of physical reads done by the underlying disk, not the number of reads that applications made from the block device. This means that 426 actual reads have completed successfully to the disk on which */dev/sda1* resides. Reads are not counted until they complete.
2. The number of reads merged because they were adjacent. In the sample, 243 reads were merged. This means that */dev/sda1* actually received 869 logical reads, but sent only 426 physical reads to the underlying physical device.

3. The number of sectors read successfully. The 426 physical reads to the disk read 3386 sectors. Sectors are 512 bytes, so a total of about 1.65MB have been read from */dev/sda1*.
4. The number of milliseconds spent reading. This counts only reads that have completed, not reads that are in progress. It counts the time spent from when requests are placed on the queue until they complete, not the time that the underlying disk spends servicing the requests. That is, it measures the total response time seen by applications, not disk response times.
5. Ditto for field 1, but for writes.
6. Ditto for field 2, but for writes.
7. Ditto for field 3, but for writes.
8. Ditto for field 4, but for writes.
9. The number of I/Os currently in progress, that is, they've been scheduled by the queue scheduler and issued to the disk (submitted to the underlying disk's queue), but not yet completed. There are bugs in some kernels that cause this number, and thus fields 10 and 11, to be wrong sometimes.
10. The total number of milliseconds spent doing I/Os. This is **not** the total response time seen by the applications; it is the total amount of time during which at least one I/O was in progress. If one I/O is issued at time 100, another comes in at 101, and both of them complete at 102, then this field increments by 2, not 3.
11. This field counts the total response time of all I/Os. In contrast to field 10, it counts double when two I/Os overlap. In our previous example, this field would increment by 3, not 2.

7.8 OPTIONS

This tool accepts additional command-line arguments. Refer to the "SYNOPSIS" and usage information for details.

`--columns-regex`
type: string; default: .

Print columns that match this Perl regex.

`--config`
type: Array

Read this comma-separated list of config files; if specified, this must be the first option on the command line.

`--devices-regex`
type: string

Print devices that match this Perl regex.

`--group-by`
type: string; default: all

Group-by mode: disk, sample, or all. In **disk** mode, each line of output shows one disk device, with the statistics computed since the tool started. In **sample** mode, each line of output shows one sample of statistics, with all disks averaged together. In **all** mode, each line of output shows one sample and one disk device.

`--headers`
type: Hash; default: group,scroll

If `group` is present, each sample will be separated by a blank line, unless the sample is only one line. If `scroll` is present, the tool will print the headers as often as needed to prevent them from scrolling out of view. Note that you can press the space bar, or the enter key, to reprint headers at will.

`--help`
Show help and exit.

--interval

type: int; default: 1

When in interactive mode, wait N seconds before printing to the screen. Also, how often the tool should sample */proc/diskstats*.

The tool attempts to gather statistics exactly on even intervals of clock time. That is, if you specify a 5-second interval, it will try to capture samples at 12:00:00, 12:00:05, and so on; it will not gather at 12:00:01, 12:00:06 and so forth.

This can lead to slightly odd delays in some circumstances, because the tool waits one full cycle before printing out the first set of lines. (Unlike iostat and vmstat, **mariadb-iostat** does not start with a line representing the averages since the computer was booted.) Therefore, the rule has an exception to avoid very long delays. Suppose you specify a 10-second interval, but you start the tool at 12:00:00.01. The tool might wait until 12:00:20 to print its first lines of output, and in the intervening 19.99 seconds, it would appear to do nothing.

To alleviate this, the tool waits until the next even interval of time to gather, unless more than 20% of that interval remains. This means the tool will never wait more than 120% of the sampling interval to produce output, e.g if you start the tool at 12:00:53 with a 10-second sampling interval, then the first sample will be only 7 seconds long, not 10 seconds.

--iterations

type: int

When in interactive mode, stop after N samples. Run forever by default.

--sample-time

type: int; default: 1

In –group-by sample mode, include N seconds of samples per group.

--save-samples

type: string

File to save diskstats samples in; these can be used for later analysis.

--show-inactive

Show inactive devices.

--show-timestamps

Show a 'HH:MM:SS' timestamp in the `#ts` column. If multiple timestamps are aggregated into one line, the first timestamp is shown.

--version

Show version and exit.

7.9 ENVIRONMENT

The environment variable `PTDEBUG` enables verbose debugging output to STDERR. To enable debugging and capture all output to a file, run the tool like:

```
PTDEBUG=1 mariadb-iostat ... > FILE 2>&1
```

Be careful: debugging output is voluminous and can generate several megabytes of output.

7.10 SYSTEM REQUIREMENTS

This tool requires Perl v5.8.0 or newer and the */proc* filesystem, unless reading from files.

7.11 AUTHORS

Cole Busby,Baron Schwartz, Brian Fraser, and Daniel Nichter

7.12 ABOUT THIS MARIADB TOOL

This tool is part of MariaDB client tools. This MariaDB Tool was forked from Percona Toolkit's pt-diskstat in August, 2019. Percona Toolkit was forked from two projects in June, 2011: Maatkit and Aspersa. Those projects were created by Baron Schwartz and primarily developed by him and Daniel Nichter.

7.13 COPYRIGHT, LICENSE, AND WARRANTY

7.14 VERSION

`mariadb-iostat` 6.0.0rc

CHAPTER

EIGHT

MARIADB-KILL

8.1 NAME

mariadb-kill - Kill MariaDB queries that match certain criteria.

8.2 SYNOPSIS

8.2.1 Usage

```
mariadb-kill [OPTIONS] [DSN]
```

mariadb-kill kills MariaDB connections. **mariadb-kill** connects to MariaDB and gets queries from SHOW PROCESSLIST if no FILE is given. Else, it reads queries from one or more FILE which contains the output of SHOW PROCESSLIST. If FILE is -, **mariadb-kill** reads from STDIN.

Kill queries running longer than 60s:

```
mariadb-kill --busy-time 60 --kill
```

Print, do not kill, queries running longer than 60s:

```
mariadb-kill --busy-time 60 --print
```

Check for sleeping processes and kill them all every 10s:

```
mariadb-kill --match-command Sleep --kill --victims all --interval 10
```

Print all login processes:

```
mariadb-kill --match-state login --print --victims all
```

See which queries in the processlist right now would match:

```
mariadb -e "SHOW PROCESSLIST" > proclist.txt
mariadb-kill --test-matching proclist.txt --busy-time 60 --print
```

8.3 RISKS

mariadb-kill is mature, proven in the real world, and well tested, but all database tools can pose a risk to the system and the database server. Before using this tool, please:

- Read the tool's documentation
- Review the tool's known "BUGS"
- Test the tool on a non-production server
- Backup your production server and verify the backups

8.4 DESCRIPTION

mariadb-kill captures queries from SHOW PROCESSLIST, filters them, and then either kills or prints them. This is also known as a "slow query sniper" in some circles. The idea is to watch for queries that might be consuming too many resources, and kill them.

For brevity, we talk about killing queries, but they may just be printed (or some other future action) depending on what options are given.

Normally **mariadb-kill** connects to MariaDB to get queries from SHOW PROCESSLIST. Alternatively, it can read SHOW PROCESSLIST output from files. In this case, **mariadb-kill** does not connect to MariaDB and *--kill* has no effect. You should use *--print* instead when reading files. The ability to read a file with *--test-matching* allows you to capture SHOW PROCESSLIST and test it later with **mariadb-kill** to make sure that your matches kill the proper queries. There are a lot of special rules to follow, such as "don't kill replication threads," so be careful not to kill something important!

Two important options to know are *--busy-time* and *--victims*. First, whereas most match/filter options match their corresponding value from SHOW PROCESSLIST (e.g. *--match-command* matches a query's Command value), the Time value is matched by *--busy-time*. See also *--interval*.

Second, *--victims* controls which matching queries from each class are killed. By default, the matching query with the highest Time value is killed (the oldest query). See the next section, "GROUP, MATCH AND KILL", for more details.

Usually you need to specify at least one `--match` option, else no queries will match. Or, you can specify *--match-all* to match all queries that aren't ignored by an `--ignore` option.

8.5 GROUP, MATCH AND KILL

Queries pass through several steps to determine which exactly will be killed (or printed–whatever action is specified). Understanding these steps will help you match precisely the queries you want.

The first step is grouping queries into classes. The *--group-by* option controls grouping. By default, this option has no value so all queries are grouped into one default class. All types of matching and filtering (the next step) are applied per-class. Therefore, you may need to group queries in order to match/filter some classes but not others.

The second step is matching. Matching implies filtering since if a query doesn't match some criteria, it is removed from its class. Matching happens for each class. First, queries are filtered from their class by the various `Query Matches` options like *--match-user*. Then, entire classes are filtered by the various `Class Matches` options like *--query-count*.

The third step is victim selection, that is, which matching queries in each class to kill. This is controlled by the *--victims* option. Although many queries in a class may match, you may only want to kill the oldest query, or all queries, etc.

The forth and final step is to take some action on all matching queries from all classes. The `Actions` options specify which actions will be taken. At this step, there are no more classes, just a single list of queries to kill, print, etc.

mariadb-kill will kill all the queries matching ANY of the specified criteria (logical OR). For example, using:

```
--busy-time 114 --match-command 'Query|Execute'
```

will kill all queries having busy-time > 114 `OR` where the command is `Query` or `Execute`

If you want to kill only the queries where `busy-time `` 114> ``AND` the command is Query or Execute, you need to use "–kill-busy-commands:

```
--busy-time 114 --kill-busy-commands 'Query|Execute'
```

8.6 OUTPUT

If only *--kill* is given, then there is no output. If only *--print* is given, then a timestamped KILL statement if printed for every query that would have been killed, like:

```
# 2009-07-15T15:04:01 KILL 8 (Query 42 sec) SELECT * FROM huge_table
```

The line shows a timestamp, the query's Id (8), its Time (42 sec) and its Info (usually the query SQL).

If both *--kill* and *--print* are given, then matching queries are killed and a line for each like the one above is printed.

Any command executed by *--execute-command* is responsible for its own output and logging. After being executed, **mariadb-kill** has no control or interaction with the command.

8.7 OPTIONS

Specify at least one of *--kill*, *--kill-query*, *--print*, *--execute-command* or *--stop*.

--any-busy-time and *--each-busy-time* are mutually exclusive.

--kill and *--kill-query* are mutually exclusive.

--daemonize and *--test-matching* are mutually exclusive.

This tool accepts additional command-line arguments. Refer to the "SYNOPSIS" and usage information for details.

--ask-pass
Prompt for a password when connecting to MariaDB.

--charset
short form: -A; type: string

Default character set. If the value is utf8, sets Perl's binmode on STDOUT to utf8, passes the mysql_enable_utf8 option to DBD::mysql, and runs SET NAMES UTF8 after connecting to MariaDB. Any other value sets binmode on STDOUT without the utf8 layer, and runs SET NAMES after connecting to MariaDB.

--config
type: Array

Read this comma-separated list of config files; if specified, this must be the first option on the command line.

--create-log-table
Create the *--log-dsn* table if it does not exist.

This option causes the table specified by *--log-dsn* to be created with the default structure shown in the documentation for that option.

--daemonize
Fork to the background and detach from the shell. POSIX operating systems only.

--database
short form: -D; type: string

The database to use for the connection.

--defaults-file
short form: -F; type: string

Only read MariaDB options from the given file. You must give an absolute pathname.

--filter
type: string

Discard events for which this Perl code doesn't return true.

This option is a string of Perl code or a file containing Perl code that gets compiled into a subroutine with one argument: $event. This is a hashref. If the given value is a readable file, then **mariadb-kill** reads the entire file and uses its contents as the code. The file should not contain a shebang (#!/usr/bin/perl) line.

If the code returns true, the chain of callbacks continues; otherwise it ends. The code is the last statement in the subroutine other than `return $event`. The subroutine template is:

```
sub { $event = shift; filter && return $event; }
```

Filters given on the command line are wrapped inside parentheses like like `( filter )`. For complex, multi-line filters, you must put the code inside a file so it will not be wrapped inside parentheses. Either way, the filter must produce syntactically valid code given the template. For example, an if-else branch given on the command line would not be valid:

```
--filter 'if () { } else { }'  # WRONG
```

Since it's given on the command line, the if-else branch would be wrapped inside parentheses which is not syntactically valid. So to accomplish something more complex like this would require putting the code in a file, for example filter.txt:

```
my $event_ok; if (...) { $event_ok=1; } else { $event_ok=0; } $event_ok
```

Then specify `--filter filter.txt` to read the code from filter.txt.

If the filter code won't compile, **mariadb-kill** will die with an error. If the filter code does compile, an error may still occur at runtime if the code tries to do something wrong (like pattern match an undefined value). **mariadb-kill** does not provide any safeguards so code carefully!

It is permissible for the code to have side effects (to alter `$event`).

--group-by
type: string

Apply matches to each class of queries grouped by this SHOW PROCESSLIST column. In addition to the basic columns of SHOW PROCESSLIST (user, host, command, state, etc.), queries can be matched by `fingerprint` which abstracts the SQL query in the `Info` column.

By default, queries are not grouped, so matches and actions apply to all queries. Grouping allows matches and actions to apply to classes of similar queries, if any queries in the class match.

For example, detecting cache stampedes (see `all-but-oldest` under *`--victims`* for an explanation of that term) requires that queries are grouped by the `arg` attribute. This creates classes of identical queries (stripped of comments). So queries `"SELECT c FROM t WHERE id=1"` and `"SELECT c FROM t WHERE id=1"` are grouped into the same class, but query c<"SELECT c FROM t WHERE id=3"> is not identical to the first two queries so it is grouped into another class. Then when *`--victims`* `all-but-oldest` is specified, all but the oldest query in each class is killed for each class of queries that matches the match criteria.

`--help`
Show help and exit.

`--host`
short form: -h; type: string; default: localhost

Connect to host.

`--interval`
type: time

How often to check for queries to kill. If *`--busy-time`* is not given, then the default interval is 30 seconds. Else the default is half as often as *`--busy-time`*. If both *`--interval`* and *`--busy-time`* are given, then the explicit *`--interval`* value is used.

See also *`--run-time`*.

`--log`
type: string

Print all output to this file when daemonized.

`--log-dsn`
type: DSN

Store each query killed in this DSN.

The argument specifies a table to store all killed queries. The DSN passed in must have the databse (D) and table (t) options. The table must have at least the following columns. You can add more columns for your own special purposes, but they won't be used by **`mariadb-kill`**. The following CREATE TABLE definition is also used for *`--create-log-table`*. MAGIC_create_log_table:

```
CREATE TABLE kill_log (
   kill_id     int(10) unsigned NOT NULL AUTO_INCREMENT,
   server_id   bigint(4) NOT NULL DEFAULT '0',
   timestamp   DATETIME,
   reason      TEXT,
   kill_error  TEXT,
   Id          bigint(4) NOT NULL DEFAULT '0',
   User        varchar(16) NOT NULL DEFAULT '',
   Host        varchar(64) NOT NULL DEFAULT '',
   db          varchar(64) DEFAULT NULL,
   Command     varchar(16) NOT NULL DEFAULT '',
   Time        int(7) NOT NULL DEFAULT '0',
   State       varchar(64) DEFAULT NULL,
   Info        longtext,
   Time_ms     bigint(21) DEFAULT '0', # NOTE, TODO: currently not used
```

(continues on next page)

(continued from previous page)

```
   PRIMARY KEY (kill_id)
) DEFAULT CHARSET=utf8
```

--password
short form: -p; type: string

Password to use when connecting. If password contains commas they must be escaped with a backslash: "exam,ple"

--pid
type: string

Create the given PID file. The tool won't start if the PID file already exists and the PID it contains is different than the current PID. However, if the PID file exists and the PID it contains is no longer running, the tool will overwrite the PID file with the current PID. The PID file is removed automatically when the tool exits.

--port
short form: -P; type: int

Port number to use for connection.

--query-id
Prints an ID of the query that was just killed. This is equivalent to the "ID" output of pt-query-digest. This allows cross-referencing the output of both tools.

Example:

```
Query ID 0xE9800998ECF8427E
```

Note that this is a digest (or hash) of the query's "fingerprint", so queries of the same form but with different values will have the same ID. See pt-query-digest for more information.

--rds
Denotes the instance in question is on Amazon RDS. By default **mariadb-kill** runs the MariaDB command "kill" for `--kill` and "kill query" `--kill-query`. On RDS these two commands are not available and are replaced by function calls. This option modifies `--kill` to use "CALL mysql.rds_kill(thread-id)" instead and `--kill-query` to use "CALL mysql.rds_kill_query(thread-id)"

--run-time
type: time

How long to run before exiting. By default **mariadb-kill** runs forever, or until its process is killed or stopped by the creation of a `--sentinel` file. If this option is specified, **mariadb-kill** runs for the specified amount of time and sleeps `--interval` seconds between each check of the PROCESSLIST.

--sentinel
type: string; default: /tmp/mariadb-kill-sentinel

Exit if this file exists.

The presence of the file specified by `--sentinel` will cause all running instances of **mariadb-kill** to exit. You might find this handy to stop cron jobs gracefully if necessary. See also `--stop`.

--slave-user
type: string

Sets the user to be used to connect to the slaves. This parameter allows you to have a different user with less privileges on the slaves but that user must exist on all slaves.

--slave-password
type: string

Sets the password to be used to connect to the slaves. It can be used with –slave-user and the password for the user must be the same on all slaves.

--set-vars

type: Array

Set the MariaDB variables in this comma-separated list of `variable=value` pairs.

By default, the tool sets:

```
wait_timeout=10000
```

Variables specified on the command line override these defaults. For example, specifying `--set-vars wait_timeout=500` overrides the defaultvalue of `10000`.

The tool prints a warning and continues if a variable cannot be set.

--socket

short form: -S; type: string

Socket file to use for connection.

--stop

Stop running instances by creating the *--sentinel* file.

Causes **mariadb-kill** to create the sentinel file specified by *--sentinel* and exit. This should have the effect of stopping all running instances which are watching the same sentinel file.

--[no]strip-comments

default: yes

Remove SQL comments from queries in the Info column of the PROCESSLIST.

--user

short form: -u; type: string

User for login if not current user.

--version

Show version and exit.

--victims

type: string; default: oldest

Which of the matching queries in each class will be killed. After classes have been matched/filtered, this option specifies which of the matching queries in each class will be killed (or printed, etc.). The following values are possible:

oldest

Only kill the single oldest query. This is to prevent killing queries that aren't really long-running, they're just long-waiting. This sorts matching queries by Time and kills the one with the highest Time value.

all

Kill all queries in the class.

all-but-oldest

Kill all but the oldest query. This is the inverse of the `oldest` value.

This value can be used to prevent "cache stampedes", the condition where several identical queries are executed and create a backlog while the first query attempts to finish. Since all queries are identical, all but the first query are killed so that it can complete and populate the cache.

--wait-after-kill
type: time

Wait after killing a query, before looking for more to kill. The purpose of this is to give blocked queries a chance to execute, so we don't kill a query that's blocking a bunch of others, and then kill the others immediately afterwards.

--wait-before-kill
type: time

Wait before killing a query. The purpose of this is to give `--execute-command` a chance to see the matching query and gather other MariaDB or system information before it's killed.

8.8 QUERY MATCHES

These options filter queries from their classes. If a query does not match, it is removed from its class. The `--ignore` options take precedence. The matches for command, db, host, etc. correspond to the columns returned by SHOW PROCESSLIST: Command, db, Host, etc. All pattern matches are case-sensitive by default, but they can be made case-insensitive by specifying a regex pattern like `(?i-xsm:select)`.

See also "GROUP, MATCH AND KILL".

--busy-time
type: time; group: Query Matches

Match queries that have been running for longer than this time. The queries must be in Command=Query status. This matches a query's Time value as reported by SHOW PROCESSLIST.

--idle-time
type: time; group: Query Matches

Match queries that have been idle/sleeping for longer than this time. The queries must be in Command=Sleep status. This matches a query's Time value as reported by SHOW PROCESSLIST.

--ignore-command
type: string; group: Query Matches

Ignore queries whose Command matches this Perl regex.

See `--match-command`.

--ignore-db
type: string; group: Query Matches

Ignore queries whose db (database) matches this Perl regex.

See `--match-db`.

--ignore-host
type: string; group: Query Matches

Ignore queries whose Host matches this Perl regex.

See `--match-host`.

--ignore-info
type: string; group: Query Matches

Ignore queries whose Info (query) matches this Perl regex.

See `--match-info`.

--[no]ignore-self
default: yes; group: Query Matches

Don't kill **mariadb-kill**'s own connection.

--ignore-state
type: string; group: Query Matches; default: Locked

Ignore queries whose State matches this Perl regex. The default is to keep threads from being killed if they are locked waiting for another thread.

See *--match-state*.

--ignore-user
type: string; group: Query Matches

Ignore queries whose user matches this Perl regex.

See *--match-user*.

--match-all
group: Query Matches

Match all queries that are not ignored. If no ignore options are specified, then every query matches (except replication threads, unless *--replication-threads* is also specified). This option allows you to specify negative matches, i.e. "match every query *except*…" where the exceptions are defined by specifying various `--ignore` options.

This option is *not* the same as *--victims* `all`. This option matches all queries within a class, whereas *--victims* `all` specifies that all matching queries in a class (however they matched) will be killed. Normally, however, the two are used together because if, for example, you specify *--victims* `oldest`, then although all queries may match, only the oldest will be killed.

--match-command
type: string; group: Query Matches

Match only queries whose Command matches this Perl regex.

Common Command values are:

```
Query
Sleep
Binlog Dump
Connect
Delayed insert
Execute
Fetch
Init DB
Kill
Prepare
Processlist
Quit
Reset stmt
Table Dump
```

See https://mariadb.com/kb/en/library/thread-command-values/ for a full list and description of Command values.

--match-db
type: string; group: Query Matches

Match only queries whose db (database) matches this Perl regex.

--match-host

type: string; group: Query Matches

Match only queries whose Host matches this Perl regex.

The Host value often time includes the port like “host:port”.

--match-info

type: string; group: Query Matches

Match only queries whose Info (query) matches this Perl regex.

The Info column of the processlist shows the query that is being executed or NULL if no query is being executed.

--match-state

type: string; group: Query Matches

Match only queries whose State matches this Perl regex.

Common State values are:

```
Locked
login
copy to tmp table
Copying to tmp table
Copying to tmp table on disk
Creating tmp table
executing
Reading from net
Sending data
Sorting for order
Sorting result
Table lock
Updating
```

See https://mariadb.com/kb/en/library/general-thread-states/ for a full list and description of State values.

--match-user

type: string; group: Query Matches

Match only queries whose User matches this Perl regex.

--replication-threads

group: Query Matches

Allow matching and killing replication threads.

By default, matches do not apply to replication threads; i.e. replication threads are completely ignored. Specifying this option allows matches to match (and potentially kill) replication threads on masters and slaves.

--test-matching

type: array; group: Query Matches

Files with processlist snapshots to test matching options against. Since the matching options can be complex, you can save snapshots of processlist in files, then test matching options against queries in those files.

This option disables *--run-time*, *--interval*, and *--[no]ignore-self*.

8.9 CLASS MATCHES

These matches apply to entire query classes. Classes are created by specifying the *--group-by* option, else all queries are members of a single, default class.

See also "GROUP, MATCH AND KILL".

--any-busy-time
type: time; group: Class Matches

Match query class if any query has been running for longer than this time. "Longer than" means that if you specify `10`, for example, the class will only match if there's at least one query that has been running for greater than 10 seconds.

See *--each-busy-time* for more details.

--each-busy-time
type: time; group: Class Matches

Match query class if each query has been running for longer than this time. "Longer than" means that if you specify `10`, for example, the class will only match if each and every query has been running for greater than 10 seconds.

See also *--any-busy-time* (to match a class if ANY query has been running longer than the specified time) and *--busy-time*.

--query-count
type: int; group: Class Matches

Match query class if it has at least this many queries. When queries are grouped into classes by specifying *--group-by*, this option causes matches to apply only to classes with at least this many queries. If *--group-by* is not specified then this option causes matches to apply only if there are at least this many queries in the entire SHOW PROCESSLIST.

--verbose
short form: -v

Print information to STDOUT about what is being done.

8.10 ACTIONS

These actions are taken for every matching query from all classes. The actions are taken in this order: *--print*, *--execute-command*, `--kill"/"--kill-query`. This order allows *--execute-command* to see the output of *--print* and the query before `--kill"/"--kill-query`. This may be helpful because **mariadb-kill** does not pass any information to *--execute-command*.

See also "GROUP, MATCH AND KILL".

--execute-command
type: string; group: Actions

Execute this command when a query matches.

After the command is executed, **mariadb-kill** has no control over it, so the command is responsible for its own info gathering, logging, interval, etc. The command is executed each time a query matches, so be careful that the command behaves well when multiple instances are ran. No information from **mariadb-kill** is passed to the command.

See also *--wait-before-kill*.

`--kill`

group: Actions

Kill the connection for matching queries.

This option makes **`mariadb-kill`** kill the connections (a.k.a. processes, threads) that have matching queries. Use *`--kill-query`* if you only want to kill individual queries and not their connections.

Unless *`--print`* is also given, no other information is printed that shows that **`mariadb-kill`** matched and killed a query.

See also *`--wait-before-kill`* and *`--wait-after-kill`*.

`--kill-busy-commands`

type: string; default: Query

group: Actions

Comma sepatated list of commands that will be watched/killed if they ran for more than *`--busy-time`* seconds. Default: `Query`

By default, *`--busy-time`* kills only `Query` commands but in some cases, it is needed to make *`--busy-time`* to watch and kill other commands. For example, a prepared statement execution command is `Execute` instead of `Query`. In this case, specifying `--kill-busy-commands=Query,Execute` will also kill the prepared stamente execution.

`--kill-query`

group: Actions

Kill matching queries.

This option makes **`mariadb-kill`** kill matching queries. This requires MariaDB 5.0 or newer. Unlike *`--kill`* which kills the connection for matching queries, this option only kills the query, not its connection.

`--print`

group: Actions

Print a KILL statement for matching queries; does not actually kill queries.

If you just want to see which queries match and would be killed without actually killing them, specify *`--print`*. To both kill and print matching queries, specify both *`--kill`* and *`--print`*.

8.11 DSN OPTIONS

These DSN options are used to create a DSN. Each option is given like `option=value`. The options are case-sensitive, so P and p are not the same option. There cannot be whitespace before or after the = and if the value contains whitespace it must be quoted. DSN options are comma-separated. See the percona-toolkit manpage for full details.

- A

  dsn: charset; copy: yes

  Default character set.

- D

  dsn: database; copy: yes

  Default database.

- F

dsn: mysql_read_default_file; copy: yes

Only read default options from the given file

- h

  dsn: host; copy: yes

  Connect to host.

- p

  dsn: password; copy: yes

  Password to use when connecting. If password contains commas they must be escaped with a backslash: "exam,ple"

- P

  dsn: port; copy: yes

  Port number to use for connection.

- S

  dsn: mariadb_socket; copy: yes

  Socket file to use for connection.

- u

  dsn: user; copy: yes

  User for login if not current user.

- t

  Table to log actions in, if passed through –log-dsn.

8.12 ENVIRONMENT

The environment variable `PTDEBUG` enables verbose debugging output to STDERR. To enable debugging and capture all output to a file, run the tool like:

```
PTDEBUG=1 mariadb-kill ... > FILE 2>&1
```

Be careful: debugging output is voluminous and can generate several megabytes of output.

8.13 SYSTEM REQUIREMENTS

You need Perl, DBI, DBD::mysql, and some core packages that ought to be installed in any reasonably new version of Perl.

8.14 AUTHORS

Cole Busby, Baron Schwartz and Daniel Nichter

8.15 ABOUT THIS MARIADB TOOL

This tool is part of MariaDB client tools. This MariaDB Tool was forked from Percona Toolkit's pt-stalk in August, 2019. Percona Toolkit was forked from two projects in June, 2011: Maatkit and Aspersa. Those projects were created by Baron Schwartz and primarily developed by him and Daniel Nichter.

8.16 COPYRIGHT, LICENSE, AND WARRANTY

8.17 VERSION

`mariadb-kill` 6.0.0rc

CHAPTER

NINE

MARIADB-PARTED

9.1 NAME

mariadb-parted - MySQL partition management script

9.2 EXAMPLES

```
# Create weekly partitions for the next quarter to test.part_table
mariadb-parted --add --interval w +1q h=localhost,D=test,t=part_table

# Create daily partitions for the next 2 weeks
# starting exactly at the beginning of every day
mariadb-parted --add --interval d +2w.startof h=localhost,D=test,t=part_table

# Email ops@example.com about partitions added
mariadb-parted --add --email-activity --email-to ops@example.com \
           --interval d +4w h=localhost,D=test,t=part_table

# Drop partitions older than 8 weeks
mariadb-parted --drop -8w h=localhost,D=test,t=part_table

# Drop partitions older than Dec 20th, 2010, but only 5 of them.
mariadb-parted --drop --limit 5 '2010-12-20 00:00:00' \
           h=localhost,D=test,t=part_table

# Drop and archive partitions older than 2 quarters ago.
mariadb-parted --drop --archive --archive-path /backups -2q \
           h=locahost,D=test,t=part_table

# Same as above, but archived to a separate database.
mariadb-parted --drop --archive --archive-database h=remotehost,D=test_archives,
↪t=part_table -2q \
           h=locahost,D=test,t=part_table

# Logging to syslog
mariadb-parted --logfile syslog:LOCAL0 --add --interval d 1y \
           h=localhost,D=test,t=part_table
```

9.3 SYNOPSIS

mariadb-parted [options] ACTION TIMESPEC DSN

This tool assists in the creation of partitions in regular intervals. It creates partitions in regular intervals up to some maximum future date.

```
--help,           -h   This help. See C<perldoc mariadb-parted> for full docs.
--dryrun,         -n   Report on actions without taking them.
--logfile,        -L   Direct output to given logfile. Default: none.

--email-activity       Send a brief email report of actions taken.
                       The email is sent to --email-to.
--use-hours           Use hours instead of days when checking partitions.
--partcol-integer      Assume no partitioning time or date function in use
--email-to,       -E   Where to send activity and failure emails.
                       Default: none.

--prefix,         -P   Partition prefix. Defaults to 'p'.

--archive              Archive partitions before dropping them.
--archive-path         Directory to place mysqldumps.
                       Default: current directory.
--archive-database     Database to archive partitions to.
                       Default: none

--limit,          -m   Limit the number of actions to be performed.
                       Default: 0 (unlimited)
```

9.4 ACTION

```
--add   Add partitions.
--drop  Remove partitions.
```

9.5 TIMESPEC

A timespec is a "natural" string to specify how far in advance to create partitions. A sampling of possible timespecs:

```
1w (create partitions one week in advance)
1m (one month)
2q (two quarters)
5h (five hours)
```

See the full documentation for a complete description of timespecs.

9.6 DSN

DSNs, such as those passed as option values, or arguments to a program are of the format: `({key}={value}(, {key}={value})*`. That is, a `key=value` pair, followed by a comma, followed by any number of additional `key=value` pairs separated by commas.

9.6.1 Examples

```
h=testdb1,u=pdb,p=frogs
h=localhost,S=/tmp/mysql.sock,u=root,F=/root/my.cnf
```

Where 'h' is a hostname, 'S' is a socket path, 'u' is a user, 'F' is a path to a defaults file, and 'p' is a password. These are non-exhaustive examples.

9.7 TIMESPEC

A timespec is one of:

```
A modifier to current local time,
A unix timestamp (assumed in UTC),
The string 'now' to refer to current local time,
An absolute time in 'YYYY-MM-DD HH:MM:SS' format,
An absolute time in 'YYYY-MD-DD HH:MM:SS TIMEZONE' format.
```

For the purposes of this module, TIMEZONE refers to zone names created and maintained by the zoneinfo database. See http://en.wikipedia.org/wiki/Tz_database for more information. Commonly used zone names are: Etc/UTC, US/Pacific and US/Eastern.

Since the last four aren't very complicated, this section describes what the modifiers are.

A modifer is, an optional plus or minus sign followed by a number, and then one of:

```
y = year, q = quarter , m = month, w = week, d = day, h = hour
```

Followed optionally by a space or a period and 'startof'. Which is described in the next section.

Some examples (the time is assumed to be 00:00:00):

```
-1y         (2010-11-01 -> 2009-11-01)
 5d         (2010-12-10 -> 2010-12-15)
-1w         (2010-12-13 -> 2010-12-07)
-1q startof (2010-05-01 -> 2010-01-01)
 1q.startof (2010-05-01 -> 2010-07-01)
```

9.8 startof

The 'startof' modifier for timespecs is a little confusing, but, is the only sane way to achieve latching like behavior. It adjusts the reference time so that it starts at the beginning of the requested type of interval. So, if you specify `-1h startof`, and the current time is: `2010-12-03 04:33:56`, first the calculation throws away `33:56` to get: `2010-12-03 04:00:00`, and then subtracts one hour to yield: `2010-12-03 03:00:00`.

Diagram of the 'startof' operator for timespec `-1q startof`, given the date `2010-05-01 00:00`.

```
        R P   C
        v v   v
 ---.---.---.---.---.--- Dec 2010
 ^   ^   ^   ^   ^   ^
 Jul Oct Jan Apr Jul Oct
2009    2010

. = quarter separator
C = current quarter
P = previous quarter
R = Resultant time (2010-01-01 00:00:00)
```

9.9 OPTIONS

--help, -h This help.

--dryrun, -n Report on actions that would be taken. Works best with the `Pdb_DEBUG` environment variable set to true.

See also: ENVIRONMENT

--logfile, -L Path to a file for logging, or, `syslog:<facility>` Where `<facility>` is a pre-defined logging facility for this machine.

See also: syslog(3), syslogd(8), syslog.conf(5)

--email-to, -E Where to send emails.

This tool can send emails on failure, and whenever it adds, drops, or archive partitions. Ordinarily, it will only send emails on failure.

`--email-activity`
If this flag is present, then this will make the tool also email whenver it adds, drops, or archives a partition.

`--use-hours`
If this flag is present, then partitions will be checked on the hour and not on the day. Useful when you need to partition by hour.

`--partcol-integer`
If this flag is present, then the tool will assume there is no partitioning function defined, e.g. if you are storing your date into an integer column

--prefix, -P Prefix for partition names. Partitions are always named like: <prefix>N. Where N is a number. Default is 'p', which was observed to be the most common prefix.

--interval, -i type: string one of: d w m y

Specifies the size of the each partition for the –add action. 'd' is day, 'w' is week, 'm' is month, and 'y' is year.

--limit
Specifies a limit to the number of partitions to add, drop, or archive. By default this is unlimited (0), so, for testing one usually wishes to set this to 1.

--archive
type: boolean

mysqldump partitions to files **in the current directory** named like <host>.<schema>.<table>.<partition_name>.sql

There is not currently a way to archive without dropping a partition.

--archive-path
What directory to place the SQL dumps of partition data in.

--archive-database
What database to place the archived partitions in.

9.10 ACTIONS

--add
Adds partitions till there are at least TIMESPEC –interval sized future buckets.

The adding of partitions is not done blindly. This will only add new partitions if there are fewer than TIMESPEC future partitions. For example:

```
Given: --interval d, today is: 2011-01-15, TIMESPEC is: +1w,
       last partition (p5) is for 2011-01-16;

Result:
  Parted will add 6 partitions to make the last partition 2011-01-22 (p11).

Before:
 |---+|
p0  p5

After:
 |---+-----|
p0  p5     p11
```

You can think of `--add` as specifying a required minimum safety zone.

--drop
Drops partitions strictly older than TIMESPEC. The partitions are not renumbered to start with p0 again.

```
Given: today is: 2011-01-15, TIMESPEC is: -1w,
       first partition (p0) is for 2011-01-06

Result: 2 partitions will be dropped.

Before: |-----+--|
        0     6  9
After : |---+--|
        2   6  9
```

9.11 ENVIRONMENT

Due to legacy reasons, this tool respond to the environment variable `Pdb_DEBUG` instead of PTDEBUG. This variable, when set to true, enables additional (very verbose) output from the tool.

9.12 SYSTEM REQUIREMENTS

You need Perl, DBI, DBD::mysql, and some core packages that ought to be installed in any reasonably new version of Perl.

9.13 ABOUT THIS MARIADB TOOL

This tool is part of MariaDB client tools. This MariaDB Tool was forked from PalominoDB's pdb-parted in 2019.

9.14 COPYRIGHT, LICENSE, AND WARRANTY

9.15 VERSION

`mariadb-parted` 6.0.0rc

CHAPTER

TEN

MARIADB-QUERY-DIGEST

10.1 NAME

mariadb-query-digest - Analyze MariaDB queries from logs, processlist, and tcpdump.

10.2 SYNOPSIS

10.2.1 Usage

```
mariadb-query-digest [OPTIONS] [FILES] [DSN]
```

mariadb-query-digest analyzes MariaDB queries from slow, general, and binary log files. It can also analyze queries from `SHOW PROCESSLIST` and MariaDB protocol data from tcpdump. By default, queries are grouped by fingerprint and reported in descending order of query time (i.e. the slowest queries first). If no `FILES` are given, the tool reads `STDIN`. The optional `DSN` is used for certain options like *--since* and *--until*.

Report the slowest queries from `slow.log`:

```
mariadb-query-digest slow.log
```

Report the slowest queries from the processlist on host1:

```
mariadb-query-digest --processlist h=host1
```

Capture MariaDB protocol data with tcppdump, then report the slowest queries:

```
tcpdump -s 65535 -x -nn -q -tttt -i any -c 1000 port 3306 > mariadb.tcp.txt

mariadb-query-digest --type tcpdump mariadb.tcp.txt
```

Save query data from `slow.log` to host2 for later review and trend analysis:

```
mariadb-query-digest --review h=host2 --no-report slow.log
```

10.3 RISKS

MariaDB Tools is mature, proven in the real world, and well tested, but all database tools can pose a risk to the system and the database server. Before using this tool, please:

- Read the tool's documentation
- Review the tool's known "BUGS"
- Test the tool on a non-production server
- Backup your production server and verify the backups

10.4 DESCRIPTION

mariadb-query-digest is a sophisticated but easy to use tool for analyzing MariaDB queries. It can analyze queries from MariaDB slow, general, and binary logs. (Binary logs must first be converted to text, see *--type*). It can also use `SHOW PROCESSLIST` and MariaDB protocol data from tcpdump. By default, the tool reports which queries are the slowest, and therefore the most important to optimize. More complex and custom-tailored reports can be created by using options like *--group-by*, *--filter*, and *--embedded-attributes*.

Query analysis is a best-practice that should be done frequently. To make this easier, **mariadb-query-digest** has two features: query review (*--review*) and query history (*--history*). When the *--review* option is used, all unique queries are saved to a database. When the tool is ran again with *--review*, queries marked as reviewed in the database are not printed in the report. This highlights new queries that need to be reviewed. When the *--history* option is used, query metrics (query time, lock time, etc.) for each unique query are saved to database. Each time the tool is ran with *--history*, the more historical data is saved which can be used to trend and analyze query performance over time.

10.5 ATTRIBUTES

mariadb-query-digest works on events, which are a collection of key-value pairs called attributes. You'll recognize most of the attributes right away: `Query_time`, `Lock_time`, and so on. You can just look at a slow log and see them. However, there are some that don't exist in the slow log, and slow logs may actually include different kinds of attributes depending upon the MariaDB Server version.

See "ATTRIBUTES REFERENCE" near the end of this documentation for a list of common and *--type* specific attributes. A familiarity with these attributes is necessary for working with *--filter*, *--ignore-attributes*, and other attribute-related options.

With creative use of *--filter*, you can create new attributes derived from existing attributes. For example, to create an attribute called `Row_ratio` for examining the ratio of `Rows_sent` to `Rows_examined`, specify a filter like:

```
--filter '($event->{Row_ratio} = $event->{Rows_sent} / ($event->{Rows_examined})) && 1
↪'
```

The `&& 1` trick is needed to create a valid one-line syntax that is always true, even if the assignment happens to evaluate false. The new attribute will automatically appears in the output:

```
# Row ratio          1.00     0.00       1     0.50       1     0.71     0.50
```

Attributes created this way can be specified for *--order-by* or any option that requires an attribute.

10.6 OUTPUT

The default `--output` is a query analysis report. The `--[no]report` option controls whether or not this report is printed. Sometimes you may want to parse all the queries but suppress the report, for example when using `--review` or `--history`.

There is one paragraph for each class of query analyzed. A "class" of queries all have the same value for the `--group-by` attribute which is `fingerprint` by default. (See "ATTRIBUTES".) A fingerprint is an abstracted version of the query text with literals removed, whitespace collapsed, and so forth. The report is formatted so it's easy to paste into emails without wrapping, and all non-query lines begin with a comment, so you can save it to a .sql file and open it in your favorite syntax-highlighting text editor. There is a response-time profile at the beginning.

The output described here is controlled by `--report-format`. That option allows you to specify what to print and in what order. The default output in the default order is described here.

The report, by default, begins with a paragraph about the entire analysis run The information is very similar to what you'll see for each class of queries in the log, but it doesn't have some information that would be too expensive to keep globally for the analysis. It also has some statistics about the code's execution itself, such as the CPU and memory usage, the local date and time of the run, and a list of input file read/parsed.

Following this is the response-time profile over the events. This is a highly summarized view of the unique events in the detailed query report that follows. It contains the following columns:

```
Column        Meaning
============  ==========================================================
Rank          The query's rank within the entire set of queries analyzed
Query ID      The query's fingerprint
Response time The total response time, and percentage of overall total
Calls         The number of times this query was executed
R/Call        The mean response time per execution
V/M           The Variance-to-mean ratio of response time
Item          The distilled query
```

A final line whose rank is shown as MISC contains aggregate statistics on the queries that were not included in the report, due to options such as `--limit` and `--outliers`. For details on the variance-to-mean ratio, please see http://en.wikipedia.org/wiki/Index_of_dispersion.

Next, the detailed query report is printed. Each query appears in a paragraph. Here is a sample, slightly reformatted so 'perldoc' will not wrap lines in a terminal. The following will all be one paragraph, but we'll break it up for commentary.

```
# Query 2: 0.01 QPS, 0.02x conc, ID 0xFDEA8D2993C9CAF3 at byte 160665
```

This line identifies the sequential number of the query in the sort order specified by `--order-by`. Then there's the queries per second, and the approximate concurrency for this query (calculated as a function of the timespan and total Query_time). Next there's a query ID. This ID is a hex version of the query's checksum in the database, if you're using `--review`. You can select the reviewed query's details from the database with a query like `SELECT .... WHERE checksum=0xFDEA8D2993C9CAF3`.

If you are investigating the report and want to print out every sample of a particular query, then the following `--filter` may be helpful:

```
mariadb-query-digest slow.log              \
   --no-report                             \
   --output slowlog                        \
   --filter '$event->{fingerprint} \
        && make_checksum($event->{fingerprint}) eq "FDEA8D2993C9CAF3"'
```

Notice that you must remove the `0x` prefix from the checksum.

Finally, in case you want to find a sample of the query in the log file, there's the byte offset where you can look. (This is not always accurate, due to some anomalies in the slow log format, but it's usually right.) The position refers to the worst sample, which we'll see more about below.

Next is the table of metrics about this class of queries.

```
#           pct   total     min     max     avg     95%  stddev  median
# Count       0       2
# Exec time  13   1105s    552s    554s    553s    554s      2s    553s
# Lock time   0   216us    99us   117us   108us   117us    12us   108us
# Rows sent  20   6.26M   3.13M   3.13M   3.13M   3.13M   12.73   3.13M
# Rows exam   0   6.26M   3.13M   3.13M   3.13M   3.13M   12.73   3.13M
```

The first line is column headers for the table. The percentage is the percent of the total for the whole analysis run, and the total is the actual value of the specified metric. For example, in this case we can see that the query executed 2 times, which is 13% of the total number of queries in the file. The min, max and avg columns are self-explanatory. The 95% column shows the 95th percentile; 95% of the values are less than or equal to this value. The standard deviation shows you how tightly grouped the values are. The standard deviation and median are both calculated from the 95th percentile, discarding the extremely large values.

The stddev, median and 95th percentile statistics are approximate. Exact statistics require keeping every value seen, sorting, and doing some calculations on them. This uses a lot of memory. To avoid this, we keep 1000 buckets, each of them 5% bigger than the one before, ranging from .000001 up to a very big number. When we see a value we increment the bucket into which it falls. Thus we have fixed memory per class of queries. The drawback is the imprecision, which typically falls in the 5 percent range.

Next we have statistics on the users, databases and time range for the query.

```
# Users       1   user1
# Databases   2     db1(1), db2(1)
# Time range 2008-11-26 04:55:18 to 2008-11-27 00:15:15
```

The users and databases are shown as a count of distinct values, followed by the values. If there's only one, it's shown alone; if there are many, we show each of the most frequent ones, followed by the number of times it appears.

```
# Query_time distribution
#   1us
#  10us
# 100us
#   1ms
#  10ms  #####
# 100ms  ####################
#    1s  ##########
#  10s+
```

The execution times show a logarithmic chart of time clustering. Each query goes into one of the "buckets" and is counted up. The buckets are powers of ten. The first bucket is all values in the "single microsecond range" – that is, less than 10us. The second is "tens of microseconds," which is from 10us up to (but not including) 100us; and so on. The charted attribute can be changed by specifying `--report-histogram` but is limited to time-based attributes.

```
# Tables
#    SHOW TABLE STATUS LIKE 'table1'\G
#    SHOW CREATE TABLE `table1`\G
# EXPLAIN
SELECT * FROM table1\G
```

This section is a convenience: if you're trying to optimize the queries you see in the slow log, you probably want to examine the table structure and size. These are copy-and-paste-ready commands to do that.

Finally, we see a sample of the queries in this class of query. This is not a random sample. It is the query that performed the worst, according to the sort order given by *--order-by*. You will normally see a commented `# EXPLAIN` line just before it, so you can copy-paste the query to examine its EXPLAIN plan. But for non-SELECT queries that isn't possible to do, so the tool tries to transform the query into a roughly equivalent SELECT query, and adds that below.

If you want to find this sample event in the log, use the offset mentioned above, and something like the following:

```
tail -c +<offset> /path/to/file | head
```

See also *--report-format*.

10.7 QUERY REVIEW

A query *--review* is the process of storing all the query fingerprints analyzed. This has several benefits:

- You can add metadata to classes of queries, such as marking them for follow-up, adding notes to queries, or marking them with an issue ID for your issue tracking system.
- You can refer to the stored values on subsequent runs so you'll know whether you've seen a query before. This can help you cut down on duplicated work.
- You can store historical data such as the row count, query times, and generally anything you can see in the report.

To use this feature, you run **mariadb-query-digest** with the *--review* option. It will store the fingerprints and other information into the table you specify. Next time you run it with the same option, it will do the following:

- It won't show you queries you've already reviewed. A query is considered to be already reviewed if you've set a value for the `reviewed_by` column. (If you want to see queries you've already reviewed, use the *--report-all* option.)
- Queries that you've reviewed, and don't appear in the output, will cause gaps in the query number sequence in the first line of each paragraph. And the value you've specified for *--limit* will still be honored. So if you've reviewed all queries in the top 10 and you ask for the top 10, you won't see anything in the output.
- If you want to see the queries you've already reviewed, you can specify *--report-all*. Then you'll see the normal analysis output, but you'll also see the information from the review table, just below the execution time graph. For example,

  ```
  # Review information
  #      comments: really bad IN() subquery, fix soon!
  #    first_seen: 2008-12-01 11:48:57
  #   jira_ticket: 1933
  #     last_seen: 2008-12-18 11:49:07
  #      priority: high
  #   reviewed_by: xaprb
  #   reviewed_on: 2008-12-18 15:03:11
  ```

  This metadata is useful because, as you analyze your queries, you get your comments integrated right into the report.

10.8 FINGERPRINTS

A query fingerprint is the abstracted form of a query, which makes it possible to group similar queries together. Abstracting a query removes literal values, normalizes whitespace, and so on. For example, consider these two queries:

```
SELECT name, password FROM user WHERE id='12823';
select name,   password from user
   where id=5;
```

Both of those queries will fingerprint to

```
select name, password from user where id=?
```

Once the query's fingerprint is known, we can then talk about a query as though it represents all similar queries.

What **mariadb-query-digest** does is analogous to a GROUP BY statement in SQL. (But note that "multiple columns" doesn't define a multi-column grouping; it defines multiple reports!) If your command-line looks like this,

```
mariadb-query-digest               \
    --group-by fingerprint    \
    --order-by Query_time:sum \
    --limit 10                \
    slow.log
```

The corresponding pseudo-SQL looks like this:

```
SELECT WORST(query BY Query_time), SUM(Query_time), ...
FROM /path/to/slow.log
GROUP BY FINGERPRINT(query)
ORDER BY SUM(Query_time) DESC
LIMIT 10
```

You can also use the value `distill`, which is a kind of super-fingerprint. See *--group-by* for more.

Query fingerprinting accommodates many special cases, which have proven necessary in the real world. For example, an `IN` list with 5 literals is really equivalent to one with 4 literals, so lists of literals are collapsed to a single one. If you find something that is not fingerprinted properly, please submit a bug report with a reproducible test case.

Here is a list of transformations during fingerprinting, which might not be exhaustive:

- Group all SELECT queries from mariadb-dump together, even if they are against different tables. The same applies to all queries from pt-table-checksum.
- Shorten multi-value INSERT statements to a single VALUES() list.
- Strip comments.
- Abstract the databases in USE statements, so all USE statements are grouped together.
- Replace all literals, such as quoted strings. For efficiency, the code that replaces literal numbers is somewhat non-selective, and might replace some things as numbers when they really are not. Hexadecimal literals are also replaced. NULL is treated as a literal. Numbers embedded in identifiers are also replaced, so tables named similarly will be fingerprinted to the same values (e.g. users_2009 and users_2010 will fingerprint identically).
- Collapse all whitespace into a single space.
- Lowercase the entire query.
- Replace all literals inside of IN() and VALUES() lists with a single placeholder, regardless of cardinality.
- Collapse multiple identical UNION queries into a single one.

10.9 OPTIONS

This tool accepts additional command-line arguments. Refer to the "SYNOPSIS" and usage information for details.

`--ask-pass`
Prompt for a password when connecting to MariaDB.

`--attribute-aliases`
type: array; default: db|Schema

List of attribute|alias,etc.

Certain attributes have multiple names, like db and Schema. If an event does not have the primary attribute, **`mariadb-query-digest`** looks for an alias attribute. If it finds an alias, it creates the primary attribute with the alias attribute's value and removes the alias attribute.

If the event has the primary attribute, all alias attributes are deleted.

This helps simplify event attributes so that, for example, there will not be report lines for both db and Schema.

`--attribute-value-limit`
type: int; default: 0

A sanity limit for attribute values.

This option deals with bugs in slow logging functionality that causes large values for attributes. If the attribute's value is bigger than this, the last-seen value for that class of query is used instead. Disabled by default.

`--charset`
short form: -A; type: string

Default character set. If the value is utf8, sets Perl's binmode on STDOUT to utf8, passes the mysql_enable_utf8 option to DBD::mysql, and runs SET NAMES UTF8 after connecting to MariaDB. Any other value sets binmode on STDOUT without the utf8 layer, and runs SET NAMES after connecting to MariaDB.

`--config`
type: Array

Read this comma-separated list of config files; if specified, this must be the first option on the command line.

`--[no]continue-on-error`
default: yes

Continue parsing even if there is an error. The tool will not continue forever: it stops once any process causes 100 errors, in which case there is probably a bug in the tool or the input is invalid.

`--[no]create-history-table`
default: yes

Create the *`--history`* table if it does not exist.

This option causes the table specified by *`--history`* to be created with the default structure shown in the documentation for *`--history`*.

`--[no]create-review-table`
default: yes

Create the *`--review`* table if it does not exist.

This option causes the table specified by *`--review`* to be created with the default structure shown in the documentation for *`--review`*.

`--daemonize`
Fork to the background and detach from the shell. POSIX operating systems only.

--database
short form: -D; type: string

Connect to this database.

--defaults-file
short form: -F; type: string

Only read mariadb options from the given file. You must give an absolute pathname.

--embedded-attributes
type: array

Two Perl regex patterns to capture pseudo-attributes embedded in queries.

Embedded attributes might be special attribute-value pairs that you've hidden in comments. The first regex should match the entire set of attributes (in case there are multiple). The second regex should match and capture attribute-value pairs from the first regex.

For example, suppose your query looks like the following:

```
SELECT * from users -- file: /login.php, line: 493;
```

You might run **mariadb-query-digest** with the following option:

```
:program:`mariadb-query-digest` --embedded-attributes ' -- .*','(\w+): ([^\,]+)'
```

The first regular expression captures the whole comment:

```
" -- file: /login.php, line: 493;"
```

The second one splits it into attribute-value pairs and adds them to the event:

```
ATTRIBUTE  VALUE
=========  ==========
file       /login.php
line       493
```

NOTE: All commas in the regex patterns must be escaped with otherwise the pattern will break.

--expected-range
type: array; default: 5,10

Explain items when there are more or fewer than expected.

Defines the number of items expected to be seen in the report given by *--[no]report*, as controlled by *--limit* and *--outliers*. If there are more or fewer items in the report, each one will explain why it was included.

--explain
type: DSN

Run EXPLAIN for the sample query with this DSN and print results.

This works only when *--group-by* includes fingerprint. It causes **mariadb-query-digest** to run EXPLAIN and include the output into the report. For safety, queries that appear to have a subquery that EXPLAIN will execute won't be EXPLAINed. Those are typically "derived table" queries of the form

```
select ... from ( select .... ) der;
```

The EXPLAIN results are printed as a full vertical format in the event report, which appears at the end of each event report in vertical style (`\G`) just like MariaDB prints it.

--filter

type: string

Discard events for which this Perl code doesn't return true.

This option is a string of Perl code or a file containing Perl code that gets compiled into a subroutine with one argument: $event. This is a hashref. If the given value is a readable file, then **mariadb-query-digest** reads the entire file and uses its contents as the code. The file should not contain a shebang (#!/usr/bin/perl) line.

If the code returns true, the chain of callbacks continues; otherwise it ends. The code is the last statement in the subroutine other than `return $event`. The subroutine template is:

```
sub { $event = shift; filter && return $event; }
```

Filters given on the command line are wrapped inside parentheses like like `( filter )`. For complex, multi-line filters, you must put the code inside a file so it will not be wrapped inside parentheses. Either way, the filter must produce syntactically valid code given the template. For example, an if-else branch given on the command line would not be valid:

```
--filter 'if () { } else { }'  # WRONG
```

Since it's given on the command line, the if-else branch would be wrapped inside parentheses which is not syntactically valid. So to accomplish something more complex like this would require putting the code in a file, for example filter.txt:

```
my $event_ok; if (...) { $event_ok=1; } else { $event_ok=0; } $event_ok
```

Then specify `--filter filter.txt` to read the code from filter.txt.

If the filter code won't compile, **mariadb-query-digest** will die with an error. If the filter code does compile, an error may still occur at runtime if the code tries to do something wrong (like pattern match an undefined value). **mariadb-query-digest** does not provide any safeguards so code carefully!

An example filter that discards everything but SELECT statements:

```
--filter '$event->{arg} =~ m/^select/i'
```

This is compiled into a subroutine like the following:

```
sub { $event = shift; ( $event->{arg} =~ m/^select/i ) && return $event; }
```

It is permissible for the code to have side effects (to alter `$event`).

See "ATTRIBUTES REFERENCE" for a list of common and *--type* specific attributes.

Here are more examples of filter code:

Host/IP matches domain.com

–filter '($event->{host} || $event->{ip} || "") =~ m/domain.com/'

Sometimes MariaDB logs the host where the IP is expected. Therefore, we check both.

User matches john

–filter '($event->{user} || "") =~ m/john/'

More than 1 warning

–filter '($event->{Warning_count} || 0) > 1'

Query does full table scan or full join

–filter '(($event->{Full_scan} || "") eq "Yes") || (($event->{Full_join} || "") eq "Yes")'

Query was not served from query cache

–filter '($event->{QC_Hit} || "") eq "No"'

Query is 1 MB or larger

–filter '$event->{bytes} >= 1_048_576'

Since `--filter` allows you to alter `$event`, you can use it to do other things, like create new attributes. See "ATTRIBUTES" for an example.

`--group-by`

type: Array; default: fingerprint

Which attribute of the events to group by.

In general, you can group queries into classes based on any attribute of the query, such as `user` or `db`, which will by default show you which users and which databases get the most `Query_time`. The default attribute, `fingerprint`, groups similar, abstracted queries into classes; see below and see also "FINGERPRINTS".

A report is printed for each `--group-by` value (unless `--no-report` is given). Therefore, `--group-by user,db` means "report on queries with the same user and report on queries with the same db"; it does not mean "report on queries with the same user and db." See also "OUTPUT".

Every value must have a corresponding value in the same position in `--order-by`. However, adding values to `--group-by` will automatically add values to `--order-by`, for your convenience.

There are several magical values that cause some extra data mining to happen before the grouping takes place:

fingerprint

This causes events to be fingerprinted to abstract queries into a canonical form, which is then used to group events together into a class. See "FINGERPRINTS" for more about fingerprinting.

tables

This causes events to be inspected for what appear to be tables, and then aggregated by that. Note that a query that contains two or more tables will be counted as many times as there are tables; so a join against two tables will count the Query_time against both tables.

distill

This is a sort of super-fingerprint that collapses queries down into a suggestion of what they do, such as `INSERT SELECT table1 table2`.

`--help`

Show help and exit.

`--history`

type: DSN

Save metrics for each query class in the given table. **`mariadb-query-digest`** saves query metrics (query time, lock time, etc.) to this table so you can see how query classes change over time.

The default table is `mariadb_tools.query_history`. Specify database (D) and table (t) DSN options to override the default. The database and table are automatically created unless `--no-create-history-table` is specified (see `--[no]create-history-table`).

`mariadb-query-digest` inspects the columns in the table. The table must have at least the following columns:

```
CREATE TABLE query_review_history (
  checksum     CHAR(32) NOT NULL,
  sample       LONGTEXT NOT NULL
);
```

Any columns not mentioned above are inspected to see if they follow a certain naming convention. The column is special if the name ends with an underscore followed by any of these values:

```
pct|avg|cnt|sum|min|max|pct_95|stddev|median|rank
```

If the column ends with one of those values, then the prefix is interpreted as the event attribute to store in that column, and the suffix is interpreted as the metric to be stored. For example, a column named `Query_time_min` will be used to store the minimum `Query_time` for the class of events.

The table should also have a primary key, but that is up to you, depending on how you want to store the historical data. We suggest adding ts_min and ts_max columns and making them part of the primary key along with the checksum. But you could also just add a ts_min column and make it a DATE type, so you'd get one row per class of queries per day.

The following table definition is used for *--[no]create-history-table*:

```
CREATE TABLE IF NOT EXISTS query_history (
  checksum             CHAR(32) NOT NULL,
  sample               LONGTEXT NOT NULL,
  ts_min               DATETIME,
  ts_max               DATETIME,
  ts_cnt               FLOAT,
  Query_time_sum       FLOAT,
  Query_time_min       FLOAT,
  Query_time_max       FLOAT,
  Query_time_pct_95    FLOAT,
  Query_time_stddev    FLOAT,
  Query_time_median    FLOAT,
  Lock_time_sum        FLOAT,
  Lock_time_min        FLOAT,
  Lock_time_max        FLOAT,
  Lock_time_pct_95     FLOAT,
  Lock_time_stddev     FLOAT,
  Lock_time_median     FLOAT,
  Rows_sent_sum        FLOAT,
  Rows_sent_min        FLOAT,
  Rows_sent_max        FLOAT,
  Rows_sent_pct_95     FLOAT,
  Rows_sent_stddev     FLOAT,
  Rows_sent_median     FLOAT,
  Rows_examined_sum    FLOAT,
  Rows_examined_min    FLOAT,
  Rows_examined_max    FLOAT,
  Rows_examined_pct_95 FLOAT,
  Rows_examined_stddev FLOAT,
  Rows_examined_median FLOAT,
  -- extended slowlog attributes
  Rows_affected_sum             FLOAT,
  Rows_affected_min             FLOAT,
  Rows_affected_max             FLOAT,
  Rows_affected_pct_95          FLOAT,
  Rows_affected_stddev          FLOAT,
```

(continues on next page)

(continued from previous page)

```
Rows_affected_median          FLOAT,
Rows_read_sum                 FLOAT,
Rows_read_min                 FLOAT,
Rows_read_max                 FLOAT,
Rows_read_pct_95              FLOAT,
Rows_read_stddev              FLOAT,
Rows_read_median              FLOAT,
Merge_passes_sum              FLOAT,
Merge_passes_min              FLOAT,
Merge_passes_max              FLOAT,
Merge_passes_pct_95           FLOAT,
Merge_passes_stddev           FLOAT,
Merge_passes_median           FLOAT,
InnoDB_IO_r_ops_min           FLOAT,
InnoDB_IO_r_ops_max           FLOAT,
InnoDB_IO_r_ops_pct_95        FLOAT,
InnoDB_IO_r_ops_stddev        FLOAT,
InnoDB_IO_r_ops_median        FLOAT,
InnoDB_IO_r_bytes_min         FLOAT,
InnoDB_IO_r_bytes_max         FLOAT,
InnoDB_IO_r_bytes_pct_95      FLOAT,
InnoDB_IO_r_bytes_stddev      FLOAT,
InnoDB_IO_r_bytes_median      FLOAT,
InnoDB_IO_r_wait_min          FLOAT,
InnoDB_IO_r_wait_max          FLOAT,
InnoDB_IO_r_wait_pct_95       FLOAT,
InnoDB_IO_r_wait_stddev       FLOAT,
InnoDB_IO_r_wait_median       FLOAT,
InnoDB_rec_lock_wait_min      FLOAT,
InnoDB_rec_lock_wait_max      FLOAT,
InnoDB_rec_lock_wait_pct_95   FLOAT,
InnoDB_rec_lock_wait_stddev   FLOAT,
InnoDB_rec_lock_wait_median   FLOAT,
InnoDB_queue_wait_min         FLOAT,
InnoDB_queue_wait_max         FLOAT,
InnoDB_queue_wait_pct_95      FLOAT,
InnoDB_queue_wait_stddev      FLOAT,
InnoDB_queue_wait_median      FLOAT,
InnoDB_pages_distinct_min     FLOAT,
InnoDB_pages_distinct_max     FLOAT,
InnoDB_pages_distinct_pct_95  FLOAT,
InnoDB_pages_distinct_stddev  FLOAT,
InnoDB_pages_distinct_median  FLOAT,
-- Boolean (Yes/No) attributes.  Only the cnt and sum are needed
-- for these.  cnt is how many times is attribute was recorded,
-- and sum is how many of those times the value was Yes.  So
-- sum/cnt * 100 equals the percentage of recorded times that
-- the value was Yes.
QC_Hit_cnt          FLOAT,
QC_Hit_sum          FLOAT,
Full_scan_cnt       FLOAT,
Full_scan_sum       FLOAT,
Full_join_cnt       FLOAT,
Full_join_sum       FLOAT,
Tmp_table_cnt       FLOAT,
Tmp_table_sum       FLOAT,
Tmp_table_on_disk_cnt FLOAT,
```

(continues on next page)

(continued from previous page)

```
  Tmp_table_on_disk_sum FLOAT,
  Filesort_cnt          FLOAT,
  Filesort_sum          FLOAT,
  Filesort_on_disk_cnt  FLOAT,
  Filesort_on_disk_sum  FLOAT,
  PRIMARY KEY(checksum, ts_min, ts_max)
);
```

Note that we store the count (cnt) for the ts attribute only; it will be redundant to store this for other attributes.

Starting from MariaDB Toolkit 3.0.11, the checksum function has been updated to use 32 chars in the MD5 sum. This causes the checksum field in the history table will have a different value than in the previous versions of the tool.

--host
short form: -h; type: string

Connect to host.

--ignore-attributes
type: array; default: arg, cmd, insert_id, ip, port, Thread_id, timestamp, exptime, flags, key, res, val, server_id, offset, end_log_pos, Xid

Do not aggregate these attributes. Some attributes are not query metrics but metadata which doesn't need to be (or can't be) aggregated.

--inherit-attributes
type: array; default: db,ts

If missing, inherit these attributes from the last event that had them.

This option sets which attributes are inherited or carried forward to events which do not have them. For example, if one event has the db attribute equal to "foo", but the next event doesn't have the db attribute, then it inherits "foo" for its db attribute.

--interval
type: float; default: .1

How frequently to poll the processlist, in seconds.

--iterations
type: int; default: 1

How many times to iterate through the collect-and-report cycle. If 0, iterate to infinity. Each iteration runs for *--run-time* amount of time. An iteration is usually determined by an amount of time and a report is printed when that amount of time elapses. With *--run-time-mode* `interval`, an interval is instead determined by the interval time you specify with *--run-time*. See *--run-time* and *--run-time-mode* for more information.

--limit
type: Array; default: 95%:20

Limit output to the given percentage or count.

If the argument is an integer, report only the top N worst queries. If the argument is an integer followed by the `%` sign, report that percentage of the worst queries. If the percentage is followed by a colon and another integer, report the top percentage or the number specified by that integer, whichever comes first.

The value is actually a comma-separated array of values, one for each item in *--group-by*. If you don't specify a value for any of those items, the default is the top 95%.

See also *--outliers*.

--log

type: string

Print all output to this file when daemonized.

--max-hostname-length

type: int; default: 10

Trim host names in reports to this length. 0=Do not trim host names.

--max-line-length

type: int; default: 74

Trim lines to this length. 0=Do not trim lines.

--order-by

type: Array; default: Query_time:sum

Sort events by this attribute and aggregate function.

This is a comma-separated list of order-by expressions, one for each *--group-by* attribute. The default `Query_time:sum` is used for *--group-by* attributes without explicitly given *--order-by* attributes (that is, if you specify more *--group-by* attributes than corresponding *--order-by* attributes). The syntax is `attribute:aggregate`. See "ATTRIBUTES" for valid attributes. Valid aggregates are:

```
Aggregate Meaning
========= ============================
sum       Sum/total attribute value
min       Minimum attribute value
max       Maximum attribute value
cnt       Frequency/count of the query
```

For example, the default `Query_time:sum` means that queries in the query analysis report will be ordered (sorted) by their total query execution time ("Exec time"). `Query_time:max` orders the queries by their maximum query execution time, so the query with the single largest `Query_time` will be list first. `cnt` refers more to the frequency of the query as a whole, how often it appears; "Count" is its corresponding line in the query analysis report. So any attribute and `cnt` should yield the same report wherein queries are sorted by the number of times they appear.

When parsing general logs (*--type* `genlog`), the default *--order-by* becomes `Query_time:cnt`. General logs do not report query times so only the `cnt` aggregate makes sense because all query times are zero.

If you specify an attribute that doesn't exist in the events, then **mariadb-query-digest** falls back to the default `Query_time:sum` and prints a notice at the beginning of the report for each query class. You can create attributes with *--filter* and order by them; see "ATTRIBUTES" for an example.

--outliers

type: array; default: Query_time:1:10

Report outliers by attribute:percentile:count.

The syntax of this option is a comma-separated list of colon-delimited strings. The first field is the attribute by which an outlier is defined. The second is a number that is compared to the attribute's 95th percentile. The third is optional, and is compared to the attribute's cnt aggregate. Queries that pass this specification are added to the report, regardless of any limits you specified in *--limit*.

For example, to report queries whose 95th percentile Query_time is at least 60 seconds and which are seen at least 5 times, use the following argument:

```
--outliers Query_time:60:5
```

You can specify an –outliers option for each value in *--group-by*.

--output

type: string; default: report

How to format and print the query analysis results. Accepted values are:

```
VALUE          FORMAT
=======        ==============================
report         Standard query analysis report
slowlog        MariaDB slow log
json           JSON, on array per query class
json-anon      JSON without example queries
secure-slowlog JSON without example queries
```

The entire `report` output can be disabled by specifying `--no-report` (see *--[no]report*), and its sections can be disabled or rearranged by specifying *--report-format*.

`json` output was introduced in 2.2.1 and is still in development, so the data structure may change in future versions.

--password

short form: -p; type: string

Password to use when connecting. If password contains commas they must be escaped with a backslash: "exam,ple"

--pid

type: string

Create the given PID file. The tool won't start if the PID file already exists and the PID it contains is different than the current PID. However, if the PID file exists and the PID it contains is no longer running, the tool will overwrite the PID file with the current PID. The PID file is removed automatically when the tool exits.

--port

short form: -P; type: int

Port number to use for connection.

--preserve-embedded-numbers

Preserve numbers in database/table names when fingerprinting queries. The standar fingeprint method replaces numbers in db/tables names, making a query like 'SELECT * FROM db1.table2' to be figerprinted as 'SELECT * FROM db?.table?'. This option changes that behaviour and the fingerprint will become 'SELECT * FROM db1.table2'.

--processlist

type: DSN

Poll this DSN's processlist for queries, with *--interval* sleep between.

If the connection fails, **mariadb-query-digest** tries to reopen it once per second.

--progress

type: array; default: time,30

Print progress reports to STDERR. The value is a comma-separated list with two parts. The first part can be percentage, time, or iterations; the second part specifies how often an update should be printed, in percentage, seconds, or number of iterations.

--read-timeout

type: time; default: 0

Wait this long for an event from the input; 0 to wait forever.

This option sets the maximum time to wait for an event from the input. It applies to all types of input except `--processlist`. If an event is not received after the specified time, the script stops reading the input and prints its reports. If `--iterations` is 0 or greater than 1, the next iteration will begin, else the script will exit.

This option requires the Perl POSIX module.

`--[no]report`
default: yes

Print query analysis reports for each `--group-by` attribute. This is the standard slow log analysis functionality. See "OUTPUT" for the description of what this does and what the results look like.

If you don't need a report (for example, when using `--review` or `--history`), it is best to specify `--no-report` because this allows the tool to skip some expensive operations.

`--report-all`
Report all queries, even ones that have been reviewed. This only affects the `report` `--output` when using `--review`. Otherwise, all queries are always printed.

`--report-format`
type: Array; default: rusage,date,hostname,files,header,profile,query_report,prepared

Print these sections of the query analysis report.

```
SECTION      PRINTS
============ ======================================================
rusage       CPU times and memory usage reported by ps
date         Current local date and time
hostname     Hostname of machine on which :program:`mariadb-query-digest` was run
files        Input files read/parse
header       Summary of the entire analysis run
profile      Compact table of queries for an overview of the report
query_report Detailed information about each unique query
prepared     Prepared statements
```

The sections are printed in the order specified. The rusage, date, files and header sections are grouped together if specified together; other sections are separated by blank lines.

See "OUTPUT" for more information on the various parts of the query report.

`--report-histogram`
type: string; default: Query_time

Chart the distribution of this attribute's values.

The distribution chart is limited to time-based attributes, so charting `Rows_examined`, for example, will produce a useless chart. Charts look like:

```
# Query_time distribution
#   1us
#  10us
# 100us
#   1ms
#  10ms  ###########################
# 100ms  ########################################################
#    1s  ########
#  10s+
```

See "OUTPUT" for more information.

`--resume`
type: string

If specified, the tool writes the last file offset, if there is one, to the given filename. When ran again with the same value for this option, the tool reads the last file offset from the file, seeks to that position in the log, and resumes parsing events from that point onward.

--review

type: DSN

Save query classes for later review, and don't report already reviewed classes.

The default table is `mariadb_tools.query_review`. Specify database (D) and table (t) DSN options to override the default. The database and table are automatically created unless `--no-create-review-table` is specified (see *--[no]create-review-table*).

If the table was created manually, it must have at least the following columns. You can add more columns for your own special purposes, but they won't be used by **mariadb-query-digest**.

```
CREATE TABLE IF NOT EXISTS query_review (
   checksum     CHAR(32) NOT NULL PRIMARY KEY,
   fingerprint  TEXT NOT NULL,
   sample       TEXT NOT NULL,
   first_seen   DATETIME,
   last_seen    DATETIME,
   reviewed_by  VARCHAR(20),
   reviewed_on  DATETIME,
   comments     TEXT
)
```

The columns are:

```
COLUMN       MEANING
===========  ====================================================
checksum     A 64-bit checksum of the query fingerprint
fingerprint  The abstracted version of the query; its primary key
sample       The query text of a sample of the class of queries
first_seen   The smallest timestamp of this class of queries
last_seen    The largest timestamp of this class of queries
reviewed_by  Initially NULL; if set, query is skipped thereafter
reviewed_on  Initially NULL; not assigned any special meaning
comments     Initially NULL; not assigned any special meaning
```

Note that the `fingerprint` column is the true primary key for a class of queries. The `checksum` is just a cryptographic hash of this value, which provides a shorter value that is very likely to also be unique.

After parsing and aggregating events, your table should contain a row for each fingerprint. This option depends on `--group-by fingerprint` (which is the default). It will not work otherwise.

--run-time

type: time

How long to run for each *--iterations*. The default is to run forever (you can interrupt with CTRL-C). Because *--iterations* defaults to 1, if you only specify *--run-time*, **mariadb-query-digest** runs for that amount of time and then exits. The two options are specified together to do collect-and-report cycles. For example, specifying *--iterations* `4` *--run-time* `15m` with a continuous input (like STDIN or *--processlist*) will cause **mariadb-query-digest** to run for 1 hour (15 minutes x 4), reporting four times, once at each 15 minute interval.

--run-time-mode

type: string; default: clock

Set what the value of *--run-time* operates on. Following are the possible values for this option:

clock

--run-time specifies an amount of real clock time during which the tool should run for each *--iterations*.

event

--run-time specifies an amount of log time. Log time is determined by timestamps in the log. The first timestamp seen is remembered, and each timestamp after that is compared to the first to determine how much log time has passed. For example, if the first timestamp seen is `12:00:00` and the next is `12:01:30`, that is 1 minute and 30 seconds of log time. The tool will read events until the log time is greater than or equal to the specified *--run-time* value.

Since timestamps in logs are not always printed, or not always printed frequently, this mode varies in accuracy.

interval

--run-time specifies interval boundaries of log time into which events are divided and reports are generated. This mode is different from the others because it doesn't specify how long to run. The value of *--run-time* must be an interval that divides evenly into minutes, hours or days. For example, `5m` divides evenly into hours (60/5=12, so 12 5 minutes intervals per hour) but `7m` does not (60/7=8.6).

Specifying `--run-time-mode interval --run-time 30m --iterations 0` is similar to specifying `--run-time-mode clock --run-time 30m --iterations 0`. In the latter case, **`mariadb-query-digest`** will run forever, producing reports every 30 minutes, but this only works effectively with continuous inputs like STDIN and the processlist. For fixed inputs, like log files, the former example produces multiple reports by dividing the log into 30 minutes intervals based on timestamps.

Intervals are calculated from the zeroth second/minute/hour in which a timestamp occurs, not from whatever time it specifies. For example, with 30 minute intervals and a timestamp of `12:10:30`, the interval is *not* `12:10:30` to `12:40:30`, it is `12:00:00` to `12:29:59`. Or, with 1 hour intervals, it is `12:00:00` to `12:59:59`. When a new timestamp exceeds the interval, a report is printed, and the next interval is recalculated based on the new timestamp.

Since *--iterations* is 1 by default, you probably want to specify a new value else **`mariadb-query-digest`** will only get and report on the first interval from the log since 1 interval = 1 iteration. If you want to get and report every interval in a log, specify *--iterations* `0`.

`--sample`
type: int

Filter out all but the first N occurrences of each query. The queries are filtered on the first value in *--group-by*, so by default, this will filter by query fingerprint. For example, `--sample 2` will permit two sample queries for each fingerprint. Useful in conjunction with `--output slowlog` to print the queries. You probably want to set `--no-report` to avoid the overhead of aggregating and reporting if you're just using this to print out samples of queries. A complete example:

```
:program:`mariadb-query-digest` --sample 2 --no-report --output slowlog slow.log
```

`--slave-user`
type: string

Sets the user to be used to connect to the slaves. This parameter allows you to have a different user with less privileges on the slaves but that user must exist on all slaves.

`--slave-password`
type: string

Sets the password to be used to connect to the slaves. It can be used with –slave-user and the password for the user must be the same on all slaves.

--set-vars

type: Array

Set the MariaDB variables in this comma-separated list of `variable=value` pairs.

By default, the tool sets:

```
wait_timeout=10000
```

Variables specified on the command line override these defaults. For example, specifying `--set-vars wait_timeout=500` overrides the defaultvalue of `10000`.

The tool prints a warning and continues if a variable cannot be set.

--show-all

type: Hash

Show all values for these attributes.

By default **`mariadb-query-digest`** only shows as many of an attribute's value that fit on a single line. This option allows you to specify attributes for which all values will be shown (line width is ignored). This only works for attributes with string values like user, host, db, etc. Multiple attributes can be specified, comma-separated.

--since

type: string

Parse only queries newer than this value (parse queries since this date).

This option allows you to ignore queries older than a certain value and parse only those queries which are more recent than the value. The value can be several types:

```
* Simple time value N with optional suffix: N[shmd], where
  s=seconds, h=hours, m=minutes, d=days (default s if no suffix
  given); this is like saying "since N[shmd] ago"
* Full date with optional hours:minutes:seconds:
  YYYY-MM-DD [HH:MM:SS]
* Short, MariaDB-style date:
  YYMMDD [HH:MM:SS]
* Any time expression evaluated by MariaDB:
  CURRENT_DATE - INTERVAL 7 DAY
```

If you give a MariaDB time expression, and you have not also specified a DSN for *`--explain`*, *`--processlist`*, or *`--review`*, then you must specify a DSN on the command line so that **`mariadb-query-digest`** can connect to MariaDB to evaluate the expression.

The MariaDB time expression is wrapped inside a query like "SELECT UNIX_TIMESTAMP(<expression>)", so be sure that the expression is valid inside this query. For example, do not use UNIX_TIMESTAMP() because UNIX_TIMESTAMP(UNIX_TIMESTAMP()) returns 0.

Events are assumed to be in chronological: older events at the beginning of the log and newer events at the end of the log. *`--since`* is strict: it ignores all queries until one is found that is new enough. Therefore, if the query events are not consistently timestamped, some may be ignored which are actually new enough.

See also *`--until`*.

--socket

short form: -S; type: string

Socket file to use for connection.

--timeline

Show a timeline of events.

This option makes **mariadb-query-digest** print another kind of report: a timeline of the events. Each query is still grouped and aggregate into classes according to *--group-by*, but then they are printed in chronological order. The timeline report prints out the timestamp, interval, count and value of each classes.

If all you want is the timeline report, then specify `--no-report` to suppress the default query analysis report. Otherwise, the timeline report will be printed at the end before the response-time profile (see *--report-format* and "OUTPUT").

For example, this:

```
:program:`mariadb-query-digest` /path/to/log --group-by distill --timeline
```

will print something like:

```
# ########################################################
# distill report
# ########################################################
# 2009-07-25 11:19:27 1+00:00:01   2 SELECT foo
# 2009-07-27 11:19:30      00:01   2 SELECT bar
# 2009-07-27 11:30:00 1+06:30:00   2 SELECT foo
```

--type

type: Array; default: slowlog

The type of input to parse. The permitted types are

binlog

Parse a binary log file that has first been converted to text using mariadb-binlog.

For example:

```
mariadb-binlog mariadb-bin.000441 > mariadb-bin.000441.txt

:program:`mariadb-query-digest` --type binlog mariadb-bin.000441.txt
```

genlog

Parse a MariaDB general log file. General logs lack a lot of "ATTRIBUTES", notably `Query_time`. The default *--order-by* for general logs changes to `Query_time:cnt`.

slowlog

Parse a log file in any variation of MariaDB slow log format.

tcpdump

Inspect network packets and decode the MariaDB client protocol, extracting queries and responses from it.

mariadb-query-digest does not actually watch the network (i.e. it does NOT "sniff packets"). Instead, it's just parsing the output of tcpdump. You are responsible for generating this output; **mariadb-query-digest** does not do it for you. Then you send this to **mariadb-query-digest** as you would any log file: as files on the command line or to STDIN.

The parser expects the input to be formatted with the following options: `-x -n -q -tttt`. For example, if you want to capture output from your local machine, you can do something like the following (the port must come last on FreeBSD):

```
tcpdump -s 65535 -x -nn -q -tttt -i any -c 1000 port 3306 \
  > mariadb.tcp.txt
:program:`mariadb-query-digest` --type tcpdump mariadb.tcp.txt
```

The other tcpdump parameters, such as -s, -c, and -i, are up to you. Just make sure the output looks like this (there is a line break in the first line to avoid man-page problems):

```
2009-04-12 09:50:16.804849 IP 127.0.0.1.42167
       > 127.0.0.1.3306: tcp 37
    0x0000:  4508 0059 6eb2 4000 4006 cde2 7f00 0001
    0x0010:  ....
```

Remember tcpdump has a handy -c option to stop after it captures some number of packets! That's very useful for testing your tcpdump command. Note that tcpdump can't capture traffic on a Unix socket. Read http://bugs.mysql.com/bug.php?id=31577 if you're confused about this.

Devananda Van Der Veen explained on the MySQL Performance Blog how to capture traffic without dropping packets on busy servers. Dropped packets cause **mariadb-query-digest** to miss the response to a request, then see the response to a later request and assign the wrong execution time to the query. You can change the filter to something like the following to help capture a subset of the queries. (See http://www.mysqlperformanceblog.com/?p=6092 for details.)

```
tcpdump -i any -s 65535 -x -n -q -tttt \
   'port 3306 and tcp[1] & 7 == 2 and tcp[3] & 7 == 2'
```

All MariaDB servers running on port 3306 are automatically detected in the tcpdump output. Therefore, if the tcpdump out contains packets from multiple servers on port 3306 (for example, 10.0.0.1:3306, 10.0.0.2:3306, etc.), all packets/queries from all these servers will be analyzed together as if they were one server.

If you're analyzing traffic for a MariaDB server that is not running on port 3306, see *--watch-server*.

Also note that **mariadb-query-digest** may fail to report the database for queries when parsing tcpdump output. The database is discovered only in the initial connect events for a new client or when <USE db> is executed. If the tcpdump output contains neither of these, then **mariadb-query-digest** cannot discover the database.

Server-side prepared statements are supported. SSL-encrypted traffic cannot be inspected and decoded.

rawlog

Raw logs are not MariaDB logs but simple text files with one SQL statement per line, like:

```
SELECT c FROM t WHERE id=1
/* Hello, world! */ SELECT * FROM t2 LIMIT 1
INSERT INTO t (a, b) VALUES ('foo', 'bar')
INSERT INTO t SELECT * FROM monkeys
```

Since raw logs do not have any metrics, many options and features of **mariadb-query-digest** do not work with them.

One use case for raw logs is ranking queries by count when the only information available is a list of queries, from polling `SHOW PROCESSLIST` for example.

--until

type: string

Parse only queries older than this value (parse queries until this date).

This option allows you to ignore queries newer than a certain value and parse only those queries which are older than the value. The value can be one of the same types listed for `--since`.

Unlike `--since`, `--until` is not strict: all queries are parsed until one has a timestamp that is equal to or greater than `--until`. Then all subsequent queries are ignored.

`--user`
short form: -u; type: string

User for login if not current user.

`--variations`
type: Array

Report the number of variations in these attributes' values.

Variations show how many distinct values an attribute had within a class. The usual value for this option is `arg` which shows how many distinct queries were in the class. This can be useful to determine a query's cacheability.

Distinct values are determined by CRC32 checksums of the attributes' values. These checksums are reported in the query report for attributes specified by this option, like:

```
# arg crc      109 (1/25%), 144 (1/25%)... 2 more
```

In that class there were 4 distinct queries. The checksums of the first two variations are shown, and each one occurred once (or, 25% of the time).

The counts of distinct variations is approximate because only 1,000 variations are saved. The mod (%) 1000 of the full CRC32 checksum is saved, so some distinct checksums are treated as equal.

`--version`
Show version and exit.

`--[no]vertical-format`
default: yes

Output a trailing "G" in the reported SQL queries.

This makes the mariadb client display the result using vertical format. Non-native MariaDB clients like phpMyAdmin do not support this.

`--watch-server`
type: string

This option tells **`mariadb-query-digest`** which server IP address and port (like "10.0.0.1:3306") to watch when parsing tcpdump (for `--type` tcpdump); all other servers are ignored. If you don't specify it, **`mariadb-query-digest`** watches all servers by looking for any IP address using port 3306 or "mariadb". If you're watching a server with a non-standard port, this won't work, so you must specify the IP address and port to watch.

If you want to watch a mix of servers, some running on standard port 3306 and some running on non-standard ports, you need to create separate tcpdump outputs for the non-standard port servers and then specify this option for each. At present **`mariadb-query-digest`** cannot auto-detect servers on port 3306 and also be told to watch a server on a non-standard port.

10.10 DSN OPTIONS

These DSN options are used to create a DSN. Each option is given like `option=value`. The options are case-sensitive, so P and p are not the same option. There cannot be whitespace before or after the = and if the value contains whitespace it must be quoted. DSN options are comma-separated. See the mariadb-tools manpage for full details.

- A

  dsn: charset; copy: yes

  Default character set.

- D

  dsn: database; copy: yes

  Default database to use when connecting to MariaDB.

- F

  dsn: mysql_read_default_file; copy: yes

  Only read default options from the given file.

- h

  dsn: host; copy: yes

  Connect to host.

- p

  dsn: password; copy: yes

  Password to use when connecting. If password contains commas they must be escaped with a backslash: "exam,ple"

- P

  dsn: port; copy: yes

  Port number to use for connection.

- S

  dsn: mysql_socket; copy: yes

  Socket file to use for connection.

- t

  The *`--review`* or *`--history`* table.

- u

  dsn: user; copy: yes

  User for login if not current user.

10.11 ENVIRONMENT

The environment variable `PTDEBUG` enables verbose debugging output to STDERR. To enable debugging and capture all output to a file, run the tool like:

```
PTDEBUG=1 mariadb-query-digest ... > FILE 2>&1
```

Be careful: debugging output is voluminous and can generate several megabytes of output.

10.12 SYSTEM REQUIREMENTS

You need Perl, DBI, DBD::mysql, and some core packages that ought to be installed in any reasonably new version of Perl.

10.13 ATTRIBUTES REFERENCE

Events may have the following attributes. If writing a *--filter*, be sure to check that an attribute is defined in each event before using it, else the filter code may crash the tool with a "use of uninitialized value" error.

You can dump event attributes for any input like:

```
$ mariadb-query-digest                   \
    slow.log                        \
    --filter 'print Dumper $event' \
    --no-report                     \
    --sample 1
```

That will produce a lot of output with "attribute => value" pairs like:

```
$VAR1 = {
  Query_time => '0.033384',
  Rows_examined => '0',
  Rows_sent => '0',
  Thread_id => '10',
  Tmp_table => 'No',
  Tmp_table_on_disk => 'No',
  arg => 'SELECT col FROM tbl WHERE id=5',
  bytes => 103,
  cmd => 'Query',
  db => 'db1',
  fingerprint => 'select col from tbl where id=?',
  host => '',
  pos_in_log => 1334,
  ts => '071218 11:48:27',
  user => '[SQL_SLAVE]'
};
```

10.14 COMMON

These attribute are common to all input *--type* and *--processlist*, except where noted.

arg

> The query text, or the command for admin commands like `Ping`.

bytes

> The byte length of the `arg`.

cmd

> "Query" or "Admin".

db

> The current database. The value comes from USE database statements. By default, `Schema` is an alias which is automatically changed to `db`; see *--attribute-aliases*.

fingerprint

> An abstracted form of the query. See "FINGERPRINTS".

host

> Client host which executed the query.

pos_in_log

> The byte offset of the event in the log or tcpdump, except for *--processlist*.

Query_time

> The total time the query took, including lock time.

ts

> The timestamp of when the query ended.

10.15 SLOW, GENERAL, AND BINARY LOGS

Events have all available attributes from the log file. Therefore, you only need to look at the log file to see which events are available, but remember: not all events have the same attributes.

10.16 TCPDUMP

These attributes are available when parsing *--type* tcpdump.

Error_no

> The MariaDB error number if the query caused an error.

ip

> The client's IP address. Certain log files may also contain this attribute.

No_good_index_used

> Yes or No if no good index existed for the query (flag set by server).

No_index_used

Yes or No if the query did not use any index (flag set by server).

port

The client's port number.

Warning_count

The number of warnings, as otherwise shown by `SHOW WARNINGS`.

10.17 PROCESSLIST

If using *--processlist*, an `id` attribute is available for the process ID, in addition to the common attributes.

10.18 AUTHORS

Cole Busby, Baron Schwartz, Daniel Nichter, and Brian Fraser

10.19 ABOUT THIS MARIADB TOOL

This tool is part of MariaDB client tools. This MariaDB Tool was forked from Percona Toolkit's pt-query-digest in November, 2019. Percona Toolkit was forked from two projects in June, 2011: Maatkit and Aspersa. Those projects were created by Baron Schwartz and primarily developed by him and Daniel Nichter.

10.20 COPYRIGHT, LICENSE, AND WARRANTY

10.21 VERSION

`mariadb-query-digest` 6.0.0rc

CHAPTER ELEVEN

MARIADB-SCHEMA-CHANGE

11.1 NAME

mariadb-schema-change - ALTER tables without locking them.

11.2 SYNOPSIS

11.2.1 Usage

```
mariadb-schema-change [OPTIONS] DSN
```

mariadb-schema-change alters a table's structure without blocking reads or writes. Specify the database and table in the DSN. Do not use this tool before reading its documentation and checking your backups carefully.

Add a column to sakila.actor:

```
mariadb-schema-change --alter "ADD COLUMN c1 INT" D=sakila,t=actor
```

Change sakila.actor to InnoDB, effectively performing OPTIMIZE TABLE in a non-blocking fashion because it is already an InnoDB table:

```
mariadb-schema-change --alter "ENGINE=InnoDB" D=sakila,t=actor
```

11.3 RISKS

mariadb-schema-change is mature, proven in the real world, and well tested, but all database tools can pose a risk to the system and the database server. Before using this tool, please:

- Read the tool's documentation
- Review the tool's known "BUGS"
- Test the tool on a non-production server
- Backup your production server and verify the backups

11.4 DESCRIPTION

mariadb-schema-change emulates the way that MariaDB alters tables internally, but it works on a copy of the table you wish to alter. This means that the original table is not locked, and clients may continue to read and change data in it.

mariadb-schema-change works by creating an empty copy of the table to alter, modifying it as desired, and then copying rows from the original table into the new table. When the copy is complete, it moves away the original table and replaces it with the new one. By default, it also drops the original table.

The data copy process is performed in small chunks of data, which are varied to attempt to make them execute in a specific amount of time (see *--chunk-time*). This process is very similar to how other tools, such as pt-table-checksum, work. Any modifications to data in the original tables during the copy will be reflected in the new table, because the tool creates triggers on the original table to update the corresponding rows in the new table. The use of triggers means that the tool will not work if any triggers are already defined on the table.

When the tool finishes copying data into the new table, it uses an atomic `RENAME TABLE` operation to simultaneously rename the original and new tables. After this is complete, the tool drops the original table.

Foreign keys complicate the tool's operation and introduce additional risk. The technique of atomically renaming the original and new tables does not work when foreign keys refer to the table. The tool must update foreign keys to refer to the new table after the schema change is complete. The tool supports two methods for accomplishing this. You can read more about this in the documentation for *--alter-foreign-keys-method*.

Foreign keys also cause some side effects. The final table will have the same foreign keys and indexes as the original table (unless you specify differently in your ALTER statement), but the names of the objects may be changed slightly to avoid object name collisions in MariaDB and InnoDB.

For safety, the tool does not modify the table unless you specify the *--execute* option, which is not enabled by default. The tool supports a variety of other measures to prevent unwanted load or other problems, including automatically detecting replicas, connecting to them, and using the following safety checks:

- In most cases the tool will refuse to operate unless a PRIMARY KEY or UNIQUE INDEX is present in the table. See *--alter* for details.
- The tool refuses to operate if it detects replication filters. See *--[no]check-replication-filters* for details.
- The tool pauses the data copy operation if it observes any replicas that are delayed in replication. See *--max-lag* for details.
- The tool pauses or aborts its operation if it detects too much load on the server. See *--max-load* and *--critical-load* for details.
- The tool sets `innodb_lock_wait_timeout=1` and (for MariaDB 5.5 and newer) `lock_wait_timeout=60` so that it is more likely to be the victim of any lock contention, and less likely to disrupt other transactions. These values can be changed by specifying *--set-vars*.
- The tool refuses to alter the table if foreign key constraints reference it, unless you specify *--alter-foreign-keys-method*.
- The tool cannot alter MyISAM tables on "Galera" nodes.

11.5 MariaDB Galera Cluster

mariadb-schema-change works with Galera Cluster 5.5.28 and newer, but there are two limitations: only InnoDB tables can be altered, and `wsrep_OSU_method` must be set to `TOI` (total order isolation). The tool exits with an error if the host is a cluster node and the table is MyISAM or is being converted to MyISAM (`ENGINE=MyISAM`), or if `wsrep_OSU_method` is not `TOI`. There is no way to disable these checks.

The tools ignores MariaDB 10.2+ `GENERATED` columns since the value for those columns is generated according to the expresion used to compute column values.

11.6 OUTPUT

The tool prints information about its activities to STDOUT so that you can see what it is doing. During the data copy phase, it prints *--progress* reports to STDERR. You can get additional information by specifying *--print*.

If *--statistics* is specified, a report of various internal event counts is printed at the end, like:

```
# Event  Count
# ====== =====
# INSERT     1
```

11.7 OPTIONS

--dry-run and *--execute* are mutually exclusive.

This tool accepts additional command-line arguments. Refer to the "SYNOPSIS" and usage information for details.

--alter

type: string

The schema modification, without the ALTER TABLE keywords. You can perform multiple modifications to the table by specifying them with commas. Please refer to the MariaDB manual for the syntax of ALTER TABLE.

The following limitations apply which, if attempted, will cause the tool to fail in unpredictable ways:

- In almost all cases a PRIMARY KEY or UNIQUE INDEX needs to be present in the table. This is necessary because the tool creates a DELETE trigger to keep the new table updated while the process is running.

  A notable exception is when a PRIMARY KEY or UNIQUE INDEX is being created from **existing columns** as part of the ALTER clause; in that case it will use these column(s) for the DELETE trigger.

- The `RENAME` clause cannot be used to rename the table.

- Columns cannot be renamed by dropping and re-adding with the new name. The tool will not copy the original column's data to the new column.

- If you add a column without a default value and make it NOT NULL, the tool will fail, as it will not try to guess a default value for you; You must specify the default.

- `DROP FOREIGN KEY constraint_name` requires specifying `_constraint_name` rather than the real `constraint_name`. Due to a limitation in MariaDB, **mariadb-schema-change** adds a leading underscore to foreign key constraint names when creating the new table. For example, to drop this constraint:

```
CONSTRAINT `fk_foo` FOREIGN KEY (`foo_id`) REFERENCES `bar` (`foo_id`)
```

You must specify `--alter "DROP FOREIGN KEY _fk_foo"`.

- The tool does not use `LOCK IN SHARE MODE` with MariaDB 5.0 because it can cause a slave error which breaks replication:

```
Query caused different errors on master and slave. Error on master:
'Deadlock found when trying to get lock; try restarting transaction' (1213),
Error on slave: 'no error' (0). Default database: 'pt_osc'.
Query: 'INSERT INTO pt_osc.t (id, c) VALUES ('730', 'new row')'
```

The error happens when converting a MyISAM table to InnoDB because MyISAM is non-transactional but InnoDB is transactional. MariaDB 5.1 and newer handle this case correctly, but testing reproduces the error 5% of the time with MariaDB 5.0.

This is a MariaDB bug, similar to http://bugs.mysql.com/bug.php?id=45694, but there is no fix or workaround in MariaDB 5.0. Without `LOCK IN SHARE MODE`, tests pass 100% of the time, so the risk of data loss or breaking replication should be negligible.

Be sure to verify the new table if using MariaDB 5.0 and converting from MyISAM to InnoDB!

`--alter-foreign-keys-method`

type: string

How to modify foreign keys so they reference the new table. Foreign keys that reference the table to be altered must be treated specially to ensure that they continue to reference the correct table. When the tool renames the original table to let the new one take its place, the foreign keys “follow” the renamed table, and must be changed to reference the new table instead.

The tool supports two techniques to achieve this. It automatically finds “child tables” that reference the table to be altered.

auto

Automatically determine which method is best. The tool uses `rebuild_constraints` if possible (see the description of that method for details), and if not, then it uses `drop_swap`.

rebuild_constraints

This method uses `ALTER TABLE` to drop and re-add foreign key constraints that reference the new table. This is the preferred technique, unless one or more of the “child” tables is so large that the `ALTER` would take too long. The tool determines that by comparing the number of rows in the child table to the rate at which the tool is able to copy rows from the old table to the new table. If the tool estimates that the child table can be altered in less time than the *`--chunk-time`*, then it will use this technique. For purposes of estimating the time required to alter the child table, the tool multiplies the row-copying rate by *`--chunk-size-limit`*, because MariaDB’s `ALTER TABLE` is typically much faster than the external process of copying rows.

Due to a limitation in MariaDB, foreign keys will not have the same names after the ALTER that they did prior to it. The tool has to rename the foreign key when it redefines it, which adds a leading underscore to the name. In some cases, MariaDB also automatically renames indexes required for the foreign key.

drop_swap

Disable foreign key checks (FOREIGN_KEY_CHECKS=0), then drop the original table before renaming the new table into its place. This is different from the normal method of swapping the old and new table, which uses an atomic `RENAME` that is undetectable to client applications.

This method is faster and does not block, but it is riskier for two reasons. First, for a short time between dropping the original table and renaming the temporary table, the table to be altered simply does not exist, and queries against it will result in an error. Secondly, if there is an error and the new table cannot be renamed into the place of the old one, then it is too late to abort, because the old table is gone permanently.

This method forces `--no-swap-tables` and `--no-drop-old-table`.

none

This method is like `drop_swap` without the "swap". Any foreign keys that referenced the original table will now reference a nonexistent table. This will typically cause foreign key violations that are visible in `SHOW ENGINE INNODB STATUS`, similar to the following:

```
Trying to add to index `idx_fk_staff_id` tuple:
DATA TUPLE: 2 fields;
0: len 1; hex 05; asc  ;;
1: len 4; hex 80000001; asc     ;;
But the parent table `sakila`.`staff_old`
or its .ibd file does not currently exist!
```

This is because the original table (in this case, sakila.staff) was renamed to sakila.staff_old and then dropped. This method of handling foreign key constraints is provided so that the database administrator can disable the tool's built-in functionality if desired.

`--[no]analyze-before-swap`
default: yes

Execute ANALYZE TABLE on the new table before swapping with the old one. By default, this happens only when running MariaDB 5.6 and newer, and `innodb_stats_persistent` is enabled. Specify the option explicitly to enable or disable it regardless of MariaDB version and `innodb_stats_persistent`.

This circumvents a potentially serious issue related to InnoDB optimizer statistics. If the table being alerted is busy and the tool completes quickly, the new table will not have optimizer statistics after being swapped. This can cause fast, index-using queries to do full table scans until optimizer statistics are updated (usually after 10 seconds). If the table is large and the server very busy, this can cause an outage.

`--ask-pass`
Prompt for a password when connecting to MariaDB.

`--channel`
type: string

Channel name used when connected to a server using replication channels. Suppose you have two masters, master_a at port 12345, master_b at port 1236 and a slave connected to both masters using channels chan_master_a and chan_master_b. If you want to run pt-table-sync to synchronize the slave against master_a, pt-table-sync won't be able to determine what's the correct master since SHOW SLAVE STATUS will return 2 rows. In this case, you can use –channel=chan_master_a to specify the channel name to use in the SHOW SLAVE STATUS command.

`--charset`
short form: -A; type: string

Default character set. If the value is utf8, sets Perl's binmode on STDOUT to utf8, passes the mysql_enable_utf8 option to DBD::mysql, and runs SET NAMES UTF8 after connecting to MariaDB. Any other value sets binmode on STDOUT without the utf8 layer, and runs SET NAMES after connecting to MariaDB.

`--[no]check-alter`
default: yes

Parses the `--alter` specified and tries to warn of possible unintended behavior. Currently, it checks for:

Column renames

In previous versions of the tool, renaming a column with `CHANGE COLUMN name new_name` would lead to that column's data being lost. The tool now parses the alter statement and tries to catch these cases, so the renamed columns should have the same data as the originals. However, the code that does this is not a full-blown SQL parser, so you should first run the tool with *`--dry-run`* and *`--print`* and verify that it detects the renamed columns correctly.

DROP PRIMARY KEY

If *`--alter`* contain `DROP PRIMARY KEY` (case- and space-insensitive), a warning is printed and the tool exits unless *`--dry-run`* is specified. Altering the primary key can be dangerous, but the tool can handle it. The tool's triggers, particularly the DELETE trigger, are most affected by altering the primary key because the tool prefers to use the primary key for its triggers. You should first run the tool with *`--dry-run`* and *`--print`* and verify that the triggers are correct.

`--[no]check-foreign-keys`
default: yes

Check for self-referencing foreign keys. Currently self referencing FKs are not full supported, so, to prevent errors, this program won't run if the table has self-referencing foreign keys. Use this parameter to disable self-referencing FK checks.

`--check-interval`
type: time; default: 1

Sleep time between checks for *`--max-lag`*.

`--[no]check-plan`
default: yes

Check query execution plans for safety. By default, this option causes the tool to run EXPLAIN before running queries that are meant to access a small amount of data, but which could access many rows if MariaDB chooses a bad execution plan. These include the queries to determine chunk boundaries and the chunk queries themselves. If it appears that MariaDB will use a bad query execution plan, the tool will skip the chunk of the table.

The tool uses several heuristics to determine whether an execution plan is bad. The first is whether EXPLAIN reports that MariaDB intends to use the desired index to access the rows. If MariaDB chooses a different index, the tool considers the query unsafe.

The tool also checks how much of the index MariaDB reports that it will use for the query. The EXPLAIN output shows this in the key_len column. The tool remembers the largest key_len seen, and skips chunks where MariaDB reports that it will use a smaller prefix of the index. This heuristic can be understood as skipping chunks that have a worse execution plan than other chunks.

The tool prints a warning the first time a chunk is skipped due to a bad execution plan in each table. Subsequent chunks are skipped silently, although you can see the count of skipped chunks in the SKIPPED column in the tool's output.

This option adds some setup work to each table and chunk. Although the work is not intrusive for MariaDB, it results in more round-trips to the server, which consumes time. Making chunks too small will cause the overhead to become relatively larger. It is therefore recommended that you not make chunks too small, because the tool may take a very long time to complete if you do.

`--[no]check-replication-filters`
default: yes

Abort if any replication filter is set on any server. The tool looks for server options that filter replication, such as binlog_ignore_db and replicate_do_db. If it finds any such filters, it aborts with an error.

If the replicas are configured with any filtering options, you should be careful not to modify any databases or tables that exist on the master and not the replicas, because it could cause replication to fail. For more information on replication rules, see http://dev.mysql.com/doc/en/replication-rules.html.

--check-slave-lag
type: string

Pause the data copy until this replica's lag is less than *--max-lag*. The value is a DSN that inherits properties from the the connection options (*--port*, *--user*, etc.). This option overrides the normal behavior of finding and continually monitoring replication lag on ALL connected replicas. If you don't want to monitor ALL replicas, but you want more than just one replica to be monitored, then use the DSN option to the *--recursion-method* option instead of this option.

--chunk-index
type: string

Prefer this index for chunking tables. By default, the tool chooses the most appropriate index for chunking. This option lets you specify the index that you prefer. If the index doesn't exist, then the tool will fall back to its default behavior of choosing an index. The tool adds the index to the SQL statements in a `FORCE INDEX` clause. Be careful when using this option; a poor choice of index could cause bad performance.

--chunk-index-columns
type: int

Use only this many left-most columns of a *--chunk-index*. This works only for compound indexes, and is useful in cases where a bug in the MariaDB query optimizer (planner) causes it to scan a large range of rows instead of using the index to locate starting and ending points precisely. This problem sometimes occurs on indexes with many columns, such as 4 or more. If this happens, the tool might print a warning related to the *--[no]check-plan* option. Instructing the tool to use only the first N columns of the index is a workaround for the bug in some cases.

--chunk-size
type: size; default: 1000

Number of rows to select for each chunk copied. Allowable suffixes are k, M, G.

This option can override the default behavior, which is to adjust chunk size dynamically to try to make chunks run in exactly *--chunk-time* seconds. When this option isn't set explicitly, its default value is used as a starting point, but after that, the tool ignores this option's value. If you set this option explicitly, however, then it disables the dynamic adjustment behavior and tries to make all chunks exactly the specified number of rows.

There is a subtlety: if the chunk index is not unique, then it's possible that chunks will be larger than desired. For example, if a table is chunked by an index that contains 10,000 of a given value, there is no way to write a WHERE clause that matches only 1,000 of the values, and that chunk will be at least 10,000 rows large. Such a chunk will probably be skipped because of *--chunk-size-limit*.

--chunk-size-limit
type: float; default: 4.0

Do not copy chunks this much larger than the desired chunk size.

When a table has no unique indexes, chunk sizes can be inaccurate. This option specifies a maximum tolerable limit to the inaccuracy. The tool uses <EXPLAIN> to estimate how many rows are in the chunk. If that estimate exceeds the desired chunk size times the limit, then the tool skips the chunk.

The minimum value for this option is 1, which means that no chunk can be larger than *--chunk-size*. You probably don't want to specify 1, because rows reported by EXPLAIN are estimates, which can be different from the real number of rows in the chunk. You can disable oversized chunk checking by specifying a value of 0.

The tool also uses this option to determine how to handle foreign keys that reference the table to be altered. See *--alter-foreign-keys-method* for details.

--chunk-time
type: float; default: 0.5

Adjust the chunk size dynamically so each data-copy query takes this long to execute. The tool tracks the copy rate (rows per second) and adjusts the chunk size after each data-copy query, so that the next query takes this amount of time (in seconds) to execute. It keeps an exponentially decaying moving average of queries per second, so that if the server's performance changes due to changes in server load, the tool adapts quickly.

If this option is set to zero, the chunk size doesn't auto-adjust, so query times will vary, but query chunk sizes will not. Another way to do the same thing is to specify a value for *`--chunk-size`* explicitly, instead of leaving it at the default.

--config
type: Array

Read this comma-separated list of config files; if specified, this must be the first option on the command line.

--critical-load
type: Array; default: Threads_running=50

Examine SHOW GLOBAL STATUS after every chunk, and abort if the load is too high. The option accepts a comma-separated list of MariaDB status variables and thresholds. An optional `=MAX_VALUE` (or `:MAX_VALUE`) can follow each variable. If not given, the tool determines a threshold by examining the current value at startup and doubling it.

See *`--max-load`* for further details. These options work similarly, except that this option will abort the tool's operation instead of pausing it, and the default value is computed differently if you specify no threshold. The reason for this option is as a safety check in case the triggers on the original table add so much load to the server that it causes downtime. There is probably no single value of Threads_running that is wrong for every server, but a default of 50 seems likely to be unacceptably high for most servers, indicating that the operation should be canceled immediately.

--database
short form: -D; type: string

Connect to this database.

--default-engine
Remove `ENGINE` from the new table.

By default the new table is created with the same table options as the original table, so if the original table uses InnoDB, then the new table will use InnoDB. In certain cases involving replication, this may cause unintended changes on replicas which use a different engine for the same table. Specifying this option causes the new table to be created with the system's default engine.

--data-dir
type: string

Create the new table on a different partition using the DATA DIRECTORY feature. Only available on 5.6+. This parameter is ignored if it is used at the same time than remove-data-dir.

--remove-data-dir
default: no

If the original table was created using the DATA DIRECTORY feature, remove it and create the new table in MariaDB default directory without creating a new isl file.

--defaults-file
short form: -F; type: string

Only read mysql options from the given file. You must give an absolute pathname.

--[no]drop-new-table
default: yes

Drop the new table if copying the original table fails.

Specifying `--no-drop-new-table` and `--no-swap-tables` leaves the new, altered copy of the table without modifying the original table. See *`--new-table-name`*.

–no-drop-new-table does not work with `alter-foreign-keys-method drop_swap`.

--[no]drop-old-table
default: yes

Drop the original table after renaming it. After the original table has been successfully renamed to let the new table take its place, and if there are no errors, the tool drops the original table by default. If there are any errors, the tool leaves the original table in place.

If `--no-swap-tables` is specified, then there is no old table to drop.

--[no]drop-triggers
default: yes

Drop triggers on the old table. `--no-drop-triggers` forces `--no-drop-old-table`.

--dry-run
Create and alter the new table, but do not create triggers, copy data, or replace the original table.

--execute
Indicate that you have read the documentation and want to alter the table. You must specify this option to alter the table. If you do not, then the tool will only perform some safety checks and exit. This helps ensure that you have read the documentation and understand how to use this tool. If you have not read the documentation, then do not specify this option.

--[no]check-unique-key-change
default: yes

Avoid **`mariadb-schema-change`** to run if the specified statement for *`--alter`* is trying to add an unique index. Since **`mariadb-schema-change`** uses `INSERT IGNORE` to copy rows to the new table, if the row being written produces a duplicate key, it will fail silently and data will be lost.

Example:

```
CREATE DATABASE test;
USE test;
CREATE TABLE `a` (
  `id` int(11) NOT NULL,
  `unique_id` varchar(32) DEFAULT NULL,
  PRIMARY KEY (`id`)
) ENGINE=InnoDB DEFAULT CHARSET=latin1;

insert into a values (1, "a");
insert into a values (2, "b");
insert into a values (3, "");
insert into a values (4, "");
insert into a values (5, NULL);
insert into a values (6, NULL);
```

Using **`mariadb-schema-change`** to add an unique index on the `unique_id` field, will cause some rows to be lost due to the use of `INSERT IGNORE` to copy rows from the source table. For this reason, **`mariadb-schema-change`** will fail if it detects that the *`--alter`* parameter is trying to add an unique key and it will show an example query to run to detect if there are rows that will produce duplicated indexes.

Even if you run the query and there are no rows that will produce duplicated indexes, take into consideration that after running this query, changes can be made to the table that can produce duplicate rows and this data will be lost.

`--force`

This options bypasses confirmation in case of using alter-foreign-keys-method = none , which might break foreign key constraints.

`--force-concat-enums`

The NibbleIterator in **`mariadb-schema-change`** can detect indexes having ENUM fields and if the items it has are sorted or not. According to documentation at https://dev.mysql.com/doc/refman/5.7/en/enum.html:

ENUM values are sorted based on their index numbers, which depend on the order in which the enumeration members were listed in the column specification. For example, 'b' sorts before 'a' for ENUM('b', 'a'). The empty string sorts before nonempty strings, and NULL values sort before all other enumeration values.

To prevent unexpected results when using the ORDER BY clause on an ENUM column, use one of these techniques: - Specify the ENUM list in alphabetic order. - Make sure that the column is sorted lexically rather than by index number by coding ORDER BY CAST(col AS CHAR) or ORDER BY CONCAT(col).

The NibbleIterator in **`mariadb-schema-change`** uses CONCAT(col) but, doing that, adds overhead since MariaDB cannot use the column directly and has to calculate the result of CONCAT for every row. To make this scenario vissible to the user, if there are indexes having ENUM fields with usorted items, it is necessary to specify the `--force-concat-enums` parameter.

`--help`

Show help and exit.

`--host`

short form: -h; type: string

Connect to host.

`--max-flow-ctl`

type: float

Somewhat similar to –max-lag but for PXC clusters. Check average time cluster spent pausing for Flow Control and make tool pause if it goes over the percentage indicated in the option. A value of 0 would make the tool pause when *any* Flow Control activity is detected. Default is no Flow Control checking. This option is available for PXC versions 5.6 or higher.

`--max-lag`

type: time; default: 1s

Pause the data copy until all replicas' lag is less than this value. After each data-copy query (each chunk), the tool looks at the replication lag of all replicas to which it connects, using Seconds_Behind_Master. If any replica is lagging more than the value of this option, then the tool will sleep for *`--check-interval`* seconds, then check all replicas again. If you specify *`--check-slave-lag`*, then the tool only examines that server for lag, not all servers. If you want to control exactly which servers the tool monitors, use the DSN value to *`--recursion-method`*.

The tool waits forever for replicas to stop lagging. If any replica is stopped, the tool waits forever until the replica is started. The data copy continues when all replicas are running and not lagging too much.

The tool prints progress reports while waiting. If a replica is stopped, it prints a progress report immediately, then again at every progress report interval.

`--max-load`

type: Array; default: Threads_running=25

Examine SHOW GLOBAL STATUS after every chunk, and pause if any status variables are higher than their thresholds. The option accepts a comma-separated list of MariaDB status variables. An optional `=MAX_VALUE` (or :MAX_VALUE) can follow each variable. If not given, the tool determines a threshold by examining the current value and increasing it by 20%.

For example, if you want the tool to pause when Threads_connected gets too high, you can specify "Threads_connected", and the tool will check the current value when it starts working and add 20% to that value. If the current value is 100, then the tool will pause when Threads_connected exceeds 120, and resume working when it is below 120 again. If you want to specify an explicit threshold, such as 110, you can use either "Threads_connected:110" or "Threads_connected=110".

The purpose of this option is to prevent the tool from adding too much load to the server. If the data-copy queries are intrusive, or if they cause lock waits, then other queries on the server will tend to block and queue. This will typically cause Threads_running to increase, and the tool can detect that by running SHOW GLOBAL STATUS immediately after each query finishes. If you specify a threshold for this variable, then you can instruct the tool to wait until queries are running normally again. This will not prevent queueing, however; it will only give the server a chance to recover from the queueing. If you notice queueing, it is best to decrease the chunk time.

--preserve-triggers

Preserves old triggers when specified. As of MariaDB 10.2.3, it is possible to define multiple triggers for a given table that have the same trigger event and action time. This allows us to add the triggers needed for **mariadb-schema-change** even if the table already has its own triggers. If this option is enabled, **mariadb-schema-change** will try to copy all the existing triggers to the new table BEFORE start copying rows from the original table to ensure the old triggers can be applied after altering the table.

Example.

```
CREATE TABLE test.t1 (
     id INT NOT NULL AUTO_INCREMENT,
     f1 INT,
     f2 VARCHAR(32),
     PRIMARY KEY (id)
);

CREATE TABLE test.log (
   ts  TIMESTAMP,
   msg VARCHAR(255)
);

CREATE TRIGGER test.after_update
 AFTER
   UPDATE ON test.t1
   FOR EACH ROW
     INSERT INTO test.log VALUES (NOW(), CONCAT("updated row row with id ", OLD.
↪id, " old f1:", OLD.f1, " new f1: ", NEW.f1 ));
```

For this table and triggers combination, it is not possible to use –preserve-triggers with an –alter like this: `"DROP COLUMN f1"` since the trigger references the column being dropped and at would make the trigger to fail.

After testing the triggers will work on the new table, the triggers are dropped from the new table until all rows have been copied and then they are re-applied.

–preserve-triggers cannot be used with these other parameters, –no-drop-triggers, –no-drop-old-table and –no-swap-tables since –preserve-triggers implies that the old triggers should be deleted and recreated in the new table. Since it is not possible to have more than one trigger with the same name, old triggers must be deleted in order to be able to recreate them into the new table.

Using `--preserve-triggers` with `--no-swap-tables` will cause triggers to remain defined for the original table. Please read the documentation for –swap-tables

If both `--no-swap-tables` and `--no-drop-new-table` is set, the trigger will remain on the original table and will be duplicated on the new table (the trigger will have a random suffix as no trigger names are unique).

`--new-table-name`
type: string; default: %T_new

New table name before it is swapped. `%T` is replaced with the original table name. When the default is used, the tool prefixes the name with up to 10 _ (underscore) to find a unique table name. If a table name is specified, the tool does not prefix it with _, so the table must not exist.

`--null-to-not-null`
Allows MODIFYing a column that allows NULL values to one that doesn't allow them. The rows which contain NULL values will be converted to the defined default value. If no explicit DEFAULT value is given MariaDB will assign a default value based on datatype, e.g. 0 for number datatypes, '' for string datatypes.

`--only-same-schema-fks`
Check foreigns keys only on tables on the same schema than the original table. This option is dangerous since if you have FKs refenrencing tables in other schemas, they won't be detected.

`--password`
short form: -p; type: string

Password to use when connecting. If password contains commas they must be escaped with a backslash: "exam,ple"

`--pause-file`
type: string

Execution will be paused while the file specified by this param exists.

`--pid`
type: string

Create the given PID file. The tool won't start if the PID file already exists and the PID it contains is different than the current PID. However, if the PID file exists and the PID it contains is no longer running, the tool will overwrite the PID file with the current PID. The PID file is removed automatically when the tool exits.

`--plugin`
type: string

Perl module file that defines a `pt_online_schema_change_plugin` class. A plugin allows you to write a Perl module that can hook into many parts of **`mariadb-schema-change`**. This requires a good knowledge of Perl and MariaDB tools conventions, which are beyond this scope of this documentation. Please contact MariaDB if you have questions or need help.

See "PLUGIN" for more information.

`--port`
short form: -P; type: int

Port number to use for connection.

`--print`
Print SQL statements to STDOUT. Specifying this option allows you to see most of the statements that the tool executes. You can use this option with *`--dry-run`*, for example.

`--progress`
type: array; default: time,30

Print progress reports to STDERR while copying rows. The value is a comma-separated list with two parts. The first part can be percentage, time, or iterations; the second part specifies how often an update should be printed, in percentage, seconds, or number of iterations.

--quiet
short form: -q

Do not print messages to STDOUT (disables *--progress*). Errors and warnings are still printed to STDERR.

--recurse
type: int

Number of levels to recurse in the hierarchy when discovering replicas. Default is infinite. See also *--recursion-method*.

--recursion-method
type: array; default: processlist,hosts

Preferred recursion method for discovering replicas. Possible methods are:

```
METHOD       USES
===========  ==================
processlist  SHOW PROCESSLIST
hosts        SHOW SLAVE HOSTS
dsn=DSN      DSNs from a table
none         Do not find slaves
```

The processlist method is the default, because SHOW SLAVE HOSTS is not reliable. However, the hosts method can work better if the server uses a non-standard port (not 3306). The tool usually does the right thing and finds all replicas, but you may give a preferred method and it will be used first.

The hosts method requires replicas to be configured with report_host, report_port, etc.

The dsn method is special: it specifies a table from which other DSN strings are read. The specified DSN must specify a D and t, or a database-qualified t. The DSN table should have the following structure:

```
CREATE TABLE `dsns` (
  `id` int(11) NOT NULL AUTO_INCREMENT,
  `parent_id` int(11) DEFAULT NULL,
  `dsn` varchar(255) NOT NULL,
  PRIMARY KEY (`id`)
);
```

To make the tool monitor only the hosts 10.10.1.16 and 10.10.1.17 for replication lag, insert the values `h=10.10.1.16` and `h=10.10.1.17` into the table. Currently, the DSNs are ordered by id, but id and parent_id are otherwise ignored.

You can change the list of hosts while OSC is executing: if you change the contents of the DSN table, OSC will pick it up very soon.

–reverse-triggers Copy the triggers added during the copy in reverse order. Commands in the new table will be reflected in the old table. You can use this as a safety feature, so that the old table continues to receive updates. This option requires `--no-drop-old-table`.

Warning! This option creates reverse triggers on the new table before it starts copying. After new table is renamed to its original name triggers will continue working. But because the name change metadata version in the table cache will also change you may start receiving "Prepared statement needs to be re-prepared" errors. The workaround for this is to re-prepare statements. If you do not use server-side prepared statements your application should not be affected.

--skip-check-slave-lag
type: DSN; repeatable: yes

DSN to skip when checking slave lag. It can be used multiple times. Example: –skip-check-slave-lag h=127.0.0.1,P=12345 –skip-check-slave-lag h=127.0.0.1,P=12346 Plase take into consideration that even when for the MariaDB driver h=127.1 is equal to h=127.0.0.1, for this parameter you need to specify the full IP address.

`--slave-user`
type: string

Sets the user to be used to connect to the slaves. This parameter allows you to have a different user with less privileges on the slaves but that user must exist on all slaves.

`--slave-password`
type: string

Sets the password to be used to connect to the slaves. It can be used with –slave-user and the password for the user must be the same on all slaves.

`--set-vars`
type: Array

Set the MariaDB variables in this comma-separated list of `variable=value` pairs.

By default, the tool sets:

```
wait_timeout=10000
innodb_lock_wait_timeout=1
lock_wait_timeout=60
```

Variables specified on the command line override these defaults. For example, specifying `--set-vars wait_timeout=500` overrides the default value of `10000`.

The tool prints a warning and continues if a variable cannot be set.

Note that setting the `sql_mode` variable requires some tricky escapes to be able to parse the quotes and commas.

Example:

```
--set-vars sql_mode=\'STRICT_ALL_TABLES\\,ALLOW_INVALID_DATES\'
```

Note the single backslash for the quotes and double backslash for the comma.

`--sleep`
type: float; default: 0

How long to sleep (in seconds) after copying each chunk. This option is useful when throttling by *`--max-lag`* and *`--max-load`* are not possible. A small, sub-second value should be used, like 0.1, else the tool could take a very long time to copy large tables.

`--socket`
short form: -S; type: string

Socket file to use for connection.

`--statistics`
Print statistics about internal counters. This is useful to see how many warnings were suppressed compared to the number of INSERT.

`--[no]swap-tables`
default: yes

Swap the original table and the new, altered table. This step completes the online schema change process by making the table with the new schema take the place of the original table. The original table becomes the "old table," and the tool drops it unless you disable *`--[no]drop-old-table`*.

Using `--no-swap-tables` will run the whole process, it will create the new table, it will copy all rows but at the end it will drop the new table. It is intended to run a more realistic –dry-run.

--tries

type: array

How many times to try critical operations. If certain operations fail due to non-fatal, recoverable errors, the tool waits and tries the operation again. These are the operations that are retried, with their default number of tries and wait time between tries (in seconds):

```
OPERATION             TRIES   WAIT
===================   =====   ====
create_triggers          10      1
drop_triggers            10      1
copy_rows                10   0.25
swap_tables              10      1
update_foreign_keys      10      1
analyze_table            10      1
```

To change the defaults, specify the new values like:

```
--tries create_triggers:5:0.5,drop_triggers:5:0.5
```

That makes the tool try `create_triggers` and `drop_triggers` 5 times with a 0.5 second wait between tries. So the format is:

```
operation:tries:wait[,operation:tries:wait]
```

All three values must be specified.

Note that most operations are affected only in MariaDB 5.5 and newer by `lock_wait_timeout` (see *--set-vars*) because of metadata locks. The `copy_rows` operation is affected in any version of MariaDB by `innodb_lock_wait_timeout`.

For creating and dropping triggers, the number of tries applies to each `CREATE TRIGGER` and `DROP TRIGGER` statement for each trigger. For copying rows, the number of tries applies to each chunk, not the entire table. For swapping tables, the number of tries usually applies once because there is usually only one `RENAME TABLE` statement. For rebuilding foreign key constraints, the number of tries applies to each statement (`ALTER` statements for the `rebuild_constraints` *--alter-foreign-keys-method*; other statements for the `drop_swap` method).

The tool retries each operation if these errors occur:

```
Lock wait timeout (innodb_lock_wait_timeout and lock_wait_timeout)
Deadlock found
Query is killed (KILL QUERY <thread_id>)
Connection is killed (KILL CONNECTION <thread_id>)
Lost connection to MariaDB
```

In the case of lost and killed connections, the tool will automatically reconnect.

Failures and retries are recorded in the *--statistics*.

--user

short form: -u; type: string

User for login if not current user.

--version

Show version and exit.

11.8 PLUGIN

The file specified by *--plugin* must define a class (i.e. a package) called `pt_online_schema_change_plugin` with a `new()` subroutine. The tool will create an instance of this class and call any hooks that it defines. No hooks are required, but a plugin isn't very useful without them.

These hooks, in this order, are called if defined:

```
init
before_create_new_table
after_create_new_table
before_alter_new_table
after_alter_new_table
before_create_triggers
after_create_triggers
before_copy_rows
after_copy_rows
before_swap_tables
after_swap_tables
before_update_foreign_keys
after_update_foreign_keys
before_drop_old_table
after_drop_old_table
before_drop_triggers
before_exit
get_slave_lag
```

Each hook is passed different arguments. To see which arguments are passed to a hook, search for the hook's name in the tool's source code, like:

```
# --plugin hook
if ( $plugin && $plugin->can('init') ) {
   $plugin->init(
      orig_tbl       => $orig_tbl,
      child_tables   => $child_tables,
      renamed_cols   => $renamed_cols,
      slaves         => $slaves,
      slave_lag_cxns => $slave_lag_cxns,
   );
}
```

The comment `# --plugin hook` precedes every hook call.

Here's a plugin file template for all hooks:

```
package pt_online_schema_change_plugin;

use strict;

sub new {
   my ($class, %args) = @_;
   my $self = { %args };
   return bless $self, $class;
}

sub init {
   my ($self, %args) = @_;
   print "PLUGIN init\n";
```

(continues on next page)

(continued from previous page)

```
 }

 sub before_create_new_table {
    my ($self, %args) = @_;
    print "PLUGIN before_create_new_table\n";
 }

 sub after_create_new_table {
    my ($self, %args) = @_;
    print "PLUGIN after_create_new_table\n";
 }

 sub before_alter_new_table {
    my ($self, %args) = @_;
    print "PLUGIN before_alter_new_table\n";
 }

 sub after_alter_new_table {
    my ($self, %args) = @_;
    print "PLUGIN after_alter_new_table\n";
 }

 sub before_create_triggers {
    my ($self, %args) = @_;
    print "PLUGIN before_create_triggers\n";
 }

sub after_create_triggers {
    my ($self, %args) = @_;
    print "PLUGIN after_create_triggers\n";
 }

 sub before_copy_rows {
    my ($self, %args) = @_;
    print "PLUGIN before_copy_rows\n";
 }

 sub after_copy_rows {
    my ($self, %args) = @_;
    print "PLUGIN after_copy_rows\n";
 }

 sub before_swap_tables {
    my ($self, %args) = @_;
    print "PLUGIN before_swap_tables\n";
 }

 sub after_swap_tables {
    my ($self, %args) = @_;
    print "PLUGIN after_swap_tables\n";
 }

 sub before_update_foreign_keys {
    my ($self, %args) = @_;
    print "PLUGIN before_update_foreign_keys\n";
 }
```

(continues on next page)

(continued from previous page)

```
sub after_update_foreign_keys {
   my ($self, %args) = @_;
   print "PLUGIN after_update_foreign_keys\n";
}

sub before_drop_old_table {
   my ($self, %args) = @_;
   print "PLUGIN before_drop_old_table\n";
}

sub after_drop_old_table {
   my ($self, %args) = @_;
   print "PLUGIN after_drop_old_table\n";
}

sub before_drop_triggers {
   my ($self, %args) = @_;
   print "PLUGIN before_drop_triggers\n";
}

sub before_exit {
   my ($self, %args) = @_;
   print "PLUGIN before_exit\n";
}

sub get_slave_lag {
   my ($self, %args) = @_;
   print "PLUGIN get_slave_lag\n";

   return sub { return 0; };
}

1;
```

Notice that `get_slave_lag` must return a function reference; ideally one that returns actual slave lag, not simply zero like in the example.

Here's an example that actually does something:

```
package pt_online_schema_change_plugin;

use strict;

sub new {
   my ($class, %args) = @_;
   my $self = { %args };
   return bless $self, $class;
}

sub after_create_new_table {
   my ($self, %args) = @_;
   my $new_tbl = $args{new_tbl};
   my $dbh     = $self->{cxn}->dbh;
   my $row = $dbh->selectrow_arrayref("SHOW CREATE TABLE $new_tbl->{name}");
   warn "after_create_new_table: $row->[1]\n\n";
}
```

(continues on next page)

(continued from previous page)

```
sub after_alter_new_table {
   my ($self, %args) = @_;
   my $new_tbl = $args{new_tbl};
   my $dbh     = $self->{cxn}->dbh;
   my $row = $dbh->selectrow_arrayref("SHOW CREATE TABLE $new_tbl->{name}");
   warn "after_alter_new_table: $row->[1]\n\n";
}

1;
```

You could use this with *--dry-run* to check how the table will look before and after.

Please contact MariaDB if you have questions or need help.

11.9 DSN OPTIONS

These DSN options are used to create a DSN. Each option is given like `option=value`. The options are case-sensitive, so P and p are not the same option. There cannot be whitespace before or after the = and if the value contains whitespace it must be quoted. DSN options are comma-separated. See the mariadb-tools manpage for full details.

- A

  dsn: charset; copy: yes

  Default character set.

- D

  dsn: database; copy: no

  Database for the old and new table.

- F

  dsn: mysql_read_default_file; copy: yes

  Only read default options from the given file

- h

  dsn: host; copy: yes

  Connect to host.

- p

  dsn: password; copy: yes

  Password to use when connecting. If password contains commas they must be escaped with a backslash: "exam,ple"

- P

  dsn: port; copy: yes

  Port number to use for connection.

- S

  dsn: mysql_socket; copy: yes

  Socket file to use for connection.

- t

  dsn: table; copy: no

  Table to alter.

- u

  dsn: user; copy: yes

  User for login if not current user.

11.10 ENVIRONMENT

The environment variable PTDEBUG enables verbose debugging output to STDERR. To enable debugging and capture all output to a file, run the tool like:

```
PTDEBUG=1 mariadb-schema-change ... > FILE 2>&1
```

Be careful: debugging output is voluminous and can generate several megabytes of output.

11.11 EXIT STATUS

```
INVALID_PARAMETERS         = 1
UNSUPORTED_MYSQL_VERSION   = 2
NO_MINIMUM_REQUIREMENTS    = 3
NO_PRIMARY_OR_UNIQUE_KEY   = 4
INVALID_PLUGIN_FILE        = 5
INVALID_ALTER_FK_METHOD    = 6
INVALID_KEY_SIZE           = 7
CANNOT_DETERMINE_KEY_SIZE  = 9
NOT_SAFE_TO_ASCEND         = 9
ERROR_CREATING_NEW_TABLE   = 10
ERROR_ALTERING_TABLE       = 11
ERROR_CREATING_TRIGGERS    = 12
ERROR_RESTORING_TRIGGERS   = 13
ERROR_SWAPPING_TABLES      = 14
ERROR_UPDATING_FKS         = 15
ERROR_DROPPING_OLD_TABLE   = 16
UNSUPORTED_OPERATION       = 17
MYSQL_CONNECTION_ERROR     = 18
LOST_MYSQL_CONNECTION      = 19
```

11.12 SYSTEM REQUIREMENTS

You need Perl, DBI, DBD::mysql, and some core packages that ought to be installed in any reasonably new version of Perl.

This tool works only on MariaDB 5.0.2 and newer versions, because earlier versions do not support triggers. Also a number of permissions should be set on MariaDB to make **mariadb-schema-change** operate as expected. PROCESS, SUPER, REPLICATION SLAVE global privileges, as well as SELECT, INSERT, UPDATE, DELETE, CREATE, DROP, ALTER, and TRIGGER table privileges should be granted on server. Slave needs only REPLICATION SLAVE and REPLICATION CLIENT privileges.

11.13 AUTHORS

Cole Busby, Daniel Nichter and Baron Schwartz

11.14 ACKNOWLEDGMENTS

The "online schema change" concept was first implemented by Shlomi Noach in his tool `oak-online-alter-table`, part of http://code.google.com/p/openarkkit/. Engineers at Facebook then built another version called `OnlineSchemaChange.php` as explained by their blog post: http://tinyurl.com/32zeb86. This tool is a hybrid of both approaches, with additional features and functionality not present in either.

11.15 ABOUT THIS MARIADB TOOL

This tool is part of MariaDB client tools. This MariaDB Tool was forked from Percona Toolkit's pt-online-schema-change in August, 2019. Percona Toolkit was forked from two projects in June, 2011: Maatkit and Aspersa. Those projects were created by Baron Schwartz and primarily developed by him and Daniel Nichter.

11.16 COPYRIGHT, LICENSE, AND WARRANTY

This program is copyright 2019-2021 MariaDB Corporation and/or its affiliates, 2011-2018 Percona LLC and/or its affiliates, 2010-2011 Baron Schwartz.

THIS PROGRAM IS PROVIDED "AS IS" AND WITHOUT ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, WITHOUT LIMITATION, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE.

This program is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation, version 2; OR the Perl Artistic License. On UNIX and similar systems, you can issue `man perlgpl' or `man perlartistic' to read these licenses.

You should have received a copy of the GNU General Public License along with this program; if not, write to the Free Software Foundation, Inc., 59 Temple Place, Suite 330, Boston, MA 02111-1307 USA.

11.17 VERSION

`mariadb-schema-change` 6.0.0rc

CHAPTER

TWELVE

MARIADB-STACKTRACE

12.1 NAME

mariadb-stacktrace - Aggregate GDB stack traces for a selected program.

12.2 SYNOPSIS

12.2.1 Usage

```
mariadb-stacktrace [OPTIONS] [FILES]
```

mariadb-stacktrace is a poor man's profiler, inspired by http://poormansprofiler.org. It can create and summarize full stack traces of processes on Linux. Summaries of stack traces can be an invaluable tool for diagnosing what a process is waiting for.

12.3 RISKS

mariadb-stacktrace is mature, proven in the real world, and well tested, but all database tools can pose a risk to the system and the database server. Before using this tool, please:

- Read the tool's documentation
- Test the tool on a non-production server
- Backup your production server and verify the backups

12.4 DESCRIPTION

mariadb-stacktrace performs two tasks: it gets a stack trace, and it summarizes the stack trace. If a file is given on the command line, the tool skips the first step and just aggregates the file.

To summarize the stack trace, the tool extracts the function name (symbol) from each level of the stack, and combines them with commas. It does this for each thread in the output. Afterwards, it sorts similar threads together and counts how many of each one there are, then sorts them most-frequent first.

mariadb-stacktrace is a read-only tool. However, collecting GDB stacktraces is achieved by attaching GDB to the program and printing stack traces from all threads. This will freeze the program for some period of time, ranging from a second or so to much longer on very busy systems with a lot of memory and many threads in the program.

In the tool's default usage as a MariaDB profiling tool, this means that MariaDB will be unresponsive while the tool runs, although if you are using the tool to diagnose an unresponsive server, there is really no reason not to do this. In addition to freezing the server, there is also some risk of the server crashing or performing badly after GDB detaches from it.

12.5 OPTIONS

`--binary`
short form: -b; type: string; default: mysqld

Which binary to trace.

`--help`
Show help and exit.

`--interval`
short form: -s; type: int; default: 0

Number of seconds to sleep between *`--iterations`*.

`--iterations`
short form: -i; type: int; default: 1

How many traces to gather and aggregate.

`--lines`
short form: -l; type: int; default: 0

Aggregate only first specified number of many functions; 0=infinity.

`--pid`
short form: -p; type: int

Process ID of the process to trace; overrides *`--binary`*.

`--save-samples`
short form: -k; type: string

Keep the raw traces in this file after aggregation.

`--version`
Show version and exit.

12.6 ENVIRONMENT

This tool does not use any environment variables.

12.7 SYSTEM REQUIREMENTS

This tool requires Bash v3 or newer. If no backtrace files are given, then gdb is also required to create backtraces for the process specified on the command line.

12.8 AUTHORS

Cole Busby, Baron Schwartz, based on a script by Domas Mituzas (http://poormansprofiler.org/)

12.9 ABOUT THIS MARIADB TOOL

This tool is part of MariaDB client tools. This MariaDB Tool was forked from Percona Toolkit's pt-pmp in August, 2019. Percona Toolkit was forked from two projects in June, 2011: Maatkit and Aspersa. Those projects were created by Baron Schwartz and primarily developed by him and Daniel Nichter.

12.10 COPYRIGHT, LICENSE, AND WARRANTY

12.11 VERSION

`mariadb-stacktrace` 6.0.0rc

CHAPTER THIRTEEN

MARIADB-STAT

13.1 NAME

mariadb-stat - Collect forensic data about MariaDB when problems occur.

13.2 SYNOPSIS

13.2.1 Usage

```
mariadb-stat [OPTIONS]
```

mariadb-stat waits for a trigger condition to occur, then collects data to help diagnose problems. The tool is designed to run as a daemon with root privileges, so that you can diagnose intermittent problems that you cannot observe directly. You can also use it to execute a custom command, or to collect data on demand without waiting for the trigger to occur.

13.3 RISKS

mariadb-stat is mature, proven in the real world, and well tested, but all database tools can pose a risk to the system and the database server. Before using this tool, please:

- Read the tool's documentation
- Review the tool's known "BUGS"
- Test the tool on a non-production server
- Backup your production server and verify the backups

13.4 DESCRIPTION

Sometimes a problem happens infrequently and for a short time, giving you no chance to see the system when it happens. How do you solve intermittent MariaDB problems when you can't observe them? That's why **mariadb-stat** exists. In addition to using it when there's a known problem on your servers, it is a good idea to run **mariadb-stat** all the time, even when you think nothing is wrong. You will appreciate the data it collects when a problem occurs, because problems such as MariaDB lockups or spikes in activity typically leave no evidence to use in root cause analysis.

mariadb-stat does two things: it watches a MariaDB server and waits for a trigger condition to occur, and it collects diagnostic data when that trigger occurs. To avoid false-positives caused by short-lived problems, the trigger condition must be true at least *--cycles* times before a *--collect* is triggered.

To use **mariadb-stat** effectively, you need to define a good trigger. A good trigger is sensitive enough to fire reliably when a problem occurs, so that you don't miss a chance to solve problems. On the other hand, a good trigger isn't prone to false positives, so you don't gather information when the server is functioning normally.

The most reliable triggers for MariaDB tend to be the number of connections to the server, and the number of queries running concurrently. These are available in the SHOW GLOBAL STATUS command as Threads_connected and Threads_running. Sometimes Threads_connected is not a reliable indicator of trouble, but Threads_running usually is. Your job, as the tool's user, is to define an appropriate trigger condition for the tool. Choose carefully, because the quality of your results will depend on the trigger you choose.

You define the trigger with the *--function*, *--variable*, *--threshold*, and *--cycles* options. The default values for these options define a reasonable trigger, but you should adjust or change them to suite your particular system and needs.

By default, **mariadb-stat** tool watches MariaDB forever until the trigger occurs, then it collects diagnostic data for a while, and sleeps afterwards to avoid repeatedly collecting data if the trigger remains true. The general order of operations is:

```
while true; do
   if --variable from --function > --threshold; then
      cycles_true++
      if cycles_true >= --cycles; then
         --notify-by-email
         if --collect; then
            if --disk-bytes-free and --disk-pct-free ok; then
               (--collect for --run-time seconds) &
            fi
            rm files in --dest older than --retention-time
         fi
         iter++
         cycles_true=0
      fi
      if iter < --iterations; then
         sleep --sleep seconds
      else
         break
      fi
   else
      if iter < --iterations; then
         sleep --interval seconds
      else
         break
      fi
   fi
```

(continues on next page)

(continued from previous page)

```
done
rm old --dest files older than --retention-time
if --collect process are still running; then
   wait up to --run-time * 3 seconds
   kill any remaining --collect processes
fi
```

The diagnostic data is written to files whose names begin with a timestamp, so you can distinguish samples from each other in case the tool collects data multiple times. The pt-sift tool is designed to help you browse and analyze the resulting data samples.

Although this sounds simple enough, in practice there are a number of subtleties, such as detecting when the disk is beginning to fill up so that the tool doesn't cause the server to run out of disk space. This tool handles these types of potential problems, so it's a good idea to use this tool instead of writing something from scratch and possibly experiencing some of the hazards this tool is designed to avoid.

13.5 CONFIGURING

You can use standard MariaDB tool configuration files to set command line options.

You will probably want to run the tool as a daemon and customize at least the *--threshold*. Here's a sample configuration file for triggering when there are more than 20 queries running at once:

```
daemonize
threshold=20
```

If you don't run the tool as root, then you will need specify several options, such as *--pid*, *--log*, and *--dest*, else the tool will probably fail to start.

13.6 OPTIONS

--ask-pass

Prompt for a password when connecting to MariaDB.

--collect

default: yes; negatable: yes

Collect diagnostic data when the trigger occurs. Specify `--no-collect` to make the tool watch the system but not collect data.

See also *--stalk*.

--collect-gdb

Collect GDB stacktraces. This is achieved by attaching to MariaDB and printing stack traces from all threads. This will freeze the server for some period of time, ranging from a second or so to much longer on very busy systems with a lot of memory and many threads in the server. For this reason, it is disabled by default. However, if you are trying to diagnose a server stall or lockup, freezing the server causes no additional harm, and the stack traces can be vital for diagnosis.

In addition to freezing the server, there is also some risk of the server crashing or performing badly after GDB detaches from it.

--collect-oprofile

Collect oprofile data. This is achieved by starting an oprofile session, letting it run for the collection time, and then stopping and saving the resulting profile data in the system's default location. Please read your system's oprofile documentation to learn more about this.

--collect-strace
Collect strace data. This is achieved by attaching strace to the server, which will make it run very slowly until strace detaches. The same cautions apply as those listed in –collect-gdb. You should not enable this option together with –collect-gdb, because GDB and strace can't attach to the server process simultaneously.

--collect-tcpdump
Collect tcpdump data. This option causes tcpdump to capture all traffic on all interfaces for the port on which MariaDB is listening. You can later use pt-query-digest to decode the MariaDB protocol and extract a log of query traffic from it.

--config
type: string

Read this comma-separated list of config files. If specified, this must be the first option on the command line.

--cycles
type: int; default: 5

How many times `--variable` must be greater than `--threshold` before triggering `--collect`. This helps prevent false positives, and makes the trigger condition less likely to fire when the problem recovers quickly.

--daemonize
Daemonize the tool. This causes the tool to fork into the background and log its output as specified in –log.

--defaults-file
short form: -F; type: string

Only read mariadb options from the given file. You must give an absolute pathname.

--dest
type: string; default: /var/lib/mariadb-stat

Where to save diagnostic data from `--collect`. Each time the tool collects data, it writes to a new set of files, which are named with the current system timestamp.

--disk-bytes-free
type: size; default: 100M

Do not `--collect` if the disk has less than this much free space. This prevents the tool from filling up the disk with diagnostic data.

If the `--dest` directory contains a previously captured sample of data, the tool will measure its size and use that as an estimate of how much data is likely to be gathered this time, too. It will then be even more pessimistic, and will refuse to collect data unless the disk has enough free space to hold the sample and still have the desired amount of free space. For example, if you'd like 100MB of free space and the previous diagnostic sample consumed 100MB, the tool won't collect any data unless the disk has 200MB free.

Valid size value suffixes are k, M, G, and T.

--disk-pct-free
type: int; default: 5

Do not `--collect` if the disk has less than this percent free space. This prevents the tool from filling up the disk with diagnostic data.

This option works similarly to `--disk-bytes-free` but specifies a percentage margin of safety instead of a bytes margin of safety. The tool honors both options, and will not collect any data unless both margins are satisfied.

--function

type: string; default: status

What to watch for the trigger. The default value watches `SHOW GLOBAL STATUS`, but you can also watch `SHOW PROCESSLIST` and specify a file with your own custom code. This function supplies the value of *--variable*, which is then compared against *--threshold* to see if the the trigger condition is met. Additional options may be required as well; see below. Possible values are:

- status

  Watch `SHOW GLOBAL STATUS` for the trigger. The value of *--variable* then defines which status counter is the trigger.

- processlist

  Watch `SHOW FULL PROCESSLIST` for the trigger. The trigger value is the count of processes whose *--variable* column matches the *--match* option. For example, to trigger *--collect* when more than 10 processes are in the "statistics" state, specify:

  ```
  --function processlist \
  --variable State       \
  --match statistics     \
  --threshold 10
  ```

In addition, you can specify a file that contains your custom trigger function, written in Unix shell script. This can be a wrapper that executes anything you wish. If the argument to *--function* is a file, then it takes precedence over built-in functions, so if there is a file in the working directory named "status" or "processlist" then the tool will use that file even though are valid built-in values.

The file works by providing a function called `trg_plugin`, and the tool simply sources the file and executes the function. For example, the file might contain:

```
trg_plugin() {
   mysql $EXT_ARGV -e "SHOW ENGINE INNODB STATUS" \
     | grep -c "has waited at"
}
```

This snippet will count the number of mutex waits inside InnoDB. It illustrates the general principle: the function must output a number, which is then compared to *--threshold* as usual. The `$EXT_ARGV` variable contains the MariaDB options mentioned in the "SYNOPSIS" above.

The file should not alter the tool's existing global variables. Prefix any file-specific global variables with "**PLUGIN_**" or make them local.

--help

Print help and exit.

--host

short form: -h; type: string

Host to connect to.

--interval

type: int; default: 1

How often to check the if trigger is true, in seconds.

--iterations

type: int

How many times to *--collect* diagnostic data. By default, the tool runs forever and collects data every time the trigger occurs. Specify *--iterations* to collect data a limited number of times. This option is also useful with `--no-stalk` to collect data once and exit, for example.

--log
type: string; default: /var/log/mariadb-stat.log

Print all output to this file when daemonized.

--match
type: string

The pattern to use when watching SHOW PROCESSLIST. See *--function* for details.

--notify-by-email
type: string

Send an email to these addresses for every *--collect*.

--password
short form: -p; type: string

Password to use when connecting. If password contains commas they must be escaped with a backslash: "exam,ple"

--pid
type: string; default: /var/run/mariadb-stat.pid

Create the given PID file. The tool won't start if the PID file already exists and the PID it contains is different than the current PID. However, if the PID file exists and the PID it contains is no longer running, the tool will overwrite the PID file with the current PID. The PID file is removed automatically when the tool exits.

--plugin
type: string

Load a plugin to hook into the tool and extend is functionality. The specified file does not need to be executable, nor does its first line need to be shebang line. It only needs to define one or more of these Bash functions:

before_stalk

Called before stalking.

before_collect

Called when the trigger occurs, before running a *--collect* subprocesses in the background.

after_collect

Called after running a collector process. The PID of the collector process is passed as the first argument. This hook is called before `after_collect_sleep`.

after_collect_sleep

Called after sleeping *--sleep* seconds for the collector process to finish. This hook is called after `after_collect`.

after_interval_sleep

Called after sleeping *--interval* seconds after each trigger check.

after_stalk

Called after stalking. Since **mariadb-stat** stalks forever by default, this hook is only called if *--iterations* is specified.

For example, a very simple plugin that touches a file when *--collect* is triggered:

```
before_collect() {
   touch /tmp/foo
}
```

Since the plugin is completely sourced (imported) into the tool's namespace, be careful not to define other functions or global variables that already exist in the tool. You should prefix all plugin-specific functions and global variables with `plugin_` or `PLUGIN_`.

Plugins have access to all command line options but they should not modify them. Each option is a global variable like `$OPT_DEST` which corresponds to *`--dest`*. Therefore, the global variable for each command line option is `OPT_` plus the option name in all caps with hyphens replaced by underscores.

Plugins can stop the tool by setting the global variable `OKTORUN` to `1`. In this case, the global variable `EXIT_REASON` should also be set to indicate why the tool was stopped.

Plugin writers should keep in mind that the file destination prefix currently in use should be accessed through the `$prefix` variable, rather than `$OPT_PREFIX`.

`--mariadb-only`
Trigger only MariaDB related captures, ignoring all others. The only not MariaDB related value being collected is the disk space, because it is needed to calculate the available free disk space to write the result files. This option is useful for RDS instances.

`--port`
short form: -P; type: int

Port number to use for connection.

`--prefix`
type: string

The filename prefix for diagnostic samples. By default, all files created by the same *`--collect`* instance have a timestamp prefix based on the current local time, like `2011_12_06_14_02_02`, which is December 6, 2011 at 14:02:02.

`--retention-count`
type: int; default: 0

Keep the data for the last N runs. If N > 0, the program will keep the data for the last N runs and will delete the older data.

`--retention-size`
type: int; default: 0

Keep up to –retention-size MB of data. It will keep at least 1 run even if the size is bigger than the specified in this parameter

`--retention-time`
type: int; default: 30

Number of days to retain collected samples. Any samples that are older will be purged.

`--run-time`
type: int; default: 30

How long to *`--collect`* diagnostic data when the trigger occurs. The value is in seconds and should not be longer than *`--sleep`*. It is usually not necessary to change this; if the default 30 seconds doesn't collect enough data, running longer is not likely to help because the system or MariaDB server is probably too busy to respond. In fact, in many cases a shorter collection period is appropriate.

This value is used two other times. After collecting, the collect subprocess will wait another *`--run-time`* seconds for its commands to finish. Some commands can take awhile if the system is running very slowly (which can likely be the case given that a collection was triggered). Since empty files are deleted, the extra wait gives commands time to finish and write their data. The value is potentially used again just before the tool exits to wait again for any collect subprocesses to finish. In most cases this won't happen because of the aforementioned extra wait. If it happens, the tool will log "Waiting up to N seconds for subprocesses to finish…" where N is three times *--run-time*. In both cases, after waiting, the tool kills all of its subprocesses.

--sleep
type: int; default: 300

How long to sleep after *--collect*. This prevents the tool from triggering continuously, which might be a problem if the collection process is intrusive. It also prevents filling up the disk or gathering too much data to analyze reasonably.

--sleep-collect
type: int; default: 1

How long to sleep between collection loop cycles. This is useful with `--no-stalk` to do long collections. For example, to collect data every minute for an hour, specify: `--no-stalk --run-time 3600 --sleep-collect 60`.

--socket
short form: -S; type: string

Socket file to use for connection.

--stalk
default: yes; negatable: yes

Watch the server and wait for the trigger to occur. Specify `--no-stalk` to collect diagnostic data immediately, that is, without waiting for the trigger to occur. You probably also want to specify values for *--interval*, *--iterations*, and *--sleep*. For example, to immediately collect data for 1 minute then exit, specify:

```
--no-stalk --run-time 60 --iterations 1
```

--cycles, *--daemonize*, *--log* and *--pid* have no effect with `--no-stalk`. Safeguard options, like *--disk-bytes-free* and *--disk-pct-free*, are still respected.

See also *--collect*.

--threshold
type: int; default: 25

The maximum acceptable value for *--variable*. *--collect* is triggered when the value of *--variable* is greater than *--threshold* for *--cycles* many times. Currently, there is no way to define a lower threshold to check for a *--variable* value that is too low.

See also *--function*.

--user
short form: -u; type: string

User for login if not current user.

--variable
type: string; default: Threads_running

The variable to compare against *--threshold*. See also *--function*.

--verbose
type: int; default: 2

Print more or less information while running. Since the tool is designed to be a long-running daemon, the default verbosity level only prints the most important information. If you run the tool interactively, you may want to use a higher verbosity level.

```
LEVEL PRINTS
===== =====================================
0     Errors
1     Warnings
2     Matching triggers and collection info
3     Non-matching triggers
```

--version

Print tool's version and exit.

13.7 ENVIRONMENT

This tool does not require any environment variables for configuration, although it can be influenced to work differently by through several variables. Keep in mind that these are expert settings, and should not be used in most cases.

Specifically, the variables that can be set are:

CMD_GDB

CMD_IOSTAT

CMD_MPSTAT

CMD_MYSQL

CMD_MYSQLADMIN

CMD_OPCONTROL

CMD_OPREPORT

CMD_PMAP

CMD_STRACE

CMD_SYSCTL

CMD_TCPDUMP

CMD_VMSTAT

For example, during collection iostat is called with a -dx argument, but because you have an NFS partition, you also need the -n flag there. Instead of editing the source, you can call **mariadb-stat** as

```
CMD_IOSTAT="iostat -n" mariadb-stat ...
```

which will do exactly what you need. Combined with the plugin hooks, this gives you a fine-grained control of what the tool does.

It is possible to enable `debug` mode in mysqladmin specifying:

`CMD_MYSQLADMIN='mysqladmin debug' :program:`mariadb-stat` params ...`

13.8 SYSTEM REQUIREMENTS

This tool requires Bash v3 or newer. Certain options require other programs:

--collect-gdb requires `gdb`

--collect-oprofile requires `opcontrol` and `opreport`

--collect-strace requires `strace`

--collect-tcpdump requires `tcpdump`

13.9 AUTHORS

Cole Busby, Baron Schwartz, Justin Swanhart, Fernando Ipar, Daniel Nichter, and Brian Fraser

13.10 ABOUT THIS MARIADB TOOL

This tool is part of MariaDB client tools. This MariaDB Tool was forked from Percona Toolkit's pt-stalk in August, 2019. Percona Toolkit was forked from two projects in June, 2011: Maatkit and Aspersa. Those projects were created by Baron Schwartz and primarily developed by him and Daniel Nichter.

13.11 COPYRIGHT, LICENSE, AND WARRANTY

This program is copyright 2019-2021 MariaDB Corporation and/or its affiliates, 2011-2018 Percona LLC and/or its affiliates, 2010-2011 Baron Schwartz.

THIS PROGRAM IS PROVIDED "AS IS" AND WITHOUT ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, WITHOUT LIMITATION, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE.

This program is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation, version 2; OR the Perl Artistic License. On UNIX and similar systems, you can issue `man perlgpl' or `man perlartistic' to read these licenses.

You should have received a copy of the GNU General Public License along with this program; if not, write to the Free Software Foundation, Inc., 59 Temple Place, Suite 330, Boston, MA 02111-1307 USA.

13.12 VERSION

`mariadb-stat` 6.0.0rc

CHAPTER

FOURTEEN

MARIADB-STAT-BROWSER

14.1 NAME

mariadb-stat-browser - Browses files created by mariadb-stat.

14.2 SYNOPSIS

14.2.1 Usage

```
mariadb-stat-browser FILE|PREFIX|DIRECTORY
```

mariadb-stat-browser browses files created by mariadb-stat. If no options are given, the tool browses all mariadb-stat files in `/var/lib/mariadb-stat` if that directory exists, else the current working directory is used. If a FILE is given, the tool browses files with the same prefix in the given file's directory. If a PREFIX is given, the tool browses files in `/var/lib/mariadb-stat` (or the current working directory) with the same prefix. If a DIRECTORY is given, the tool browses all mariadb-stat files in it.

14.3 RISKS

Percona Toolkit is mature, proven in the real world, and well tested, but all database tools can pose a risk to the system and the database server. Before using this tool, please:

- Read the tool's documentation
- Review the tool's known "BUGS"
- Test the tool on a non-production server
- Backup your production server and verify the backups

14.4 DESCRIPTION

`mariadb-stat-browser` downloads other tools that it might need, such as mariadb-iostat, and then makes a list of the unique timestamp prefixes of all the files in the directory, as written by the mariadb-stat tool. If the user specified a timestamp on the command line, then it begins with that sample of data; otherwise it begins by showing a list of the timestamps and prompting for a selection. Thereafter, it displays a summary of the selected sample, and the user can navigate and inspect with keystrokes. The keystroke commands you can use are as follows:

- d

  Sets the action to start the mariadb-iostat tool on the sample's disk performance statistics.

- i

  Sets the action to view the first INNODB STATUS sample in less.

- m

  Displays the first 4 samples of SHOW STATUS counters side by side with the mariadb-status-diff tool.

- n

  Summarizes the first sample of netstat data in two ways: by originating host, and by connection state.

- j

  Select the next timestamp as the active sample.

- k

  Select the previous timestamp as the active sample.

- q

  Quit the program.

- 1

  Sets the action for each sample to the default, which is to view a summary of the sample.

- 0

  Sets the action to just list the files in the sample.

- **`-`**

  Sets the action to view all of the sample's files in the less program.

14.5 OPTIONS

`--help`
  Show help and exit.

`--version`
  Show version and exit.

14.6 ENVIRONMENT

This tool does not use any environment variables.

14.7 SYSTEM REQUIREMENTS

This tool requires Bash v3 and the following programs: mariadb-iostat, mariadb-stacktrace, mariadb-status-diff, and mariadb-align-output. If these programs are not in your PATH, they will be fetched from the Internet if curl is available.

14.8 AUTHORS

Cole Busby, Baron Schwartz

14.9 ABOUT THIS MARIADB TOOL

This tool is part of MariaDB client tools. This MariaDB Tool was forked from Percona Toolkit's pt-stalk in August, 2019. Percona Toolkit was forked from two projects in June, 2011: Maatkit and Aspersa. Those projects were created by Baron Schwartz and primarily developed by him and Daniel Nichter.

14.10 COPYRIGHT, LICENSE, AND WARRANTY

This program is copyright 2019-2021 MariaDB Corporation and/or its affiliates, 2011-2018 Percona LLC and/or its affiliates, 2010-2011 Baron Schwartz.

THIS PROGRAM IS PROVIDED "AS IS" AND WITHOUT ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, WITHOUT LIMITATION, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE.

This program is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation, version 2; OR the Perl Artistic License. On UNIX and similar systems, you can issue `man perlgpl' or `man perlartistic' to read these licenses.

You should have received a copy of the GNU General Public License along with this program; if not, write to the Free Software Foundation, Inc., 59 Temple Place, Suite 330, Boston, MA 02111-1307 USA.

14.11 VERSION

`mariadb-stat-browser` 6.0.0rc

CHAPTER

FIFTEEN

MARIADB-STATUS-DIFF

15.1 NAME

mariadb-status-diff - Look at many samples of MariaDB `SHOW GLOBAL STATUS` side-by-side.

15.2 SYNOPSIS

15.2.1 Usage

```
mariadb-status-diff [OPTIONS] -- COMMAND
```

mariadb-status-diff columnizes repeated output from a program like mariadb-admin extended.

Get output from `mariadb-admin`:

```
mariadb-status-diff -r -- mariadb-admin ext -i10 -c3
```

Get output from a file:

```
mariadb-status-diff -r -- cat mariadb-admin-output.txt
```

15.3 RISKS

pt-mext is mature, proven in the real world, and well tested, but all database tools can pose a risk to the system and the database server. Before using this tool, please:

- Read the tool's documentation
- Review the tool's known "BUGS"
- Test the tool on a non-production server
- Backup your production server and verify the backups

15.4 DESCRIPTION

mariadb-status-diff executes the `COMMAND` you specify, and reads through the result one line at a time. It places each line into a temporary file. When it finds a blank line, it assumes that a new sample of SHOW GLOBAL STATUS is starting, and it creates a new temporary file. At the end of this process, it has a number of temporary files. It joins the temporary files together side-by-side and prints the result. If *--relative* option is given, it first subtracts each sample from the one after it before printing results.

15.5 OPTIONS

--help
Show help and exit.

--relative
short form: -r

Subtract each column from the previous column.

--version
Show version and exit.

15.6 ENVIRONMENT

This tool does not use any environment variables.

15.7 SYSTEM REQUIREMENTS

This tool requires the Bourne shell (*/bin/sh*) and the seq program.

15.8 AUTHORS

Cole Busby, Baron Schwartz

15.9 ABOUT THIS MARIADB TOOL

This tool is part of MariaDB client tools. This MariaDB Tool was forked from Percona Toolkit's pt-mext in August, 2019. Percona Toolkit was forked from two projects in June, 2011: Maatkit and Aspersa. Those projects were created by Baron Schwartz and primarily developed by him and Daniel Nichter.

15.10 COPYRIGHT, LICENSE, AND WARRANTY

15.11 VERSION

`mariadb-status-diff` 6.0.0rc

CHAPTER SIXTEEN

MARIADB-SUMMARY

16.1 NAME

mariadb-summary - Summarize system information nicely.

16.2 SYNOPSIS

16.2.1 Usage

```
mariadb-summary
```

mariadb-summary conveniently summarizes the status and configuration of a database and its underlying server. It is not a tuning tool or diagnosis tool. It produces a report that is easy to diff and can be pasted into emails without losing the formatting. This tool works well on many types of Unix systems.

16.3 RISKS

mariadb-summary is mature, proven in the real world, and well tested, but all database tools can pose a risk to the system and the database server. Before using this tool, please:

- Read the tool's documentation
- Test the tool on a non-production server
- Backup your production server and verify the backups

16.4 DESCRIPTION

mariadb-summary runs mariadb-system-summary and mariadb-database-summary together.

These tools run a large variety of commands to inspect system and MariaDB status and configuration.

It works best when executed as a privileged user, but will also work without privileges, although some output might not be possible to generate without root.

16.5 OUTPUT

See mariadb-system-summary and mariadb-database-summary documentation for output details.

16.6 OPTIONS

`--all-databases`
: mariadb-dump and summarize all databases. See *`--databases`*.

`--ask-pass`
: Prompt for a password when connecting to MariaDB.

`--config`
: type: string

  Read this comma-separated list of config files. If specified, this must be the first option on the command line.

`--databases`
: type: string

  mariadb-dump and summarize this comma-separated list of databases. Specify *`--all-databases`* instead if you want to dump and summary all databases.

`--defaults-file`
: short form: -F; type: string

  Only read mariadb options from the given file. You must give an absolute pathname.

`--help`
: Print help and exit.

`--host`
: short form: -h; type: string

  Host to connect to.

`--password`
: short form: -p; type: string

  Password to use when connecting. If password contains commas they must be escaped with a backslash: “exam,ple”

`--port`
: short form: -P; type: int

  Port number to use for connection.

`--read-samples`
: type: string

  Create a report from the files in this directory.

`--save-samples`
: type: string

  Save the collected data in this directory.

`--sleep`
: type: int; default: 5

  How long to sleep when gathering samples from vmstat.

`--socket`
short form: -S; type: string

Socket file to use for connection.

`--summarize-mounts`
default: yes; negatable: yes

Report on mounted filesystems and disk usage.

`--summarize-network`
default: yes; negatable: yes

Report on network controllers and configuration.

`--summarize-processes`
default: yes; negatable: yes

Report on top processes and `vmstat` output.

`--user`
short form: -u; type: string

User for login if not current user.

`--version`
Print tool's version and exit.

16.7 ENVIRONMENT

This tool does not use any environment variables.

16.8 SYSTEM REQUIREMENTS

This tool requires the Bourne shell (*/bin/sh*).

16.9 AUTHORS

Cole Busby, Manjot Singh

16.10 ABOUT THIS MARIADB TOOL

This tool is part of MariaDB client tools. This MariaDB Tool was created in August, 2019, based on Percona Toolkit which was forked from two projects in June, 2011: Maatkit and Aspersa. Those projects were created by Baron Schwartz and primarily developed by him and Daniel Nichter.

16.11 COPYRIGHT, LICENSE, AND WARRANTY

16.12 VERSION

`mariadb-summary` 6.0.0rc

CHAPTER

SEVENTEEN

MARIADB-SYSTEM-SUMMARY

17.1 NAME

mariadb-system-summary - Summarize system information nicely.

17.2 SYNOPSIS

17.2.1 Usage

```
mariadb-system-summary
```

mariadb-system-summary conveniently summarizes the status and configuration of a server. It is not a tuning tool or diagnosis tool. It produces a report that is easy to diff and can be pasted into emails without losing the formatting. This tool works well on many types of Unix systems.

17.3 RISKS

mariadb-system-summary is mature, proven in the real world, and well tested, but all database tools can pose a risk to the system and the database server. Before using this tool, please:

- Read the tool's documentation
- Test the tool on a non-production server
- Backup your production server and verify the backups

17.4 DESCRIPTION

mariadb-system-summary runs a large variety of commands to inspect system status and configuration, saves the output into files in a temporary directory, and then runs Unix commands on these results to format them nicely. It works best when executed as a privileged user, but will also work without privileges, although some output might not be possible to generate without root.

17.5 OUTPUT

Many of the outputs from this tool are deliberately rounded to show their magnitude but not the exact detail. This is called fuzzy-rounding. The idea is that it doesn't matter whether a particular counter is 918 or 921; such a small variation is insignificant, and only makes the output hard to compare to other servers. Fuzzy-rounding rounds in larger increments as the input grows. It begins by rounding to the nearest 5, then the nearest 10, nearest 25, and then repeats by a factor of 10 larger (50, 100, 250), and so on, as the input grows.

The following is a simple report generated from a CentOS virtual machine, broken into sections with commentary following each section. Some long lines are reformatted for clarity when reading this documentation as a manual page in a terminal.

```
# MariaDB System Summary Report ##############################
        Date | 2012-03-30 00:58:07 UTC (local TZ: EDT -0400)
    Hostname | localhost.localdomain
      Uptime | 20:58:06 up 1 day, 20 min, 1 user,
               load average: 0.14, 0.18, 0.18
      System | innotek GmbH; VirtualBox; v1.2 ()
 Service Tag | 0
    Platform | Linux
     Release | CentOS release 5.5 (Final)
      Kernel | 2.6.18-194.el5
Architecture | CPU = 32-bit, OS = 32-bit
   Threading | NPTL 2.5
    Compiler | GNU CC version 4.1.2 20080704 (Red Hat 4.1.2-48).
     SELinux | Enforcing
 Virtualized | VirtualBox
```

This section shows the current date and time, and a synopsis of the server and operating system.

```
# Processor ##################################################
  Processors | physical = 1, cores = 0, virtual = 1, hyperthreading = no
      Speeds | 1x2510.626
      Models | 1xIntel(R) Core(TM) i5-2400S CPU @ 2.50GHz
      Caches | 1x6144 KB
```

This section is derived from */proc/cpuinfo*.

```
# Memory #####################################################
       Total | 503.2M
        Free | 29.0M
        Used | physical = 474.2M, swap allocated = 1.0M,
               swap used = 16.0k, virtual = 474.3M
     Buffers | 33.9M
      Caches | 262.6M
       Dirty | 396 kB
     UsedRSS | 201.9M
  Swappiness | 60
 DirtyPolicy | 40, 10
 Locator  Size  Speed    Form Factor  Type    Type Detail
 =======  ====  =====    ===========  ====    ===========
```

Information about memory is gathered from `free`. The Used statistic is the total of the rss sizes displayed by `ps`. The Dirty statistic for the cached value comes from */proc/meminfo*. On Linux, the swappiness settings are gathered from `sysctl`. The final portion of this section is a table of the DIMMs, which comes from `dmidecode`. In this example there is no output.

```
# Mounted Filesystems ########################################
  Filesystem                         Size Used Type  Opts Mountpoint
  /dev/mapper/VolGroup00-LogVol00     15G  17% ext3  rw   /
  /dev/sda1                           99M  13% ext3  rw   /boot
  tmpfs                              252M   0% tmpfs rw   /dev/shm
```

The mounted filesystem section is a combination of information from `mount` and `df`. This section is skipped if you disable *--summarize-mounts*.

```
# Disk Schedulers And Queue Size #############################
        dm-0 | UNREADABLE
        dm-1 | UNREADABLE
         hdc | [cfq] 128
         md0 | UNREADABLE
         sda | [cfq] 128
```

The disk scheduler information is extracted from the */sys* filesystem in Linux.

```
# Disk Partioning ############################################
Device       Type      Start        End               Size
============ ==== ========== ========== ==================
/dev/sda     Disk                              17179869184
/dev/sda1    Part          1         13           98703360
/dev/sda2    Part         14       2088        17059230720
```

Information about disk partitioning comes from `fdisk -l`.

```
# Kernel Inode State #########################################
dentry-state | 10697 8559  45 0  0  0
     file-nr | 960   0  50539
    inode-nr | 14059 8139
```

These lines are from the files of the same name in the */proc/sys/fs* directory on Linux. Read the `proc` man page to learn about the meaning of these files on your system.

```
# LVM Volumes ################################################
LV       VG         Attr   LSize   Origin Snap% Move Log Copy% Convert
LogVol00 VolGroup00 -wi-ao 269.00G
LogVol01 VolGroup00 -wi-ao   9.75G
```

This section shows the output of `lvs`.

```
# RAID Controller ############################################
  Controller | No RAID controller detected
```

The tool can detect a variety of RAID controllers by examining `lspci` and `dmesg` information. If the controller software is installed on the system, in many cases it is able to execute status commands and show a summary of the RAID controller's status and configuration. If your system is not supported, please file a bug report.

```
# Network Config #############################################
  Controller | Intel Corporation 82540EM Gigabit Ethernet Controller
 FIN Timeout | 60
  Port Range | 61000
```

The network controllers attached to the system are detected from `lspci`. The TCP/IP protocol configuration parameters are extracted from `sysctl`. You can skip this section by disabling the *--summarize-network* option.

```
# Interface Statistics ########################################
interface rx_bytes rx_packets rx_errors tx_bytes tx_packets tx_errors
========= ======== ========== ========= ======== ========== =========
lo        60000000      12500         0 60000000      12500         0
eth0      15000000      80000         0  1500000      10000         0
sit0             0          0         0        0          0         0
```

Interface statistics are gathered from `ip -s link` and are fuzzy-rounded. The columns are received and transmitted bytes, packets, and errors. You can skip this section by disabling the *--summarize-network* option.

```
# Network Connections #########################################
  Connections from remote IP addresses
    127.0.0.1           2
  Connections to local IP addresses
    127.0.0.1           2
  Connections to top 10 local ports
    38346               1
    60875               1
  States of connections
    ESTABLISHED         5
    LISTEN              8
```

This section shows a summary of network connections, retrieved from `netstat` and "fuzzy-rounded" to make them easier to compare when the numbers grow large. There are two sub-sections showing how many connections there are per origin and destination IP address, and a sub-section showing the count of ports in use. The section ends with the count of the network connections' states. You can skip this section by disabling the *--summarize-network* option.

```
# Top Processes ###############################################
  PID USER  PR  NI  VIRT  RES  SHR S %CPU %MEM    TIME+  COMMAND
    1 root  15   0  2072  628  540 S  0.0  0.1   0:02.55 init
    2 root  RT  -5     0    0    0 S  0.0  0.0   0:00.00 migration/0
    3 root  34  19     0    0    0 S  0.0  0.0   0:00.03 ksoftirqd/0
    4 root  RT  -5     0    0    0 S  0.0  0.0   0:00.00 watchdog/0
    5 root  10  -5     0    0    0 S  0.0  0.0   0:00.97 events/0
    6 root  10  -5     0    0    0 S  0.0  0.0   0:00.00 khelper
    7 root  10  -5     0    0    0 S  0.0  0.0   0:00.00 kthread
   10 root  10  -5     0    0    0 S  0.0  0.0   0:00.13 kblockd/0
   11 root  20  -5     0    0    0 S  0.0  0.0   0:00.00 kacpid
# Notable Processes ###########################################
  PID    OOM    COMMAND
 2028    +0    sshd
```

This section shows the first few lines of `top` so that you can see what processes are actively using CPU time. The notable processes include the SSH daemon and any process whose out-of-memory-killer priority is set to 17. You can skip this section by disabling the *--summarize-processes* option.

```
# Simplified and fuzzy rounded vmstat (wait please) ##########
  procs  ---swap-- -----io---- ---system---- --------cpu--------
   r  b    si   so    bi    bo     ir     cs  us  sy  il  wa  st
   2  0     0    0     3    15     30    125   0   0  99   0   0
   0  0     0    0     0     0   1250    800   6  10  84   0   0
   0  0     0    0     0     0   1000    125   0   0 100   0   0
   0  0     0    0     0     0   1000    125   0   0 100   0   0
   0  0     0    0     0   450   1000    125   0   1  88  11   0
# The End ####################################################
```

This section is a trimmed-down sample of `vmstat 1 5`, so you can see the general status of the system at present. The values in the table are fuzzy-rounded, except for the CPU columns. You can skip this section by disabling the *--summarize-processes* option.

17.6 OPTIONS

--config
type: string

Read this comma-separated list of config files. If specified, this must be the first option on the command line.

--help
Print help and exit.

--read-samples
type: string

Create a report from the files in this directory.

--save-samples
type: string

Save the collected data in this directory.

--sleep
type: int; default: 5

How long to sleep when gathering samples from vmstat.

--summarize-mounts
default: yes; negatable: yes

Report on mounted filesystems and disk usage.

--summarize-network
default: yes; negatable: yes

Report on network controllers and configuration.

--summarize-processes
default: yes; negatable: yes

Report on top processes and `vmstat` output.

--version
Print tool's version and exit.

17.7 ENVIRONMENT

This tool does not use any environment variables.

17.8 SYSTEM REQUIREMENTS

This tool requires the Bourne shell (*/bin/sh*).

17.9 AUTHORS

Cole Busby, Baron Schwartz, Kevin van Zonneveld, and Brian Fraser

17.10 ABOUT THIS MARIADB TOOL

This tool is part of MariaDB client tools. This MariaDB Tool was forked from Percona Toolkit's **`mariadb-system-summary`** in August, 2019. Percona Toolkit was forked from two projects in June, 2011: Maatkit and Aspersa. Those projects were created by Baron Schwartz and primarily developed by him and Daniel Nichter.

17.11 COPYRIGHT, LICENSE, AND WARRANTY

17.12 VERSION

`mariadb-system-summary` 6.0.0rc

Part III

Configuration

Part IV

Miscellaneous

CHAPTER

EIGHTEEN

V6.0.0-RC RELEASED 2021-11-04

CHAPTER

NINETEEN

V6.0.0RC RELEASED 2021-11-04

CHAPTER

TWENTY

V6.0.0-RC RELEASED 2021-11-04

20.1 Changelog

20.2 Changelog

20.3 Changelog

INDEX

Symbols

--add
  mariadb-parted command line option, 79
--all-databases
  mariadb-database-summary command line option, 42
  mariadb-summary command line option, 154
--all-structs
  mariadb-index-checker command line option, 46
--alter
  mariadb-schema-change command line option, 111
--alter-foreign-keys-method
  mariadb-schema-change command line option, 112
--analyze
  mariadb-archiver command line option, 15
--any-busy-time
  mariadb-kill command line option, 71
--archive
  mariadb-parted command line option, 79
--archive-database
  mariadb-parted command line option, 79
--archive-path
  mariadb-parted command line option, 79
--ascend-first
  mariadb-archiver command line option, 15
--ask-pass
  mariadb-archiver command line option, 16
  mariadb-database-summary command line option, 42
  mariadb-index-checker command line option, 46
  mariadb-kill command line option, 63
  mariadb-query-digest command line option, 87
  mariadb-schema-change command line option, 113
  mariadb-stat command line option, 137
  mariadb-summary command line option, 154
--attribute-aliases
  mariadb-query-digest command line option, 87
--attribute-value-limit
  mariadb-query-digest command line option, 87
--binary
  mariadb-stacktrace command line option, 132
--buffer
  mariadb-archiver command line option, 16
--bulk-delete
  mariadb-archiver command line option, 16
--bulk-insert
  mariadb-archiver command line option, 16
--busy-time
  mariadb-kill command line option, 68
--channel
  mariadb-archiver command line option, 16
  mariadb-schema-change command line option, 113
--charset
  mariadb-archiver command line option, 17
  mariadb-index-checker command line option, 46
  mariadb-kill command line option, 63
  mariadb-query-digest command line option, 87

mariadb-schema-change command line option, 113
--check-interval
mariadb-archiver command line option, 17
mariadb-schema-change command line option, 114
--check-slave-lag
mariadb-archiver command line option, 17
mariadb-schema-change command line option, 115
--chunk-index
mariadb-schema-change command line option, 115
--chunk-index-columns
mariadb-schema-change command line option, 115
--chunk-size
mariadb-schema-change command line option, 115
--chunk-size-limit
mariadb-schema-change command line option, 115
--chunk-time
mariadb-schema-change command line option, 115
--collect
mariadb-stat command line option, 137
--collect-gdb
mariadb-stat command line option, 137
--collect-oprofile
mariadb-stat command line option, 137
--collect-strace
mariadb-stat command line option, 138
--collect-tcpdump
mariadb-stat command line option, 138
--columns
mariadb-archiver command line option, 17
--columns-regex
mariadb-iostat command line option, 57
--commit-each
mariadb-archiver command line option, 17
--config
mariadb-archiver command line option, 18
mariadb-database-summary command line option, 42
mariadb-index-checker command line option, 47
mariadb-iostat command line option, 57
mariadb-kill command line option, 63
mariadb-query-digest command line option, 87
mariadb-schema-change command line option, 116
mariadb-stat command line option, 138
mariadb-summary command line option, 154
mariadb-system-summary command line option, 161
--create-log-table
mariadb-kill command line option, 64
--critical-load
mariadb-schema-change command line option, 116
--cycles
mariadb-stat command line option, 138
--daemonize
mariadb-kill command line option, 64
mariadb-query-digest command line option, 87
mariadb-stat command line option, 138
--data-dir
mariadb-schema-change command line option, 116
--database
mariadb-archiver command line option, 18
mariadb-kill command line option, 64
mariadb-query-digest command line option, 87
mariadb-schema-change command line option, 116
--databases
mariadb-database-summary command line option, 42
mariadb-index-checker command line option, 47
mariadb-summary command line option, 154
--default-engine
mariadb-schema-change command line option, 116
--defaults-file

mariadb-database-summary command line option, 42
mariadb-index-checker command line option, 47
mariadb-kill command line option, 64
mariadb-query-digest command line option, 88
mariadb-schema-change command line option, 116
mariadb-stat command line option, 138
mariadb-summary command line option, 154

--delayed-insert
mariadb-archiver command line option, 18

--dest
mariadb-archiver command line option, 18
mariadb-stat command line option, 138

--devices-regex
mariadb-iostat command line option, 57

--disk-bytes-free
mariadb-stat command line option, 138

--disk-pct-free
mariadb-stat command line option, 138

--drop
mariadb-parted command line option, 79

--dry-run
mariadb-archiver command line option, 18
mariadb-schema-change command line option, 117

--each-busy-time
mariadb-kill command line option, 71

--email-activity
mariadb-parted command line option, 78

--embedded-attributes
mariadb-query-digest command line option, 88

--engines
mariadb-index-checker command line option, 47

--execute
mariadb-schema-change command line option, 117

--execute-command
mariadb-kill command line option, 71

--expected-range
mariadb-query-digest command line option, 88

--explain
mariadb-query-digest command line option, 88

--file
mariadb-archiver command line option, 18

--filter
mariadb-kill command line option, 64
mariadb-query-digest command line option, 88

--for-update
mariadb-archiver command line option, 19

--force
mariadb-schema-change command line option, 118

--force-concat-enums
mariadb-schema-change command line option, 118

--function
mariadb-stat command line option, 138

--group-by
mariadb-iostat command line option, 57
mariadb-kill command line option, 64
mariadb-query-digest command line option, 90

--header
mariadb-archiver command line option, 19

--headers
mariadb-iostat command line option, 57

--help
mariadb-align-output command line option, 10
mariadb-archiver command line option, 19
mariadb-database-summary command line option, 42
mariadb-index-checker command line option, 47
mariadb-iostat command line option, 57
mariadb-kill command line option, 65
mariadb-query-digest command line option, 90
mariadb-schema-change command line option, 118

mariadb-stacktrace command line option, 132
mariadb-stat command line option, 139
mariadb-stat-browser command line option, 146
mariadb-status-diff command line option, 150
mariadb-summary command line option, 154
mariadb-system-summary command line option, 161

--high-priority-select
mariadb-archiver command line option, 19

--history
mariadb-query-digest command line option, 90

--host
mariadb-archiver command line option, 19
mariadb-database-summary command line option, 42
mariadb-index-checker command line option, 47
mariadb-kill command line option, 65
mariadb-query-digest command line option, 93
mariadb-schema-change command line option, 118
mariadb-stat command line option, 139
mariadb-summary command line option, 154

--idle-time
mariadb-kill command line option, 68

--ignore
mariadb-archiver command line option, 19

--ignore-attributes
mariadb-query-digest command line option, 93

--ignore-command
mariadb-kill command line option, 68

--ignore-databases
mariadb-index-checker command line option, 47

--ignore-db
mariadb-kill command line option, 68

--ignore-engines
mariadb-index-checker command line option, 47

--ignore-host
mariadb-kill command line option, 68

--ignore-info
mariadb-kill command line option, 68

--ignore-order
mariadb-index-checker command line option, 47

--ignore-state
mariadb-kill command line option, 69

--ignore-tables
mariadb-index-checker command line option, 47

--ignore-user
mariadb-kill command line option, 69

--inherit-attributes
mariadb-query-digest command line option, 93

--interval
mariadb-iostat command line option, 57
mariadb-kill command line option, 65
mariadb-query-digest command line option, 93
mariadb-stacktrace command line option, 132
mariadb-stat command line option, 139

--iterations
mariadb-iostat command line option, 58
mariadb-query-digest command line option, 93
mariadb-stacktrace command line option, 132
mariadb-stat command line option, 139

--key-types
mariadb-index-checker command line option, 47

--kill
mariadb-kill command line option, 71

--kill-busy-commands
mariadb-kill command line option, 72

--kill-query
mariadb-kill command line option, 72

--limit
mariadb-archiver command line option, 19
mariadb-parted command line option, 78
mariadb-query-digest command line option, 93

--lines
mariadb-stacktrace command line option, 132

--list-encrypted-tables

mariadb-database-summary command line option, 42
--local
mariadb-archiver command line option, 19
--log
mariadb-kill command line option, 65
mariadb-query-digest command line option, 93
mariadb-stat command line option, 140
--log-dsn
mariadb-kill command line option, 65
--low-priority-delete
mariadb-archiver command line option, 20
--low-priority-insert
mariadb-archiver command line option, 20
--mariadb-only
mariadb-stat command line option, 141
--match
mariadb-stat command line option, 140
--match-all
mariadb-kill command line option, 69
--match-command
mariadb-kill command line option, 69
--match-db
mariadb-kill command line option, 69
--match-host
mariadb-kill command line option, 69
--match-info
mariadb-kill command line option, 70
--match-state
mariadb-kill command line option, 70
--match-user
mariadb-kill command line option, 70
--max-flow-ctl
mariadb-archiver command line option, 20
mariadb-schema-change command line option, 118
--max-hostname-length
mariadb-query-digest command line option, 94
--max-lag
mariadb-archiver command line option, 20
mariadb-schema-change command line option, 118
--max-line-length
mariadb-query-digest command line option, 94
--max-load
mariadb-schema-change command line option, 118
--new-table-name
mariadb-schema-change command line option, 120
--no-ascend
mariadb-archiver command line option, 20
--no-delete
mariadb-archiver command line option, 20
--notify-by-email
mariadb-stat command line option, 140
--null-to-not-null
mariadb-schema-change command line option, 120
--only-same-schema-fks
mariadb-schema-change command line option, 120
--optimize
mariadb-archiver command line option, 21
--order-by
mariadb-query-digest command line option, 94
--outliers
mariadb-query-digest command line option, 94
--output
mariadb-query-digest command line option, 95
--output-format
mariadb-archiver command line option, 21
--partcol-integer
mariadb-parted command line option, 78
--password
mariadb-archiver command line option, 21
mariadb-database-summary command line option, 42
mariadb-index-checker command line option, 47
mariadb-kill command line option, 66
mariadb-query-digest command line option, 95
mariadb-schema-change command line option, 120

mariadb-stat command line option, 140
mariadb-summary command line option, 154
--pause-file
mariadb-schema-change command line option, 120
--pid
mariadb-archiver command line option, 21
mariadb-index-checker command line option, 47
mariadb-kill command line option, 66
mariadb-query-digest command line option, 95
mariadb-schema-change command line option, 120
mariadb-stacktrace command line option, 132
mariadb-stat command line option, 140
--plugin
mariadb-archiver command line option, 21
mariadb-schema-change command line option, 120
mariadb-stat command line option, 140
--port
mariadb-archiver command line option, 21
mariadb-database-summary command line option, 43
mariadb-index-checker command line option, 48
mariadb-kill command line option, 66
mariadb-query-digest command line option, 95
mariadb-schema-change command line option, 120
mariadb-stat command line option, 141
mariadb-summary command line option, 154
--prefix
mariadb-stat command line option, 141
--preserve-embedded-numbers
mariadb-query-digest command line option, 95
--preserve-triggers
mariadb-schema-change command line option, 119
--primary-key-only
mariadb-archiver command line option, 21
--print
mariadb-kill command line option, 72
mariadb-schema-change command line option, 120
--processlist
mariadb-query-digest command line option, 95
--progress
mariadb-archiver command line option, 22
mariadb-query-digest command line option, 95
mariadb-schema-change command line option, 120
--purge
mariadb-archiver command line option, 22
--query-count
mariadb-kill command line option, 71
--query-id
mariadb-kill command line option, 66
--quick-delete
mariadb-archiver command line option, 22
--quiet
mariadb-archiver command line option, 22
mariadb-schema-change command line option, 120
--rds
mariadb-kill command line option, 66
--read-samples
mariadb-database-summary command line option, 43
mariadb-summary command line option, 154
mariadb-system-summary command line option, 161
--read-timeout
mariadb-query-digest command line option, 95
--recurse
mariadb-schema-change command line option, 121
--recursion-method
mariadb-schema-change command line option, 121
--relative
mariadb-status-diff command line option, 150
--remove-data-dir

mariadb-schema-change command line option, 116
--replace
mariadb-archiver command line option, 22
--replication-threads
mariadb-kill command line option, 70
--report-all
mariadb-query-digest command line option, 96
--report-format
mariadb-query-digest command line option, 96
--report-histogram
mariadb-query-digest command line option, 96
--resume
mariadb-query-digest command line option, 96
--retention-count
mariadb-stat command line option, 141
--retention-size
mariadb-stat command line option, 141
--retention-time
mariadb-stat command line option, 141
--retries
mariadb-archiver command line option, 22
--review
mariadb-query-digest command line option, 97
--run-time
mariadb-archiver command line option, 22
mariadb-kill command line option, 66
mariadb-query-digest command line option, 97
mariadb-stat command line option, 141
--run-time-mode
mariadb-query-digest command line option, 97
--sample
mariadb-query-digest command line option, 98
--sample-time
mariadb-iostat command line option, 58
--save-samples
mariadb-database-summary command line option, 43
mariadb-iostat command line option, 58
mariadb-stacktrace command line option, 132
mariadb-summary command line option, 154
mariadb-system-summary command line option, 161
--sentinel
mariadb-archiver command line option, 23
mariadb-kill command line option, 66
--set-vars
mariadb-archiver command line option, 23
mariadb-index-checker command line option, 48
mariadb-kill command line option, 67
mariadb-query-digest command line option, 99
mariadb-schema-change command line option, 122
--share-lock
mariadb-archiver command line option, 23
--show-all
mariadb-query-digest command line option, 99
--show-inactive
mariadb-iostat command line option, 58
--show-timestamps
mariadb-iostat command line option, 58
--since
mariadb-query-digest command line option, 99
--skip-check-slave-lag
mariadb-schema-change command line option, 121
--skip-foreign-key-checks
mariadb-archiver command line option, 23
--slave-password
mariadb-archiver command line option, 23
mariadb-kill command line option, 66
mariadb-query-digest command line option, 98
mariadb-schema-change command line option, 122
--slave-user
mariadb-archiver command line option, 23

mariadb-kill command line option, 66
mariadb-query-digest command line option, 98
mariadb-schema-change command line option, 122

--sleep
mariadb-archiver command line option, 23
mariadb-database-summary command line option, 43
mariadb-schema-change command line option, 122
mariadb-stat command line option, 142
mariadb-summary command line option, 154
mariadb-system-summary command line option, 161

--sleep-coef
mariadb-archiver command line option, 23

--sleep-collect
mariadb-stat command line option, 142

--socket
mariadb-archiver command line option, 24
mariadb-database-summary command line option, 43
mariadb-index-checker command line option, 48
mariadb-kill command line option, 67
mariadb-query-digest command line option, 99
mariadb-schema-change command line option, 122
mariadb-stat command line option, 142
mariadb-summary command line option, 154

--source
mariadb-archiver command line option, 24

--stalk
mariadb-stat command line option, 142

--statistics
mariadb-archiver command line option, 24
mariadb-schema-change command line option, 122

--stop
mariadb-archiver command line option, 25

mariadb-kill command line option, 67

--summarize-mounts
mariadb-summary command line option, 155
mariadb-system-summary command line option, 161

--summarize-network
mariadb-summary command line option, 155
mariadb-system-summary command line option, 161

--summarize-processes
mariadb-summary command line option, 155
mariadb-system-summary command line option, 161

--tables
mariadb-index-checker command line option, 48

--test-matching
mariadb-kill command line option, 70

--threshold
mariadb-stat command line option, 142

--timeline
mariadb-query-digest command line option, 99

--tries
mariadb-schema-change command line option, 123

--txn-size
mariadb-archiver command line option, 25

--type
mariadb-query-digest command line option, 100

--until
mariadb-query-digest command line option, 101

--use-hours
mariadb-parted command line option, 78

--user
mariadb-archiver command line option, 25
mariadb-database-summary command line option, 43
mariadb-index-checker command line option, 48
mariadb-kill command line option, 67
mariadb-query-digest command line option, 102
mariadb-schema-change command line option, 123

mariadb-stat command line option, 142
mariadb-summary command line option, 155
--variable
mariadb-stat command line option, 142
--variations
mariadb-query-digest command line option, 102
--verbose
mariadb-index-checker command line option, 48
mariadb-kill command line option, 71
mariadb-stat command line option, 142
--version
mariadb-align-output command line option, 10
mariadb-archiver command line option, 25
mariadb-database-summary command line option, 43
mariadb-index-checker command line option, 48
mariadb-iostat command line option, 58
mariadb-kill command line option, 67
mariadb-query-digest command line option, 102
mariadb-schema-change command line option, 123
mariadb-stacktrace command line option, 132
mariadb-stat command line option, 143
mariadb-stat-browser command line option, 146
mariadb-status-diff command line option, 150
mariadb-summary command line option, 155
mariadb-system-summary command line option, 161
--victims
mariadb-kill command line option, 67
--wait-after-kill
mariadb-kill command line option, 67
--wait-before-kill
mariadb-kill command line option, 68
--watch-server
mariadb-query-digest command line option, 102
--where
mariadb-archiver command line option, 26
--why-quit
mariadb-archiver command line option, 26
--[no]analyze-before-swap
mariadb-schema-change command line option, 113
--[no]bulk-delete-limit
mariadb-archiver command line option, 16
--[no]check-alter
mariadb-schema-change command line option, 113
--[no]check-charset
mariadb-archiver command line option, 17
--[no]check-columns
mariadb-archiver command line option, 17
--[no]check-foreign-keys
mariadb-schema-change command line option, 114
--[no]check-plan
mariadb-schema-change command line option, 114
--[no]check-replication-filters
mariadb-schema-change command line option, 114
--[no]check-unique-key-change
mariadb-schema-change command line option, 117
--[no]clustered
mariadb-index-checker command line option, 46
--[no]continue-on-error
mariadb-query-digest command line option, 87
--[no]create-history-table
mariadb-query-digest command line option, 87
--[no]create-review-table
mariadb-query-digest command line option, 87
--[no]drop-new-table
mariadb-schema-change command line option, 116
--[no]drop-old-table
mariadb-schema-change command line option, 117
--[no]drop-triggers
mariadb-schema-change command line option, 117
--[no]ignore-self

mariadb-kill command line option, 68
--[no]report
mariadb-query-digest command line option, 96
--[no]safe-auto-increment
mariadb-archiver command line option, 22
--[no]sql
mariadb-index-checker command line option, 48
--[no]strip-comments
mariadb-kill command line option, 67
--[no]summary
mariadb-index-checker command line option, 48
--[no]swap-tables
mariadb-schema-change command line option, 122
--[no]vertical-format
mariadb-query-digest command line option, 102

M

mariadb-align-output command line option
--help, 10
--version, 10
mariadb-archiver command line option
--analyze, 15
--ascend-first, 15
--ask-pass, 16
--buffer, 16
--bulk-delete, 16
--bulk-insert, 16
--channel, 16
--charset, 17
--check-interval, 17
--check-slave-lag, 17
--columns, 17
--commit-each, 17
--config, 18
--database, 18
--delayed-insert, 18
--dest, 18
--dry-run, 18
--file, 18
--for-update, 19
--header, 19
--help, 19
--high-priority-select, 19
--host, 19
--ignore, 19
--limit, 19
--local, 19
--low-priority-delete, 20
--low-priority-insert, 20
--max-flow-ctl, 20
--max-lag, 20
--no-ascend, 20
--no-delete, 20
--optimize, 21
--output-format, 21
--password, 21
--pid, 21
--plugin, 21
--port, 21
--primary-key-only, 21
--progress, 22
--purge, 22
--quick-delete, 22
--quiet, 22
--replace, 22
--retries, 22
--run-time, 22
--sentinel, 23
--set-vars, 23
--share-lock, 23
--skip-foreign-key-checks, 23
--slave-password, 23
--slave-user, 23
--sleep, 23
--sleep-coef, 23
--socket, 24
--source, 24
--statistics, 24
--stop, 25
--txn-size, 25
--user, 25
--version, 25
--where, 26
--why-quit, 26
--[no]bulk-delete-limit, 16
--[no]check-charset, 17
--[no]check-columns, 17
--[no]safe-auto-increment, 22
mariadb-database-summary command line option
--all-databases, 42
--ask-pass, 42
--config, 42
--databases, 42
--defaults-file, 42
--help, 42
--host, 42
--list-encrypted-tables, 42
--password, 42
--port, 43
--read-samples, 43

--save-samples, 43
--sleep, 43
--socket, 43
--user, 43
--version, 43

mariadb-index-checker command line option
--all-structs, 46
--ask-pass, 46
--charset, 46
--config, 47
--databases, 47
--defaults-file, 47
--engines, 47
--help, 47
--host, 47
--ignore-databases, 47
--ignore-engines, 47
--ignore-order, 47
--ignore-tables, 47
--key-types, 47
--password, 47
--pid, 47
--port, 48
--set-vars, 48
--socket, 48
--tables, 48
--user, 48
--verbose, 48
--version, 48
--[no]clustered, 46
--[no]sql, 48
--[no]summary, 48

mariadb-iostat command line option
--columns-regex, 57
--config, 57
--devices-regex, 57
--group-by, 57
--headers, 57
--help, 57
--interval, 57
--iterations, 58
--sample-time, 58
--save-samples, 58
--show-inactive, 58
--show-timestamps, 58
--version, 58

mariadb-kill command line option
--any-busy-time, 71
--ask-pass, 63
--busy-time, 68
--charset, 63
--config, 63
--create-log-table, 64
--daemonize, 64
--database, 64
--defaults-file, 64
--each-busy-time, 71
--execute-command, 71
--filter, 64
--group-by, 64
--help, 65
--host, 65
--idle-time, 68
--ignore-command, 68
--ignore-db, 68
--ignore-host, 68
--ignore-info, 68
--ignore-state, 69
--ignore-user, 69
--interval, 65
--kill, 71
--kill-busy-commands, 72
--kill-query, 72
--log, 65
--log-dsn, 65
--match-all, 69
--match-command, 69
--match-db, 69
--match-host, 69
--match-info, 70
--match-state, 70
--match-user, 70
--password, 66
--pid, 66
--port, 66
--print, 72
--query-count, 71
--query-id, 66
--rds, 66
--replication-threads, 70
--run-time, 66
--sentinel, 66
--set-vars, 67
--slave-password, 66
--slave-user, 66
--socket, 67
--stop, 67
--test-matching, 70
--user, 67
--verbose, 71
--version, 67
--victims, 67
--wait-after-kill, 67
--wait-before-kill, 68
--[no]ignore-self, 68
--[no]strip-comments, 67

mariadb-parted command line option

--add, 79
--archive, 79
--archive-database, 79
--archive-path, 79
--drop, 79
--email-activity, 78
--limit, 78
--partcol-integer, 78
--use-hours, 78
mariadb-query-digest command line option
--ask-pass, 87
--attribute-aliases, 87
--attribute-value-limit, 87
--charset, 87
--config, 87
--daemonize, 87
--database, 87
--defaults-file, 88
--embedded-attributes, 88
--expected-range, 88
--explain, 88
--filter, 88
--group-by, 90
--help, 90
--history, 90
--host, 93
--ignore-attributes, 93
--inherit-attributes, 93
--interval, 93
--iterations, 93
--limit, 93
--log, 93
--max-hostname-length, 94
--max-line-length, 94
--order-by, 94
--outliers, 94
--output, 95
--password, 95
--pid, 95
--port, 95
--preserve-embedded-numbers, 95
--processlist, 95
--progress, 95
--read-timeout, 95
--report-all, 96
--report-format, 96
--report-histogram, 96
--resume, 96
--review, 97
--run-time, 97
--run-time-mode, 97
--sample, 98
--set-vars, 99
--show-all, 99
--since, 99
--slave-password, 98
--slave-user, 98
--socket, 99
--timeline, 99
--type, 100
--until, 101
--user, 102
--variations, 102
--version, 102
--watch-server, 102
--[no]continue-on-error, 87
--[no]create-history-table, 87
--[no]create-review-table, 87
--[no]report, 96
--[no]vertical-format, 102
mariadb-schema-change command line option
--alter, 111
--alter-foreign-keys-method, 112
--ask-pass, 113
--channel, 113
--charset, 113
--check-interval, 114
--check-slave-lag, 115
--chunk-index, 115
--chunk-index-columns, 115
--chunk-size, 115
--chunk-size-limit, 115
--chunk-time, 115
--config, 116
--critical-load, 116
--data-dir, 116
--database, 116
--default-engine, 116
--defaults-file, 116
--dry-run, 117
--execute, 117
--force, 118
--force-concat-enums, 118
--help, 118
--host, 118
--max-flow-ctl, 118
--max-lag, 118
--max-load, 118
--new-table-name, 120
--null-to-not-null, 120
--only-same-schema-fks, 120
--password, 120
--pause-file, 120
--pid, 120
--plugin, 120
--port, 120

--preserve-triggers, 119
--print, 120
--progress, 120
--quiet, 120
--recurse, 121
--recursion-method, 121
--remove-data-dir, 116
--set-vars, 122
--skip-check-slave-lag, 121
--slave-password, 122
--slave-user, 122
--sleep, 122
--socket, 122
--statistics, 122
--tries, 123
--user, 123
--version, 123
--[no]analyze-before-swap, 113
--[no]check-alter, 113
--[no]check-foreign-keys, 114
--[no]check-plan, 114
--[no]check-replication-filters, 114
--[no]check-unique-key-change, 117
--[no]drop-new-table, 116
--[no]drop-old-table, 117
--[no]drop-triggers, 117
--[no]swap-tables, 122

mariadb-stacktrace command line option
--binary, 132
--help, 132
--interval, 132
--iterations, 132
--lines, 132
--pid, 132
--save-samples, 132
--version, 132

mariadb-stat command line option
--ask-pass, 137
--collect, 137
--collect-gdb, 137
--collect-oprofile, 137
--collect-strace, 138
--collect-tcpdump, 138
--config, 138
--cycles, 138
--daemonize, 138
--defaults-file, 138
--dest, 138
--disk-bytes-free, 138
--disk-pct-free, 138
--function, 138
--help, 139
--host, 139
--interval, 139
--iterations, 139
--log, 140
--mariadb-only, 141
--match, 140
--notify-by-email, 140
--password, 140
--pid, 140
--plugin, 140
--port, 141
--prefix, 141
--retention-count, 141
--retention-size, 141
--retention-time, 141
--run-time, 141
--sleep, 142
--sleep-collect, 142
--socket, 142
--stalk, 142
--threshold, 142
--user, 142
--variable, 142
--verbose, 142
--version, 143

mariadb-stat-browser command line option
--help, 146
--version, 146

mariadb-status-diff command line option
--help, 150
--relative, 150
--version, 150

mariadb-summary command line option
--all-databases, 154
--ask-pass, 154
--config, 154
--databases, 154
--defaults-file, 154
--help, 154
--host, 154
--password, 154
--port, 154
--read-samples, 154
--save-samples, 154
--sleep, 154
--socket, 154
--summarize-mounts, 155
--summarize-network, 155
--summarize-processes, 155
--user, 155
--version, 155

mariadb-system-summary command line option
--config, 161

--help, 161
--read-samples, 161
--save-samples, 161
--sleep, 161
--summarize-mounts, 161
--summarize-network, 161
--summarize-processes, 161
--version, 161