PERCONA

www.percona.com

Percona Toolkit Documentation

Release 3.0.4

Aug 02, 2017

CONTENTS

Percona Toolkit is a collection of advanced command-line tools used by Percona) support staff to perform a variety of MySQL, MongoDB, 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.

Percona Toolkit was derived from Maatkit and Aspersa, two of the best-known toolkits for MySQL server administration. It is developed and supported by Percona. For more information and other free, open-source software developed by Percona, visit http://www.percona.com/software/.

Part I

Getting Percona Toolkit

CHAPTER

ONE

INSTALLING PERCONA TOOLKIT

Percona 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: Percona Toolkit 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 Percona software from official repositories:

1. Configure Percona repositories as described in Percona Software Repositories Documentation.
2. Install Percona Toolkit using the corresponding package manager:
   - For Debian or Ubuntu:

     ```
     sudo apt-get install percona-toolkit
     ```

   - For RHEL or CentOS:

     ```
     sudo yum install percona-toolkit
     ```

Alternative Install Methods

You can also download the packages from the Percona web site and install it using tools like `dpkg` and `rpm`, depending on your system. For example, to download the package for Debian 8 ("jessie"), run the following:

```
wget https://www.percona.com/downloads/percona-toolkit/3.0.3/binary/debian/jessie/x86_
↪64/percona-toolkit_3.0.3-1.jessie_amd64.deb
```

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

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

```
wget percona.com/get/pt-summary
```

Part II

Tools

CHAPTER

TWO

PT-ALIGN

NAME

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

SYNOPSIS

Usage

```
pt-align [FILES]
```

pt-align 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 **pt-align** reprints the output as,

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

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

DESCRIPTION

pt-align 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.

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.

ENVIRONMENT

This tool does not use any environment variables.

SYSTEM REQUIREMENTS

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

BUGS

For a list of known bugs, see http://www.percona.com/bugs/pt-align.

Please report bugs at https://bugs.launchpad.net/percona-toolkit. Include the following information in your bug report:

- Complete command-line used to run the tool
- Tool *--version*
- MySQL version of all servers involved
- Output from the tool including STDERR
- Input files (log/dump/config files, etc.)

If possible, include debugging output by running the tool with PTDEBUG; see "ENVIRONMENT".

DOWNLOADING

Visit http://www.percona.com/software/percona-toolkit/ to download the latest release of Percona Toolkit. Or, get the latest release from the command line:

```
wget percona.com/get/percona-toolkit.tar.gz

wget percona.com/get/percona-toolkit.rpm

wget percona.com/get/percona-toolkit.deb
```

You can also get individual tools from the latest release:

```
wget percona.com/get/TOOL
```

Replace `TOOL` with the name of any tool.

AUTHORS

Baron Schwartz, Brian Fraser, and Daniel Nichter

ABOUT PERCONA TOOLKIT

This tool is part of Percona Toolkit, a collection of advanced command-line tools for MySQL developed by Percona. 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. Visit http://www.percona.com/software/ to learn about other free, open-source software from Percona.

COPYRIGHT, LICENSE, AND WARRANTY

VERSION

pt-align 3.0.4

CHAPTER

THREE

PT-ARCHIVER

NAME

pt-archiver - Archive rows from a MySQL table into another table or a file.

SYNOPSIS

Usage

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

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

Examples

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

```
pt-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:

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

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

DESCRIPTION

pt-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.

pt-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 **pt-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.

Percona XtraDB Cluster

pt-archiver works with Percona XtraDB Cluster (PXC) 5.5.28-23.7 and newer, but there are three limitations you should consider before archiving on a cluster:

Error on commit

> **pt-archiver** does not check for error when it commits transactions. Commits on PXC 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 PXC is still experimental at the time of this release. There are several known bugs with PXC, 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 PXC 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.

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 **pt-archiver** has been running, and how many rows it has archived.

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

ERROR-HANDLING

pt-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.

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 http://dev.mysql.com/doc/en/analyze-table.html 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 **pt-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 MySQL.

--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 **pt-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 MySQL version`, refer to the documentation for the `L` DSN option.

--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 MySQL. Any other value sets binmode on STDOUT without the utf8 layer, and runs SET NAMES after connecting to MySQL.

Note that only charsets as known by MySQL 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 **pt-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, **pt-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, **pt-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, **pt-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 http://dev.mysql.com/doc/en/insert.html for details.

--dest
type: DSN

DSN specifying the table to archive to.

This item specifies a table into which **pt-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 **pt-archiver** to connect to *--dest* using that socket unless another socket for *--dest* is specified. This means that **pt-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 **pt-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 **pt-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 MySQL'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 MySQL 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 http://dev.mysql.com/doc/en/select.html 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 http://dev.mysql.com/doc/en/delete.html for details.

--low-priority-insert
Adds the LOW_PRIORITY modifier to INSERT or REPLACE statements.

See http://dev.mysql.com/doc/en/insert.html for details.

--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. Default is no Flow Control checking. This option is available for PXC 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 **pt-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 **pt-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 MySQL 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 **pt-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 **pt-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 http://dev.mysql.com/doc/en/optimize-table.html for details on OPTIMIZE TABLE.

--output-format
type: string

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

Valid formats are: dump: MySQL 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 http://dev.mysql.com/doc/en/delete.html 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 **pt-archiver** should retry when there is an InnoDB lock wait timeout or deadlock. When retries are exhausted, **pt-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, **pt-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 **pt-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 **pt-archiver** is running, it will not see them.

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

Exit if this file exists.

The presence of the file specified by *--sentinel* will cause **pt-archiver** to stop archiving and exit. The default is /tmp/pt-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 MySQL 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, **pt-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 **pt-archiver** connects to MySQL, 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 **pt-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, **pt-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; **pt-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 MySQL 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 **pt-archiver** to connect to *--dest* using that socket unless another socket for *--dest* is specified. This means that **pt-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 **pt-archiver** connects to *--dest*, host2, it will connect via the *--source*, host1, socket defined in host1.cnf.

--statistics

Collect and print timing statistics.

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

Unless you specify *--quiet*, **pt-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 **pt-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 **pt-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 **pt-archiver**, a value of 500 caused archiving to take about 2 seconds per 1000 rows on an otherwise quiet MySQL 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 **pt-archiver** doesn't try to commit.

--user

short form: -u; type: string

User for login if not current user.

--version

Show version and exit.

--[no]version-check

default: yes

Check for the latest version of Percona Toolkit, MySQL, and other programs.

This is a standard "check for updates automatically" feature, with two additional features. First, the tool checks the version of other programs on the local system in addition to its own version. For example, it checks the version of every MySQL server it connects to, Perl, and the Perl module DBD::mysql. Second, it checks for and warns about versions with known problems. For example, MySQL 5.5.25 had a critical bug and was re-released as 5.5.25a.

Any updates or known problems are printed to STDOUT before the tool's normal output. This feature should never interfere with the normal operation of the tool.

For more information, visit https://www.percona.com/version-check.

--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 **pt-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 **pt-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 **pt-archiver** in a `cron` job and get an email if there's an abnormal exit.

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

  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://dev.mysql.com/doc/refman/5.0/en/load-data-local.html 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.

EXTENDING

pt-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 **pt-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 **pt-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 **pt-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 **pt-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 **pt-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 **pt-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(), **pt-archiver** could get into an infinite loop because the row still exists. Therefore, when you specify a plugin for the *--source* argument, **pt-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.
>
> **pt-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, **pt-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 **pt-archiver** exits the archiving loop, commits all database handles, closes *--file*, and prints the final statistics, but before **pt-archiver** runs ANALYZE or OPTIMIZE (see *--analyze* and *--optimize*).

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

pt-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 **pt-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);
```

```
}

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;
```

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 pt-archiver ... > FILE 2>&1
```

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

SYSTEM REQUIREMENTS

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

BUGS

For a list of known bugs, see http://www.percona.com/bugs/pt-archiver.

Please report bugs at https://bugs.launchpad.net/percona-toolkit. Include the following information in your bug report:

- Complete command-line used to run the tool
- Tool *--version*
- MySQL version of all servers involved
- Output from the tool including STDERR
- Input files (log/dump/config files, etc.)

If possible, include debugging output by running the tool with `PTDEBUG`; see "ENVIRONMENT".

DOWNLOADING

Visit http://www.percona.com/software/percona-toolkit/ to download the latest release of Percona Toolkit. Or, get the latest release from the command line:

```
wget percona.com/get/percona-toolkit.tar.gz

wget percona.com/get/percona-toolkit.rpm

wget percona.com/get/percona-toolkit.deb
```

You can also get individual tools from the latest release:

```
wget percona.com/get/TOOL
```

Replace `TOOL` with the name of any tool.

AUTHORS

Baron Schwartz

ACKNOWLEDGMENTS

Andrew O'Brien

ABOUT PERCONA TOOLKIT

This tool is part of Percona Toolkit, a collection of advanced command-line tools for MySQL developed by Percona. 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. Visit http://www.percona.com/software/ to learn about other free, open-source software from Percona.

COPYRIGHT, LICENSE, AND WARRANTY

VERSION

pt-archiver 3.0.4

CHAPTER

FOUR

PT-CONFIG-DIFF

NAME

pt-config-diff - Diff MySQL configuration files and server variables.

SYNOPSIS

Usage

```
pt-config-diff [OPTIONS] CONFIG CONFIG [CONFIG...]
```

pt-config-diff diffs MySQL configuration files and server variables. CONFIG can be a filename or a DSN. At least two CONFIG sources must be given. Like standard Unix diff, there is no output if there are no differences.

Diff host1 config from SHOW VARIABLES against host2:

```
pt-config-diff h=host1 h=host2
```

Diff config from [mysqld] section in my.cnf against host1 config:

```
pt-config-diff /etc/my.cnf h=host1
```

Diff the [mysqld] section of two option files:

```
pt-config-diff /etc/my-small.cnf /etc/my-large.cnf
```

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

DESCRIPTION

pt-config-diff diffs MySQL configurations by examining the values of server system variables from two or more CONFIG sources specified on the command line. A CONFIG source can be a DSN or a filename containing the output of `mysqld --help --verbose`, `my_print_defaults`, `SHOW VARIABLES`, or an option file (e.g. my.cnf).

For each DSN CONFIG, **pt-config-diff** connects to MySQL and gets variables and values by executing `SHOW /*!40103 GLOBAL*/ VARIABLES`. This is an "active config" because it shows what server values MySQL is actively (currently) running with.

Only variables that all CONFIG sources have are compared because if a variable is not present then we cannot know or safely guess its value. For example, if you compare an option file (e.g. my.cnf) to an active config (i.e. SHOW VARIABLES from a DSN CONFIG), the option file will probably only have a few variables, whereas the active config has every variable. Only values of the variables present in both configs are compared.

Option file and DSN configs provide the best results.

OUTPUT

There is no output when there are no differences. When there are differences, **pt-config-diff** prints a report to STDOUT that looks similar to the following:

```
2 config differences
Variable                  my.master.cnf   my.slave.cnf
========================= =============== ===============
datadir                   /tmp/12345/data /tmp/12346/data
port                      12345           12346
```

Comparing MySQL variables is difficult because there are many variations and subtleties across the many versions and distributions of MySQL. When a comparison fails, the tool prints a warning to STDERR, such as the following:

```
Comparing log_error values (mysqld.log, /tmp/12345/data/mysqld.log)
caused an error: Argument "/tmp/12345/data/mysqld.log" isn't numeric
in numeric eq (==) at ./pt-config-diff line 2311.
```

Please report these warnings so the comparison functions can be improved.

EXIT STATUS

pt-config-diff exits with a zero exit status when there are no differences, and 1 if there are.

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 MySQL.

--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 MySQL. Any other value sets binmode on STDOUT without the utf8 layer, and runs SET NAMES after connecting to MySQL.

--config
type: Array

Read this comma-separated list of config files; if specified, this must be the first option on the command line. (This option does not specify a CONFIG; it's equivalent to `--defaults-file`.)

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

Connect to this database.

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

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

--help
Show help and exit.

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

Connect to host.

--`[no]ignore-case`
default: yes

Compare the variables case-insensitively.

--ignore-variables
type: array

Ignore, do not compare, these variables.

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

Password to use for connection.

--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.

--`[no]report`
default: yes

Print the MySQL config diff report to STDOUT. If you just want to check if the given configs are different or not by examining the tool's exit status, then specify `--no-report` to suppress the report.

--report-width
type: int; default: 78

Truncate report lines to this many characters. Since some variable values can be long, or when comparing multiple configs, it may help to increase the report width so values are not truncated beyond readability.

--set-vars
type: Array

Set the MySQL 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.

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

MySQL user if not current user.

--version
Show version and exit.

--`[no]version-check`
default: yes

Check for the latest version of Percona Toolkit, MySQL, and other programs.

This is a standard "check for updates automatically" feature, with two additional features. First, the tool checks the version of other programs on the local system in addition to its own version. For example, it checks the version of every MySQL server it connects to, Perl, and the Perl module DBD::mysql. Second, it checks for and warns about versions with known problems. For example, MySQL 5.5.25 had a critical bug and was re-released as 5.5.25a.

Any updates or known problems are printed to STDOUT before the tool's normal output. This feature should never interfere with the normal operation of the tool.

For more information, visit https://www.percona.com/version-check.

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: mysql_socket; copy: yes

  Socket file to use for connection.

- u

  dsn: user; copy: yes

  User for login if not current user.

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 pt-config-diff ... > FILE 2>&1
```

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

SYSTEM REQUIREMENTS

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

BUGS

For a list of known bugs, see http://www.percona.com/bugs/pt-config-diff.

Please report bugs at https://bugs.launchpad.net/percona-toolkit. Include the following information in your bug report:

- Complete command-line used to run the tool
- Tool *--version*
- MySQL version of all servers involved
- Output from the tool including STDERR
- Input files (log/dump/config files, etc.)

If possible, include debugging output by running the tool with `PTDEBUG`; see "ENVIRONMENT".

DOWNLOADING

Visit http://www.percona.com/software/percona-toolkit/ to download the latest release of Percona Toolkit. Or, get the latest release from the command line:

```
wget percona.com/get/percona-toolkit.tar.gz

wget percona.com/get/percona-toolkit.rpm

wget percona.com/get/percona-toolkit.deb
```

You can also get individual tools from the latest release:

```
wget percona.com/get/TOOL
```

Replace `TOOL` with the name of any tool.

AUTHORS

Baron Schwartz and Daniel Nichter

ABOUT PERCONA TOOLKIT

This tool is part of Percona Toolkit, a collection of advanced command-line tools for MySQL developed by Percona. 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. Visit http://www.percona.com/software/ to learn about other free, open-source software from Percona.

COPYRIGHT, LICENSE, AND WARRANTY

VERSION

pt-config-diff 3.0.4

CHAPTER

FIVE

PT-DEADLOCK-LOGGER

NAME

pt-deadlock-logger - Log MySQL deadlocks.

SYNOPSIS

Usage

```
pt-deadlock-logger [OPTIONS] DSN
```

pt-deadlock-logger logs information about MySQL deadlocks on the given DSN. Information is printed to STDOUT, and it can also be saved to a table by specifying *--dest*. The tool runs for forever unless *--run-time* or *--iterations* is specified.

Print deadlocks on host1:

```
pt-deadlock-logger h=host1
```

Print deadlocks on host1 once then exit:

```
pt-deadlock-logger h=host1 --iterations 1
```

Save deadlocks on host1 to percona_schema.deadlocks on host2:

```
pt-deadlock-logger h=host1 --dest h=host2,D=percona_schema,t=deadlocks
```

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

DESCRIPTION

pt-deadlock-logger prints information about MySQL deadlocks by polling and parsing `SHOW ENGINE INNODB STATUS`. When a new deadlock occurs, it's printed to `STDOUT` and, if specified, saved to *--dest*.

Only new deadlocks are printed. A fingerprint for each deadlock is created using the deadlock's server, ts, and thread values (even if these columns are not specified by *--columns*). A deadlock is printed if its fingerprint is different than the last deadlock's fingerprint.

The *--dest* statement uses `INSERT IGNORE` to eliminate duplicate deadlocks, so every deadlock is saved for every *--iterations*.

OUTPUT

New deadlocks are printed to `STDOUT`, unless *--quiet* is specified. Errors and warnings are printed to `STDERR`.

See also *--columns* and *--tab*.

INNODB CAVEATS AND DETAILS

InnoDB's output is hard to parse and sometimes there's no way to do it right.

Sometimes not all information (for example, username or IP address) is included in the deadlock information. In this case there's nothing for the tool to put in those columns. It may also be the case that the deadlock output is so long (because there were a lot of locks) that the whole thing is truncated.

Though there are usually two transactions involved in a deadlock, there are more locks than that; at a minimum, one more lock than transactions is necessary to create a cycle in the waits-for graph. **pt-deadlock-logger** prints the transactions (always two in the InnoDB output, even when there are more transactions in the waits-for graph than that) and fills in locks. It prefers waited-for over held when choosing lock information to output, but you can figure out the rest with a moment's thought. If you see one wait-for and one held lock, you're looking at the same lock, so of course you'd prefer to see both wait-for locks and get more information. If the two waited-for locks are not on the same table, more than two transactions were involved in the deadlock.

Finally, keep in mind that, because usernames with spaces are not quoted by InnoDB, the tool will generally misreport the second word of these usernames as the hostname.

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 MySQL.

--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 MySQL. Any other value sets binmode on STDOUT without the utf8 layer, and runs SET NAMES after connecting to MySQL.

--clear-deadlocks

type: string

Use this table to create a small deadlock. This usually has the effect of clearing out a huge deadlock, which otherwise consumes the entire output of `SHOW INNODB STATUS`. The table must not exist. **pt-deadlock-logger** will create it with the following structure:

```
CREATE TABLE percona_schema.clear_deadlocks (
  a INT PRIMARY KEY
) ENGINE=InnoDB
```

After creating the table and causing a small deadlock, the tool will drop the table again.

--columns

type: Array; default: server, ts, thread, txn_id, txn_time, user, hostname, ip, db, tbl, idx, lock_type, lock_mode, wait_hold, victim, query

The columns are:

server

The (source) server on which the deadlock occurred. This might be useful if you're tracking deadlocks on many servers.

ts

The date and time of the last detected deadlock.

thread

The MySQL thread number, which is the same as the connection ID in SHOW FULL PROCESSLIST.

txn_id

The InnoDB transaction ID, which InnoDB expresses as two unsigned integers. I have multiplied them out to be one number.

txn_time

How long the transaction was active when the deadlock happened.

user

The connection's database username.

hostname

The connection's host.

ip

The connection's IP address. If you specify *--numeric-ip*, this is converted to an unsigned integer.

db

The database in which the deadlock occurred.

tbl

The table on which the deadlock occurred.

idx

The index on which the deadlock occurred.

lock_type

The lock type the transaction held on the lock that caused the deadlock.

lock_mode

The lock mode of the lock that caused the deadlock.

wait_hold

Whether the transaction was waiting for the lock or holding the lock. Usually you will see the two waited-for locks.

victim

Whether the transaction was selected as the deadlock victim and rolled back.

query

The query that caused the deadlock.

--config

type: Array

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

--create-dest-table

Create the table specified by *--dest*.

Normally the *--dest* table is expected to exist already. This option causes **pt-deadlock-logger** to create the table automatically using the suggested table structure.

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

--dest

type: DSN

DSN for where to store deadlocks; specify at least a database (D) and table (t).

Missing values are filled in with the same values from the source host, so you can usually omit most parts of this argument if you're storing deadlocks on the same server on which they happen.

The following table structure is suggested if you want to store all the information **pt-deadlock-logger** can extract about deadlocks:

```
CREATE TABLE deadlocks (
  server char(20) NOT NULL,
  ts timestamp NOT NULL DEFAULT CURRENT_TIMESTAMP,
  thread int unsigned NOT NULL,
  txn_id bigint unsigned NOT NULL,
  txn_time smallint unsigned NOT NULL,
  user char(16) NOT NULL,
  hostname char(20) NOT NULL,
  ip char(15) NOT NULL, -- alternatively, ip int unsigned NOT NULL
  db char(64) NOT NULL,
```

```
  tbl char(64) NOT NULL,
  idx char(64) NOT NULL,
  lock_type char(16) NOT NULL,
  lock_mode char(1) NOT NULL,
  wait_hold char(1) NOT NULL,
  victim tinyint unsigned NOT NULL,
  query text NOT NULL,
  PRIMARY KEY  (server,ts,thread)
) ENGINE=InnoDB
```

If you use *--columns*, you can omit whichever columns you don't want to store.

--help
Show help and exit.

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

Connect to host.

--interval
type: time; default: 30

How often to check for deadlocks. If no *--run-time* is specified, **pt-deadlock-logger** runs forever, checking for deadlocks at every interval. See also *--run-time*.

--iterations
type: int

How many times to check for deadlocks. By default, this option is undefined which means an infinite number of iterations. The tool always exits for *--run-time*, regardless of the value specified for this option. For example, the tool will exit after 1 minute with `--run-time 1m --iterations 4 --interval 30` because 4 iterations at 30 second intervals would take 2 minutes, longer than the 1 mintue run-time.

--log
type: string

Print all output to this file when daemonized.

--numeric-ip
Express IP addresses as integers.

--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.

--quiet
Do not deadlocks; only print errors and warnings to `STDERR`.

--run-time
type: time

How long to run before exiting. By default **pt-deadlock-logger** runs forever, checking for deadlocks every *--interval* seconds.

--set-vars
type: Array

Set the MySQL 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.

--tab
Use tabs to separate columns instead of spaces.

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

User for login if not current user.

--version
Show version and exit.

`--[no]version-check`
default: yes

Check for the latest version of Percona Toolkit, MySQL, and other programs.

This is a standard "check for updates automatically" feature, with two additional features. First, the tool checks the version of other programs on the local system in addition to its own version. For example, it checks the version of every MySQL server it connects to, Perl, and the Perl module DBD::mysql. Second, it checks for and warns about versions with known problems. For example, MySQL 5.5.25 had a critical bug and was re-released as 5.5.25a.

Any updates or known problems are printed to STDOUT before the tool's normal output. This feature should never interfere with the normal operation of the tool.

For more information, visit https://www.percona.com/version-check.

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: mysql_socket; copy: yes

  Socket file to use for connection.

- t

  Table in which to store deadlock information.

- u

  dsn: user; copy: yes

  User for login if not current user.

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 pt-deadlock-logger ... > FILE 2>&1
```

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

SYSTEM REQUIREMENTS

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

BUGS

For a list of known bugs, see http://www.percona.com/bugs/pt-deadlock-logger.

Please report bugs at https://bugs.launchpad.net/percona-toolkit. Include the following information in your bug report:

- Complete command-line used to run the tool
- Tool `--version`
- MySQL version of all servers involved
- Output from the tool including STDERR
- Input files (log/dump/config files, etc.)

If possible, include debugging output by running the tool with `PTDEBUG`; see "ENVIRONMENT".

DOWNLOADING

Visit http://www.percona.com/software/percona-toolkit/ to download the latest release of Percona Toolkit. Or, get the latest release from the command line:

```
wget percona.com/get/percona-toolkit.tar.gz

wget percona.com/get/percona-toolkit.rpm

wget percona.com/get/percona-toolkit.deb
```

You can also get individual tools from the latest release:

```
wget percona.com/get/TOOL
```

Replace `TOOL` with the name of any tool.

AUTHORS

Baron Schwartz and Daniel Nichter

ABOUT PERCONA TOOLKIT

This tool is part of Percona Toolkit, a collection of advanced command-line tools for MySQL developed by Percona. 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. Visit http://www.percona.com/software/ to learn about other free, open-source software from Percona.

COPYRIGHT, LICENSE, AND WARRANTY

VERSION

pt-deadlock-logger 3.0.4

CHAPTER

SIX

PT-DISKSTATS

NAME

pt-diskstats - An interactive I/O monitoring tool for GNU/Linux.

SYNOPSIS

Usage

```
pt-diskstats [OPTIONS] [FILES]
```

pt-diskstats 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.

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

DESCRIPTION

The **pt-diskstats** 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 pt-stalk 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.

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 **pt-diskstats**. 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 **pt-diskstats**.

avgrq-sz

This column is in sectors in iostat, and is a combination of reads and writes. The **pt-diskstats** 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 **pt-diskstats**, 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 **pt-diskstats** 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 **pt-diskstats**. Instead, **pt-diskstats** 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 **pt-diskstats**.

%util

This column is called busy in **pt-diskstats**. 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.

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 **pt-diskstats** with *--save-samples* to collect this data for you. If you wish to capture samples as part of some other tool, and use **pt-diskstats** 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
```

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.

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, **pt-diskstats** 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.

--`[no]version-check`
default: yes

Check for the latest version of Percona Toolkit, MySQL, and other programs.

This is a standard "check for updates automatically" feature, with two additional features. First, the tool checks the version of other programs on the local system in addition to its own version. For example, it checks the version of every MySQL server it connects to, Perl, and the Perl module DBD::mysql. Second, it checks for and warns about versions with known problems. For example, MySQL 5.5.25 had a critical bug and was re-released as 5.5.25a.

Any updates or known problems are printed to STDOUT before the tool's normal output. This feature should never interfere with the normal operation of the tool.

For more information, visit https://www.percona.com/version-check.

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 pt-diskstats ... > FILE 2>&1
```

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

SYSTEM REQUIREMENTS

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

BUGS

For a list of known bugs, see http://www.percona.com/bugs/pt-diskstats.

Please report bugs at https://bugs.launchpad.net/percona-toolkit. Include the following information in your bug report:

- Complete command-line used to run the tool
- Tool *--version*
- MySQL version of all servers involved
- Output from the tool including STDERR
- Input files (log/dump/config files, etc.)

If possible, include debugging output by running the tool with `PTDEBUG`; see "ENVIRONMENT".

DOWNLOADING

Visit http://www.percona.com/software/percona-toolkit/ to download the latest release of Percona Toolkit. Or, get the latest release from the command line:

```
wget percona.com/get/percona-toolkit.tar.gz

wget percona.com/get/percona-toolkit.rpm

wget percona.com/get/percona-toolkit.deb
```

You can also get individual tools from the latest release:

```
wget percona.com/get/TOOL
```

Replace `TOOL` with the name of any tool.

AUTHORS

Baron Schwartz, Brian Fraser, and Daniel Nichter

ABOUT PERCONA TOOLKIT

This tool is part of Percona Toolkit, a collection of advanced command-line tools for MySQL developed by Percona. 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. Visit http://www.percona.com/software/ to learn about other free, open-source software from Percona.

COPYRIGHT, LICENSE, AND WARRANTY

VERSION

pt-diskstats 3.0.4

CHAPTER

SEVEN

PT-DUPLICATE-KEY-CHECKER

NAME

pt-duplicate-key-checker - Find duplicate indexes and foreign keys on MySQL tables.

SYNOPSIS

Usage

```
pt-duplicate-key-checker [OPTIONS] [DSN]
```

pt-duplicate-key-checker examines MySQL tables for duplicate or redundant indexes and foreign keys. Connection options are read from MySQL option files.

```
pt-duplicate-key-checker --host host1
```

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

DESCRIPTION

This program examines the output of SHOW CREATE TABLE on MySQL 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.

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 MySQL.

--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 MySQL. Any other value sets binmode on STDOUT without the utf8 layer, and runs SET NAMES after connecting to MySQL.

--`[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 MySQL 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 MySQL.

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.

--[no]version-check
default: yes

Check for the latest version of Percona Toolkit, MySQL, and other programs.

This is a standard "check for updates automatically" feature, with two additional features. First, the tool checks the version of other programs on the local system in addition to its own version. For example, it checks the version of every MySQL server it connects to, Perl, and the Perl module DBD::mysql. Second, it checks for and warns about versions with known problems. For example, MySQL 5.5.25 had a critical bug and was re-released as 5.5.25a.

Any updates or known problems are printed to STDOUT before the tool's normal output. This feature should never interfere with the normal operation of the tool.

For more information, visit https://www.percona.com/version-check.

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: mysql_socket; copy: yes

  Socket file to use for connection.

- u

  dsn: user; copy: yes

  User for login if not current user.

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 pt-duplicate-key-checker ... > FILE 2>&1
```

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

SYSTEM REQUIREMENTS

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

BUGS

For a list of known bugs, see http://www.percona.com/bugs/pt-duplicate-key-checker.

Please report bugs at https://bugs.launchpad.net/percona-toolkit. Include the following information in your bug report:

- Complete command-line used to run the tool
- Tool *--version*
- MySQL version of all servers involved
- Output from the tool including STDERR
- Input files (log/dump/config files, etc.)

If possible, include debugging output by running the tool with `PTDEBUG`; see "ENVIRONMENT".

DOWNLOADING

Visit http://www.percona.com/software/percona-toolkit/ to download the latest release of Percona Toolkit. Or, get the latest release from the command line:

```
wget percona.com/get/percona-toolkit.tar.gz

wget percona.com/get/percona-toolkit.rpm

wget percona.com/get/percona-toolkit.deb
```

You can also get individual tools from the latest release:

```
wget percona.com/get/TOOL
```

Replace `TOOL` with the name of any tool.

AUTHORS

Baron Schwartz and Daniel Nichter

ABOUT PERCONA TOOLKIT

This tool is part of Percona Toolkit, a collection of advanced command-line tools for MySQL developed by Percona. 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. Visit http://www.percona.com/software/ to learn about other free, open-source software from Percona.

COPYRIGHT, LICENSE, AND WARRANTY

VERSION

pt-duplicate-key-checker 3.0.4

CHAPTER

EIGHT

PT-FIFO-SPLIT

NAME

pt-fifo-split - Split files and pipe lines to a fifo without really splitting.

SYNOPSIS

Usage

```
pt-fifo-split [OPTIONS] [FILE]
```

pt-fifo-split splits FILE and pipes lines to a fifo. With no FILE, or when FILE is -, read standard input.

Read hugefile.txt in chunks of a million lines without physically splitting it:

```
pt-fifo-split --lines 1000000 hugefile.txt
while [ -e /tmp/pt-fifo-split ]; do cat /tmp/pt-fifo-split; done
```

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

DESCRIPTION

pt-fifo-split lets you read from a file as though it contains only some of the lines in the file. When you read from it again, it contains the next set of lines; when you have gone all the way through it, the file disappears. This works only on Unix-like operating systems.

You can specify multiple files on the command line. If you don't specify any, or if you use the special filename –, lines are read from standard input.

OPTIONS

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

--config
type: Array

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

--fifo
type: string; default: /tmp/pt-fifo-split

The name of the fifo from which the lines can be read.

--force
Remove the fifo if it exists already, then create it again.

--help
Show help and exit.

--lines
type: int; default: 1000

The number of lines to read in each chunk.

--offset
type: int; default: 0

Begin at the Nth line. If the argument is 0, all lines are printed to the fifo. If 1, then beginning at the first line, lines are printed (exactly the same as 0). If 2, the first line is skipped, and the 2nd and subsequent lines are printed to the fifo.

--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.

--statistics
Print out statistics between chunks. The statistics are the number of chunks, the number of lines, elapsed time, and lines per second overall and during the last chunk.

--version
Show version and exit.

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 pt-fifo-split ... > FILE 2>&1
```

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

SYSTEM REQUIREMENTS

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

BUGS

For a list of known bugs, see http://www.percona.com/bugs/pt-fifo-split.

Please report bugs at https://bugs.launchpad.net/percona-toolkit. Include the following information in your bug report:

- Complete command-line used to run the tool
- Tool `--version`
- MySQL version of all servers involved
- Output from the tool including STDERR
- Input files (log/dump/config files, etc.)

If possible, include debugging output by running the tool with `PTDEBUG`; see "ENVIRONMENT".

DOWNLOADING

Visit http://www.percona.com/software/percona-toolkit/ to download the latest release of Percona Toolkit. Or, get the latest release from the command line:

```
wget percona.com/get/percona-toolkit.tar.gz

wget percona.com/get/percona-toolkit.rpm

wget percona.com/get/percona-toolkit.deb
```

You can also get individual tools from the latest release:

```
wget percona.com/get/TOOL
```

Replace `TOOL` with the name of any tool.

AUTHORS

Baron Schwartz

ABOUT PERCONA TOOLKIT

This tool is part of Percona Toolkit, a collection of advanced command-line tools for MySQL developed by Percona. 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. Visit http://www.percona.com/software/ to learn about other free, open-source software from Percona.

COPYRIGHT, LICENSE, AND WARRANTY

VERSION

pt-fifo-split 3.0.4

CHAPTER

NINE

PT-FIND

NAME

pt-find - Find MySQL tables and execute actions, like GNU find.

SYNOPSIS

Usage

```
pt-find [OPTIONS] [DATABASES]
```

pt-find searches for MySQL tables and executes actions, like GNU find. The default action is to print the database and table name.

Find all tables created more than a day ago, which use the MyISAM engine, and print their names:

```
pt-find --ctime +1 --engine MyISAM
```

Find InnoDB tables and convert them to MyISAM:

```
pt-find --engine InnoDB --exec "ALTER TABLE %D.%N ENGINE=MyISAM"
```

Find tables created by a process that no longer exists, following the name_sid_pid naming convention, and remove them.

```
pt-find --connection-id '\D_\d+_(\d+)$' --server-id '\D_(\d+)_\d+$' --exec-plus "DROP␣
↪TABLE %s"
```

Find empty tables in the test and junk databases, and delete them:

```
pt-find --empty junk test --exec-plus "DROP TABLE %s"
```

Find tables more than five gigabytes in total size:

```
pt-find --tablesize +5G
```

Find all tables and print their total data and index size, and sort largest tables first (sort is a different program, by the way).

```
pt-find --printf "%T\t%D.%N\n" | sort -rn
```

As above, but this time, insert the data back into the database for posterity:

```
pt-find --noquote --exec "INSERT INTO sysdata.tblsize(db, tbl, size) VALUES('%D', '%N
↪', %T)"
```

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

DESCRIPTION

pt-find looks for MySQL tables that pass the tests you specify, and executes the actions you specify. The default action is to print the database and table name to STDOUT.

pt-find is simpler than GNU find. It doesn't allow you to specify complicated expressions on the command line.

pt-find uses SHOW TABLES when possible, and SHOW TABLE STATUS when needed.

OPTION TYPES

There are three types of options: normal options, which determine some behavior or setting; tests, which determine whether a table should be included in the list of tables found; and actions, which do something to the tables **pt-find** finds.

pt-find uses standard Getopt::Long option parsing, so you should use double dashes in front of long option names, unlike GNU find.

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 MySQL.

--case-insensitive
: Specifies that all regular expression searches are case-insensitive.

--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 MySQL. Any other value sets binmode on STDOUT without the utf8 layer, and runs SET NAMES after connecting to MySQL.

--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.

--day-start
Measure times (for *--mmin*, etc) from the beginning of today rather than from the current time.

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

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

--help
Show help and exit.

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

Connect to host.

--or
Combine tests with OR, not AND.

By default, tests are evaluated as though there were an AND between them. This option switches it to OR.

Option parsing is not implemented by **pt-find** itself, so you cannot specify complicated expressions with parentheses and mixtures of OR and AND.

--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.

--`[no]quote`
default: yes

Quotes MySQL identifier names with MySQL's standard backtick character.

Quoting happens after tests are run, and before actions are run.

--set-vars
type: Array

Set the MySQL 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.

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

User for login if not current user.

--version
Show version and exit.

--[no]version-check
default: yes

Check for the latest version of Percona Toolkit, MySQL, and other programs.

This is a standard "check for updates automatically" feature, with two additional features. First, the tool checks the version of other programs on the local system in addition to its own version. For example, it checks the version of every MySQL server it connects to, Perl, and the Perl module DBD::mysql. Second, it checks for and warns about versions with known problems. For example, MySQL 5.5.25 had a critical bug and was re-released as 5.5.25a.

Any updates or known problems are printed to STDOUT before the tool's normal output. This feature should never interfere with the normal operation of the tool.

For more information, visit https://www.percona.com/version-check.

TESTS

Most tests check some criterion against a column of SHOW TABLE STATUS output. Numeric arguments can be specified as +n for greater than n, -n for less than n, and n for exactly n. All numeric options can take an optional suffix multiplier of k, M or G (1_024, 1_048_576, and 1_073_741_824 respectively). All patterns are Perl regular expressions (see 'man perlre') unless specified as SQL LIKE patterns.

Dates and times are all measured relative to the same instant, when **pt-find** first asks the database server what time it is. All date and time manipulation is done in SQL, so if you say to find tables modified 5 days ago, that translates to SELECT DATE_SUB(CURRENT_TIMESTAMP, INTERVAL 5 DAY). If you specify *--day-start*, if course it's relative to CURRENT_DATE instead.

However, table sizes and other metrics are not consistent at an instant in time. It can take some time for MySQL to process all the SHOW queries, and **pt-find** can't do anything about that. These measurements are as of the time they're taken.

If you need some test that's not in this list, file a bug report and I'll enhance **pt-find** for you. It's really easy.

--autoinc
type: string; group: Tests

Table's next AUTO_INCREMENT is n. This tests the Auto_increment column.

--avgrowlen
type: size; group: Tests

Table avg row len is n bytes. This tests the Avg_row_length column. The specified size can be "NULL" to test where Avg_row_length IS NULL.

--checksum
type: string; group: Tests

Table checksum is n. This tests the Checksum column.

--cmin
type: size; group: Tests

Table was created n minutes ago. This tests the Create_time column.

--collation
type: string; group: Tests

Table collation matches pattern. This tests the Collation column.

--column-name
type: string; group: Tests

A column name in the table matches pattern.

--column-type
type: string; group: Tests

A column in the table matches this type (case-insensitive).

Examples of types are: varchar, char, int, smallint, bigint, decimal, year, timestamp, text, enum.

--comment
type: string; group: Tests

Table comment matches pattern. This tests the Comment column.

--connection-id
type: string; group: Tests

Table name has nonexistent MySQL connection ID. This tests the table name for a pattern. The argument to this test must be a Perl regular expression that captures digits like this: (d+). If the table name matches the pattern, these captured digits are taken to be the MySQL connection ID of some process. If the connection doesn't exist according to SHOW FULL PROCESSLIST, the test returns true. If the connection ID is greater than **pt-find**'s own connection ID, the test returns false for safety.

Why would you want to do this? If you use MySQL statement-based replication, you probably know the trouble temporary tables can cause. You might choose to work around this by creating real tables with unique names, instead of temporary tables. One way to do this is to append your connection ID to the end of the table, thusly: scratch_table_12345. This assures the table name is unique and lets you have a way to find which connection it was associated with. And perhaps most importantly, if the connection no longer exists, you can assume the connection died without cleaning up its tables, and this table is a candidate for removal.

This is how I manage scratch tables, and that's why I included this test in **pt-find**.

The argument I use to *--connection-id* is "D_(d+)$". That finds tables with a series of numbers at the end, preceded by an underscore and some non-number character (the latter criterion prevents me from examining tables with a date at the end, which people tend to do: baron_scratch_2007_05_07 for example). It's better to keep the scratch tables separate of course.

If you do this, make sure the user **pt-find** runs as has the PROCESS privilege! Otherwise it will only see connections from the same user, and might think some tables are ready to remove when they're still in use. For safety, **pt-find** checks this for you.

See also *--server-id*.

--createopts
type: string; group: Tests

Table create option matches pattern. This tests the Create_options column.

--ctime
type: size; group: Tests

Table was created n days ago. This tests the Create_time column.

--datafree
type: size; group: Tests

Table has n bytes of free space. This tests the Data_free column. The specified size can be "NULL" to test where Data_free IS NULL.

--datasize
type: size; group: Tests

Table data uses n bytes of space. This tests the Data_length column. The specified size can be "NULL" to test where Data_length IS NULL.

--dblike
type: string; group: Tests

Database name matches SQL LIKE pattern.

--dbregex
type: string; group: Tests

Database name matches this pattern.

--empty
group: Tests

Table has no rows. This tests the Rows column.

--engine
type: string; group: Tests

Table storage engine matches this pattern. This tests the Engine column, or in earlier versions of MySQL, the Type column.

--function
type: string; group: Tests

Function definition matches pattern.

--indexsize
type: size; group: Tests

Table indexes use n bytes of space. This tests the Index_length column. The specified size can be "NULL" to test where Index_length IS NULL.

--kmin
type: size; group: Tests

Table was checked n minutes ago. This tests the Check_time column.

--ktime
type: size; group: Tests

Table was checked n days ago. This tests the Check_time column.

--mmin
type: size; group: Tests

Table was last modified n minutes ago. This tests the Update_time column.

--mtime
type: size; group: Tests

Table was last modified n days ago. This tests the Update_time column.

--procedure
type: string; group: Tests

Procedure definition matches pattern.

--rowformat
type: string; group: Tests

Table row format matches pattern. This tests the Row_format column.

--rows
type: size; group: Tests

Table has n rows. This tests the Rows column. The specified size can be "NULL" to test where Rows IS NULL.

--server-id
type: string; group: Tests

Table name contains the server ID. If you create temporary tables with the naming convention explained in *--connection-id*, but also add the server ID of the server on which the tables are created, then you can use this pattern match to ensure tables are dropped only on the server they're created on. This prevents a table from being accidentally dropped on a slave while it's in use (provided that your server IDs are all unique, which they should be for replication to work).

For example, on the master (server ID 22) you create a table called scratch_table_22_12345. If you see this table on the slave (server ID 23), you might think it can be dropped safely if there's no such connection 12345. But if you also force the name to match the server ID with `--server-id '\D_(\d+)_\d+$'`, the table won't be dropped on the slave.

--tablesize
type: size; group: Tests

Table uses n bytes of space. This tests the sum of the Data_length and Index_length columns.

--tbllike
type: string; group: Tests

Table name matches SQL LIKE pattern.

--tblregex
type: string; group: Tests

Table name matches this pattern.

--tblversion
type: size; group: Tests

Table version is n. This tests the Version column.

--trigger
type: string; group: Tests

Trigger action statement matches pattern.

--trigger-table
type: string; group: Tests

--trigger is defined on table matching pattern.

--view
type: string; group: Tests

CREATE VIEW matches this pattern.

ACTIONS

The *--exec-plus* action happens after everything else, but otherwise actions happen in an indeterminate order. If you need determinism, file a bug report and I'll add this feature.

--exec
type: string; group: Actions

Execute this SQL with each item found. The SQL can contain escapes and formatting directives (see *--printf*).

--exec-dsn
type: string; group: Actions

Specify a DSN in key-value format to use when executing SQL with *--exec* and *--exec-plus*. Any values not specified are inherited from command-line arguments.

--exec-plus
type: string; group: Actions

Execute this SQL with all items at once. This option is unlike *--exec*. There are no escaping or formatting directives; there is only one special placeholder for the list of database and table names, %s. The list of tables found will be joined together with commas and substituted wherever you place %s.

You might use this, for example, to drop all the tables you found:

```
DROP TABLE %s
```

This is sort of like GNU find's "-exec command {} +" syntax. Only it's not totally cryptic. And it doesn't require me to write a command-line parser.

--print
group: Actions

Print the database and table name, followed by a newline. This is the default action if no other action is specified.

--printf
type: string; group: Actions

Print format on the standard output, interpreting '' escapes and '%' directives. Escapes are backslashed characters, like n and t. Perl interprets these, so you can use any escapes Perl knows about. Directives are replaced by %s, and as of this writing, you can't add any special formatting instructions, like field widths or alignment (though I'm musing over ways to do that).

Here is a list of the directives. Note that most of them simply come from columns of SHOW TABLE STATUS. If the column is NULL or doesn't exist, you get an empty string in the output. A % character followed by any character not in the following list is discarded (but the other character is printed).

```
CHAR DATA SOURCE        NOTES
---- ------------------ ----------------------------------------
a    Auto_increment
A    Avg_row_length
c    Checksum
C    Create_time
D    Database           The database name in which the table lives
d    Data_length
E    Engine             In older versions of MySQL, this is Type
F    Data_free
f    Innodb_free        Parsed from the Comment field
I    Index_length
K    Check_time
L    Collation
M    Max_data_length
N    Name
O    Comment
P    Create_options
R    Row_format
S    Rows
T    Table_length       Data_length+Index_length
U    Update_time
V    Version
```

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: mysql_socket; copy: yes

  Socket file to use for connection.

- u

  dsn: user; copy: yes

  User for login if not current user.

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 pt-find ... > FILE 2>&1
```

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

SYSTEM REQUIREMENTS

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

BUGS

For a list of known bugs, see http://www.percona.com/bugs/pt-find.

Please report bugs at https://bugs.launchpad.net/percona-toolkit. Include the following information in your bug report:

- Complete command-line used to run the tool
- Tool *--version*
- MySQL version of all servers involved
- Output from the tool including STDERR
- Input files (log/dump/config files, etc.)

If possible, include debugging output by running the tool with `PTDEBUG`; see "ENVIRONMENT".

DOWNLOADING

Visit http://www.percona.com/software/percona-toolkit/ to download the latest release of Percona Toolkit. Or, get the latest release from the command line:

```
wget percona.com/get/percona-toolkit.tar.gz

wget percona.com/get/percona-toolkit.rpm

wget percona.com/get/percona-toolkit.deb
```

You can also get individual tools from the latest release:

```
wget percona.com/get/TOOL
```

Replace `TOOL` with the name of any tool.

AUTHORS

Baron Schwartz

ABOUT PERCONA TOOLKIT

This tool is part of Percona Toolkit, a collection of advanced command-line tools for MySQL developed by Percona. 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. Visit http://www.percona.com/software/ to learn about other free, open-source software from Percona.

COPYRIGHT, LICENSE, AND WARRANTY

VERSION

pt-find 3.0.4

CHAPTER

TEN

PT-FINGERPRINT

NAME

pt-fingerprint - Convert queries into fingerprints.

SYNOPSIS

Usage

```
pt-fingerprint [OPTIONS] [FILES]
```

pt-fingerprint converts queries into fingerprints. With the –query option, converts the option's value into a fingerprint. With no options, treats command-line arguments as FILEs and reads and converts semicolon-separated queries from the FILEs. When FILE is -, it read standard input.

Convert a single query:

```
pt-fingerprint --query "select a, b, c from users where id = 500"
```

Convert a file full of queries:

```
pt-fingerprint /path/to/file.txt
```

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

DESCRIPTION

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.

Query fingerprinting accommodates a great 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 want to understand more about how and why all of these cases are handled, please review the test cases in the Subversion repository. 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 mysqldump together, even if they are against different tables. Ditto for all of pt-table-checksum's checksum queries.
- 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.

OPTIONS

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

--config
type: Array

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

--help
Show help and exit.

--match-embedded-numbers
Match numbers embedded in words and replace as single values. This option causes the tool to be more careful about matching numbers so that words with numbers, like `catch22` are matched and replaced as a single `?` placeholder. Otherwise the default number matching pattern will replace `catch22` as `catch?`.

This is helpful if database or table names contain numbers.

--match-md5-checksums
Match MD5 checksums and replace as single values. This option causes the tool to be more careful about matching numbers so that MD5 checksums like `fbc5e685a5d3d45aa1d0347fdb7c4d35` are matched and replaced as a single `?` placeholder. Otherwise, the default number matching pattern will replace `fbc5e685a5d3d45aa1d0347fdb7c4d35` as `fbc?`.

--query
type: string

The query to convert into a fingerprint.

--version
Show version and exit.

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 pt-fingerprint ... > FILE 2>&1
```

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

SYSTEM REQUIREMENTS

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

BUGS

For a list of known bugs, see http://www.percona.com/bugs/pt-fingerprint.

Please report bugs at https://bugs.launchpad.net/percona-toolkit. Include the following information in your bug report:

- Complete command-line used to run the tool
- Tool *--version*
- MySQL version of all servers involved
- Output from the tool including STDERR
- Input files (log/dump/config files, etc.)

If possible, include debugging output by running the tool with `PTDEBUG`; see "ENVIRONMENT".

DOWNLOADING

Visit http://www.percona.com/software/percona-toolkit/ to download the latest release of Percona Toolkit. Or, get the latest release from the command line:

```
wget percona.com/get/percona-toolkit.tar.gz

wget percona.com/get/percona-toolkit.rpm

wget percona.com/get/percona-toolkit.deb
```

You can also get individual tools from the latest release:

```
wget percona.com/get/TOOL
```

Replace `TOOL` with the name of any tool.

AUTHORS

Baron Schwartz and Daniel Nichter

ABOUT PERCONA TOOLKIT

This tool is part of Percona Toolkit, a collection of advanced command-line tools for MySQL developed by Percona. 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. Visit http://www.percona.com/software/ to learn about other free, open-source software from Percona.

COPYRIGHT, LICENSE, AND WARRANTY

VERSION

pt-fingerprint 3.0.4

CHAPTER

ELEVEN

PT-FK-ERROR-LOGGER

NAME

pt-fk-error-logger - Log MySQL foreign key errors.

SYNOPSIS

Usage

```
pt-fk-error-logger [OPTIONS] [DSN]
```

pt-fk-error-logger logs information about foreign key errors on the given DSN. Information is printed to STDOUT, and it can also be saved to a table by specifying *--dest*. The tool runs for forever unless *--run-time* or *--iterations* is specified.

Print foreign key errors on host1:

```
pt-fk-error-logger h=host1
```

Print foreign key errors on host1 once then exit:

```
pt-fk-error-logger h=host1 --iterations 1
```

Save foreign key errors on host1 to percona_schema.fke on host2:

```
pt-fk-error-logger h=host1 --dest h=host2,D=percona_schema,t=fke
```

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

DESCRIPTION

pt-fk-error-logger prints or saves the foreign key errors text from `SHOW INNODB STATUS`. The errors are not parsed or interpreted in any way. Foreign key errors are uniquely identified by their timestamp. Only new (more recent) errors are printed or saved.

By default the tool runs forever, checking every *--interval* seconds for new foreign key errors. Specify *--run-time* and/or *--iterations* to limit how long the tool runs.

OUTPUT

The foreign key error text from `SHOW ENGINE INNODB STATUS` is printed to `STDOUT`, unless *--quiet* is specified. Errors and warnings are printed to `STDERR`.

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 MySQL.

--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 MySQL. Any other value sets binmode on STDOUT without the utf8 layer, and runs SET NAMES after connecting to MySQL.

--config
type: Array

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

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

--dest
type: DSN

Save foreign key errors in this table. The DSN must specify a database (D) and table (t).

Missing DSN values are inherited from the DSN being monitored, so you can omit most values if you're saving foreign key errors on the same host.

The following table is suggested:

```
CREATE TABLE foreign_key_errors (
  ts datetime NOT NULL,
  error text NOT NULL,
  PRIMARY KEY (ts)
)
```

The only information saved is the timestamp and the foreign key error text.

--help
Show help and exit.

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

Connect to host.

--interval
type: time; default: 30

How often to check for foreign key errors.

--iterations
type: int

How many times to check for foreign key errors. By default, this option is undefined which means an infinite number of iterations. The tool always exits for *--run-time*, regardless of the value specified for this option. For example, the tool will exit after 1 minute with `--run-time 1m --iterations 4 --interval 30` because 4 iterations at 30 second intervals would take 2 minutes, longer than the 1 mintue run-time.

--log
type: string

Print all output to this file when daemonized.

--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.

--quiet
Do not print foreign key errors; only print errors and warnings to `STDERR`.

--run-time
type: time

How long to run before exiting. By default, the tool runs forever.

--set-vars
type: Array

Set the MySQL 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.

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

User for login if not current user.

--version
Show version and exit.

--`[no]version-check`
default: yes

Check for the latest version of Percona Toolkit, MySQL, and other programs.

This is a standard "check for updates automatically" feature, with two additional features. First, the tool checks the version of other programs on the local system in addition to its own version. For example, it checks the version of every MySQL server it connects to, Perl, and the Perl module DBD::mysql. Second, it checks for and warns about versions with known problems. For example, MySQL 5.5.25 had a critical bug and was re-released as 5.5.25a.

Any updates or known problems are printed to STDOUT before the tool's normal output. This feature should never interfere with the normal operation of the tool.

For more information, visit https://www.percona.com/version-check.

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: mysql_socket; copy: yes

  Socket file to use for connection.

- t

  Table in which to store foreign key errors.

- u

  dsn: user; copy: yes

  User for login if not current user.

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 pt-fk-error-logger ... > FILE 2>&1
```

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

SYSTEM REQUIREMENTS

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

BUGS

For a list of known bugs, see http://www.percona.com/bugs/pt-fk-error-logger.

Please report bugs at https://bugs.launchpad.net/percona-toolkit. Include the following information in your bug report:

- Complete command-line used to run the tool
- Tool *--version*

- MySQL version of all servers involved
- Output from the tool including STDERR
- Input files (log/dump/config files, etc.)

If possible, include debugging output by running the tool with `PTDEBUG`; see "ENVIRONMENT".

DOWNLOADING

Visit http://www.percona.com/software/percona-toolkit/ to download the latest release of Percona Toolkit. Or, get the latest release from the command line:

```
wget percona.com/get/percona-toolkit.tar.gz

wget percona.com/get/percona-toolkit.rpm

wget percona.com/get/percona-toolkit.deb
```

You can also get individual tools from the latest release:

```
wget percona.com/get/TOOL
```

Replace `TOOL` with the name of any tool.

AUTHORS

Daniel Nichter

ABOUT PERCONA TOOLKIT

This tool is part of Percona Toolkit, a collection of advanced command-line tools for MySQL developed by Percona. 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. Visit http://www.percona.com/software/ to learn about other free, open-source software from Percona.

COPYRIGHT, LICENSE, AND WARRANTY

VERSION

pt-fk-error-logger 3.0.4

CHAPTER

TWELVE

PT-HEARTBEAT

NAME

pt-heartbeat - Monitor MySQL replication delay.

SYNOPSIS

Usage

```
pt-heartbeat [OPTIONS] [DSN] --update|--monitor|--check|--stop
```

pt-heartbeat measures replication lag on a MySQL or PostgreSQL server. You can use it to update a master or monitor a replica. If possible, MySQL connection options are read from your .my.cnf file.

Start daemonized process to update test.heartbeat table on master:

```
pt-heartbeat -D test --update -h master-server --daemonize
```

Monitor replication lag on slave:

```
pt-heartbeat -D test --monitor -h slave-server

pt-heartbeat -D test --monitor -h slave-server --dbi-driver Pg
```

Check slave lag once and exit (using optional DSN to specify slave host):

```
pt-heartbeat -D test --check h=slave-server
```

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

DESCRIPTION

pt-heartbeat is a two-part MySQL and PostgreSQL replication delay monitoring system that measures delay by looking at actual replicated data. This avoids reliance on the replication mechanism itself, which is unreliable. (For example, `SHOW SLAVE STATUS` on MySQL).

The first part is an *--update* instance of **pt-heartbeat** that connects to a master and updates a timestamp ("heartbeat record") every *--interval* seconds. Since the heartbeat table may contain records from multiple masters (see "MULTI-SLAVE HIERARCHY"), the server's ID (@@server_id) is used to identify records.

The second part is a *--monitor* or *--check* instance of **pt-heartbeat** that connects to a slave, examines the replicated heartbeat record from its immediate master or the specified *--master-server-id*, and computes the difference from the current system time. If replication between the slave and the master is delayed or broken, the computed difference will be greater than zero and potentially increase if *--monitor* is specified.

You must either manually create the heartbeat table on the master or use *--create-table*. See *--create-table* for the proper heartbeat table structure. The `MEMORY` storage engine is suggested, but not required of course, for MySQL.

The heartbeat table must contain a heartbeat row. By default, a heartbeat row is inserted if it doesn't exist. This feature can be disabled with the `--[no]insert-heartbeat-row` option in case the database user does not have INSERT privileges.

pt-heartbeat depends only on the heartbeat record being replicated to the slave, so it works regardless of the replication mechanism (built-in replication, a system such as Continuent Tungsten, etc). It works at any depth in the replication hierarchy; for example, it will reliably report how far a slave lags its master's master's master. And if replication is stopped, it will continue to work and report (accurately!) that the slave is falling further and further behind the master.

pt-heartbeat has a maximum resolution of 0.01 second. The clocks on the master and slave servers must be closely synchronized via NTP. By default, *--update* checks happen on the edge of the second (e.g. 00:01) and *--monitor* checks happen halfway between seconds (e.g. 00:01.5). As long as the servers' clocks are closely synchronized and replication events are propagating in less than half a second, **pt-heartbeat** will report zero seconds of delay.

pt-heartbeat will try to reconnect if the connection has an error, but will not retry if it can't get a connection when it first starts.

The *--dbi-driver* option lets you use **pt-heartbeat** to monitor PostgreSQL as well. It is reported to work well with Slony-1 replication.

MULTI-SLAVE HIERARCHY

If the replication hierarchy has multiple slaves which are masters of other slaves, like "master -> slave1 -> slave2", *--update* instances can be ran on the slaves as well as the master. The default heartbeat table (see *--create-table*) is keyed on the `server_id` column, so each server will update the row where `server_id=@@server_id`.

For *--monitor* and *--check*, if *--master-server-id* is not specified, the tool tries to discover and use the slave's immediate master. If this fails, or if you want monitor lag from another master, then you can specify the *--master-server-id* to use.

For example, if the replication hierarchy is "master -> slave1 -> slave2" with corresponding server IDs 1, 2 and 3, you can:

```
pt-heartbeat --daemonize -D test --update -h master
pt-heartbeat --daemonize -D test --update -h slave1
```

Then check (or monitor) the replication delay from master to slave2:

```
pt-heartbeat -D test --master-server-id 1 --check slave2
```

Or check the replication delay from slave1 to slave2:

```
pt-heartbeat -D test --master-server-id 2 --check slave2
```

Stopping the *--update* instance one slave1 will not affect the instance on master.

MASTER AND SLAVE STATUS

The default heartbeat table (see *--create-table*) has columns for saving information from `SHOW MASTER STATUS` and `SHOW SLAVE STATUS`. These columns are optional. If any are present, their corresponding information will be saved.

Percona XtraDB Cluster

Although **pt-heartbeat** should work with all supported versions of Percona XtraDB Cluster (PXC), we recommend using 5.5.28-23.7 and newer.

If you are setting up heartbeat instances between cluster nodes, keep in mind that, since the speed of the cluster is determined by its slowest node, **pt-heartbeat** will not report how fast the cluster itself is, but only how fast events are replicating from one node to another.

You must specify *--master-server-id* for *--monitor* and *--check* instances.

OPTIONS

Specify at least one of *--stop*, *--update*, *--monitor*, or *--check*.

--update, *--monitor*, and *--check* are mutually exclusive.

--daemonize and *--check* 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 MySQL.

--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 MySQL. Any other value sets binmode on STDOUT without the utf8 layer, and runs SET NAMES after connecting to MySQL.

--check
Check slave delay once and exit. If you also specify *--recurse*, the tool will try to discover slave's of the given slave and check and print their lag, too. The hostname or IP and port for each slave is printed before its delay. *--recurse* only works with MySQL.

--check-read-only
Check if the server has read_only enabled; If it does, the tool skips doing any inserts.

--config
type: Array

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

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

This option causes the table specified by *--database* and *--table* to be created with the following MAGIC_create_heartbeat table definition:

```
CREATE TABLE heartbeat (
  ts                    varchar(26) NOT NULL,
  server_id             int unsigned NOT NULL PRIMARY KEY,
  file                  varchar(255) DEFAULT NULL,     -- SHOW MASTER STATUS
  position              bigint unsigned DEFAULT NULL, -- SHOW MASTER STATUS
  relay_master_log_file varchar(255) DEFAULT NULL,     -- SHOW SLAVE STATUS
  exec_master_log_pos   bigint unsigned DEFAULT NULL  -- SHOW SLAVE STATUS
);
```

The heartbeat table requires at least one row. If you manually create the heartbeat table, then you must insert a row by doing:

```
INSERT INTO heartbeat (ts, server_id) VALUES (NOW(), N);
```

or if using *--utc*:

```
INSERT INTO heartbeat (ts, server_id) VALUES (UTC_TIMESTAMP(), N);
```

where `N` is the server's ID; do not use @@server_id because it will replicate and slaves will insert their own server ID instead of the master's server ID.

This is done automatically by *--create-table*.

A legacy version of the heartbeat table is still supported:

```
CREATE TABLE heartbeat (
  id int NOT NULL PRIMARY KEY,
  ts datetime NOT NULL
);
```

Legacy tables do not support *--update* instances on each slave of a multi-slave hierarchy like "master -> slave1 -> slave2". To manually insert the one required row into a legacy table:

```
INSERT INTO heartbeat (id, ts) VALUES (1, NOW());
```

or if using *--utc*:

```
INSERT INTO heartbeat (id, ts) VALUES (1, UTC_TIMESTAMP());
```

The tool automatically detects if the heartbeat table is legacy.

See also "MULTI-SLAVE HIERARCHY".

--create-table-engine
type: string

Sets the engine to be used for the heartbeat table. The default storage engine is InnoDB as of MySQL 5.5.5.

--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.

--dbi-driver
default: mysql; type: string

Specify a driver for the connection; `mysql` and `Pg` are supported.

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

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

--file
type: string

Print latest *--monitor* output to this file.

When *--monitor* is given, prints output to the specified file instead of to STDOUT. The file is opened, truncated, and closed every interval, so it will only contain the most recent statistics. Useful when *--daemonize* is given.

--frames
type: string; default: 1m,5m,15m

Timeframes for averages.

Specifies the timeframes over which to calculate moving averages when *--monitor* is given. Specify as a comma-separated list of numbers with suffixes. The suffix can be s for seconds, m for minutes, h for hours, or d for days. The size of the largest frame determines the maximum memory usage, as up to the specified number of per-second samples are kept in memory to calculate the averages. You can specify as many timeframes as you like.

--help
Show help and exit.

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

Connect to host.

--`[no]insert-heartbeat-row`
default: yes

Insert a heartbeat row in the *--table* if one doesn't exist.

The heartbeat *--table* requires a heartbeat row, else there's nothing to *--update*, *--monitor*, or *--check*! By default, the tool will insert a heartbeat row if one is not already present. You can disable this feature by specifying `--no-insert-heartbeat-row` in case the database user does not have INSERT privileges.

--interval
type: float; default: 1.0

How often to update or check the heartbeat *--table*. Updates and checks begin on the first whole second then repeat every *--interval* seconds for *--update* and every *--interval* plus *--skew* seconds for *--monitor*.

For example, if at 00:00.4 an *--update* instance is started at 0.5 second intervals, the first update happens at 00:01.0, the next at 00:01.5, etc. If at 00:10.7 a *--monitor* instance is started at 0.05 second intervals with the default 0.5 second *--skew*, then the first check happens at 00:11.5 (00:11.0 + 0.5) which will be *--skew*

seconds after the last update which, because the instances are checking at synchronized intervals, happened at 00:11.0.

The tool waits for and begins on the first whole second just to make the interval calculations simpler. Therefore, the tool could wait up to 1 second before updating or checking.

The minimum (fastest) interval is 0.01, and the maximum precision is two decimal places, so 0.015 will be rounded to 0.02.

If a legacy heartbeat table (see *--create-table*) is used, then the maximum precision is 1s because the `ts` column is type `datetime`.

--log
type: string

Print all output to this file when daemonized.

--master-server-id
type: string

Calculate delay from this master server ID for *--monitor* or *--check*. If not given, **pt-heartbeat** attempts to connect to the server's master and determine its server id.

--monitor
Monitor slave delay continuously.

Specifies that **pt-heartbeat** should check the slave's delay every second and report to STDOUT (or if *--file* is given, to the file instead). The output is the current delay followed by moving averages over the timeframe given in *--frames*. For example,

```
5s [  0.25s,  0.05s,  0.02s ]
```

--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.

--print-master-server-id
Print the auto-detected or given *--master-server-id*. If *--check* or *--monitor* is specified, specifying this option will print the auto-detected or given *--master-server-id* at the end of each line.

--recurse
type: int

Check slaves recursively to this depth in *--check* mode.

Try to discover slave servers recursively, to the specified depth. After discovering servers, run the check on each one of them and print the hostname (if possible), followed by the slave delay.

This currently works only with MySQL. See *--recursion-method*.

--recursion-method
type: array; default: processlist,hosts

Preferred recursion method used to find slaves.

Possible methods are:

```
METHOD       USES
===========  ==================
processlist  SHOW PROCESSLIST
hosts        SHOW SLAVE HOSTS
none         Do not find slaves
```

The processlist method is preferred because SHOW SLAVE HOSTS is not reliable. However, the hosts method is required if the server uses a non-standard port (not 3306). Usually **pt-heartbeat** does the right thing and finds the slaves, but you may give a preferred method and it will be used first. If it doesn't find any slaves, the other methods will be tried.

--replace
Use `REPLACE` instead of `UPDATE` for –update.

When running in *--update* mode, use `REPLACE` instead of `UPDATE` to set the heartbeat table's timestamp. The `REPLACE` statement is a MySQL extension to SQL. This option is useful when you don't know whether the table contains any rows or not. It must be used in conjunction with –update.

--run-time
type: time

Time to run before exiting.

--sentinel
type: string; default: /tmp/pt-heartbeat-sentinel

Exit if this file exists.

--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 MySQL 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.

--skew
type: float; default: 0.5

How long to delay checks.

The default is to delay checks one half second. Since the update happens as soon as possible after the beginning of the second on the master, this allows one half second of replication delay before reporting that the slave lags the master by one second. If your clocks are not completely accurate or there is some other reason you'd like to delay the slave more or less, you can tweak this value. Try setting the `PTDEBUG` environment variable to see the effect this has.

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

Socket file to use for connection.

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

This should have the effect of stopping all running instances which are watching the same sentinel file. If none of *--update*, *--monitor* or *--check* is specified, **pt-heartbeat** will exit after creating the file. If one of these is specified, **pt-heartbeat** will wait the interval given by *--interval*, then remove the file and continue working.

You might find this handy to stop cron jobs gracefully if necessary, or to replace one running instance with another. For example, if you want to stop and restart **pt-heartbeat** every hour (just to make sure that it is restarted every hour, in case of a server crash or some other problem), you could use a `crontab` line like this:

```
0 * * * * :program:`pt-heartbeat` --update -D test --stop \
  --sentinel /tmp/pt-heartbeat-hourly
```

The non-default *--sentinel* will make sure the hourly `cron` job stops only instances previously started with the same options (that is, from the same `cron` job).

See also *--sentinel*.

--table
type: string; default: heartbeat

The table to use for the heartbeat.

Don't specify database.table; use *--database* to specify the database.

See *--create-table*.

--update
Update a master's heartbeat.

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

User for login if not current user.

--utc
Ignore system time zones and use only UTC. By default **pt-heartbeat** does not check or adjust for different system or MySQL time zones which can cause the tool to compute the lag incorrectly. Specifying this option is a good idea because it ensures that the tool works correctly regardless of time zones.

If used, this option must be used for all **pt-heartbeat** instances: *--update*, *--monitor*, *--check*, etc. You should probably set the option in a *--config* file. Mixing this option with **pt-heartbeat** instances not using this option will cause false-positive lag readings due to different time zones (unless all your systems are set to use UTC, in which case this option isn't required).

--version
Show version and exit.

--[no]version-check
default: yes

Check for the latest version of Percona Toolkit, MySQL, and other programs.

This is a standard "check for updates automatically" feature, with two additional features. First, the tool checks the version of other programs on the local system in addition to its own version. For example, it checks the version of every MySQL server it connects to, Perl, and the Perl module DBD::mysql. Second, it checks for and warns about versions with known problems. For example, MySQL 5.5.25 had a critical bug and was re-released as 5.5.25a.

Any updates or known problems are printed to STDOUT before the tool's normal output. This feature should never interfere with the normal operation of the tool.

For more information, visit https://www.percona.com/version-check.

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: mysql_socket; copy: yes

  Socket file to use for connection.

- u

  dsn: user; copy: yes

  User for login if not current user.

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 pt-heartbeat ... > FILE 2>&1
```

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

SYSTEM REQUIREMENTS

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

BUGS

For a list of known bugs, see http://www.percona.com/bugs/pt-heartbeat.

Please report bugs at https://bugs.launchpad.net/percona-toolkit. Include the following information in your bug report:

- Complete command-line used to run the tool
- Tool *--version*
- MySQL version of all servers involved
- Output from the tool including STDERR
- Input files (log/dump/config files, etc.)

If possible, include debugging output by running the tool with `PTDEBUG`; see "ENVIRONMENT".

DOWNLOADING

Visit http://www.percona.com/software/percona-toolkit/ to download the latest release of Percona Toolkit. Or, get the latest release from the command line:

```
wget percona.com/get/percona-toolkit.tar.gz

wget percona.com/get/percona-toolkit.rpm

wget percona.com/get/percona-toolkit.deb
```

You can also get individual tools from the latest release:

```
wget percona.com/get/TOOL
```

Replace `TOOL` with the name of any tool.

AUTHORS

Proven Scaling LLC, SixApart Ltd, Baron Schwartz, and Daniel Nichter

ABOUT PERCONA TOOLKIT

This tool is part of Percona Toolkit, a collection of advanced command-line tools for MySQL developed by Percona. 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. Visit http://www.percona.com/software/ to learn about other free, open-source software from Percona.

COPYRIGHT, LICENSE, AND WARRANTY

VERSION

pt-heartbeat 3.0.4

CHAPTER

THIRTEEN

PT-INDEX-USAGE

NAME

pt-index-usage - Read queries from a log and analyze how they use indexes.

SYNOPSIS

Usage

```
pt-index-usage [OPTIONS] [FILES]
```

pt-index-usage reads queries from logs and analyzes how they use indexes.

Analyze queries in slow.log and print reports:

```
pt-index-usage /path/to/slow.log --host localhost
```

Disable reports and save results to percona database for later analysis:

```
pt-index-usage slow.log --no-report --save-results-database percona
```

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

DESCRIPTION

This tool connects to a MySQL database server, reads through a query log, and uses EXPLAIN to ask MySQL how it will use each query. When it is finished, it prints out a report on indexes that the queries didn't use.

The query log needs to be in MySQL's slow query log format. If you need to input a different format, you can use pt-query-digest to translate the formats. If you don't specify a filename, the tool reads from STDIN.

The tool runs two stages. In the first stage, the tool takes inventory of all the tables and indexes in your database, so it can compare the existing indexes to those that were actually used by the queries in the log. In the second stage, it runs EXPLAIN on each query in the query log. It uses separate database connections to inventory the tables and run EXPLAIN, so it opens two connections to the database.

If a query is not a SELECT, it tries to transform it to a roughly equivalent SELECT query so it can be EXPLAINed. This is not a perfect process, but it is good enough to be useful.

The tool skips the EXPLAIN step for queries that are exact duplicates of those seen before. It assumes that the same query will generate the same EXPLAIN plan as it did previously (usually a safe assumption, and generally good for performance), and simply increments the count of times that the indexes were used. However, queries that have the same fingerprint but different checksums will be re-EXPLAINed. Queries that have different literal constants can have different execution plans, and this is important to measure.

After EXPLAIN-ing the query, it is necessary to try to map aliases in the query back to the original table names. For example, consider the EXPLAIN plan for the following query:

```
SELECT * FROM tbl1 AS foo;
```

The EXPLAIN output will show access to table `foo`, and that must be translated back to `tbl1`. This process involves complex parsing. It is generally very accurate, but there is some chance that it might not work right. If you find cases where it fails, submit a bug report and a reproducible test case.

Queries that cannot be EXPLAINed will cause all subsequent queries with the same fingerprint to be blacklisted. This is to reduce the work they cause, and prevent them from continuing to print error messages. However, at least in this stage of the tool's development, it is my opinion that it's not a good idea to preemptively silence these, or prevent them from being EXPLAINed at all. I am looking for lots of feedback on how to improve things like the query parsing. So please submit your test cases based on the errors the tool prints!

OUTPUT

After it reads all the events in the log, the tool prints out DROP statements for every index that was not used. It skips indexes for tables that were never accessed by any queries in the log, to avoid false-positive results.

If you don't specify *--quiet*, the tool also outputs warnings about statements that cannot be EXPLAINed and similar. These go to standard error.

Progress reports are enabled by default (see *--progress*). These also go to standard error.

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 MySQL.

--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 MySQL. Any other value sets binmode on STDOUT without the utf8 layer, and runs SET NAMES after connecting to MySQL.

--config
type: Array

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

--create-save-results-database
Create the *--save-results-database* if it does not exist.

If the *--save-results-database* already exists and this option is specified, the database is used and the necessary tables are created if they do not already exist.

--[no]create-views
Create views for *--save-results-database* example queries.

Several example queries are given for querying the tables in the *--save-results-database*. These example queries are, by default, created as views. Specifying --no-create-views prevents these views from being created.

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

The database to use for the connection.

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

Only get tables and indexes from this comma-separated list of databases.

--databases-regex
type: string

Only get tables and indexes from database whose names match this Perl regex.

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

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

--drop
type: Hash; default: non-unique

Suggest dropping only these types of unused indexes.

By default **pt-index-usage** will only suggest to drop unused secondary indexes, not primary or unique indexes. You can specify which types of unused indexes the tool suggests to drop: primary, unique, non-unique, all.

A separate ALTER TABLE statement for each type is printed. So if you specify --drop all and there is a primary key and a non-unique index, the ALTER TABLE ... DROP for each will be printed on separate lines.

--empty-save-results-tables
Drop and re-create all pre-existing tables in the *--save-results-database*. This allows information from previous runs to be removed before the current run.

--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-databases-regex
type: string

Ignore databases whose names match this Perl regex.

--ignore-tables
type: Hash

Ignore this comma-separated list of table names.

Table names may be qualified with the database name.

--ignore-tables-regex
type: string

Ignore tables whose names match the Perl regex.

--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.

--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.

--quiet
short form: -q

Do not print any warnings. Also disables *--progress*.

--[no]report
default: yes

Print the reports for *--report-format*.

You may want to disable the reports by specifying `--no-report` if, for example, you also specify *--save-results-database* and you only want to query the results tables later.

--report-format
type: Array; default: drop_unused_indexes

Right now there is only one report: drop_unused_indexes. This report prints SQL statements for dropping any unused indexes. See also *--drop*.

See also `--[no]report`.

--save-results-database
type: DSN

Save results to tables in this database. Information about indexes, queries, tables and their usage is stored in several tables in the specified database. The tables are auto-created if they do not exist. If the database doesn't exist, it can be auto-created with *--create-save-results-database*. In this case the connection is initially created with no default database, then after the database is created, it is USE'ed.

pt-index-usage executes INSERT statements to save the results. Therefore, you should be careful if you use this feature on a production server. It might increase load, or cause trouble if you don't want the server to be written to, or so on.

This is a new feature. It may change in future releases.

After a run, you can query the usage tables to answer various questions about index usage. The tables have the following CREATE TABLE definitions:

MAGIC_create_indexes:

```
CREATE TABLE IF NOT EXISTS indexes (
  db           VARCHAR(64) NOT NULL,
  tbl          VARCHAR(64) NOT NULL,
  idx          VARCHAR(64) NOT NULL,
  cnt          BIGINT UNSIGNED NOT NULL DEFAULT 0,
  PRIMARY KEY  (db, tbl, idx)
)
```

MAGIC_create_queries:

```
CREATE TABLE IF NOT EXISTS queries (
  query_id     BIGINT UNSIGNED NOT NULL,
  fingerprint  TEXT NOT NULL,
  sample       TEXT NOT NULL,
  PRIMARY KEY  (query_id)
)
```

MAGIC_create_tables:

```
CREATE TABLE IF NOT EXISTS tables (
  db           VARCHAR(64) NOT NULL,
  tbl          VARCHAR(64) NOT NULL,
  cnt          BIGINT UNSIGNED NOT NULL DEFAULT 0,
  PRIMARY KEY  (db, tbl)
)
```

MAGIC_create_index_usage:

```
CREATE TABLE IF NOT EXISTS index_usage (
  query_id      BIGINT UNSIGNED NOT NULL,
  db            VARCHAR(64) NOT NULL,
  tbl           VARCHAR(64) NOT NULL,
  idx           VARCHAR(64) NOT NULL,
  cnt           BIGINT UNSIGNED NOT NULL DEFAULT 1,
  UNIQUE INDEX  (query_id, db, tbl, idx)
)
```

MAGIC_create_index_alternatives:

```
CREATE TABLE IF NOT EXISTS index_alternatives (
  query_id      BIGINT UNSIGNED NOT NULL, -- This query used
  db            VARCHAR(64) NOT NULL,     -- this index, but...
  tbl           VARCHAR(64) NOT NULL,     --
  idx           VARCHAR(64) NOT NULL,     --
  alt_idx       VARCHAR(64) NOT NULL,     -- was an alternative
  cnt           BIGINT UNSIGNED NOT NULL DEFAULT 1,
  UNIQUE INDEX  (query_id, db, tbl, idx, alt_idx),
  INDEX         (db, tbl, idx),
```

```
  INDEX          (db, tbl, alt_idx)
)
```

The following are some queries you can run against these tables to answer common questions you might have. Each query is also created as a view (with MySQL v5.0 and newer) if :option:`--[no]create-views` is true (it is by default). The view names are the strings after the MAGIC_view_ prefix.

Question: which queries sometimes use different indexes, and what fraction of the time is each index chosen? MAGIC_view_query_uses_several_indexes:

```
SELECT iu.query_id, CONCAT_WS('.', iu.db, iu.tbl, iu.idx) AS idx,
   variations, iu.cnt, iu.cnt / total_cnt * 100 AS pct
FROM index_usage AS iu
   INNER JOIN (
      SELECT query_id, db, tbl, SUM(cnt) AS total_cnt,
        COUNT(*) AS variations
      FROM index_usage
      GROUP BY query_id, db, tbl
      HAVING COUNT(*) > 1
   ) AS qv USING(query_id, db, tbl);
```

Question: which indexes have lots of alternatives, i.e. are chosen instead of other indexes, and for what queries? MAGIC_view_index_has_alternates:

```
SELECT CONCAT_WS('.', db, tbl, idx) AS idx_chosen,
   GROUP_CONCAT(DISTINCT alt_idx) AS alternatives,
   GROUP_CONCAT(DISTINCT query_id) AS queries, SUM(cnt) AS cnt
FROM index_alternatives
GROUP BY db, tbl, idx
HAVING COUNT(*) > 1;
```

Question: which indexes are considered as alternates for other indexes, and for what queries? MAGIC_view_index_alternates:

```
SELECT CONCAT_WS('.', db, tbl, alt_idx) AS idx_considered,
   GROUP_CONCAT(DISTINCT idx) AS alternative_to,
   GROUP_CONCAT(DISTINCT query_id) AS queries, SUM(cnt) AS cnt
FROM index_alternatives
GROUP BY db, tbl, alt_idx
HAVING COUNT(*) > 1;
```

Question: which of those are never chosen by any queries, and are therefore superfluous? MAGIC_view_unused_index_alternates:

```
SELECT CONCAT_WS('.', i.db, i.tbl, i.idx) AS idx,
   alt.alternative_to, alt.queries, alt.cnt
FROM indexes AS i
   INNER JOIN (
      SELECT db, tbl, alt_idx, GROUP_CONCAT(DISTINCT idx) AS alternative_to,
         GROUP_CONCAT(DISTINCT query_id) AS queries, SUM(cnt) AS cnt
      FROM index_alternatives
      GROUP BY db, tbl, alt_idx
      HAVING COUNT(*) > 1
   ) AS alt ON i.db = alt.db AND i.tbl = alt.tbl
     AND i.idx = alt.alt_idx
WHERE i.cnt = 0;
```

Question: given a table, which indexes were used, by how many queries, with how many distinct fingerprints?

Were there alternatives? Which indexes were not used? You can edit the following query's SELECT list to also see the query IDs in question. MAGIC_view_index_usage:

```
SELECT i.idx, iu.usage_cnt, iu.usage_total,
   ia.alt_cnt, ia.alt_total
FROM indexes AS i
   LEFT OUTER JOIN (
      SELECT db, tbl, idx, COUNT(*) AS usage_cnt,
         SUM(cnt) AS usage_total, GROUP_CONCAT(query_id) AS used_by
      FROM index_usage
      GROUP BY db, tbl, idx
   ) AS iu ON i.db=iu.db AND i.tbl=iu.tbl AND i.idx = iu.idx
   LEFT OUTER JOIN (
      SELECT db, tbl, idx, COUNT(*) AS alt_cnt,
         SUM(cnt) AS alt_total,
         GROUP_CONCAT(query_id) AS alt_queries
      FROM index_alternatives
      GROUP BY db, tbl, idx
   ) AS ia ON i.db=ia.db AND i.tbl=ia.tbl AND i.idx = ia.idx;
```

Question: which indexes on a given table are vital for at least one query (there is no alternative)? MAGIC_view_required_indexes:

```
SELECT i.db, i.tbl, i.idx, no_alt.queries
FROM indexes AS i
   INNER JOIN (
      SELECT iu.db, iu.tbl, iu.idx,
         GROUP_CONCAT(iu.query_id) AS queries
      FROM index_usage AS iu
         LEFT OUTER JOIN index_alternatives AS ia
            USING(db, tbl, idx)
      WHERE ia.db IS NULL
      GROUP BY iu.db, iu.tbl, iu.idx
   ) AS no_alt ON no_alt.db = i.db AND no_alt.tbl = i.tbl
      AND no_alt.idx = i.idx
ORDER BY i.db, i.tbl, i.idx, no_alt.queries;
```

--set-vars

type: Array

Set the MySQL 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.

--tables

short form: -t; type: hash

Only get indexes from this comma-separated list of tables.

--tables-regex
type: string

Only get indexes from tables whose names match this Perl regex.

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

User for login if not current user.

--version
Show version and exit.

--[no]version-check
default: yes

Check for the latest version of Percona Toolkit, MySQL, and other programs.

This is a standard "check for updates automatically" feature, with two additional features. First, the tool checks the version of other programs on the local system in addition to its own version. For example, it checks the version of every MySQL server it connects to, Perl, and the Perl module DBD::mysql. Second, it checks for and warns about versions with known problems. For example, MySQL 5.5.25 had a critical bug and was re-released as 5.5.25a.

Any updates or known problems are printed to STDOUT before the tool's normal output. This feature should never interfere with the normal operation of the tool.

For more information, visit https://www.percona.com/version-check.

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

  Database to connect to.

- 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.

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 pt-index-usage ... > FILE 2>&1
```

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

SYSTEM REQUIREMENTS

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

BUGS

For a list of known bugs, see http://www.percona.com/bugs/pt-index-usage.

Please report bugs at https://bugs.launchpad.net/percona-toolkit. Include the following information in your bug report:

- Complete command-line used to run the tool
- Tool *--version*
- MySQL version of all servers involved
- Output from the tool including STDERR
- Input files (log/dump/config files, etc.)

If possible, include debugging output by running the tool with `PTDEBUG`; see "ENVIRONMENT".

DOWNLOADING

Visit http://www.percona.com/software/percona-toolkit/ to download the latest release of Percona Toolkit. Or, get the latest release from the command line:

```
wget percona.com/get/percona-toolkit.tar.gz

wget percona.com/get/percona-toolkit.rpm

wget percona.com/get/percona-toolkit.deb
```

You can also get individual tools from the latest release:

```
wget percona.com/get/TOOL
```

Replace `TOOL` with the name of any tool.

AUTHORS

Baron Schwartz and Daniel Nichter

ABOUT PERCONA TOOLKIT

This tool is part of Percona Toolkit, a collection of advanced command-line tools for MySQL developed by Percona. 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. Visit http://www.percona.com/software/ to learn about other free, open-source software from Percona.

COPYRIGHT, LICENSE, AND WARRANTY

VERSION

pt-index-usage 3.0.4

CHAPTER

FOURTEEN

PT-IOPROFILE

NAME

pt-ioprofile - Watch process IO and print a table of file and I/O activity.

SYNOPSIS

Usage

```
pt-ioprofile [OPTIONS] [FILE]
```

pt-ioprofile does two things: 1) get lsof+strace for -s seconds, 2) aggregate the result. If you specify a FILE, then step 1) is not performed.

RISKS

WARNING: **pt-ioprofile** freezes the server and may crash the process, or make it perform badly after detaching, or leave it in a sleeping state! 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

  pt-ioprofile should be considered an intrusive tool, and should not be used on production servers unless you understand and accept the risks.

DESCRIPTION

pt-ioprofile uses `strace` and `lsof` to watch a process's IO and print out a table of files and I/O activity. By default, it watches the mysqld process for 30 seconds. The output is like:

```
Tue Dec 27 15:33:57 PST 2011
Tracing process ID 1833
     total       read      write      lseek  ftruncate filename
  0.000150   0.000029   0.000068   0.000038   0.000015 /tmp/ibBE5opS
```

You probably need to run this tool as root.

pt-ioprofile works by attaching `strace` to the process using `ptrace()`, which will make it run very slowly until `strace` detaches. In addition to freezing the server, there is some risk of the process crashing or performing badly after `strace` detaches from it, or of `strace` not detaching cleanly and leaving the process in a sleeping state. As a result, this should be considered an intrusive tool, and should not be used on production servers unless you are comfortable with that.

OPTIONS

--aggregate
short form: -a; type: string; default: sum

The aggregate function, either `sum` or `avg`.

If sum, then each cell will contain the sum of the values in it. If avg, then each cell will contain the average of the values in it.

--cell
short form: -c; type: string; default: times

The cell contents.

Valid values are:

```
VALUE  CELLS CONTAIN
=====  ======================
count  Count of I/O operations
sizes  Sizes of I/O operations
times  I/O operation timing
```

--group-by
short form: -g; type: string; default: filename

The group-by item.

Valid values are:

```
VALUE     GROUPING
=====     ====================================
all       Summarize into a single line of output
filename  One line of output per filename
pid       One line of output per process ID
```

--help
Print help and exit.

--profile-pid
short form: -p; type: int

The PID to profile, overrides *--profile-process*.

--profile-process
short form: -b; type: string; default: mysqld

The process name to profile.

--run-time
type: int; default: 30

How long to profile.

--save-samples
type: string

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

--version
Print the tool's version and exit.

ENVIRONMENT

This tool does not use any environment variables.

SYSTEM REQUIREMENTS

This tool requires the Bourne shell (*/bin/sh*).

BUGS

For a list of known bugs, see http://www.percona.com/bugs/pt-ioprofile.

Please report bugs at https://bugs.launchpad.net/percona-toolkit. Include the following information in your bug report:

- Complete command-line used to run the tool
- Tool `--version`
- MySQL version of all servers involved
- Output from the tool including STDERR
- Input files (log/dump/config files, etc.)

If possible, include debugging output by running the tool with `PTDEBUG`; see "ENVIRONMENT".

DOWNLOADING

Visit http://www.percona.com/software/percona-toolkit/ to download the latest release of Percona Toolkit. Or, get the latest release from the command line:

```
wget percona.com/get/percona-toolkit.tar.gz

wget percona.com/get/percona-toolkit.rpm

wget percona.com/get/percona-toolkit.deb
```

You can also get individual tools from the latest release:

```
wget percona.com/get/TOOL
```

Replace `TOOL` with the name of any tool.

AUTHORS

Baron Schwartz

ABOUT PERCONA TOOLKIT

This tool is part of Percona Toolkit, a collection of advanced command-line tools for MySQL developed by Percona. 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. Visit http://www.percona.com/software/ to learn about other free, open-source software from Percona.

COPYRIGHT, LICENSE, AND WARRANTY

VERSION

pt-ioprofile 3.0.4

CHAPTER

FIFTEEN

PT-KILL

NAME

pt-kill - Kill MySQL queries that match certain criteria.

SYNOPSIS

Usage

```
pt-kill [OPTIONS] [DSN]
```

pt-kill kills MySQL connections. **pt-kill** connects to MySQL 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 -, **pt-kill** reads from STDIN.

Kill queries running longer than 60s:

```
pt-kill --busy-time 60 --kill
```

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

```
pt-kill --busy-time 60 --print
```

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

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

Print all login processes:

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

See which queries in the processlist right now would match:

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

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

DESCRIPTION

pt-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 **pt-kill** connects to MySQL to get queries from SHOW PROCESSLIST. Alternatively, it can read SHOW PROCESSLIST output from files. In this case, **pt-kill** does not connect to MySQL 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 **pt-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.

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.

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, **pt-kill** has no control or interaction with the command.

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 MySQL.

--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 MySQL. Any other value sets binmode on STDOUT without the utf8 layer, and runs SET NAMES after connecting to MySQL.

--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 mysql 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 **pt-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, **pt-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). **pt-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 **pt-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
   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 **pt-kill** runs the MySQL 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 **pt-kill** runs forever, or until its process is killed or stopped by the creation of a *--sentinel* file. If this option is specified, **pt-kill** runs for the specified amount of time and sleeps *--interval* seconds between each check of the PROCESSLIST.

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

Exit if this file exists.

The presence of the file specified by *--sentinel* will cause all running instances of **pt-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 MySQL 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 **pt-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.

--[no]version-check
default: yes

Check for the latest version of Percona Toolkit, MySQL, and other programs.

This is a standard "check for updates automatically" feature, with two additional features. First, the tool checks the version of other programs on the local system in addition to its own version. For example, it checks the version of every MySQL server it connects to, Perl, and the Perl module DBD::mysql. Second, it checks for and warns about versions with known problems. For example, MySQL 5.5.25 had a critical bug and was re-released as 5.5.25a.

Any updates or known problems are printed to STDOUT before the tool's normal output. This feature should never interfere with the normal operation of the tool.

For more information, visit https://www.percona.com/version-check.

--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 MySQL or system information before it's killed.

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 **pt-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 http://dev.mysql.com/doc/refman/5.1/en/thread-commands.html 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 http://dev.mysql.com/doc/refman/5.1/en/general-thread-states.html 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.

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.

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 **pt-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, **pt-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 **pt-kill** is passed to the command.

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

--kill
group: Actions

Kill the connection for matching queries.

This option makes **pt-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 **pt-kill** matched and killed a query.

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

--kill-query
group: Actions

Kill matching queries.

This option makes **pt-kill** kill matching queries. This requires MySQL 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*.

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: mysql_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.

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 pt-kill ... > FILE 2>&1
```

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

SYSTEM REQUIREMENTS

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

BUGS

For a list of known bugs, see http://www.percona.com/bugs/pt-kill.

Please report bugs at https://bugs.launchpad.net/percona-toolkit. Include the following information in your bug report:

- Complete command-line used to run the tool
- Tool *--version*
- MySQL version of all servers involved
- Output from the tool including STDERR
- Input files (log/dump/config files, etc.)

If possible, include debugging output by running the tool with `PTDEBUG`; see "ENVIRONMENT".

DOWNLOADING

Visit http://www.percona.com/software/percona-toolkit/ to download the latest release of Percona Toolkit. Or, get the latest release from the command line:

```
wget percona.com/get/percona-toolkit.tar.gz

wget percona.com/get/percona-toolkit.rpm

wget percona.com/get/percona-toolkit.deb
```

You can also get individual tools from the latest release:

```
wget percona.com/get/TOOL
```

Replace `TOOL` with the name of any tool.

AUTHORS

Baron Schwartz and Daniel Nichter

ABOUT PERCONA TOOLKIT

This tool is part of Percona Toolkit, a collection of advanced command-line tools for MySQL developed by Percona. 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. Visit http://www.percona.com/software/ to learn about other free, open-source software from Percona.

COPYRIGHT, LICENSE, AND WARRANTY

VERSION

pt-kill 3.0.4

CHAPTER

SIXTEEN

PT-MEXT

NAME

pt-mext - Look at many samples of MySQL `SHOW GLOBAL STATUS` side-by-side.

SYNOPSIS

Usage

```
pt-mext [OPTIONS] -- COMMAND
```

pt-mext columnizes repeated output from a program like mysqladmin extended.

Get output from `mysqladmin`:

```
pt-mext -r -- mysqladmin ext -i10 -c3
```

Get output from a file:

```
pt-mext -r -- cat mysqladmin-output.txt
```

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

DESCRIPTION

pt-mext 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.

OPTIONS

--help
Show help and exit.

--relative
short form: -r

Subtract each column from the previous column.

--version
Show version and exit.

ENVIRONMENT

This tool does not use any environment variables.

SYSTEM REQUIREMENTS

This tool requires the Bourne shell (*/bin/sh*) and the seq program.

BUGS

For a list of known bugs, see http://www.percona.com/bugs/pt-mext.

Please report bugs at https://bugs.launchpad.net/percona-toolkit. Include the following information in your bug report:

- Complete command-line used to run the tool
- Tool *--version*
- MySQL version of all servers involved
- Output from the tool including STDERR
- Input files (log/dump/config files, etc.)

If possible, include debugging output by running the tool with `PTDEBUG`; see "ENVIRONMENT".

DOWNLOADING

Visit http://www.percona.com/software/percona-toolkit/ to download the latest release of Percona Toolkit. Or, get the latest release from the command line:

```
wget percona.com/get/percona-toolkit.tar.gz

wget percona.com/get/percona-toolkit.rpm

wget percona.com/get/percona-toolkit.deb
```

You can also get individual tools from the latest release:

```
wget percona.com/get/TOOL
```

Replace `TOOL` with the name of any tool.

AUTHORS

Baron Schwartz

ABOUT PERCONA TOOLKIT

This tool is part of Percona Toolkit, a collection of advanced command-line tools for MySQL developed by Percona. 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. Visit http://www.percona.com/software/ to learn about other free, open-source software from Percona.

COPYRIGHT, LICENSE, AND WARRANTY

VERSION

pt-mext 3.0.4

CHAPTER

SEVENTEEN

PT-MONGODB-QUERY-DIGEST

`pt-mongodb-query-digest` reports query usage statistics by aggregating queries from MongoDB query profiler.

Usage

```
pt-mongodb-query-digest [OPTIONS]
```

It runs the following command:

```
db.getSiblingDB("samples").system.profile.find({"op":{"$nin":["getmore", "delete"]}});
```

Then the results are grouped by fingerprint and namespace (database.collection). The fingerprint is calculated as a sorted list of keys in the document with a maximum depth level of 10. By default, the results are sorted by ascending query count.

Note: `pt-mongodb-query-digest` cannot collect statistics from MongoDB instances that require connection via SSL. Support for SSL will be added in the future.

Options

-?, --help Show help and exit

-a, --authenticationDatabase Specifies the database used to establish credentials and privileges with a MongoDB server. By default, the `admin` database is used.

-c, --no-version-check Don't check for updates

-d, --database Specifies which database to profile

-l, --log-level Specifies the log level: `panic`, `fatal`, `error`, `warn`, `info`, `debug error`

-n, --limit Limits the number of queries to show

-o, --order-by Specifies the sorting order using fields: `count`, `ratio`, `query-time`, `docs-scanned`, `docs-returned`.

Adding a hypen (`-`) in front of a field denotes reverse order. For example: `--order-by="count,-ratio"`.

-p, --password Specifies the password to use when connecting to a server with authentication enabled.

Do not add a space between the option and its value: `-p<password>`.

If you specify the option without any value, you will be prompted for the password.

-u, --user Specifies the user name for connecting to a server with authentication enabled.

-v, --version Show version and exit

Output Example

```
# Query 2:  0.00 QPS, ID 1a6443c2db9661f3aad8edb6b877e45d
# Ratio    1.00  (docs scanned/returned)
# Time range: 2017-01-11 12:58:26.519 -0300 ART to 2017-01-11 12:58:26.686 -0300 ART
# Attribute            pct     total        min         max        avg         95%
↪    stddev      median
# ==================   ===   ========    ========    ========    ========    ========
↪    =======    ========
# Count (docs)                    36
# Exec Time ms           0          0           0           0           0           0
↪         0           0
# Docs Scanned           0    148.00        0.00       74.00        4.11       74.00
↪     16.95        0.00
# Docs Returned          2    148.00        0.00       74.00        4.11       74.00
↪     16.95        0.00
# Bytes recv             0      2.11M     215.00        1.05M      58.48K       1.05M
↪    240.22K     215.00
# String:
# Namespaces          samples.col1
# Fingerprint         $gte,$lt,$meta,$sortKey,filter,find,projection,shardVersion,
↪sort,user_id,user_id
```

CHAPTER

EIGHTEEN

PT-MONGODB-SUMMARY

`pt-mongodb-summary` collects information about a MongoDB cluster. It collects information from several sources to provide an overview of the cluster.

Usage

```
pt-mongodb-summary [OPTIONS] [HOST[:PORT]]
```

By default, if you run `pt-mongodb-summary` without any parameters, it will try to connect to `localhost` on port `27017`. The program collects information about MongoDB instances by running administration commands and formatting the output.

Note: `pt-mongodb-summary` requires to be run by user with the `clusterAdmin` or `root` built-in roles.

Note: `pt-mongodb-summary` cannot collect statistics from MongoDB instances that require connection via SSL. Support for SSL will be added in the future.

Options

-a, --auth-db Specifies the database used to establish credentials and privileges with a MongoDB server. By default, the `admin` database is used.

-p, --password Specifies the password to use when connecting to a server with authentication enabled.

Do not add a space between the option and its value: `-p<password>`.

If you specify the option without any value, `pt-mongodb-summary` will ask for password interactively.

-u, --user Specifies the user name for connecting to a server with authentication enabled.

Output Example

```
# Instances ##################################################################
↪##########
ID    Host                      Type                                ReplSet
  0 localhost:17001             PRIMARY                               r1
  1 localhost:17002             SECONDARY                             r1
```

```
  2 localhost:17003               SECONDARY                                r1
  0 localhost:18001               PRIMARY                                  r2
  1 localhost:18002               SECONDARY                                r2
  2 localhost:18003               SECONDARY                                r2

# This host
# Mongo Executable ############################################################
↪##########
       Path to executable | /home/karl/tmp/MongoDB32Labs/3.0/bin/mongos
# Report On 0 #########################################
                     User | karl
                   PID Owner | mongos
                        Time | 2016-10-30 00:18:49 -0300 ART
                    Hostname | karl-HP-ENVY
                     Version | 3.0.11
                    Built On | Linux x86_64
                     Started | 2016-10-30 00:18:49 -0300 ART
                     Datadir | /data/db
               Process Type | mongos

# Running Ops #################################################################
↪##########

Type         Min        Max        Avg
Insert           0          0          0/5s
Query            0          0          0/5s
Update           0          0          0/5s
Delete           0          0          0/5s
GetMore          0          0          0/5s
Command          0         22         16/5s

# Security ####################################################################
↪##########
Users 0
Roles 0
Auth  disabled
SSL   disabled


# Oplog #######################################################################
↪##########
Oplog Size     18660 Mb
Oplog Used     55 Mb
Oplog Length   0.91 hours
Last Election  2016-10-30 00:18:44 -0300 ART


# Cluster wide ################################################################
↪##########
            Databases: 3
          Collections: 17
  Sharded Collections: 1
Unsharded Collections: 16
    Sharded Data Size: 68 GB
  Unsharded Data Size: 0 KB
# Balancer (per day)
              Success: 6
               Failed: 0
```

```
              Splits: 0
               Drops: 0
```

Sections

Output is separated into the following sections:

- **Instances**

  This section lists all hosts connected to the current MongoDB instance. For this, `pt-mongodb-summary` runs the `listShards` command and then the `replSetGetStatus` on every instance to collect its ID, type, and replica set.

- **This host**

  This section provides an overview of the current MongoDB instance and the underlying OS. For this, `pt-mongodb-summary` groups information collected from `hostInfo`, `getCmdLineOpts`, `serverStatus`, and the OS process (by process ID).

- **Running Ops**

  This section provides minimum, maximum, and average operation counters for `insert`, `query`, `update`, `delete`, `getMore`, and `command` operations. For this, `pt-mongodb-summary` runs the `serverStatus` command 5 times at regular intervals (every second).

- **Security**

  This section provides information about the security settings. For this, `pt-mongodb-summary`, parses `getCmdLineOpts` output and queries the `admin.system.users` and `admin.system.roles` collections.

- **Oplog**

  This section contains details about the MongoDB operations log (oplog). For this, `pt-mongodb-summary` collects statistics from the oplog on every host in the cluster, and returns those with the smallest `TimeDiffHours` value.

- **Cluster wide**

  This section provides information about the number of sharded and unsharded databases, collections, and their size. For this, `pt-mongodb-summary` runs the `listDatabases` command and then runs `collStats` for every collection in every database.

CHAPTER

NINETEEN

PT-MYSQL-SUMMARY

NAME

pt-mysql-summary - Summarize MySQL information nicely.

SYNOPSIS

Usage

```
pt-mysql-summary [OPTIONS]
```

pt-mysql-summary conveniently summarizes the status and configuration of a MySQL 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.

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

DESCRIPTION

pt-mysql-summary works by connecting to a MySQL 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 MySQL, such as the following:

```
pt-mysql-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 MySQL 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 MySQL database server running on another server.

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:

```
# Percona Toolkit MySQL 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 MySQL 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 MySQL 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 | 5.5.20-log MySQL Community Server (GPL)
                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 MySQL instance: version, uptime, and other very basic parameters. The Time output is generated from the MySQL 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 Percona 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 Percona Server and whether they are enabled or not. In the example shown, the server is standard MySQL, not Percona Server, so the features are generally not supported.

```
# 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

                      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 `mysqldump --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
              NDB 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
              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).

```
# 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
             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                             = mysql-bin
relay_log                           = mysql-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.

OPTIONS

All options after – are passed to `mysql`.

--all-databases
mysqldump and summarize all databases. See *--databases*.

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

--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

mysqldump 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 mysql 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 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.

ENVIRONMENT

This tool does not use any environment variables.

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.

BUGS

For a list of known bugs, see http://www.percona.com/bugs/pt-mysql-summary.

Please report bugs at https://bugs.launchpad.net/percona-toolkit. Include the following information in your bug report:

- Complete command-line used to run the tool
- Tool *--version*
- MySQL version of all servers involved
- Output from the tool including STDERR
- Input files (log/dump/config files, etc.)

If possible, include debugging output by running the tool with `PTDEBUG`; see "ENVIRONMENT".

DOWNLOADING

Visit http://www.percona.com/software/percona-toolkit/ to download the latest release of Percona Toolkit. Or, get the latest release from the command line:

```
wget percona.com/get/percona-toolkit.tar.gz

wget percona.com/get/percona-toolkit.rpm

wget percona.com/get/percona-toolkit.deb
```

You can also get individual tools from the latest release:

```
wget percona.com/get/TOOL
```

Replace `TOOL` with the name of any tool.

AUTHORS

Baron Schwartz, Brian Fraser, and Daniel Nichter

ABOUT PERCONA TOOLKIT

This tool is part of Percona Toolkit, a collection of advanced command-line tools for MySQL developed by Percona. 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. Visit http://www.percona.com/software/ to learn about other free, open-source software from Percona.

COPYRIGHT, LICENSE, AND WARRANTY

VERSION

pt-mysql-summary 3.0.4

CHAPTER

TWENTY

PT-ONLINE-SCHEMA-CHANGE

NAME

pt-online-schema-change - ALTER tables without locking them.

SYNOPSIS

Usage

```
pt-online-schema-change [OPTIONS] DSN
```

pt-online-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:

```
pt-online-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:

```
pt-online-schema-change --alter "ENGINE=InnoDB" D=sakila,t=actor
```

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

DESCRIPTION

pt-online-schema-change emulates the way that MySQL 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.

pt-online-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 MySQL 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 MySQL 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 "Percona XtraDB Cluster" nodes.

Percona XtraDB Cluster

pt-online-schema-change works with Percona XtraDB Cluster (PXC) 5.5.28-23.7 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.

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
```

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 MySQL 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 MySQL, **pt-online-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 MySQL 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. MySQL 5.1 and newer handle this case correctly, but testing reproduces the error 5% of the time with MySQL 5.0.

This is a MySQL bug, similar to http://bugs.mysql.com/bug.php?id=45694, but there is no fix or workaround in MySQL 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 MySQL 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 MySQL's `ALTER TABLE` is typically much faster than the external process of copying rows.

Due to a limitation in MySQL, 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, MySQL 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 MySQL 5.6 and newer, and `innodb_stats_persistent` is enabled. Specify the option explicitly to enable or disable it regardless of MySQL 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 MySQL.

--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 MySQL. Any other value sets binmode on STDOUT without the utf8 layer, and runs SET NAMES after connecting to MySQL.

--`[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.

--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 MySQL chooses a bad execution plan. These include the queries to determine chunk boundaries and the chunk queries themselves. If it appears that MySQL 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 MySQL intends to use the desired index to access the rows. If MySQL chooses a different index, the tool considers the query unsafe.

The tool also checks how much of the index MySQL 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 MySQL 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 MySQL, 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 MySQL 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 MySQL 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 MySQL 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 **pt-online-schema-change** to run if the specified statement for *--alter* is trying to add an unique index. Since **pt-online-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 **pt-online-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, **pt-online-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.

--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 MySQL 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 MySQL 5.7.2, 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 **pt-online-schema-change** even if the table already has its own triggers. If this option is enabled, **pt-online-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 MySQL 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 **pt-online-schema-change**. This requires a good knowledge of Perl and Percona Toolkit conventions, which are beyond this scope of this documentation. Please contact Percona 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.

--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 MySQL 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 MySQL 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 MySQL 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 MySQL 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 MySQL
```

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.

--[no]version-check
default: yes

Check for the latest version of Percona Toolkit, MySQL, and other programs.

This is a standard "check for updates automatically" feature, with two additional features. First, the tool checks the version of other programs on the local system in addition to its own version. For example, it checks the version of every MySQL server it connects to, Perl, and the Perl module DBD::mysql. Second, it checks for and warns about versions with known problems. For example, MySQL 5.5.25 had a critical bug and was re-released as 5.5.25a.

Any updates or known problems are printed to STDOUT before the tool's normal output. This feature should never interfere with the normal operation of the tool.

For more information, visit https://www.percona.com/version-check.

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";
}

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";
}

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";
}

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 Percona if you have questions or need help.

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

  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.

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 pt-online-schema-change ... > FILE 2>&1
```

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

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 MySQL 5.0.2 and newer versions, because earlier versions do not support triggers.

BUGS

For a list of known bugs, see http://www.percona.com/bugs/pt-online-schema-change.

Please report bugs at https://bugs.launchpad.net/percona-toolkit. Include the following information in your bug report:

- Complete command-line used to run the tool
- Tool *--version*
- MySQL version of all servers involved
- Output from the tool including STDERR
- Input files (log/dump/config files, etc.)

If possible, include debugging output by running the tool with PTDEBUG; see "ENVIRONMENT".

DOWNLOADING

Visit http://www.percona.com/software/percona-toolkit/ to download the latest release of Percona Toolkit. Or, get the latest release from the command line:

```
wget percona.com/get/percona-toolkit.tar.gz

wget percona.com/get/percona-toolkit.rpm

wget percona.com/get/percona-toolkit.deb
```

You can also get individual tools from the latest release:

```
wget percona.com/get/TOOL
```

Replace TOOL with the name of any tool.

AUTHORS

Daniel Nichter and Baron Schwartz

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.

ABOUT PERCONA TOOLKIT

This tool is part of Percona Toolkit, a collection of advanced command-line tools for MySQL developed by Percona. 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. Visit http://www.percona.com/software/ to learn about other free, open-source software from Percona.

COPYRIGHT, LICENSE, AND WARRANTY

VERSION

pt-online-schema-change 3.0.4

CHAPTER

TWENTYONE

PT-PMP

NAME

pt-pmp - Aggregate GDB stack traces for a selected program.

SYNOPSIS

Usage

```
pt-pmp [OPTIONS] [FILES]
```

pt-pmp 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.

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

DESCRIPTION

pt-pmp 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.

pt-pmp 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 MySQL profiling tool, this means that MySQL 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.

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.

ENVIRONMENT

This tool does not use any environment variables.

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.

BUGS

For a list of known bugs, see http://www.percona.com/bugs/pt-pmp.

Please report bugs at https://bugs.launchpad.net/percona-toolkit. Include the following information in your bug report:

- Complete command-line used to run the tool
- Tool *--version*
- MySQL version of all servers involved
- Output from the tool including STDERR
- Input files (log/dump/config files, etc.)

If possible, include debugging output by running the tool with `PTDEBUG`; see "ENVIRONMENT".

DOWNLOADING

Visit http://www.percona.com/software/percona-toolkit/ to download the latest release of Percona Toolkit. Or, get the latest release from the command line:

```
wget percona.com/get/percona-toolkit.tar.gz

wget percona.com/get/percona-toolkit.rpm

wget percona.com/get/percona-toolkit.deb
```

You can also get individual tools from the latest release:

```
wget percona.com/get/TOOL
```

Replace `TOOL` with the name of any tool.

AUTHORS

Baron Schwartz, based on a script by Domas Mituzas (http://poormansprofiler.org/)

ABOUT PERCONA TOOLKIT

This tool is part of Percona Toolkit, a collection of advanced command-line tools for MySQL developed by Percona. 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. Visit http://www.percona.com/software/ to learn about other free, open-source software from Percona.

COPYRIGHT, LICENSE, AND WARRANTY

VERSION

pt-pmp 3.0.4

CHAPTER

TWENTYTWO

PT-QUERY-DIGEST

NAME

pt-query-digest - Analyze MySQL queries from logs, processlist, and tcpdump.

SYNOPSIS

Usage

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

pt-query-digest analyzes MySQL queries from slow, general, and binary log files. It can also analyze queries from `SHOW PROCESSLIST` and MySQL 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`:

```
pt-query-digest slow.log
```

Report the slowest queries from the processlist on host1:

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

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

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

pt-query-digest --type tcpdump mysql.tcp.txt
```

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

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

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

DESCRIPTION

pt-query-digest is a sophisticated but easy to use tool for analyzing MySQL queries. It can analyze queries from MySQL slow, general, and binary logs. (Binary logs must first be converted to text, see *--type*). It can also use `SHOW PROCESSLIST` and MySQL 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, **pt-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.

ATTRIBUTES

pt-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 (for example, you may have a server with the Percona patches).

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.

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:

```
pt-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*.

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 **pt-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.

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 **pt-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,

```
pt-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 mysqldump 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.

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 MySQL.

--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, **pt-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 MySQL. Any other value sets binmode on STDOUT without the utf8 layer, and runs SET NAMES after connecting to MySQL.

--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 mysql 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 **pt-query-digest** with the following option:

```
:program:`pt-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 **pt-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 MySQL 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 **pt-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, **pt-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). **pt-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 MySQL 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. **pt-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 `percona_schema.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`).

pt-query-digest inspects the columns in the table. The table must have at least the following columns:

```
CREATE TABLE query_review_history (
  checksum     BIGINT UNSIGNED NOT NULL,
  sample       TEXT 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             BIGINT UNSIGNED NOT NULL,
  sample               TEXT 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,
  -- Percona extended slowlog attributes
  -- http://www.percona.com/docs/wiki/patches:slow_extended
  Rows_affected_sum              FLOAT,
  Rows_affected_min              FLOAT,
  Rows_affected_max              FLOAT,
  Rows_affected_pct_95           FLOAT,
  Rows_affected_stddev           FLOAT,
  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,
  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.

--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.

--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 **`pt-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    MySQL slow log
json       JSON, on array per query class
json-anon  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, **pt-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:`pt-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 `percona_schema.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 **pt-query-digest**.

```
CREATE TABLE IF NOT EXISTS query_review (
   checksum     BIGINT UNSIGNED 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*, **pt-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 **pt-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, **pt-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 **pt-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:`pt-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 MySQL 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 **pt-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, MySQL-style date:
  YYMMDD [HH:MM:SS]
* Any time expression evaluated by MySQL:
  CURRENT_DATE - INTERVAL 7 DAY
```

If you give a MySQL 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 **pt-query-digest** can connect to MySQL to evaluate the expression.

The MySQL 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 **pt-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:`pt-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 mysqlbinlog.

For example:

```
mysqlbinlog mysql-bin.000441 > mysql-bin.000441.txt

:program:`pt-query-digest` --type binlog mysql-bin.000441.txt
```

genlog

Parse a MySQL 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 MySQL slow log format.

tcpdump

Inspect network packets and decode the MySQL client protocol, extracting queries and responses from it.

pt-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; **pt-query-digest** does not do it for you. Then you send this to **pt-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 \
  > mysql.tcp.txt
:program:`pt-query-digest` --type tcpdump mysql.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 **pt-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 MySQL 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 MySQL server that is not running on port 3306, see *--watch-server*.

Also note that **pt-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 **pt-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 MySQL 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 **pt-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]version-check
default: yes

Check for the latest version of Percona Toolkit, MySQL, and other programs.

This is a standard "check for updates automatically" feature, with two additional features. First, the tool checks the version of other programs on the local system in addition to its own version. For example, it checks the version of every MySQL server it connects to, Perl, and the Perl module DBD::mysql. Second, it checks for and warns about versions with known problems. For example, MySQL 5.5.25 had a critical bug and was re-released as 5.5.25a.

Any updates or known problems are printed to STDOUT before the tool's normal output. This feature should never interfere with the normal operation of the tool.

For more information, visit https://www.percona.com/version-check.

--[no]vertical-format
default: yes

Output a trailing "G" in the reported SQL queries.

This makes the mysql client display the result using vertical format. Non-native MySQL clients like phpMyAdmin do not support this.

--watch-server
type: string

This option tells **pt-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, **pt-query-digest** watches all servers by looking for any IP address using port 3306 or "mysql". 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 **pt-query-digest** cannot auto-detect servers on port 3306 and also be told to watch a server on a non-standard port.

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 to use when connecting to MySQL.

- 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.

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 pt-query-digest ... > FILE 2>&1
```

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

SYSTEM REQUIREMENTS

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

BUGS

For a list of known bugs, see http://www.percona.com/bugs/pt-query-digest.

Please report bugs at https://bugs.launchpad.net/percona-toolkit. Include the following information in your bug report:

- Complete command-line used to run the tool
- Tool *--version*
- MySQL version of all servers involved
- Output from the tool including STDERR
- Input files (log/dump/config files, etc.)

If possible, include debugging output by running the tool with `PTDEBUG`; see "ENVIRONMENT".

DOWNLOADING

Visit http://www.percona.com/software/percona-toolkit/ to download the latest release of Percona Toolkit. Or, get the latest release from the command line:

```
wget percona.com/get/percona-toolkit.tar.gz

wget percona.com/get/percona-toolkit.rpm

wget percona.com/get/percona-toolkit.deb
```

You can also get individual tools from the latest release:

```
wget percona.com/get/TOOL
```

Replace `TOOL` with the name of any tool.

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:

```
$ pt-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]'
};
```

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.

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.

Percona Server adds many attributes to the slow log; see http://www.percona.com/docs/wiki/patches:slow_extended for more information.

TCPDUMP

These attributes are available when parsing *--type* tcpdump.

Error_no

The MySQL 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`.

PROCESSLIST

If using *--processlist*, an `id` attribute is available for the process ID, in addition to the common attributes.

AUTHORS

Baron Schwartz, Daniel Nichter, and Brian Fraser

ABOUT PERCONA TOOLKIT

This tool is part of Percona Toolkit, a collection of advanced command-line tools for MySQL developed by Percona. 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. Visit http://www.percona.com/software/ to learn about other free, open-source software from Percona.

COPYRIGHT, LICENSE, AND WARRANTY

VERSION

pt-query-digest 3.0.4

CHAPTER

TWENTYTHREE

PT-SHOW-GRANTS

NAME

pt-show-grants - Canonicalize and print MySQL grants so you can effectively replicate, compare and version-control them.

SYNOPSIS

Usage

```
pt-show-grants [OPTIONS] [DSN]
```

pt-show-grants shows grants (user privileges) from a MySQL server.

Examples

```
pt-show-grants

pt-show-grants --separate --revoke | diff othergrants.sql -
```

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

DESCRIPTION

pt-show-grants extracts, orders, and then prints grants for MySQL user accounts.

Why would you want this? There are several reasons.

The first is to easily replicate users from one server to another; you can simply extract the grants from the first server and pipe the output directly into another server.

The second use is to place your grants into version control. If you do a daily automated grant dump into version control, you'll get lots of spurious changesets for grants that don't change, because MySQL prints the actual grants out in a seemingly random order. For instance, one day it'll say

```
GRANT DELETE, INSERT, UPDATE ON `test`.* TO 'foo'@'%';
```

And then another day it'll say

```
GRANT INSERT, DELETE, UPDATE ON `test`.* TO 'foo'@'%';
```

The grants haven't changed, but the order has. This script sorts the grants within the line, between 'GRANT' and 'ON'. If there are multiple rows from SHOW GRANTS, it sorts the rows too, except that it always prints the row with the user's password first, if it exists. This removes three kinds of inconsistency you'll get from running SHOW GRANTS, and avoids spurious changesets in version control.

Third, if you want to diff grants across servers, it will be hard without "canonicalizing" them, which **pt-show-grants** does. The output is fully diff-able.

With the *--revoke*, *--separate* and other options, **pt-show-grants** also makes it easy to revoke specific privileges from users. This is tedious otherwise.

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 MySQL.

--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 MySQL. Any other value sets binmode on STDOUT without the utf8 layer, and runs SET NAMES after connecting to MySQL.

--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

The database to use for the connection.

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

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

--drop
Add DROP USER before each user in the output.

--flush
Add FLUSH PRIVILEGES after output.

You might need this on pre-4.1.1 servers if you want to drop a user completely.

--[no]header
default: yes

Print dump header.

The header precedes the dumped grants. It looks like:

```
-- Grants dumped by :program:`pt-show-grants` 1.0.19
-- Dumped from server Localhost via UNIX socket, MySQL 5.0.82-log at 2009-10-26␣
↪10:01:04
```

See also --[no]timestamp.

--help
Show help and exit.

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

Connect to host.

--ignore
type: array

Ignore this comma-separated list of users.

--only
type: array

Only show grants for this comma-separated list of users.

--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.

--revoke
Add REVOKE statements for each GRANT statement.

--separate
List each GRANT or REVOKE separately.

The default output from MySQL's SHOW GRANTS command lists many privileges on a single line. With *--flush*, places a FLUSH PRIVILEGES after each user, instead of once at the end of all the output.

--set-vars
type: Array

Set the MySQL 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]timestamp`
default: yes

Add timestamp to the dump header.

See also `--[no]header`.

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

User for login if not current user.

--version
Show version and exit.

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: mysql_socket; copy: yes

  Socket file to use for connection.

- u

  dsn: user; copy: yes

  User for login if not current user.

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 pt-show-grants ... > FILE 2>&1
```

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

SYSTEM REQUIREMENTS

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

BUGS

For a list of known bugs, see http://www.percona.com/bugs/pt-show-grants.

Please report bugs at https://bugs.launchpad.net/percona-toolkit. Include the following information in your bug report:

- Complete command-line used to run the tool
- Tool *--version*
- MySQL version of all servers involved
- Output from the tool including STDERR
- Input files (log/dump/config files, etc.)

If possible, include debugging output by running the tool with `PTDEBUG`; see "ENVIRONMENT".

DOWNLOADING

Visit http://www.percona.com/software/percona-toolkit/ to download the latest release of Percona Toolkit. Or, get the latest release from the command line:

```
wget percona.com/get/percona-toolkit.tar.gz

wget percona.com/get/percona-toolkit.rpm

wget percona.com/get/percona-toolkit.deb
```

You can also get individual tools from the latest release:

```
wget percona.com/get/TOOL
```

Replace `TOOL` with the name of any tool.

AUTHORS

Baron Schwartz

ABOUT PERCONA TOOLKIT

This tool is part of Percona Toolkit, a collection of advanced command-line tools for MySQL developed by Percona. 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. Visit http://www.percona.com/software/ to learn about other free, open-source software from Percona.

COPYRIGHT, LICENSE, AND WARRANTY

VERSION

pt-show-grants 3.0.4

CHAPTER

TWENTYFOUR

PT-SIFT

NAME

pt-sift - Browses files created by pt-stalk.

SYNOPSIS

Usage

```
pt-sift FILE|PREFIX|DIRECTORY
```

pt-sift browses files created by pt-stalk. If no options are given, the tool browses all pt-stalk files in `/var/lib/pt-stalk` 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/pt-stalk` (or the current working directory) with the same prefix. If a DIRECTORY is given, the tool browses all pt-stalk files in it.

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

DESCRIPTION

pt-sift downloads other tools that it might need, such as pt-diskstats, and then makes a list of the unique timestamp prefixes of all the files in the directory, as written by the pt-stalk 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 pt-diskstats 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 pt-mext 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.

OPTIONS

--help
Show help and exit.

--version
Show version and exit.

ENVIRONMENT

This tool does not use any environment variables.

SYSTEM REQUIREMENTS

This tool requires Bash v3 and the following programs: pt-diskstats, pt-pmp, pt-mext, and pt-align. If these programs are not in your PATH, they will be fetched from the Internet if curl is available.

BUGS

For a list of known bugs, see http://www.percona.com/bugs/pt-sift.

Please report bugs at https://bugs.launchpad.net/percona-toolkit. Include the following information in your bug report:

- Complete command-line used to run the tool
- Tool *--version*
- MySQL version of all servers involved
- Output from the tool including STDERR
- Input files (log/dump/config files, etc.)

If possible, include debugging output by running the tool with `PTDEBUG`; see "ENVIRONMENT".

DOWNLOADING

Visit http://www.percona.com/software/percona-toolkit/ to download the latest release of Percona Toolkit. Or, get the latest release from the command line:

```
wget percona.com/get/percona-toolkit.tar.gz

wget percona.com/get/percona-toolkit.rpm

wget percona.com/get/percona-toolkit.deb
```

You can also get individual tools from the latest release:

```
wget percona.com/get/TOOL
```

Replace `TOOL` with the name of any tool.

AUTHORS

Baron Schwartz

ABOUT PERCONA TOOLKIT

This tool is part of Percona Toolkit, a collection of advanced command-line tools for MySQL developed by Percona. 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. Visit http://www.percona.com/software/ to learn about other free, open-source software from Percona.

COPYRIGHT, LICENSE, AND WARRANTY

VERSION

pt-sift 3.0.4

CHAPTER

TWENTYFIVE

PT-SLAVE-DELAY

NAME

pt-slave-delay - Make a MySQL slave server lag behind its master.

SYNOPSIS

Usage

```
pt-slave-delay [OPTIONS] SLAVE_DSN [MASTER_DSN]
```

pt-slave-delay starts and stops a slave server as needed to make it lag behind the master. The SLAVE_DSN and MASTER_DSN use DSN syntax, and values are copied from the SLAVE_DSN to the MASTER_DSN if omitted.

To hold slavehost one minute behind its master for ten minutes:

```
pt-slave-delay --delay 1m --interval 15s --run-time 10m slavehost
```

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

DESCRIPTION

pt-slave-delay watches a slave and starts and stops its replication SQL thread as necessary to hold it at least as far behind the master as you request. In practice, it will typically cause the slave to lag between *--delay* and `--delay"+"--interval` behind the master.

It bases the delay on binlog positions in the slave's relay logs by default, so there is no need to connect to the master. This works well if the IO thread doesn't lag the master much, which is typical in most replication setups; the IO thread lag is usually milliseconds on a fast network. If your IO thread's lag is too large for your purposes, **pt-slave-delay** can also connect to the master for information about binlog positions.

If the slave's I/O thread reports that it is waiting for the SQL thread to free some relay log space, **pt-slave-delay** will automatically connect to the master to find binary log positions. If *--ask-pass* and *--daemonize* are given, it is possible that this could cause it to ask for a password while daemonized. In this case, it exits. Therefore, if you think your slave might encounter this condition, you should be sure to either specify *--use-master* explicitly when daemonizing, or don't specify *--ask-pass*.

The SLAVE_DSN and optional MASTER_DSN are both DSNs. See "DSN OPTIONS". Missing MASTER_DSN values are filled in with values from SLAVE_DSN, so you don't need to specify them in both places. **pt-slave-delay** reads all normal MySQL option files, such as ~/.my.cnf, so you may not need to specify username, password and other common options at all.

pt-slave-delay tries to exit gracefully by trapping signals such as Ctrl-C. You cannot bypass `--[no]continue` with a trappable signal.

PRIVILEGES

pt-slave-delay requires the following privileges: PROCESS, REPLICATION CLIENT, and SUPER.

OUTPUT

If you specify *--quiet*, there is no output. Otherwise, the normal output is a status message consisting of a timestamp and information about what **pt-slave-delay** is doing: starting the slave, stopping the slave, or just observing.

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 MySQL.

--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 MySQL. Any other value sets binmode on STDOUT without the utf8 layer, and runs SET NAMES after connecting to MySQL.

--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`
default: yes

Continue replication normally on exit. After exiting, restart the slave's SQL thread with no UNTIL condition, so it will run as usual and catch up to the master. This is enabled by default and works even if you terminate **pt-slave-delay** with Control-C.

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

--delay
type: time; default: 1h

How far the slave should lag its master.

--help
Show help and exit.

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

Connect to host.

--interval
type: time; default: 1m

How frequently **pt-slave-delay** should check whether the slave needs to be started or stopped.

--log
type: string

Print all output to this file when daemonized.

--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.

--quiet
short form: -q

Don't print informational messages about operation. See OUTPUT for details.

--run-time
type: time

How long **pt-slave-delay** should run before exiting. The default is to run forever.

--set-vars
type: Array

Set the MySQL 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.

--use-master
Get binlog positions from master, not slave. Don't trust the binlog positions in the slave's relay log. Connect to the master and get binlog positions instead. If you specify this option without giving a MASTER_DSN on the command line, **pt-slave-delay** examines the slave's SHOW SLAVE STATUS to determine the hostname and port for connecting to the master.

pt-slave-delay uses only the MASTER_HOST and MASTER_PORT values from SHOW SLAVE STATUS for the master connection. It does not use the MASTER_USER value. If you want to specify a different username for the master than the one you use to connect to the slave, you should specify the MASTER_DSN option explicitly on the command line.

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

User for login if not current user.

--version
Show version and exit.

--`[no]version-check`
default: yes

Check for the latest version of Percona Toolkit, MySQL, and other programs.

This is a standard "check for updates automatically" feature, with two additional features. First, the tool checks the version of other programs on the local system in addition to its own version. For example, it checks the version of every MySQL server it connects to, Perl, and the Perl module DBD::mysql. Second, it checks for and warns about versions with known problems. For example, MySQL 5.5.25 had a critical bug and was re-released as 5.5.25a.

Any updates or known problems are printed to STDOUT before the tool's normal output. This feature should never interfere with the normal operation of the tool.

For more information, visit https://www.percona.com/version-check.

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: mysql_socket; copy: yes

  Socket file to use for connection.

- u

  dsn: user; copy: yes

  User for login if not current user.

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 pt-slave-delay ... > FILE 2>&1
```

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

SYSTEM REQUIREMENTS

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

BUGS

For a list of known bugs, see http://www.percona.com/bugs/pt-slave-delay.

Please report bugs at https://bugs.launchpad.net/percona-toolkit. Include the following information in your bug report:

- Complete command-line used to run the tool
- Tool *--version*
- MySQL version of all servers involved
- Output from the tool including STDERR
- Input files (log/dump/config files, etc.)

If possible, include debugging output by running the tool with `PTDEBUG`; see "ENVIRONMENT".

DOWNLOADING

Visit http://www.percona.com/software/percona-toolkit/ to download the latest release of Percona Toolkit. Or, get the latest release from the command line:

```
wget percona.com/get/percona-toolkit.tar.gz

wget percona.com/get/percona-toolkit.rpm

wget percona.com/get/percona-toolkit.deb
```

You can also get individual tools from the latest release:

```
wget percona.com/get/TOOL
```

Replace `TOOL` with the name of any tool.

AUTHORS

Sergey Zhuravlev and Baron Schwartz

ABOUT PERCONA TOOLKIT

This tool is part of Percona Toolkit, a collection of advanced command-line tools for MySQL developed by Percona. 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. Visit http://www.percona.com/software/ to learn about other free, open-source software from Percona.

COPYRIGHT, LICENSE, AND WARRANTY

VERSION

pt-slave-delay 3.0.4

CHAPTER

TWENTYSIX

PT-SLAVE-FIND

NAME

pt-slave-find - Find and print replication hierarchy tree of MySQL slaves.

SYNOPSIS

Usage

```
pt-slave-find [OPTIONS] [DSN]
```

pt-slave-find finds and prints a hierarchy tree of MySQL slaves.

Examples

```
pt-slave-find --host master-host
```

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

DESCRIPTION

pt-slave-find connects to a MySQL replication master and finds its slaves. Currently the only thing it can do is print a tree-like view of the replication hierarchy.

The master host can be specified using one of two methods. The first method is to use the standard connection-related command line options: *--defaults-file*, *--password*, *--host*, *--port*, *--socket* or *--user*.

The second method to specify the master host is a DSN. A DSN is a special syntax that can be either just a hostname (like `server.domain.com` or `1.2.3.4`), or a `key=value,key=value` string. Keys are a single letter:

```
KEY MEANING
=== =======
h   Connect to host
P   Port number to use for connection
S   Socket file to use for connection
u   User for login if not current user
p   Password to use when connecting
F   Only read default options from the given file
```

pt-slave-find reads all normal MySQL option files, such as ~/.my.cnf, so you may not need to specify username, password and other common options at all.

EXIT STATUS

An exit status of 0 (sometimes also called a return value or return code) indicates success. Any other value represents the exit status of the Perl process itself.

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 MySQL.

--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 MySQL. Any other value sets binmode on STDOUT without the utf8 layer, and runs SET NAMES after connecting to MySQL.

--config
type: Array

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

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

Database to use.

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

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

--help
Show help and exit.

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

Connect to host.

--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.

--recurse
type: int

Number of levels to recurse in the hierarchy. Default is infinite.

See *--recursion-method*.

--recursion-method
type: array; default: processlist,hosts

Preferred recursion method used to find slaves.

Possible methods are:

```
METHOD       USES
===========  ==================
processlist  SHOW PROCESSLIST
hosts        SHOW SLAVE HOSTS
none         Do not find slaves
```

The processlist method is preferred because SHOW SLAVE HOSTS is not reliable. However, the hosts method is required if the server uses a non-standard port (not 3306). Usually **pt-slave-find** does the right thing and finds the slaves, but you may give a preferred method and it will be used first. If it doesn't find any slaves, the other methods will be tried.

--report-format
type: string; default: summary

Set what information about the slaves is printed. The report format can be one of the following:

•hostname

Print just the hostname name of the slaves. It looks like:

```
127.0.0.1:12345
+- 127.0.0.1:12346
   +- 127.0.0.1:12347
```

•summary

Print a summary of each slave's settings. This report shows more information about each slave, like:

```
127.0.0.1:12345
Version         5.1.34-log
Server ID       12345
Uptime          04:56 (started 2010-06-17T11:21:22)
Replication     Is not a slave, has 1 slaves connected
Filters
Binary logging  STATEMENT
Slave status
Slave mode      STRICT
Auto-increment  increment 1, offset 1
+- 127.0.0.1:12346
   Version         5.1.34-log
   Server ID       12346
   Uptime          04:54 (started 2010-06-17T11:21:24)
   Replication     Is a slave, has 1 slaves connected
   Filters
   Binary logging  STATEMENT
   Slave status    0 seconds behind, running, no errors
   Slave mode      STRICT
   Auto-increment  increment 1, offset 1
```

--resolve-address

Resolve ip-address to hostname. Report will print both IP and hostname.

Example:

```
10.10.7.14 (dbase1.sample.net)
```

Might delay runtime a few seconds.

--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 MySQL 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.

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

User for login if not current user.

--version
Show version and exit.

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: mysql_socket; copy: yes

  Socket file to use for connection.

- u

  dsn: user; copy: yes

  User for login if not current user.

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 pt-slave-find ... > FILE 2>&1
```

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

SYSTEM REQUIREMENTS

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

BUGS

For a list of known bugs, see http://www.percona.com/bugs/pt-slave-find.

Please report bugs at https://bugs.launchpad.net/percona-toolkit. Include the following information in your bug report:

- Complete command-line used to run the tool
- Tool *--version*
- MySQL version of all servers involved
- Output from the tool including STDERR
- Input files (log/dump/config files, etc.)

If possible, include debugging output by running the tool with `PTDEBUG`; see "ENVIRONMENT".

DOWNLOADING

Visit http://www.percona.com/software/percona-toolkit/ to download the latest release of Percona Toolkit. Or, get the latest release from the command line:

```
wget percona.com/get/percona-toolkit.tar.gz

wget percona.com/get/percona-toolkit.rpm

wget percona.com/get/percona-toolkit.deb
```

You can also get individual tools from the latest release:

```
wget percona.com/get/TOOL
```

Replace `TOOL` with the name of any tool.

AUTHORS

Baron Schwartz and Daniel Nichter

ABOUT PERCONA TOOLKIT

This tool is part of Percona Toolkit, a collection of advanced command-line tools for MySQL developed by Percona. 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. Visit http://www.percona.com/software/ to learn about other free, open-source software from Percona.

COPYRIGHT, LICENSE, AND WARRANTY

VERSION

pt-slave-find 3.0.4

CHAPTER

TWENTYSEVEN

PT-SLAVE-RESTART

NAME

pt-slave-restart - Watch and restart MySQL replication after errors.

SYNOPSIS

Usage

```
pt-slave-restart [OPTIONS] [DSN]
```

pt-slave-restart watches one or more MySQL replication slaves for errors, and tries to restart replication if it stops.

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

DESCRIPTION

pt-slave-restart watches one or more MySQL replication slaves and tries to skip statements that cause errors. It polls slaves intelligently with an exponentially varying sleep time. You can specify errors to skip and run the slaves until a certain binlog position.

Although this tool can help a slave advance past errors, you should not rely on it to "fix" replication. If slave errors occur frequently or unexpectedly, you should identify and fix the root cause.

OUTPUT

pt-slave-restart prints a line every time it sees the slave has an error. By default this line is: a timestamp, connection information, relay_log_file, relay_log_pos, and last_errno. You can add more information using the *--verbose* option. You can suppress all output using the *--quiet* option.

SLEEP

pt-slave-restart sleeps intelligently between polling the slave. The current sleep time varies.

- The initial sleep time is given by *--sleep*.
- If it checks and finds an error, it halves the previous sleep time.
- If it finds no error, it doubles the previous sleep time.
- The sleep time is bounded below by *--min-sleep* and above by *--max-sleep*.
- Immediately after finding an error, **pt-slave-restart** assumes another error is very likely to happen next, so it sleeps the current sleep time or the initial sleep time, whichever is less.

GLOBAL TRANSACTION IDS

As of Percona Toolkit 2.2.8, **pt-slave-restart** supports Global Transaction IDs introduced in MySQL 5.6.5. It's important to keep in mind that:

- **pt-slave-restart** will not skip transactions when multiple replication threads are being used (slave_parallel_workers > 0). **pt-slave-restart** does not know what the GTID event is of the failed transaction of a specific slave thread.
- The default behavior is to skip the next transaction from the slave's master. Writes can originate on different servers, each with their own UUID.

  See *--master-uuid*.

EXIT STATUS

An exit status of 0 (sometimes also called a return value or return code) indicates success. Any other value represents the exit status of the Perl process itself, or of the last forked process that exited if there were multiple servers to monitor.

COMPATIBILITY

pt-slave-restart should work on many versions of MySQL. Lettercase of many output columns from SHOW SLAVE STATUS has changed over time, so it treats them all as lowercase.

OPTIONS

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

--always
Start slaves even when there is no error. With this option enabled, **pt-slave-restart** will not let you stop the slave manually if you want to!

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

--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 MySQL. Any other value sets binmode on STDOUT without the utf8 layer, and runs SET NAMES after connecting to MySQL.

--`[no]check-relay-log`
default: yes

Check the last relay log file and position before checking for slave errors.

By default **pt-slave-restart** will not doing anything (it will just sleep) if neither the relay log file nor the relay log position have changed since the last check. This prevents infinite loops (i.e. restarting the same error in the same relay log file at the same relay log position).

For certain slave errors, however, this check needs to be disabled by specifying `--no-check-relay-log`. Do not do this unless you know what you are doing!

--config
type: Array

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

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

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

Database to use.

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

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

--error-length
type: int

Max length of error message to print. When *--verbose* is set high enough to print the error, this option will truncate the error text to the specified length. This can be useful to prevent wrapping on the terminal.

--error-numbers
type: hash

Only restart this comma-separated list of errors. Makes **pt-slave-restart** only try to restart if the error number is in this comma-separated list of errors. If it sees an error not in the list, it will exit.

The error number is in the `last_errno` column of `SHOW SLAVE STATUS`.

--error-text
type: string

Only restart errors that match this pattern. A Perl regular expression against which the error text, if any, is matched. If the error text exists and matches, **pt-slave-restart** will try to restart the slave. If it exists but doesn't match, **pt-slave-restart** will exit.

The error text is in the `last_error` column of `SHOW SLAVE STATUS`.

--help
Show help and exit.

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

Connect to host.

--log
type: string

Print all output to this file when daemonized.

--max-sleep
type: float; default: 64

Maximum sleep seconds.

The maximum time **pt-slave-restart** will sleep before polling the slave again. This is also the time that **pt-slave-restart** will wait for all other running instances to quit if both *--stop* and *--monitor* are specified.

See "SLEEP".

--min-sleep
type: float; default: 0.015625

The minimum time **pt-slave-restart** will sleep before polling the slave again. See "SLEEP".

--monitor
Whether to monitor the slave (default). Unless you specify –monitor explicitly, *--stop* will disable it.

--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.

--quiet
short form: -q

Suppresses normal output (disables *--verbose*).

--recurse
type: int; default: 0

Watch slaves of the specified server, up to the specified number of servers deep in the hierarchy. The default depth of 0 means "just watch the slave specified."

pt-slave-restart examines `SHOW PROCESSLIST` and tries to determine which connections are from slaves, then connect to them. See *--recursion-method*.

Recursion works by finding all slaves when the program starts, then watching them. If there is more than one slave, **pt-slave-restart** uses `fork()` to monitor them.

This also works if you have configured your slaves to show up in `SHOW SLAVE HOSTS`. The minimal configuration for this is the `report_host` parameter, but there are other "report" parameters as well for the port, username, and password.

--recursion-method
type: array; default: processlist,hosts

Preferred recursion method used to find slaves.

Possible methods are:

```
METHOD       USES
===========  ==================
processlist  SHOW PROCESSLIST
hosts        SHOW SLAVE HOSTS
none         Do not find slaves
```

The processlist method is preferred because SHOW SLAVE HOSTS is not reliable. However, the hosts method is required if the server uses a non-standard port (not 3306). Usually **pt-slave-restart** does the right thing and finds the slaves, but you may give a preferred method and it will be used first. If it doesn't find any slaves, the other methods will be tried.

--run-time
type: time

Time to run before exiting. Causes **pt-slave-restart** to stop after the specified time has elapsed. Optional suffix: s=seconds, m=minutes, h=hours, d=days; if no suffix, s is used.

--sentinel
type: string; default: /tmp/pt-slave-restart-sentinel

Exit if this file exists.

--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 MySQL 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.

--skip-count
type: int; default: 1

Number of statements to skip when restarting the slave.

--master-uuid
type: string

When using GTID, an empty transaction should be created in order to skip it. If writes are coming from different nodes in the replication tree above, it is not possible to know which event from which UUID to skip.

By default, transactions from the slave's master (`'Master_UUID'` from `SHOW SLAVE STATUS`) are skipped.

For example, with

```
master1 -> slave1 -> slave2
```

When skipping events on slave2 that were written to master1, you must specify the UUID of master1, else the tool will use the UUID of slave1 by default.

See "GLOBAL TRANSACTION IDS".

--sleep
type: int; default: 1

Initial sleep seconds between checking the slave.

See "SLEEP".

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

Socket file to use for connection.

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

Causes **pt-slave-restart** to create the sentinel file specified by *--sentinel*. This should have the effect of stopping all running instances which are watching the same sentinel file. If *--monitor* isn't specified, **pt-slave-restart** will exit after creating the file. If it is specified, **pt-slave-restart** will wait the interval given by *--max-sleep*, then remove the file and continue working.

You might find this handy to stop cron jobs gracefully if necessary, or to replace one running instance with another. For example, if you want to stop and restart **pt-slave-restart** every hour (just to make sure that it is restarted every hour, in case of a server crash or some other problem), you could use a `crontab` line like this:

```
0 * * * * :program:`pt-slave-restart` --monitor --stop --sentinel /tmp/pt-slave-
↪restartup
```

The non-default *--sentinel* will make sure the hourly `cron` job stops only instances previously started with the same options (that is, from the same `cron` job).

See also *--sentinel*.

--until-master
type: string

Run until this master log file and position. Start the slave, and retry if it fails, until it reaches the given replication coordinates. The coordinates are the logfile and position on the master, given by relay_master_log_file, exec_master_log_pos. The argument must be in the format "file,pos". Separate the filename and position with a single comma and no space.

This will also cause an UNTIL clause to be given to START SLAVE.

After reaching this point, the slave should be stopped and **pt-slave-restart** will exit.

--until-relay
type: string

Run until this relay log file and position. Like *--until-master*, but in the slave's relay logs instead. The coordinates are given by relay_log_file, relay_log_pos.

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

User for login if not current user.

--verbose
short form: -v; cumulative: yes; default: 1

Adds more information to the output. This flag can be specified multiple times. e.g. -v -v OR -vv. By default (no verbose flag) the tool outputs connection information, a timestamp, relay_log_file, relay_log_pos, and last_errno. One flag (-v) adds last_error. See also *--error-length*. Two flags (-vv) prints the current sleep time each time **pt-slave-restart** sleeps. To suppress all output use the *--quiet* option.

--version
Show version and exit.

--[no]version-check
default: yes

Check for the latest version of Percona Toolkit, MySQL, and other programs.

This is a standard "check for updates automatically" feature, with two additional features. First, the tool checks the version of other programs on the local system in addition to its own version. For example, it checks the version of every MySQL server it connects to, Perl, and the Perl module DBD::mysql. Second, it checks for and warns about versions with known problems. For example, MySQL 5.5.25 had a critical bug and was re-released as 5.5.25a.

Any updates or known problems are printed to STDOUT before the tool's normal output. This feature should never interfere with the normal operation of the tool.

For more information, visit https://www.percona.com/version-check.

Show version and exit.

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: mysql_socket; copy: yes

  Socket file to use for connection.

- u

  dsn: user; copy: yes

  User for login if not current user.

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 pt-slave-restart ... > FILE 2>&1
```

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

SYSTEM REQUIREMENTS

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

BUGS

For a list of known bugs, see http://www.percona.com/bugs/pt-slave-restart.

Please report bugs at https://bugs.launchpad.net/percona-toolkit. Include the following information in your bug report:

- Complete command-line used to run the tool
- Tool *--version*
- MySQL version of all servers involved
- Output from the tool including STDERR
- Input files (log/dump/config files, etc.)

If possible, include debugging output by running the tool with `PTDEBUG`; see "ENVIRONMENT".

DOWNLOADING

Visit http://www.percona.com/software/percona-toolkit/ to download the latest release of Percona Toolkit. Or, get the latest release from the command line:

```
wget percona.com/get/percona-toolkit.tar.gz

wget percona.com/get/percona-toolkit.rpm

wget percona.com/get/percona-toolkit.deb
```

You can also get individual tools from the latest release:

```
wget percona.com/get/TOOL
```

Replace `TOOL` with the name of any tool.

AUTHORS

Baron Schwartz

ABOUT PERCONA TOOLKIT

This tool is part of Percona Toolkit, a collection of advanced command-line tools for MySQL developed by Percona. 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. Visit http://www.percona.com/software/ to learn about other free, open-source software from Percona.

COPYRIGHT, LICENSE, AND WARRANTY

VERSION

pt-slave-restart 3.0.4

CHAPTER

TWENTYEIGHT

PT-STALK

NAME

pt-stalk - Collect forensic data about MySQL when problems occur.

SYNOPSIS

Usage

```
pt-stalk [OPTIONS]
```

pt-stalk 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.

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

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 MySQL problems when you can't observe them? That's why **pt-stalk** exists. In addition to using it when there's a known problem on your servers, it is a good idea to run **pt-stalk** 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 MySQL lockups or spikes in activity typically leave no evidence to use in root cause analysis.

pt-stalk does two things: it watches a MySQL 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 **pt-stalk** 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 MySQL 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, **pt-stalk** tool watches MySQL 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
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.

CONFIGURING

You can use standard Percona Toolkit 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.

OPTIONS

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

--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 MySQL 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 MySQL is listening. You can later use pt-query-digest to decode the MySQL 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 mysql options from the given file. You must give an absolute pathname.

--dest
type: string; default: /var/lib/pt-stalk

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 MySQL 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/pt-stalk.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/pt-stalk.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 **pt-stalk** 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`.

--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-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 MySQL 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.

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 **pt-stalk** as

```
CMD_IOSTAT="iostat -n" pt-stalk ...
```

which will do exactly what you need. Combined with the plugin hooks, this gives you a fine-grained control of what the tool does.

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`

BUGS

For a list of known bugs, see http://www.percona.com/bugs/pt-stalk.

Please report bugs at https://bugs.launchpad.net/percona-toolkit. Include the following information in your bug report:

- Complete command-line used to run the tool
- Tool *--version*
- MySQL version of all servers involved
- Output from the tool including STDERR
- Input files (log/dump/config files, etc.)

If possible, include debugging output by running the tool with `PTDEBUG`; see "ENVIRONMENT".

DOWNLOADING

Visit http://www.percona.com/software/percona-toolkit/ to download the latest release of Percona Toolkit. Or, get the latest release from the command line:

```
wget percona.com/get/percona-toolkit.tar.gz

wget percona.com/get/percona-toolkit.rpm

wget percona.com/get/percona-toolkit.deb
```

You can also get individual tools from the latest release:

```
wget percona.com/get/TOOL
```

Replace `TOOL` with the name of any tool.

AUTHORS

Baron Schwartz, Justin Swanhart, Fernando Ipar, Daniel Nichter, and Brian Fraser

ABOUT PERCONA TOOLKIT

This tool is part of Percona Toolkit, a collection of advanced command-line tools for MySQL developed by Percona. 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. Visit http://www.percona.com/software/ to learn about other free, open-source software from Percona.

COPYRIGHT, LICENSE, AND WARRANTY

VERSION

pt-stalk 3.0.4

CHAPTER

TWENTYNINE

PT-SUMMARY

NAME

pt-summary - Summarize system information nicely.

SYNOPSIS

Usage

```
pt-summary
```

pt-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.

Download and run:

```
wget http://percona.com/get/pt-summary
bash ./pt-summary
```

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

DESCRIPTION

pt-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.

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.

```
# Percona Toolkit 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.

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.

ENVIRONMENT

This tool does not use any environment variables.

SYSTEM REQUIREMENTS

This tool requires the Bourne shell (*/bin/sh*).

BUGS

For a list of known bugs, see http://www.percona.com/bugs/pt-summary.

Please report bugs at https://bugs.launchpad.net/percona-toolkit. Include the following information in your bug report:

- Complete command-line used to run the tool
- Tool *--version*
- MySQL version of all servers involved
- Output from the tool including STDERR
- Input files (log/dump/config files, etc.)

If possible, include debugging output by running the tool with `PTDEBUG`; see "ENVIRONMENT".

DOWNLOADING

Visit http://www.percona.com/software/percona-toolkit/ to download the latest release of Percona Toolkit. Or, get the latest release from the command line:

```
wget percona.com/get/percona-toolkit.tar.gz

wget percona.com/get/percona-toolkit.rpm

wget percona.com/get/percona-toolkit.deb
```

You can also get individual tools from the latest release:

```
wget percona.com/get/TOOL
```

Replace `TOOL` with the name of any tool.

AUTHORS

Baron Schwartz, Kevin van Zonneveld, and Brian Fraser

ABOUT PERCONA TOOLKIT

This tool is part of Percona Toolkit, a collection of advanced command-line tools for MySQL developed by Percona. 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. Visit http://www.percona.com/software/ to learn about other free, open-source software from Percona.

COPYRIGHT, LICENSE, AND WARRANTY

VERSION

pt-summary 3.0.4

CHAPTER

THIRTY

PT-TABLE-CHECKSUM

NAME

pt-table-checksum - Verify MySQL replication integrity.

SYNOPSIS

Usage

```
pt-table-checksum [OPTIONS] [DSN]
```

pt-table-checksum performs an online replication consistency check by executing checksum queries on the master, which produces different results on replicas that are inconsistent with the master. The optional DSN specifies the master host. The tool's "EXIT STATUS" is non-zero if any differences are found, or if any warnings or errors occur.

The following command will connect to the replication master on localhost, checksum every table, and report the results on every detected replica:

```
pt-table-checksum
```

This tool is focused on finding data differences efficiently. If any data is different, you can resolve the problem with pt-table-sync.

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

See also "LIMITATIONS".

DESCRIPTION

pt-table-checksum is designed to do the right thing by default in almost every case. When in doubt, use *--explain* to see how the tool will checksum a table. The following is a high-level overview of how the tool functions.

In contrast to older versions of **pt-table-checksum**, this tool is focused on a single purpose, and does not have a lot of complexity or support many different checksumming techniques. It executes checksum queries on only one server, and these flow through replication to re-execute on replicas. If you need the older behavior, you can use Percona Toolkit version 1.0.

pt-table-checksum connects to the server you specify, and finds databases and tables that match the filters you specify (if any). It works one table at a time, so it does not accumulate large amounts of memory or do a lot of work before beginning to checksum. This makes it usable on very large servers. We have used it on servers with hundreds of thousands of databases and tables, and trillions of rows. No matter how large the server is, **pt-table-checksum** works equally well.

One reason it can work on very large tables is that it divides each table into chunks of rows, and checksums each chunk with a single REPLACE..SELECT query. It varies the chunk size to make the checksum queries run in the desired amount of time. The goal of chunking the tables, instead of doing each table with a single big query, is to ensure that checksums are unintrusive and don't cause too much replication lag or load on the server. That's why the target time for each chunk is 0.5 seconds by default.

The tool keeps track of how quickly the server is able to execute the queries, and adjusts the chunks as it learns more about the server's performance. It uses an exponentially decaying weighted average to keep the chunk size stable, yet remain responsive if the server's performance changes during checksumming for any reason. This means that the tool will quickly throttle itself if your server becomes heavily loaded during a traffic spike or a background task, for example.

Chunking is accomplished by a technique that we used to call "nibbling" in other tools in Percona Toolkit. It is the same technique used for pt-archiver, for example. The legacy chunking algorithms used in older versions of **pt-table-checksum** are removed, because they did not result in predictably sized chunks, and didn't work well on many tables. All that is required to divide a table into chunks is an index of some sort (preferably a primary key or unique index). If there is no index, and the table contains a suitably small number of rows, the tool will checksum the table in a single chunk.

pt-table-checksum has many other safeguards to ensure that it does not interfere with any server's operation, including replicas. To accomplish this, **pt-table-checksum** detects replicas and connects to them automatically. (If this fails, you can give it a hint with the *--recursion-method* option.)

The tool monitors replicas continually. If any replica falls too far behind in replication, **pt-table-checksum** pauses to allow it to catch up. If any replica has an error, or replication stops, **pt-table-checksum** pauses and waits. In addition, **pt-table-checksum** looks for common causes of problems, such as replication filters, and refuses to operate unless you force it to. Replication filters are dangerous, because the queries that **pt-table-checksum** executes could potentially conflict with them and cause replication to fail.

pt-table-checksum verifies that chunks are not too large to checksum safely. It performs an EXPLAIN query on each chunk, and skips chunks that might be larger than the desired number of rows. You can configure the sensitivity of this safeguard with the *--chunk-size-limit* option. If a table will be checksummed in a single chunk because it has a small number of rows, then **pt-table-checksum** additionally verifies that the table isn't oversized on replicas. This avoids the following scenario: a table is empty on the master but is very large on a replica, and is checksummed in a single large query, which causes a very long delay in replication.

There are several other safeguards. For example, **pt-table-checksum** sets its session-level innodb_lock_wait_timeout to 1 second, so that if there is a lock wait, it will be the victim instead of causing other queries to time out. Another safeguard checks the load on the database server, and pauses if the load is too high. There is no single right answer for how to do this, but by default **pt-table-checksum** will pause if there are more

than 25 concurrently executing queries. You should probably set a sane value for your server with the *--max-load* option.

Checksumming usually is a low-priority task that should yield to other work on the server. However, a tool that must be restarted constantly is difficult to use. Thus, **pt-table-checksum** is very resilient to errors. For example, if the database administrator needs to kill **pt-table-checksum**'s queries for any reason, that is not a fatal error. Users often run pt-kill to kill any long-running checksum queries. The tool will retry a killed query once, and if it fails again, it will move on to the next chunk of that table. The same behavior applies if there is a lock wait timeout. The tool will print a warning if such an error happens, but only once per table. If the connection to any server fails, **pt-table-checksum** will attempt to reconnect and continue working.

If **pt-table-checksum** encounters a condition that causes it to stop completely, it is easy to resume it with the *--resume* option. It will begin from the last chunk of the last table that it processed. You can also safely stop the tool with CTRL-C. It will finish the chunk it is currently processing, and then exit. You can resume it as usual afterwards.

After **pt-table-checksum** finishes checksumming all of the chunks in a table, it pauses and waits for all detected replicas to finish executing the checksum queries. Once that is finished, it checks all of the replicas to see if they have the same data as the master, and then prints a line of output with the results. You can see a sample of its output later in this documentation.

The tool prints progress indicators during time-consuming operations. It prints a progress indicator as each table is checksummed. The progress is computed by the estimated number of rows in the table. It will also print a progress report when it pauses to wait for replication to catch up, and when it is waiting to check replicas for differences from the master. You can make the output less verbose with the *--quiet* option.

If you wish, you can query the checksum tables manually to get a report of which tables and chunks have differences from the master. The following query will report every database and table with differences, along with a summary of the number of chunks and rows possibly affected:

```
SELECT db, tbl, SUM(this_cnt) AS total_rows, COUNT(*) AS chunks
FROM percona.checksums
WHERE (
 master_cnt <> this_cnt
 OR master_crc <> this_crc
 OR ISNULL(master_crc) <> ISNULL(this_crc))
GROUP BY db, tbl;
```

The table referenced in that query is the checksum table, where the checksums are stored. Each row in the table contains the checksum of one chunk of data from some table in the server.

Version 2.0 of **pt-table-checksum** is not backwards compatible with pt-table-sync version 1.0. In some cases this is not a serious problem. Adding a "boundaries" column to the table, and then updating it with a manually generated WHERE clause, may suffice to let pt-table-sync version 1.0 interoperate with **pt-table-checksum** version 2.0. Assuming an integer primary key named 'id', You can try something like the following:

```
ALTER TABLE checksums ADD boundaries VARCHAR(500);
UPDATE checksums
 SET boundaries = COALESCE(CONCAT('id BETWEEN ', lower_boundary,
    ' AND ', upper_boundary), '1=1');
```

LIMITATIONS

Replicas using row-based replication

> **pt-table-checksum** requires statement-based replication, and it sets `binlog_format=STATEMENT` on the master, but due to a MySQL limitation replicas do not

honor this change. Therefore, checksums will not replicate past any replicas using row-based replication that are masters for further replicas.

The tool automatically checks the `binlog_format` on all servers. See `--[no]check-binlog-format`.

(Bug 899415)

Schema and table differences

The tool presumes that schemas and tables are identical on the master and all replicas. Replication will break if, for example, a replica does not have a schema that exists on the master (and that schema is checksummed), or if the structure of a table on a replica is different than on the master.

Percona XtraDB Cluster

pt-table-checksum works with Percona XtraDB Cluster (PXC) 5.5.28-23.7 and newer. The number of possible Percona XtraDB Cluster setups is large given that it can be used with regular replication as well. Therefore, only the setups listed below are supported and known to work. Other setups, like cluster to cluster, are not support and probably don't work.

Except where noted, all of the following supported setups require that you use the `dsn` method for *`--recursion-method`* to specify cluster nodes. Also, the lag check (see "REPLICA CHECKS") is not performed for cluster nodes.

Single cluster

The simplest PXC setup is a single cluster: all servers are cluster nodes, and there are no regular replicas. If all nodes are specified in the DSN table (see *`--recursion-method`*), then you can run the tool on any node and any diffs on any other nodes will be detected.

All nodes must be in the same cluster (have the same `wsrep_cluster_name` value), else the tool exits with an error. Although it's possible to have different clusters with the same name, this should not be done and is not supported. This applies to all supported setups.

Single cluster with replicas

Cluster nodes can also be regular masters and replicate to regular replicas. However, the tool can only detect diffs on a replica if ran on the replica's "master node". For example, if the cluster setup is,

```
node1 <-> node2 <-> node3
            |         |
            |         +-> replica3
            +-> replica2
```

you can detect diffs on replica3 by running the tool on node3, but to detect diffs on replica2 you must run the tool again on node2. If you run the tool on node1, it will not detect diffs on either replica.

Currently, the tool does not detect this setup or warn about replicas that cannot be checked (e.g. replica2 when running on node3).

Replicas in this setup are still subject to `--[no]check-binlog-format`.

Master to single cluster

It is possible for a regular master to replicate to a cluster, as if the cluster were one logical slave, like:

```
master -> node1 <-> node2 <-> node3
```

The tool supports this setup but only if ran on the master and if all nodes in the cluster are consistent with the "direct replica" (node1 in this example) of the master. For example, if all nodes have value "foo" for row 1 but the master has value "bar" for the same row, this diff will be detected. Or if only node1 has this diff, it will also be detected. But if only node2 or node3 has this diff, it will not be detected. Therefore, this setup is used to check that the master and the cluster as a whole are consistent.

In this setup, the tool can automatically detect the "direct replica" (node1) when ran on the master, so you do not have to use the `dsn` method for *`--recursion-method`* because node1 will represent the entire cluster, which is why all other nodes must be consistent with it.

The tool warns when it detects this setup to remind you that it only works when used as described above. These warnings do not affect the exit status of the tool; they're only reminders to help avoid false-positive results.

OUTPUT

The tool prints tabular results, one line per table:

```
            TS ERRORS  DIFFS  ROWS  CHUNKS SKIPPED    TIME TABLE
10-20T08:36:50      0      0   200       1       0   0.005 db1.tbl1
10-20T08:36:50      0      0   603       7       0   0.035 db1.tbl2
10-20T08:36:50      0      0    16       1       0   0.003 db2.tbl3
10-20T08:36:50      0      0   600       6       0   0.024 db2.tbl4
```

Errors, warnings, and progress reports are printed to standard error. See also *`--quiet`*.

Each table's results are printed when the tool finishes checksumming the table. The columns are as follows:

TS The timestamp (without the year) when the tool finished checksumming the table.

ERRORS The number of errors and warnings that occurred while checksumming the table. Errors and warnings are printed to standard error while the table is in progress.

DIFFS The number of chunks that differ from the master on one or more replicas. If `--no-replicate-check` is specified, this column will always have zeros. If *`--replicate-check-only`* is specified, then only tables with differences are printed.

ROWS The number of rows selected and checksummed from the table. It might be different from the number of rows in the table if you use the –where option.

CHUNKS The number of chunks into which the table was divided.

SKIPPED The number of chunks that were skipped due one or more of these problems:

```
* MySQL not using the --chunk-index
* MySQL not using the full chunk index (--[no]check-plan)
* Chunk size is greater than --chunk-size * --chunk-size-limit
* Lock wait timeout exceeded (--retries)
* Checksum query killed (--retries)
```

As of **`pt-table-checksum`** 2.2.5, skipped chunks cause a non-zero "EXIT STATUS".

TIME The time elapsed while checksumming the table.

TABLE The database and table that was checksummed.

If *`--replicate-check-only`* is specified, only checksum differences on detected replicas are printed. The output is different: one paragraph per replica, one checksum difference per line, and values are separated by spaces:

```
Differences on h=127.0.0.1,P=12346
TABLE CHUNK CNT_DIFF CRC_DIFF CHUNK_INDEX LOWER_BOUNDARY UPPER_BOUNDARY
db1.tbl1 1 0 1 PRIMARY 1 100
db1.tbl1 6 0 1 PRIMARY 501 600

Differences on h=127.0.0.1,P=12347
TABLE CHUNK CNT_DIFF CRC_DIFF CHUNK_INDEX LOWER_BOUNDARY UPPER_BOUNDARY
db1.tbl1 1 0 1 PRIMARY 1 100
db2.tbl2 9 5 0 PRIMARY 101 200
```

The first line of a paragraph indicates the replica with differences. In this example there are two: h=127.0.0.1,P=12346 and h=127.0.0.1,P=12347. The columns are as follows:

TABLE The database and table that differs from the master.

CHUNK The chunk number of the table that differs from the master.

CNT_DIFF The number of chunk rows on the replica minus the number of chunk rows on the master.

CRC_DIFF 1 if the CRC of the chunk on the replica is different than the CRC of the chunk on the master, else 0.

CHUNK_INDEX The index used to chunk the table.

LOWER_BOUNDARY The index values that define the lower boundary of the chunk.

UPPER_BOUNDARY The index values that define the upper boundary of the chunk.

EXIT STATUS

pt-table-checksum has three possible exit statuses: zero, 255, and any other value is a bitmask with flags for different problems.

A zero exit status indicates no errors, warnings, or checksum differences, or skipped chunks or tables.

A 255 exit status indicates a fatal error. In other words: the tool died or crashed. The error is printed to `STDERR`.

If the exit status is not zero or 255, then its value functions as a bitmask with these flags:

```
FLAG              BIT VALUE  MEANING
================  =========  ========================================
ERROR                     1  A non-fatal error occurred
ALREADY_RUNNING           2  --pid file exists and the PID is running
CAUGHT_SIGNAL             4  Caught SIGHUP, SIGINT, SIGPIPE, or SIGTERM
NO_SLAVES_FOUND           8  No replicas or cluster nodes were found
TABLE_DIFF               16  At least one diff was found
SKIP_CHUNK               32  At least one chunk was skipped
SKIP_TABLE               64  At least one table was skipped
```

If any flag is set, the exit status will be non-zero. Use the bitwise `AND` operation to check for a particular flag. For example, if `$exit_status & 16` is true, then at least one diff was found.

As of **pt-table-checksum** 2.2.5, skipped chunks cause a non-zero exit status. An exit status of zero or 32 is equivalent to a zero exit status with skipped chunks in previous versions of the tool.

OPTIONS

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

--ask-pass
group: Connection

Prompt for a password when connecting to MySQL.

--[no]check-binlog-format
default: yes

Check that the `binlog_format` is the same on all servers.

See "Replicas using row-based replication" under "LIMITATIONS".

--binary-index
This option modifies the behavior of `--create-replicate-table` such that the replicate table's upper and lower boundary columns are created with the BLOB data type. This is useful in cases where you have trouble checksuming tables with keys that include a binary data type or that have non-standard character sets. See *--replicate*.

--check-interval
type: time; default: 1; group: Throttle

Sleep time between checks for *--max-lag*.

--[no]check-plan
default: yes

Check query execution plans for safety. By default, this option causes **pt-table-checksum** to run EXPLAIN before running queries that are meant to access a small amount of data, but which could access many rows if MySQL chooses a bad execution plan. These include the queries to determine chunk boundaries and the chunk queries themselves. If it appears that MySQL 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 MySQL intends to use the desired index to access the rows. If MySQL chooses a different index, the tool considers the query unsafe.

The tool also checks how much of the index MySQL 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 MySQL 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 MySQL, 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; group: Safety

Do not checksum if any replication filters are set on any replicas. 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 checksum any databases or tables that exist on the master and not the replicas. Changes to such tables might normally be skipped on the replicas because of the filtering options, but the checksum queries modify the contents of the table that stores the checksums, not the tables whose data you are checksumming. Therefore, these queries will be executed on the replica, and if the table or database you're checksumming does not exist, the queries will cause replication to fail. For more information on replication rules, see http://dev.mysql.com/doc/en/replication-rules.html.

Replication filtering makes it impossible to be sure that the checksum queries won't break replication (or simply fail to replicate). If you are sure that it's OK to run the checksum queries, you can negate this option to disable the checks. See also *--replicate-database*.

See also "REPLICA CHECKS".

--check-slave-lag
type: string; group: Throttle

Pause checksumming until this replica's lag is less than *--max-lag*. The value is a DSN that inherits properties from the master host and the connection options (*--port*, *--user*, etc.). By default, **pt-table-checksum** monitors lag on all connected replicas, but this option limits lag monitoring to the specified replica. This is useful if certain replicas are intentionally lagged (with pt-slave-delay for example), in which case you can specify a normal replica to monitor.

See also "REPLICA CHECKS".

--[no]check-slave-tables
default: yes; group: Safety

Checks that tables on slaves exist and have all the checksum *--columns*. Tables missing on slaves or not having all the checksum *--columns* can cause the tool to break replication when it tries to check for differences. Only disable this check if you are aware of the risks and are sure that all tables on all slaves exist and are identical to the master.

--chunk-index
type: string

Prefer this index for chunking tables. By default, **pt-table-checksum** chooses the most appropriate index for chunking. This option lets you specify the index that you prefer. If the index doesn't exist, then **pt-table-checksum** will fall back to its default behavior of choosing an index. **pt-table-checksum** adds the index to the checksum SQL statements in a `FORCE INDEX` clause. Be careful when using this option; a poor choice of index could cause bad performance. This is probably best to use when you are checksumming only a single table, not an entire server.

--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 MySQL 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 checksum query. Allowable suffixes are k, M, G. You should not use this option in most cases; prefer *--chunk-time* instead.

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*.

Selecting a small chunk size will cause the tool to become much slower, in part because of the setup work required for `--[no]check-plan`.

--chunk-size-limit
type: float; default: 2.0; group: Safety

Do not checksum 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 (twice as large, by default), 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. If the tool skips too many chunks because they are oversized, you might want to specify a value larger than the default of 2.

You can disable oversized chunk checking by specifying a value of 0.

--chunk-time
type: float; default: 0.5

Adjust the chunk size dynamically so each checksum query takes this long to execute.

The tool tracks the checksum rate (rows per second) for all tables and each table individually. It uses these rates to adjust the chunk size after each checksum query, so that the next checksum query takes this amount of time (in seconds) to execute.

The algorithm is as follows: at the beginning of each table, the chunk size is initialized from the overall average rows per second since the tool began working, or the value of *--chunk-size* if the tool hasn't started working yet. For each subsequent chunk of a table, the tool adjusts the chunk size to try to make queries run in the desired amount of time. 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. This allows the tool to achieve predictably timed queries for each table, and for the server overall.

If this option is set to zero, the chunk size doesn't auto-adjust, so query checksum times will vary, but query checksum 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.

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

Checksum only this comma-separated list of columns. If a table doesn't have any of the specified columns it will be skipped.

This option applies to all tables, so it really only makes sense when checksumming one table unless the tables have a common set of columns.

--config
type: Array; group: Config

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

See the *--help* output for a list of default config files.

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

Create the *--replicate* database and table if they do not exist. The structure of the replicate table is the same as the suggested table mentioned in *--replicate*.

--databases
short form: -d; type: hash; group: Filter

Only checksum this comma-separated list of databases.

--databases-regex
type: string; group: Filter

Only checksum databases whose names match this Perl regex.

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

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

--[no]empty-replicate-table
default: yes

Delete previous checksums for each table before checksumming the table. This option does not truncate the entire table, it only deletes rows (checksums) for each table just before checksumming the table. Therefore, if checksumming stops prematurely and there was preexisting data, there will still be rows for tables that were not checksummed before the tool was stopped.

If you're resuming from a previous checksum run, then the checksum records for the table from which the tool resumes won't be emptied.

To empty the entire replicate table, you must manually execute `TRUNCATE TABLE` before running the tool.

--engines
short form: -e; type: hash; group: Filter

Only checksum tables which use these storage engines.

--explain
cumulative: yes; default: 0; group: Output

Show, but do not execute, checksum queries (disables `--[no]empty-replicate-table`). If specified twice, the tool actually iterates through the chunking algorithm, printing the upper and lower boundary values for each chunk, but not executing the checksum queries.

--float-precision
type: int

Precision for FLOAT and DOUBLE number-to-string conversion. Causes FLOAT and DOUBLE values to be rounded to the specified number of digits after the decimal point, with the ROUND() function in MySQL. This can help avoid checksum mismatches due to different floating-point representations of the same values on different MySQL versions and hardware. The default is no rounding; the values are converted to strings by the CONCAT() function, and MySQL chooses the string representation. If you specify a value of 2, for example, then the values 1.008 and 1.009 will be rounded to 1.01, and will checksum as equal.

--function
type: string

Hash function for checksums (FNV1A_64, MURMUR_HASH, SHA1, MD5, CRC32, etc).

The default is to use CRC32(), but MD5() and SHA1() also work, and you can use your own function, such as a compiled UDF, if you wish. The function you specify is run in SQL, not in Perl, so it must be available to MySQL.

MySQL doesn't have good built-in hash functions that are fast. CRC32() is too prone to hash collisions, and MD5() and SHA1() are very CPU-intensive. The FNV1A_64() UDF that is distributed with Percona Server is a faster alternative. It is very simple to compile and install; look at the header in the source code for instructions. If it is installed, it is preferred over MD5(). You can also use the MURMUR_HASH() function if you compile and install that as a UDF; the source is also distributed with Percona Server, and it might be better than FNV1A_64().

--help
group: Help

Show help and exit.

--host
short form: -h; type: string; default: localhost; group: Connection

Host to connect to.

--ignore-columns
type: Hash; group: Filter

Ignore this comma-separated list of columns when calculating the checksum. If a table has all of its columns filtered by –ignore-columns, it will be skipped.

--ignore-databases
type: Hash; group: Filter

Ignore this comma-separated list of databases.

--ignore-databases-regex
type: string; group: Filter

Ignore databases whose names match this Perl regex.

--ignore-engines
type: Hash; default: FEDERATED,MRG_MyISAM; group: Filter

Ignore this comma-separated list of storage engines.

--ignore-tables
type: Hash; group: Filter

Ignore this comma-separated list of tables. Table names may be qualified with the database name. The *--replicate* table is always automatically ignored.

--ignore-tables-regex
type: string; group: Filter

Ignore tables whose names match the Perl regex.

--max-lag
type: time; default: 1s; group: Throttle

Pause checksumming until all replicas' lag is less than this value. After each checksum query (each chunk), **pt-table-checksum** 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 **pt-table-checksum** 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.

The tool waits forever for replicas to stop lagging. If any replica is stopped, the tool waits forever until the replica is started. Checksumming continues once 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.

See also "REPLICA CHECKS".

--max-load
type: Array; default: Threads_running=25; group: Throttle

Examine SHOW GLOBAL STATUS after every chunk, and pause if any status variables are higher than the threshold. The option accepts a comma-separated list of MySQL status variables to check for a threshold.

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 checksum 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 checksum 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.

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

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_table_checksum_plugin class. A plugin allows you to write a Perl module that can hook into many parts of **pt-table-checksum**. This requires a good knowledge of Perl and Percona Toolkit conventions, which are beyond this scope of this documentation. Please contact Percona if you have questions or need help.

See "PLUGIN" for more information.

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

Port number to use for connection.

--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. The tool prints progress reports for a variety of time-consuming operations, including waiting for replicas to catch up if they become lagged.

--quiet
short form: -q; cumulative: yes; default: 0

Print only the most important information (disables *--progress*). Specifying this option once causes the tool to print only errors, warnings, and tables that have checksum differences.

Specifying this option twice causes the tool to print only errors. In this case, you can use the tool's exit status to determine if there were any warnings or checksum differences.

--recurse

type: int

Number of levels to recurse in the hierarchy when discovering replicas. Default is infinite. See also *--recursion-method* and "REPLICA CHECKS".

--recursion-method

type: array; default: processlist,hosts

Preferred recursion method for discovering replicas. **pt-table-checksum** performs several "REPLICA CHECKS" before and while running.

Although replicas are not required to run **pt-table-checksum**, the tool cannot detect diffs on slaves that it cannot discover. Therefore, a warning is printed and the "EXIT STATUS" is non-zero if no replicas are found and the method is not `none`. If this happens, try a different recursion method, or use the `dsn` method to specify the replicas to check.

Possible methods are:

```
METHOD       USES
===========  ===============================================
processlist  SHOW PROCESSLIST
hosts        SHOW SLAVE HOSTS
cluster      SHOW STATUS LIKE 'wsrep\_incoming\_addresses'
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, if the server uses a non-standard port (not 3306), then the `hosts` method becomes the default because it works better in this case.

The `hosts` method requires replicas to be configured with `report_host`, `report_port`, etc.

The `cluster` method requires a cluster based on Galera 23.7.3 or newer, such as Percona XtraDB Cluster versions 5.5.29 and above. This will auto-discover nodes in a cluster using `SHOW STATUS LIKE 'wsrep\_incoming\_addresses'`. You can combine `cluster` with `processlist` and `hosts` to auto-discover cluster nodes and replicas, but this functionality is experimental.

The `dsn` method is special: rather than automatically discovering replicas, this method specifies a table with replica DSNs. The tool will only connect to these replicas. This method works best when replicas do not use the same MySQL username or password as the master, or when you want to prevent the tool from connecting to certain replicas. The `dsn` method is specified like: `--recursion-method dsn=h=host,D=percona,t=dsns`. The specified DSN must have D and t parts, or just a database-qualified t part, which specify the DSN table. The DSN table must 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`)
);
```

DSNs are ordered by `id`, but `id` and `parent_id` are otherwise ignored. The `dsn` column contains a replica

DSN like it would be given on the command line, for example: "h=replica_host,u=repl_user,p=repl_pass".

The none method makes the tool ignore all slaves and cluster nodes. This method is not recommended because it effectively disables the "REPLICA CHECKS" and no differences can be found. It is useful, however, if you only need to write checksums on the master or a single cluster node. The safer alternative is --no-replicate-check: the tool finds replicas and cluster nodes, performs the "REPLICA CHECKS", but does not check for differences. See --[no]replicate-check.

--replicate

type: string; default: percona.checksums

Write checksum results to this table. The replicate table must have this structure (MAGIC_create_replicate):

```
CREATE TABLE checksums (
   db             CHAR(64)     NOT NULL,
   tbl            CHAR(64)     NOT NULL,
   chunk          INT          NOT NULL,
   chunk_time     FLOAT            NULL,
   chunk_index    VARCHAR(200)     NULL,
   lower_boundary TEXT             NULL,
   upper_boundary TEXT             NULL,
   this_crc       CHAR(40)     NOT NULL,
   this_cnt       INT          NOT NULL,
   master_crc     CHAR(40)         NULL,
   master_cnt     INT              NULL,
   ts             TIMESTAMP    NOT NULL DEFAULT CURRENT_TIMESTAMP ON UPDATE␣
↪CURRENT_TIMESTAMP,
   PRIMARY KEY (db, tbl, chunk),
   INDEX ts_db_tbl (ts, db, tbl)
) ENGINE=InnoDB DEFAULT CHARSET=utf8;
```

Note: lower_boundary and upper_boundary data type can be BLOB. See *--binary-index*.

By default, --[no]create-replicate-table is true, so the database and the table specified by this option are created automatically if they do not exist.

Be sure to choose an appropriate storage engine for the replicate table. If you are checksumming InnoDB tables, and you use MyISAM for this table, a deadlock will break replication, because the mixture of transactional and non-transactional tables in the checksum statements will cause it to be written to the binlog even though it had an error. It will then replay without a deadlock on the replicas, and break replication with "different error on master and slave." This is not a problem with **pt-table-checksum**; it's a problem with MySQL replication, and you can read more about it in the MySQL manual.

The replicate table is never checksummed (the tool automatically adds this table to *--ignore-tables*).

--[no]replicate-check

default: yes

Check replicas for data differences after finishing each table. The tool finds differences by executing a simple SELECT statement on all detected replicas. The query compares the replica's checksum results to the master's checksum results. It reports differences in the DIFFS column of the output.

--replicate-check-only

Check replicas for consistency without executing checksum queries. This option is used only with --[no]replicate-check. If specified, **pt-table-checksum** doesn't checksum any tables. It checks replicas for differences found by previous checksumming, and then exits. It might be useful if you run **pt-table-checksum** quietly in a cron job, for example, and later want a report on the results of the cron job, perhaps to implement a Nagios check.

--replicate-check-retries
type: int; default: 1

Retry checksum comparison this many times when a difference is encountered. Only when a difference persists after this number of checks is it considered valid. Using this option with a value of 2 or more alleviates spurious differences that arise when using the –resume option.

--replicate-database
type: string

USE only this database. By default, **pt-table-checksum** executes USE to select the database that contains the table it's currently working on. This is is a best effort to avoid problems with replication filters such as binlog_ignore_db and replicate_ignore_db. However, replication filters can create a situation where there simply is no one right way to do things. Some statements might not be replicated, and others might cause replication to fail. In such cases, you can use this option to specify a default database that **pt-table-checksum** selects with USE, and never changes. See also `--[no]check-replication-filters`.

--resume
Resume checksumming from the last completed chunk (disables `--[no]empty-replicate-table`). If the tool stops before it checksums all tables, this option makes checksumming resume from the last chunk of the last table that it finished.

--retries
type: int; default: 2

Retry a chunk this many times when there is a nonfatal error. Nonfatal errors are problems such as a lock wait timeout or the query being killed.

--run-time
type: time

How long to run. Default is to run until all tables have been checksummed. These time value suffixes are allowed: s (seconds), m (minutes), h (hours), and d (days). Combine this option with *`--resume`* to checksum as many tables within an allotted time, resuming from where the tool left off next time it is ran.

--separator
type: string; default: #

The separator character used for CONCAT_WS(). This character is used to join the values of columns when checksumming.

--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=127.1,P=12345 –skip-check-slave-lag h=127.1,P=12346

--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; group: Connection

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

By default, the tool sets:

```
wait_timeout=10000
innodb_lock_wait_timeout=1
```

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; group: Connection

Socket file to use for connection.

--slave-skip-tolerance
type: float; default: 1.0

When a master table is marked to be checksumed in only one chunk but a slave table exceeds the maximum accepted size for this, the table is skipped. Since number of rows are often rough estimates, many times tables are skipped needlessly for very small differences. This option provides a max row excess tolerance to prevent this. For example a value of 1.2 will tolerate slave tables with up to 20% excess rows.

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

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

--tables-regex
type: string; group: Filter

Checksum only tables whose names match this Perl regex.

--trim
Add TRIM() to VARCHAR columns (helps when comparing 4.1 to >= 5.0). This is useful when you don't care about the trailing space differences between MySQL versions that vary in their handling of trailing spaces. MySQL 5.0 and later all retain trailing spaces in VARCHAR, while previous versions would remove them. These differences will cause false checksum differences.

--truncate-replicate-table
Truncate the replicate table before starting the checksum. This parameter differs from –empty-replicate-table which only deletes the rows for the table being checksumed when starting the checksum for that table, while –truncate-replicate-table will truncate the replicate table at the beginning of the process and thus, all previous checksum information will be losti, even if the process stops due to an error.

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

User for login if not current user.

--version
group: Help

Show version and exit.

--`[no]version-check`
default: yes

Check for the latest version of Percona Toolkit, MySQL, and other programs.

This is a standard "check for updates automatically" feature, with two additional features. First, the tool checks the version of other programs on the local system in addition to its own version. For example, it checks the version of every MySQL server it connects to, Perl, and the Perl module DBD::mysql. Second, it checks for and

warns about versions with known problems. For example, MySQL 5.5.25 had a critical bug and was re-released as 5.5.25a.

Any updates or known problems are printed to STDOUT before the tool's normal output. This feature should never interfere with the normal operation of the tool.

For more information, visit https://www.percona.com/version-check.

--where

type: string

Do only rows matching this WHERE clause. You can use this option to limit the checksum to only part of the table. This is particularly useful if you have append-only tables and don't want to constantly re-check all rows; you could run a daily job to just check yesterday's rows, for instance.

This option is much like the -w option to mysqldump. Do not specify the WHERE keyword. You might need to quote the value. Here is an example:

```
:program:`pt-table-checksum` --where "ts > CURRENT_DATE - INTERVAL 1 DAY"
```

REPLICA CHECKS

By default, **pt-table-checksum** attempts to find and connect to all replicas connected to the master host. This automated process is called "slave recursion" and is controlled by the *--recursion-method* and *--recurse* options. The tool performs these checks on all replicas:

1. `--[no]check-replication-filters`

   pt-table-checksum checks for replication filters on all replicas because they can complicate or break the checksum process. By default, the tool will exit if any replication filters are found, but this check can be disabled by specifying `--no-check-replication-filters`.

2. *--replicate* table

   pt-table-checksum checks that the *--replicate* table exists on all replicas, else checksumming can break replication when updates to the table on the master replicate to a replica that doesn't have the table. This check cannot be disabled, and the tool waits forever until the table exists on all replicas, printing *--progress* messages while it waits.

3. Single chunk size

   If a table can be checksummed in a single chunk on the master, **pt-table-checksum** will check that the table size on all replicas is less than *--chunk-size* * *--chunk-size-limit*. This prevents a rare problem where the table on the master is empty or small, but on a replica it is much larger. In this case, the single chunk checksum on the master would overload the replica.

   Another rare problem occurs when the table size on a replica is close to *--chunk-size* * *--chunk-size-limit*. In such cases, the table is more likely to be skipped even though it's safe to checksum in a single chunk. This happens because table sizes are estimates. When those estimates and *--chunk-size* * *--chunk-size-limit* are almost equal, this check becomes more sensitive to the estimates' margin of error rather than actual significant differences in table sizes. Specifying a larger value for *--chunk-size-limit* helps avoid this problem.

   This check cannot be disabled.

4. Lag

   After each chunk, **pt-table-checksum** checks the lag on all replicas, or only the replica specified by *--check-slave-lag*. This helps the tool not to overload the replicas with checksum data. There is

no way to disable this check, but you can specify a single replica to check with *--check-slave-lag*, and if that replica is the fastest, it will help prevent the tool from waiting too long for replica lag to abate.

5. Checksum chunks

   When **pt-table-checksum** finishes checksumming a table, it waits for the last checksum chunk to replicate to all replicas so it can perform the `--[no]replicate-check`. Disabling that option by specifying –no-replicate-check disables this check, but it also disables immediate reporting of checksum differences, thereby requiring a second run of the tool with *--replicate-check-only* to find and print checksum differences.

PLUGIN

The file specified by *--plugin* must define a class (i.e. a package) called `pt_table_checksum_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_replicate_check
after_replicate_check
get_slave_lag
before_checksum_table
after_checksum_table
```

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(
      slaves         => $slaves,
      slave_lag_cxns => $slave_lag_cxns,
      repl_table     => $repl_table,
   );
}
```

The comment `# --plugin hook` precedes every hook call.

Please contact Percona if you have questions or need help.

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

copy: no

DSN table database.

- F

  dsn: mysql_read_default_file; copy: yes

  Defaults file for connection values.

- 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: no

  Socket file to use for connection.

- t

  copy: no

  DSN table table.

- u

  dsn: user; copy: yes

  User for login if not current user.

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 pt-table-checksum ... > FILE 2>&1
```

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

SYSTEM REQUIREMENTS

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

BUGS

For a list of known bugs, see http://www.percona.com/bugs/pt-table-checksum.

Please report bugs at https://bugs.launchpad.net/percona-toolkit. Include the following information in your bug report:

- Complete command-line used to run the tool
- Tool *--version*
- MySQL version of all servers involved
- Output from the tool including STDERR
- Input files (log/dump/config files, etc.)

If possible, include debugging output by running the tool with `PTDEBUG`; see "ENVIRONMENT".

DOWNLOADING

Visit http://www.percona.com/software/percona-toolkit/ to download the latest release of Percona Toolkit. Or, get the latest release from the command line:

```
wget percona.com/get/percona-toolkit.tar.gz

wget percona.com/get/percona-toolkit.rpm

wget percona.com/get/percona-toolkit.deb
```

You can also get individual tools from the latest release:

```
wget percona.com/get/TOOL
```

Replace `TOOL` with the name of any tool.

AUTHORS

Baron Schwartz and Daniel Nichter

ACKNOWLEDGMENTS

Claus Jeppesen, Francois Saint-Jacques, Giuseppe Maxia, Heikki Tuuri, James Briggs, Martin Friebe, and Sergey Zhuravlev

ABOUT PERCONA TOOLKIT

This tool is part of Percona Toolkit, a collection of advanced command-line tools for MySQL developed by Percona. 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. Visit http://www.percona.com/software/ to learn about other free, open-source software from Percona.

COPYRIGHT, LICENSE, AND WARRANTY

VERSION

pt-table-checksum 3.0.4

CHAPTER

THIRTYONE

PT-TABLE-SYNC

NAME

pt-table-sync - Synchronize MySQL table data efficiently.

SYNOPSIS

Usage

```
pt-table-sync [OPTIONS] DSN [DSN]
```

pt-table-sync synchronizes data efficiently between MySQL tables.

This tool changes data, so for maximum safety, you should back up your data before using it. When synchronizing a server that is a replication slave with the *--replicate* or *--sync-to-master* methods, it **always** makes the changes on the replication master, **never** the replication slave directly. This is in general the only safe way to bring a replica back in sync with its master; changes to the replica are usually the source of the problems in the first place. However, the changes it makes on the master should be no-op changes that set the data to their current values, and actually affect only the replica.

Sync db.tbl on host1 to host2:

```
pt-table-sync --execute h=host1,D=db,t=tbl h=host2
```

Sync all tables on host1 to host2 and host3:

```
pt-table-sync --execute host1 host2 host3
```

Make slave1 have the same data as its replication master:

```
pt-table-sync --execute --sync-to-master slave1
```

Resolve differences that pt-table-checksum found on all slaves of master1:

```
pt-table-sync --execute --replicate test.checksum master1
```

Same as above but only resolve differences on slave1:

```
pt-table-sync --execute --replicate test.checksum \
  --sync-to-master slave1
```

Sync master2 in a master-master replication configuration, where master2's copy of db.tbl is known or suspected to be incorrect:

```
pt-table-sync --execute --sync-to-master h=master2,D=db,t=tbl
```

Note that in the master-master configuration, the following will NOT do what you want, because it will make changes directly on master2, which will then flow through replication and change master1's data:

```
# Don't do this in a master-master setup!
pt-table-sync --execute h=master1,D=db,t=tbl master2
```

RISKS

WARNING: **pt-table-sync** changes data! 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

  pt-table-sync is mature, proven in the real world, and well tested, but if used improperly it can have adverse consequences. Always test syncing first with *--dry-run* and *--print*.

DESCRIPTION

pt-table-sync does one-way and bidirectional synchronization of table data. It does **not** synchronize table structures, indexes, or any other schema objects. The following describes one-way synchronization. "BIDIRECTIONAL SYNCING" is described later.

This tool is complex and functions in several different ways. To use it safely and effectively, you should understand three things: the purpose of *--replicate*, finding differences, and specifying hosts. These three concepts are closely related and determine how the tool will run. The following is the abbreviated logic:

```
if DSN has a t part, sync only that table:
   if 1 DSN:
      if --sync-to-master:
         The DSN is a slave.  Connect to its master and sync.
   if more than 1 DSN:
      The first DSN is the source.  Sync each DSN in turn.
else if --replicate:
   if --sync-to-master:
      The DSN is a slave.  Connect to its master, find records
      of differences, and fix.
   else:
      The DSN is the master.  Find slaves and connect to each,
      find records of differences, and fix.
else:
   if only 1 DSN and --sync-to-master:
      The DSN is a slave.  Connect to its master, find tables and
      filter with --databases etc, and sync each table to the master.
   else:
      find tables, filtering with --databases etc, and sync each
      DSN to the first.
```

pt-table-sync can run in one of two ways: with *--replicate* or without. The default is to run without *--replicate* which causes **pt-table-sync** to automatically find differences efficiently with one of several algorithms (see "ALGORITHMS"). Alternatively, the value of *--replicate*, if specified, causes **pt-table-sync** to use the differences already found by having previously ran pt-table-checksum with its own `--replicate` option. Strictly speaking, you don't need to use *--replicate* because **pt-table-sync** can find differences, but many people use *--replicate* if, for example, they checksum regularly using pt-table-checksum then fix differences as needed with **pt-table-sync**. If you're unsure, read each tool's documentation carefully and decide for yourself, or consult with an expert.

Regardless of whether *--replicate* is used or not, you need to specify which hosts to sync. There are two ways: with *--sync-to-master* or without. Specifying *--sync-to-master* makes **pt-table-sync** expect one and only slave DSN on the command line. The tool will automatically discover the slave's master and sync it so that its data is the same as its master. This is accomplished by making changes on the master which then flow through replication and update the slave to resolve its differences. **Be careful though**: although this option specifies and syncs a single slave, if there are other slaves on the same master, they will receive via replication the changes intended for the slave that you're trying to sync.

Alternatively, if you do not specify *--sync-to-master*, the first DSN given on the command line is the source host. There is only ever one source host. If you do not also specify *--replicate*, then you must specify at least one other DSN as the destination host. There can be one or more destination hosts. Source and destination hosts must be independent; they cannot be in the same replication topology. **pt-table-sync** will die with an error if it detects that a destination host is a slave because changes are written directly to destination hosts (and it's not safe to write directly to slaves). Or, if you specify *--replicate* (but not *--sync-to-master*) then **pt-table-sync** expects one and only one master DSN on the command line. The tool will automatically discover all the master's slaves and sync them to the master. This is the only way to sync several (all) slaves at once (because *--sync-to-master* only specifies one slave).

Each host on the command line is specified as a DSN. The first DSN (or only DSN for cases like *--sync-to-master*) provides default values for other DSNs, whether those other DSNs are specified on the command line or auto-discovered by the tool. So in this example,

```
pt-table-sync --execute h=host1,u=msandbox,p=msandbox h=host2
```

the host2 DSN inherits the `u` and `p` DSN parts from the host1 DSN. Use the *--explain-hosts* option to see how **pt-table-sync** will interpret the DSNs given on the command line.

LIMITATIONS

Replicas using row-based replication

> **pt-table-sync** requires statement-based replication when used with the *--sync-to-master* or *--replicate* option. Therefore it will set `binlog_format=STATEMENT` on the master for its session if required. To do this user must have `SUPER` privilege.

OUTPUT

If you specify the *--verbose* option, you'll see information about the differences between the tables. There is one row per table. Each server is printed separately. For example,

```
# Syncing h=host1,D=test,t=test1
# DELETE REPLACE INSERT UPDATE ALGORITHM START    END      EXIT DATABASE.TABLE
#      0       0      3      0 Chunk     13:00:00 13:00:17 2    test.test1
```

Table test.test1 on host1 required 3 INSERT statements to synchronize and it used the Chunk algorithm (see "ALGORITHMS"). The sync operation for this table started at 13:00:00 and ended 17 seconds later (times taken from NOW() on the source host). Because differences were found, its "EXIT STATUS" was 2.

If you specify the *--print* option, you'll see the actual SQL statements that the script uses to synchronize the table if *--execute* is also specified.

If you want to see the SQL statements that **pt-table-sync** is using to select chunks, nibbles, rows, etc., then specify *--print* once and *--verbose* twice. Be careful though: this can print a lot of SQL statements.

There are cases where no combination of INSERT, UPDATE or DELETE statements can resolve differences without violating some unique key. For example, suppose there's a primary key on column a and a unique key on column b. Then there is no way to sync these two tables with straightforward UPDATE statements:

```
+---+---+  +---+---+
| a | b |  | a | b |
+---+---+  +---+---+
| 1 | 2 |  | 1 | 1 |
| 2 | 1 |  | 2 | 2 |
+---+---+  +---+---+
```

The tool rewrites queries to DELETE and REPLACE in this case. This is automatically handled after the first index violation, so you don't have to worry about it.

Be careful when using **pt-table-sync** in any master-master setup. Master-master replication is inherently tricky, and it's easy to make mistakes. You need to be sure you're using the tool correctly for master-master replication. See the "SYNOPSIS" for the overview of the correct usage.

Also be careful with tables that have foreign key constraints with ON DELETE or ON UPDATE definitions because these might cause unintended changes on the child tables. See --[no]check-child-tables.

In general, this tool is best suited when your tables have a primary key or unique index. Although it can synchronize data in tables lacking a primary key or unique index, it might be best to synchronize that data by another means.

REPLICATION SAFETY

Synchronizing a replication master and slave safely is a non-trivial problem, in general. There are all sorts of issues to think about, such as other processes changing data, trying to change data on the slave, whether the destination and source are a master-master pair, and much more.

In general, the safe way to do it is to change the data on the master, and let the changes flow through replication to the slave like any other changes. However, this works only if it's possible to REPLACE into the table on the master. REPLACE works only if there's a unique index on the table (otherwise it just acts like an ordinary INSERT).

If your table has unique keys, you should use the *--sync-to-master* and/or *--replicate* options to sync a slave to its master. This will generally do the right thing. When there is no unique key on the table, there is no choice but to change the data on the slave, and **pt-table-sync** will detect that you're trying to do so. It will complain and die unless you specify --no-check-slave (see --[no]check-slave).

If you're syncing a table without a primary or unique key on a master-master pair, you must change the data on the destination server. Therefore, you need to specify --no-bin-log for safety (see --[no]bin-log). If you don't, the changes you make on the destination server will replicate back to the source server and change the data there!

The generally safe thing to do on a master-master pair is to use the *--sync-to-master* option so you don't change the data on the destination server. You will also need to specify --no-check-slave to keep **pt-table-sync** from complaining that it is changing data on a slave.

ALGORITHMS

pt-table-sync has a generic data-syncing framework which uses different algorithms to find differences. The tool automatically chooses the best algorithm for each table based on indexes, column types, and the algorithm preferences specified by *--algorithms*. The following algorithms are available, listed in their default order of preference:

Chunk

Finds an index whose first column is numeric (including date and time types), and divides the column's range of values into chunks of approximately *--chunk-size* rows. Syncs a chunk at a time by checksumming the entire chunk. If the chunk differs on the source and destination, checksums each chunk's rows individually to find the rows that differ.

It is efficient when the column has sufficient cardinality to make the chunks end up about the right size.

The initial per-chunk checksum is quite small and results in minimal network traffic and memory consumption. If a chunk's rows must be examined, only the primary key columns and a checksum are sent over the network, not the entire row. If a row is found to be different, the entire row will be fetched, but not before.

Note that this algorithm will not work if chunking a char column where all the values start with the same character. In that case, the tool will exit and suggest picking a different algorithm.

Nibble

Finds an index and ascends the index in fixed-size nibbles of *--chunk-size* rows, using a non-backtracking algorithm (see pt-archiver for more on this algorithm). It is very similar to "Chunk", but instead of pre-calculating the boundaries of each piece of the table based on index cardinality, it uses `LIMIT` to define each nibble's upper limit, and the previous nibble's upper limit to define the lower limit.

It works in steps: one query finds the row that will define the next nibble's upper boundary, and the next query checksums the entire nibble. If the nibble differs between the source and destination, it examines the nibble row-by-row, just as "Chunk" does.

GroupBy

Selects the entire table grouped by all columns, with a COUNT(*) column added. Compares all columns, and if they're the same, compares the COUNT(*) column's value to determine how many rows to insert or delete into the destination. Works on tables with no primary key or unique index.

Stream

Selects the entire table in one big stream and compares all columns. Selects all columns. Much less efficient than the other algorithms, but works when there is no suitable index for them to use.

Future Plans

Possibilities for future algorithms are TempTable (what I originally called bottom-up in earlier versions of this tool), DrillDown (what I originally called top-down), and GroupByPrefix (similar to how SqlYOG Job Agent works). Each algorithm has strengths and weaknesses. If you'd like to implement your favorite technique for finding differences between two sources of data on possibly different servers, I'm willing to help. The algorithms adhere to a simple interface that makes it pretty easy to write your own.

BIDIRECTIONAL SYNCING

Bidirectional syncing is a new, experimental feature. To make it work reliably there are a number of strict limitations:

```
* only works when syncing one server to other independent servers
* does not work in any way with replication
* requires that the table(s) are chunkable with the Chunk algorithm
* is not N-way, only bidirectional between two servers at a time
* does not handle DELETE changes
```

For example, suppose we have three servers: c1, r1, r2. c1 is the central server, a pseudo-master to the other servers (viz. r1 and r2 are not slaves to c1). r1 and r2 are remote servers. Rows in table foo are updated and inserted on all three servers and we want to synchronize all the changes between all the servers. Table foo has columns:

```
id    int PRIMARY KEY
ts    timestamp auto updated
name  varchar
```

Auto-increment offsets are used so that new rows from any server do not create conflicting primary key (id) values. In general, newer rows, as determined by the ts column, take precedence when a same but differing row is found during the bidirectional sync. "Same but differing" means that two rows have the same primary key (id) value but different values for some other column, like the name column in this example. Same but differing conflicts are resolved by a "conflict". A conflict compares some column of the competing rows to determine a "winner". The winning row becomes the source and its values are used to update the other row.

There are subtle differences between three columns used to achieve bidirectional syncing that you should be familiar with: chunk column (*--chunk-column*), comparison column(s) (*--columns*), and conflict column (*--conflict-column*). The chunk column is only used to chunk the table; e.g. "WHERE id >= 5 AND id < 10". Chunks are checksummed and when chunk checksums reveal a difference, the tool selects the rows in that chunk and checksums the *--columns* for each row. If a column checksum differs, the rows have one or more conflicting column values. In a traditional unidirectional sync, the conflict is a moot point because it can be resolved simply by updating the entire destination row with the source row's values. In a bidirectional sync, however, the *--conflict-column* (in accordance with other --conflict-* options list below) is compared to determine which row is "correct" or "authoritative"; this row becomes the "source".

To sync all three servers completely, two runs of **pt-table-sync** are required. The first run syncs c1 and r1, then syncs c1 and r2 including any changes from r1. At this point c1 and r2 are completely in sync, but r1 is missing any changes from r2 because c1 didn't have these changes when it and r1 were synced. So a second run is needed which syncs the servers in the same order, but this time when c1 and r1 are synced r1 gets r2's changes.

The tool does not sync N-ways, only bidirectionally between the first DSN given on the command line and each subsequent DSN in turn. So the tool in this example would be ran twice like:

```
pt-table-sync --bidirectional h=c1 h=r1 h=r2
```

The *--bidirectional* option enables this feature and causes various sanity checks to be performed. You must specify other options that tell **pt-table-sync** how to resolve conflicts for same but differing rows. These options are:

```
* --conflict-column
* --conflict-comparison
* --conflict-value
* --conflict-threshold
* --conflict-error">  (optional)
```

Use *--print* to test this option before *--execute*. The printed SQL statements will have comments saying on which host the statement would be executed if you used *--execute*.

Technical side note: the first DSN is always the "left" server and the other DSNs are always the "right" server. Since either server can become the source or destination it's confusing to think of them as "src" and "dst". Therefore, they're

generically referred to as left and right. It's easy to remember this because the first DSN is always to the left of the other server DSNs on the command line.

EXIT STATUS

The following are the exit statuses (also called return values, or return codes) when **pt-table-sync** finishes and exits.

```
STATUS  MEANING
======  ========================================================
0       Success.
1       Internal error.
2       At least one table differed on the destination.
3       Combination of 1 and 2.
```

OPTIONS

Specify at least one of *--print*, *--execute*, or *--dry-run*.

--where and *--replicate* are mutually exclusive.

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

--algorithms
type: string; default: Chunk,Nibble,GroupBy,Stream

Algorithm to use when comparing the tables, in order of preference.

For each table, **pt-table-sync** will check if the table can be synced with the given algorithms in the order that they're given. The first algorithm that can sync the table is used. See "ALGORITHMS".

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

--bidirectional
Enable bidirectional sync between first and subsequent hosts.

See "BIDIRECTIONAL SYNCING" for more information.

--`[no]bin-log`
default: yes

Log to the binary log (`SET SQL_LOG_BIN=1`).

Specifying `--no-bin-log` will `SET SQL_LOG_BIN=0`.

--buffer-in-mysql
Instruct MySQL to buffer queries in its memory.

This option adds the `SQL_BUFFER_RESULT` option to the comparison queries. This causes MySQL to execute the queries and place them in a temporary table internally before sending the results back to **pt-table-sync**. The advantage of this strategy is that **pt-table-sync** can fetch rows as desired without using a lot of memory inside the Perl process, while releasing locks on the MySQL table (to reduce contention with other queries). The disadvantage is that it uses more memory on the MySQL server instead.

You probably want to leave `--[no]buffer-to-client` enabled too, because buffering into a temp table and then fetching it all into Perl's memory is probably a silly thing to do. This option is most useful for the GroupBy and Stream algorithms, which may fetch a lot of data from the server.

--[no]buffer-to-client
default: yes

Fetch rows one-by-one from MySQL while comparing.

This option enables `mysql_use_result` which causes MySQL to hold the selected rows on the server until the tool fetches them. This allows the tool to use less memory but may keep the rows locked on the server longer.

If this option is disabled by specifying `--no-buffer-to-client` then `mysql_store_result` is used which causes MySQL to send all selected rows to the tool at once. This may result in the results "cursor" being held open for a shorter time on the server, but if the tables are large, it could take a long time anyway, and use all your memory.

For most non-trivial data sizes, you want to leave this option enabled.

This option is disabled when *--bidirectional* is used.

--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 MySQL. Any other value sets binmode on STDOUT without the utf8 layer, and runs SET NAMES after connecting to MySQL.

--[no]check-child-tables
default: yes

Check if *--execute* will adversely affect child tables. When *--replace*, *--replicate*, or *--sync-to-master* is specified, the tool may sync tables using `REPLACE` statements. If a table being synced has child tables with `ON DELETE CASCADE`, `ON UPDATE CASCADE`, or `ON UPDATE SET NULL`, the tool prints an error and skips the table because `REPLACE` becomes `DELETE` then `INSERT`, so the `DELETE` will cascade to the child table and delete its rows. In the worst case, this can delete all rows in child tables!

Specify `--no-check-child-tables` to disable this check. To completely avoid affecting child tables, also specify `--no-foreign-key-checks` so MySQL will not cascade any operations from the parent to child tables.

This check is only preformed if *--execute* and one of *--replace*, *--replicate*, or *--sync-to-master* is specified. *--print* does not check child tables.

The error message only prints the first child table found with an `ON DELETE CASCADE`, `ON UPDATE CASCADE`, or `ON UPDATE SET NULL` foreign key constraint. There could be other affected child tables.

--[no]check-master
default: yes

With *--sync-to-master*, try to verify that the detected master is the real master.

--[no]check-slave
default: yes

Check whether the destination server is a slave.

If the destination server is a slave, it's generally unsafe to make changes on it. However, sometimes you have to; *--replace* won't work unless there's a unique index, for example, so you can't make changes on the master in that scenario. By default **pt-table-sync** will complain if you try to change data on a slave. Specify `--no-check-slave` to disable this check. Use it at your own risk.

--[no]check-triggers
default: yes

Check that no triggers are defined on the destination table.

Triggers were introduced in MySQL v5.0.2, so for older versions this option has no effect because triggers will not be checked.

--chunk-column
type: string

Chunk the table on this column.

--chunk-index
type: string

Chunk the table using this index.

--chunk-size
type: string; default: 1000

Number of rows or data size per chunk.

The size of each chunk of rows for the "Chunk" and "Nibble" algorithms. The size can be either a number of rows, or a data size. Data sizes are specified with a suffix of k=kibibytes, M=mebibytes, G=gibibytes. Data sizes are converted to a number of rows by dividing by the average row length.

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

Compare this comma-separated list of columns.

--config
type: Array

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

--conflict-column
type: string

Compare this column when rows conflict during a *--bidirectional* sync.

When a same but differing row is found the value of this column from each row is compared according to *--conflict-comparison*, *--conflict-value* and *--conflict-threshold* to determine which row has the correct data and becomes the source. The column can be any type for which there is an appropriate *--conflict-comparison* (this is almost all types except, for example, blobs).

This option only works with *--bidirectional*. See "BIDIRECTIONAL SYNCING" for more information.

--conflict-comparison
type: string

Choose the *--conflict-column* with this property as the source.

The option affects how the *--conflict-column* values from the conflicting rows are compared. Possible comparisons are one of these MAGIC_comparisons:

```
newest|oldest|greatest|least|equals|matches

COMPARISON  CHOOSES ROW WITH
==========  ========================================================
newest      Newest temporal --conflict-column value
oldest      Oldest temporal --conflict-column value
greatest    Greatest numerical "--conflict-column value
least       Least numerical --conflict-column value
equals      --conflict-column value equal to --conflict-value
```

```
matches     --conflict-column value matching Perl regex pattern
            --conflict-value
```

This option only works with *--bidirectional*. See "BIDIRECTIONAL SYNCING" for more information.

--conflict-error
type: string; default: warn

How to report unresolvable conflicts and conflict errors

This option changes how the user is notified when a conflict cannot be resolved or causes some kind of error. Possible values are:

```
* warn: Print a warning to STDERR about the unresolvable conflict
* die:  Die, stop syncing, and print a warning to STDERR
```

This option only works with *--bidirectional*. See "BIDIRECTIONAL SYNCING" for more information.

--conflict-threshold
type: string

Amount by which one *--conflict-column* must exceed the other.

The *--conflict-threshold* prevents a conflict from being resolved if the absolute difference between the two *--conflict-column* values is less than this amount. For example, if two *--conflict-column* have timestamp values "2009-12-01 12:00:00" and "2009-12-01 12:05:00" the difference is 5 minutes. If *--conflict-threshold* is set to "5m" the conflict will be resolved, but if *--conflict-threshold* is set to "6m" the conflict will fail to resolve because the difference is not greater than or equal to 6 minutes. In this latter case, *--conflict-error* will report the failure.

This option only works with *--bidirectional*. See "BIDIRECTIONAL SYNCING" for more information.

--conflict-value
type: string

Use this value for certain *--conflict-comparison*.

This option gives the value for `equals` and `matches` *--conflict-comparison*.

This option only works with *--bidirectional*. See "BIDIRECTIONAL SYNCING" for more information.

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

Sync only this comma-separated list of databases.

A common request is to sync tables from one database with tables from another database on the same or different server. This is not yet possible. *--databases* will not do it, and you can't do it with the D part of the DSN either because in the absence of a table name it assumes the whole server should be synced and the D part controls only the connection's default database.

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

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

--dry-run
Analyze, decide the sync algorithm to use, print and exit.

Implies *--verbose* so you can see the results. The results are in the same output format that you'll see from actually running the tool, but there will be zeros for rows affected. This is because the tool actually executes, but stops before it compares any data and just returns zeros. The zeros do not mean there are no changes to be made.

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

Sync only this comma-separated list of storage engines.

--execute
Execute queries to make the tables have identical data.

This option makes **pt-table-sync** actually sync table data by executing all the queries that it created to resolve table differences. Therefore, **the tables will be changed!** And unless you also specify *--verbose*, the changes will be made silently. If this is not what you want, see *--print* or *--dry-run*.

--explain-hosts
Print connection information and exit.

Print out a list of hosts to which **pt-table-sync** will connect, with all the various connection options, and exit.

--float-precision
type: int

Precision for `FLOAT` and `DOUBLE` number-to-string conversion. Causes FLOAT and DOUBLE values to be rounded to the specified number of digits after the decimal point, with the ROUND() function in MySQL. This can help avoid checksum mismatches due to different floating-point representations of the same values on different MySQL versions and hardware. The default is no rounding; the values are converted to strings by the CONCAT() function, and MySQL chooses the string representation. If you specify a value of 2, for example, then the values 1.008 and 1.009 will be rounded to 1.01, and will checksum as equal.

--`[no]foreign-key-checks`
default: yes

Enable foreign key checks (`SET FOREIGN_KEY_CHECKS=1`).

Specifying `--no-foreign-key-checks` will `SET FOREIGN_KEY_CHECKS=0`.

--function
type: string

Which hash function you'd like to use for checksums.

The default is `CRC32`. Other good choices include `MD5` and `SHA1`. If you have installed the `FNV_64` user-defined function, **pt-table-sync** will detect it and prefer to use it, because it is much faster than the built-ins. You can also use MURMUR_HASH if you've installed that user-defined function. Both of these are distributed with Percona Server. See pt-table-checksum for more information and benchmarks.

--help
Show help and exit.

--`[no]hex-blob`
default: yes

`HEX()` `BLOB`, `TEXT` and `BINARY` columns.

When row data from the source is fetched to create queries to sync the data (i.e. the queries seen with *--print* and executed by *--execute*), binary columns are wrapped in HEX() so the binary data does not produce an invalid SQL statement. You can disable this option but you probably shouldn't.

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

Connect to host.

--ignore-columns
type: Hash

Ignore this comma-separated list of column names in comparisons.

This option causes columns not to be compared. However, if a row is determined to differ between tables, all columns in that row will be synced, regardless. (It is not currently possible to exclude columns from the sync process itself, only from the comparison.)

--ignore-databases
type: Hash

Ignore this comma-separated list of databases.

(system databases such as **information_schema** and **performance_schema** are ignored by default)

--ignore-engines
type: Hash; default: FEDERATED,MRG_MyISAM

Ignore this comma-separated list of storage engines.

--ignore-tables
type: Hash

Ignore this comma-separated list of tables.

Table names may be qualified with the database name.

--ignore-tables-regex
type: string; group: Filter

Ignore tables whose names match the Perl regex.

--[no]index-hint
default: yes

Add FORCE/USE INDEX hints to the chunk and row queries.

By default **pt-table-sync** adds a FORCE/USE INDEX hint to each SQL statement to coerce MySQL into using the index chosen by the sync algorithm or specified by *--chunk-index*. This is usually a good thing, but in rare cases the index may not be the best for the query so you can suppress the index hint by specifying --no-index-hint and let MySQL choose the index.

This does not affect the queries printed by *--print*; it only affects the chunk and row queries that **pt-table-sync** uses to select and compare rows.

--lock
type: int

Lock tables: 0=none, 1=per sync cycle, 2=per table, or 3=globally.

This uses LOCK TABLES. This can help prevent tables being changed while you're examining them. The possible values are as follows:

```
VALUE  MEANING
=====  ==================================================
0      Never lock tables.
1      Lock and unlock one time per sync cycle (as implemented
       by the syncing algorithm).  This is the most granular
       level of locking available.  For example, the Chunk
       algorithm will lock each chunk of C<N> rows, and then
       unlock them if they are the same on the source and the
       destination, before moving on to the next chunk.
2      Lock and unlock before and after each table.
3      Lock and unlock once for every server (DSN) synced, with
       C<FLUSH TABLES WITH READ LOCK>.
```

A replication slave is never locked if *--replicate* or *--sync-to-master* is specified, since in theory locking the table on the master should prevent any changes from taking place. (You are not changing data on your slave, right?) If *--wait* is given, the master (source) is locked and then the tool waits for the slave to catch up to the master before continuing.

If `--transaction` is specified, `LOCK TABLES` is not used. Instead, lock and unlock are implemented by beginning and committing transactions. The exception is if *--lock* is 3.

If `--no-transaction` is specified, then `LOCK TABLES` is used for any value of *--lock*. See `--[no]transaction`.

--lock-and-rename

Lock the source and destination table, sync, then swap names. This is useful as a less-blocking ALTER TABLE, once the tables are reasonably in sync with each other (which you may choose to accomplish via any number of means, including dump and reload or even something like pt-archiver). It requires exactly two DSNs and assumes they are on the same server, so it does no waiting for replication or the like. Tables are locked with LOCK TABLES.

--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.

--print

Print queries that will resolve differences.

If you don't trust **pt-table-sync**, or just want to see what it will do, this is a good way to be safe. These queries are valid SQL and you can run them yourself if you want to sync the tables manually.

--recursion-method

type: array; default: processlist,hosts

Preferred recursion method used to find slaves.

Possible methods are:

```
METHOD       USES
===========  ==================
processlist  SHOW PROCESSLIST
hosts        SHOW SLAVE HOSTS
none         Do not find slaves
```

The processlist method is preferred because SHOW SLAVE HOSTS is not reliable. However, the hosts method is required if the server uses a non-standard port (not 3306). Usually **pt-table-sync** does the right thing and finds the slaves, but you may give a preferred method and it will be used first. If it doesn't find any slaves, the other methods will be tried.

--replace

Write all `INSERT` and `UPDATE` statements as `REPLACE`.

This is automatically switched on as needed when there are unique index violations.

--replicate
type: string

Sync tables listed as different in this table.

Specifies that **pt-table-sync** should examine the specified table to find data that differs. The table is exactly the same as the argument of the same name to pt-table-checksum. That is, it contains records of which tables (and ranges of values) differ between the master and slave.

For each table and range of values that shows differences between the master and slave, `pt-table-checksum` will sync that table, with the appropriate `WHERE` clause, to its master.

This automatically sets *`--wait`* to 60 and causes changes to be made on the master instead of the slave.

If *`--sync-to-master`* is specified, the tool will assume the server you specified is the slave, and connect to the master as usual to sync.

Otherwise, it will try to use `SHOW PROCESSLIST` to find slaves of the server you specified. If it is unable to find any slaves via `SHOW PROCESSLIST`, it will inspect `SHOW SLAVE HOSTS` instead. You must configure each slave's `report-host`, `report-port` and other options for this to work right. After finding slaves, it will inspect the specified table on each slave to find data that needs to be synced, and sync it.

The tool examines the master's copy of the table first, assuming that the master is potentially a slave as well. Any table that shows differences there will **NOT** be synced on the slave(s). For example, suppose your replication is set up as A->B, B->C, B->D. Suppose you use this argument and specify server B. The tool will examine server B's copy of the table. If it looks like server B's data in table `test.tbl1` is different from server A's copy, the tool will not sync that table on servers C and D.

--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 MySQL 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.

--sync-to-master
Treat the DSN as a slave and sync it to its master.

Treat the server you specified as a slave. Inspect `SHOW SLAVE STATUS`, connect to the server's master, and treat the master as the source and the slave as the destination. Causes changes to be made on the master. Sets *`--wait`* to 60 by default, sets *`--lock`* to 1 by default, and disables `--[no]transaction` by default. See also *`--replicate`*, which changes this option's behavior.

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

Sync only this comma-separated list of tables.

Table names may be qualified with the database name.

`--timeout-ok`
Keep going if *`--wait`* fails.

If you specify *`--wait`* and the slave doesn't catch up to the master's position before the wait times out, the default behavior is to abort. This option makes the tool keep going anyway. **Warning**: if you are trying to get a consistent comparison between the two servers, you probably don't want to keep going after a timeout.

`--[no]transaction`
Use transactions instead of `LOCK TABLES`.

The granularity of beginning and committing transactions is controlled by *`--lock`*. This is enabled by default, but since *`--lock`* is disabled by default, it has no effect.

Most options that enable locking also disable transactions by default, so if you want to use transactional locking (via `LOCK IN SHARE MODE` and `FOR UPDATE`, you must specify `--transaction` explicitly.

If you don't specify `--transaction` explicitly **`pt-table-sync`** will decide on a per-table basis whether to use transactions or table locks. It currently uses transactions on InnoDB tables, and table locks on all others.

If `--no-transaction` is specified, then **`pt-table-sync`** will not use transactions at all (not even for InnoDB tables) and locking is controlled by *`--lock`*.

When enabled, either explicitly or implicitly, the transaction isolation level is set `REPEATABLE READ` and transactions are started `WITH CONSISTENT SNAPSHOT`.

`--trim`
`TRIM() VARCHAR` columns in `BIT_XOR` and `ACCUM` modes. Helps when comparing MySQL 4.1 to >= 5.0.

This is useful when you don't care about the trailing space differences between MySQL versions which vary in their handling of trailing spaces. MySQL 5.0 and later all retain trailing spaces in `VARCHAR`, while previous versions would remove them.

`--[no]unique-checks`
default: yes

Enable unique key checks (`SET UNIQUE_CHECKS=1`).

Specifying `--no-unique-checks` will `SET UNIQUE_CHECKS=0`.

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

User for login if not current user.

`--verbose`
short form: -v; cumulative: yes

Print results of sync operations.

See "OUTPUT" for more details about the output.

`--version`
Show version and exit.

--[no]version-check
default: yes

Check for the latest version of Percona Toolkit, MySQL, and other programs.

This is a standard "check for updates automatically" feature, with two additional features. First, the tool checks the version of other programs on the local system in addition to its own version. For example, it checks the version of every MySQL server it connects to, Perl, and the Perl module DBD::mysql. Second, it checks for and warns about versions with known problems. For example, MySQL 5.5.25 had a critical bug and was re-released as 5.5.25a.

Any updates or known problems are printed to STDOUT before the tool's normal output. This feature should never interfere with the normal operation of the tool.

For more information, visit https://www.percona.com/version-check.

--wait
short form: -w; type: time

How long to wait for slaves to catch up to their master.

Make the master wait for the slave to catch up in replication before comparing the tables. The value is the number of seconds to wait before timing out (see also *--timeout-ok*). Sets *--lock* to 1 and --[no]transaction to 0 by default. If you see an error such as the following,

```
MASTER_POS_WAIT returned -1
```

It means the timeout was exceeded and you need to increase it.

The default value of this option is influenced by other options. To see what value is in effect, run with *--help*.

To disable waiting entirely (except for locks), specify *--wait* 0. This helps when the slave is lagging on tables that are not being synced.

--where
type: string

WHERE clause to restrict syncing to part of the table.

--[no]zero-chunk
default: yes

Add a chunk for rows with zero or zero-equivalent values. The only has an effect when *--chunk-size* is specified. The purpose of the zero chunk is to capture a potentially large number of zero values that would imbalance the size of the first chunk. For example, if a lot of negative numbers were inserted into an unsigned integer column causing them to be stored as zeros, then these zero values are captured by the zero chunk instead of the first chunk and all its non-zero values.

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

  Database containing the table to be synced.

- 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

  copy: yes

  Table to be synced.

- u

  dsn: user; copy: yes

  User for login if not current user.

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 pt-table-sync ... > FILE 2>&1
```

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

SYSTEM REQUIREMENTS

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

BUGS

For a list of known bugs, see http://www.percona.com/bugs/pt-table-sync.

Please report bugs at https://bugs.launchpad.net/percona-toolkit. Include the following information in your bug report:

- Complete command-line used to run the tool
- Tool *--version*
- MySQL version of all servers involved
- Output from the tool including STDERR
- Input files (log/dump/config files, etc.)

If possible, include debugging output by running the tool with PTDEBUG; see "ENVIRONMENT".

DOWNLOADING

Visit http://www.percona.com/software/percona-toolkit/ to download the latest release of Percona Toolkit. Or, get the latest release from the command line:

```
wget percona.com/get/percona-toolkit.tar.gz

wget percona.com/get/percona-toolkit.rpm

wget percona.com/get/percona-toolkit.deb
```

You can also get individual tools from the latest release:

```
wget percona.com/get/TOOL
```

Replace TOOL with the name of any tool.

AUTHORS

Baron Schwartz

ACKNOWLEDGMENTS

My work is based in part on Giuseppe Maxia's work on distributed databases, http://www.sysadminmag.com/articles/2004/0408/ and code derived from that article. There is more explanation, and a link to the code, at http://www.perlmonks.org/?node_id=381053.

Another programmer extended Maxia's work even further. Fabien Coelho changed and generalized Maxia's technique, introducing symmetry and avoiding some problems that might have caused too-frequent checksum collisions. This work grew into pg_comparator, http://www.coelho.net/pg_comparator/. Coelho also explained the technique further in a paper titled "Remote Comparison of Database Tables" (http://cri.ensmp.fr/classement/doc/A-375.pdf).

This existing literature mostly addressed how to find the differences between the tables, not how to resolve them once found. I needed a tool that would not only find them efficiently, but would then resolve them. I first began thinking about how to improve the technique further with my article http://tinyurl.com/mysql-data-diff-algorithm, where I discussed a number of problems with the Maxia/Coelho "bottom-up" algorithm. After writing that article, I began to

write this tool. I wanted to actually implement their algorithm with some improvements so I was sure I understood it completely. I discovered it is not what I thought it was, and is considerably more complex than it appeared to me at first. Fabien Coelho was kind enough to address some questions over email.

The first versions of this tool implemented a version of the Coelho/Maxia algorithm, which I called "bottom-up", and my own, which I called "top-down." Those algorithms are considerably more complex than the current algorithms and I have removed them from this tool, and may add them back later. The improvements to the bottom-up algorithm are my original work, as is the top-down algorithm. The techniques to actually resolve the differences are also my own work.

Another tool that can synchronize tables is the SQLyog Job Agent from webyog. Thanks to Rohit Nadhani, SJA's author, for the conversations about the general techniques. There is a comparison of **pt-table-sync** and SJA at http://tinyurl.com/maatkit-vs-sqlyog

Thanks to the following people and organizations for helping in many ways:

The Rimm-Kaufman Group http://www.rimmkaufman.com/, MySQL AB http://www.mysql.com/, Blue Ridge InternetWorks http://www.briworks.com/, Percona http://www.percona.com/, Fabien Coelho, Giuseppe Maxia and others at MySQL AB, Kristian Koehntopp (MySQL AB), Rohit Nadhani (WebYog), The helpful monks at Perlmonks, And others too numerous to mention.

ABOUT PERCONA TOOLKIT

This tool is part of Percona Toolkit, a collection of advanced command-line tools for MySQL developed by Percona. 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. Visit http://www.percona.com/software/ to learn about other free, open-source software from Percona.

COPYRIGHT, LICENSE, AND WARRANTY

VERSION

pt-table-sync 3.0.4

CHAPTER

THIRTYTWO

PT-TABLE-USAGE

NAME

pt-table-usage - Analyze how queries use tables.

SYNOPSIS

Usage

```
pt-table-usage [OPTIONS] [FILES]
```

pt-table-usage reads queries from a log and analyzes how they use tables. If no FILE is specified, it reads STDIN. It prints a report for each query.

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

DESCRIPTION

pt-table-usage reads queries from a log and analyzes how they use tables. The log should be in MySQL's slow query log format.

Table usage is more than simply an indication of which tables the query reads or writes. It also indicates data flow: data in and data out. The tool determines the data flow by the contexts in which tables appear. A single query can use a table in several different contexts simultaneously. The tool's output lists every context for every table. This CONTEXT-TABLE list indicates how data flows between tables. The "OUTPUT" section lists the possible contexts and describes how to read a table usage report.

The tool analyzes data flow down to the level of individual columns, so it is helpful if columns are identified unambiguously in the query. If a query uses only one table, then all columns must be from that table, and there's no difficulty. But if a query uses multiple tables and the column names are not table-qualified, then it is necessary to use `EXPLAIN EXTENDED`, followed by `SHOW WARNINGS`, to determine to which tables the columns belong.

If the tool does not know the query's default database, which can occur when the database is not printed in the log, then `EXPLAIN EXTENDED` can fail. In this case, you can specify a default database with *`--database`*. You can also use the *`--create-table-definitions`* option to help resolve ambiguities.

OUTPUT

The tool prints a usage report for each table in every query, similar to the following:

```
Query_id: 0x1CD27577D202A339.1
UPDATE t1
SELECT DUAL
JOIN t1
JOIN t2
WHERE t1

Query_id: 0x1CD27577D202A339.2
UPDATE t2
SELECT DUAL
JOIN t1
JOIN t2
WHERE t1
```

The first line contains the query ID, which by default is the same as those shown in pt-query-digest reports. It is an MD5 checksum of the query's "fingerprint," which is what remains after removing literals, collapsing white space, and a variety of other transformations. The query ID has two parts separated by a period: the query ID and the table number. If you wish to use a different value to identify the query, you can specify the *`--id-attribute`* option.

The previous example shows two paragraphs for a single query, not two queries. Note that the query ID is identical for the two, but the table number differs. The table number increments by 1 for each table that the query updates. Only multi-table UPDATE queries can update multiple tables with a single query, so the table number is 1 for all other types of queries. (The tool does not support multi-table DELETE queries.) The example output above is from this query:

```
UPDATE t1 AS a JOIN t2 AS b USING (id)
SET a.foo="bar", b.foo="bat"
WHERE a.id=1;
```

The `SET` clause indicates that the query updates two tables: `a` aliased as `t1`, and `b` aliased as `t2`.

After the first line, the tool prints a variable number of CONTEXT-TABLE lines. Possible contexts are as follows:

- SELECT

  SELECT means that the query retrieves data from the table for one of two reasons. The first is to be returned to the user as part of a result set. Only SELECT queries return result sets, so the report always shows a SELECT context for SELECT queries.

  The second case is when data flows to another table as part of an INSERT or UPDATE. For example, the UPDATE query in the example above has the usage:

  ```
  SELECT DUAL
  ```

  This refers to:

```
SET a.foo="bar", b.foo="bat"
```

The tool uses DUAL for any values that do not originate in a table, in this case the literal values "bar" and "bat". If that `SET` clause were `SET a.foo=b.foo` instead, then the complete usage would be:

```
Query_id: 0x1CD27577D202A339.1
UPDATE t1
SELECT t2
JOIN t1
JOIN t2
WHERE t1
```

The presence of a SELECT context after another context, such as UPDATE or INSERT, indicates where the UPDATE or INSERT retrieves its data. The example immediately above reflects an UPDATE query that updates rows in table `t1` with data from table `t2`.

- Any other verb

  Any other verb, such as INSERT, UPDATE, DELETE, etc. may be a context. These verbs indicate that the query modifies data in some way. If a SELECT context follows one of these verbs, then the query reads data from the SELECT table and writes it to this table. This happens, for example, with INSERT..SELECT or UPDATE queries that use values from tables instead of constant values.

  These query types are not supported: SET, LOAD, and multi-table DELETE.

- JOIN

  The JOIN context lists tables that are joined, either with an explicit JOIN in the FROM clause, or implicitly in the WHERE clause, such as `t1.id = t2.id`.

- WHERE

  The WHERE context lists tables that are used in the WHERE clause to filter results. This does not include tables that are implicitly joined in the WHERE clause; those are listed as JOIN contexts. For example:

  ```
  WHERE t1.id > 100 AND t1.id < 200 AND t2.foo IS NOT NULL
  ```

  Results in:

  ```
  WHERE t1
  WHERE t2
  ```

  The tool lists only distinct tables; that is why table `t1` is listed only once.

- TLIST

  The TLIST context lists tables that the query accesses, but which do not appear in any other context. These tables are usually an implicit cartesian join. For example, the query `SELECT * FROM t1, t2` results in:

  ```
  Query_id: 0xBDDEB6EDA41897A8.1
  SELECT t1
  SELECT t2
  TLIST t1
  TLIST t2
  ```

  First of all, there are two SELECT contexts, because `SELECT *` selects rows from all tables; `t1` and `t2` in this case. Secondly, the tables are implicitly joined, but without any kind of join condition, which results in a cartesian join as indicated by the TLIST context for each.

EXIT STATUS

pt-table-usage exits 1 on any kind of error, or 0 if no errors.

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 MySQL.

--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 MySQL. Any other value sets binmode on STDOUT without the utf8 layer, and runs SET NAMES after connecting to MySQL.

--config
type: Array

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

--constant-data-value
type: string; default: DUAL

Table to print as the source for constant data (literals). This is any data not retrieved from tables (or subqueries, because subqueries are not supported). This includes literal values such as strings ("foo") and numbers (42), or functions such as `NOW()`. For example, in the query `INSERT INTO t (c) VALUES ('a')`, the string 'a' is constant data, so the table usage report is:

```
INSERT t
SELECT DUAL
```

The first line indicates that the query inserts data into table `t`, and the second line indicates that the inserted data comes from some constant value.

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

Continue to work even if there is an error.

--create-table-definitions
type: array

Read `CREATE TABLE` definitions from this list of comma-separated files. If you cannot use *`--explain-extended`* to fully qualify table and column names, you can save the output of `mysqldump --no-data` to one or more files and specify those files with this option. The tool will parse all `CREATE TABLE` definitions from the files and use this information to qualify table and column names. If a column name appears in multiple tables, or a table name appears in multiple databases, the ambiguities cannot be resolved.

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

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

Default database.

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

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

--explain-extended
type: DSN

A server to execute EXPLAIN EXTENDED queries. This may be necessary to resolve ambiguous (unqualified) column and table names.

--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 is compiled into a subroutine with one argument: $event. If the given value is a readable file, then **pt-table-usage** reads the entire file and uses its contents as the code.

Filters are implemented in the same fashion as in the pt-query-digest tool, so please refer to its documentation for more information.

--help
Show help and exit.

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

Connect to host.

--id-attribute
type: string

Identify each event using this attribute. The default is to use a query ID, which is an MD5 checksum of the query's fingerprint.

--log
type: string

Print all output to this file when daemonized.

--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.

--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.

--query
type: string

Analyze the specified query instead of reading a log file.

--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. If an event is not received after the specified time, the tool stops reading the input and prints its reports.

This option requires the Perl POSIX module.

--run-time
type: time

How long to run before exiting. The default is to run forever (you can interrupt with CTRL-C).

--set-vars
type: Array

Set the MySQL 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.

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

User for login if not current user.

--version
Show version and exit.

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

  copy: no

  Default database.

- F

  dsn: mysql_read_default_file; copy: no

  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: no

  Socket file to use for connection.

- u

  dsn: user; copy: yes

  User for login if not current user.

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 pt-table-usage ... > FILE 2>&1
```

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

SYSTEM REQUIREMENTS

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

BUGS

For a list of known bugs, see http://www.percona.com/bugs/pt-table-usage.

Please report bugs at https://bugs.launchpad.net/percona-toolkit. Include the following information in your bug report:

- Complete command-line used to run the tool
- Tool *--version*
- MySQL version of all servers involved
- Output from the tool including STDERR
- Input files (log/dump/config files, etc.)

If possible, include debugging output by running the tool with `PTDEBUG`; see "ENVIRONMENT".

DOWNLOADING

Visit http://www.percona.com/software/percona-toolkit/ to download the latest release of Percona Toolkit. Or, get the latest release from the command line:

```
wget percona.com/get/percona-toolkit.tar.gz

wget percona.com/get/percona-toolkit.rpm

wget percona.com/get/percona-toolkit.deb
```

You can also get individual tools from the latest release:

```
wget percona.com/get/TOOL
```

Replace `TOOL` with the name of any tool.

AUTHORS

Daniel Nichter

ABOUT PERCONA TOOLKIT

This tool is part of Percona Toolkit, a collection of advanced command-line tools for MySQL developed by Percona. 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. Visit http://www.percona.com/software/ to learn about other free, open-source software from Percona.

COPYRIGHT, LICENSE, AND WARRANTY

VERSION

pt-table-usage 3.0.4

CHAPTER

THIRTYTHREE

PT-UPGRADE

NAME

pt-upgrade - Verify that query results are identical on different servers.

SYNOPSIS

Usage

```
pt-upgrade [OPTIONS] LOGS|RESULTS DSN [DSN]
```

pt-upgrade executes queries in the given MySQL `LOGS` on each `DSN`, compares the results, and reports any significant differences. The tool can also save the results for later analyses. `LOGS` can be slow, general, binary, tcpdump, and "raw".

Compare host2 to host1 using queries in `slow.log`:

```
pt-upgrade h=host1 h=host2 slow.log
```

Compare host2 to saved results from host1:

```
pt-upgrade h=host1 --save-results host1_results/ slow.log

pt-upgrade host1_results1/ h=host2
```

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

DESCRIPTION

pt-upgrade helps determine if it is safe to upgrade (or downgrade) to a new version of MySQL. A safe and conservative upgrade plan has several steps, one of which is ensuring that queries will produce identical results on the new version of MySQL.

pt-upgrade executes queries from slow, general, binary, tcpdump, and "raw" logs on two servers, compares many aspects of each query's exeuction and results, and reports any signficant differences. The two servers are typically development servers, one running the current production version of MySQL and the other running the new version of MySQL.

USE CASES

pt-upgrade has two use cases. The first, canonical case is running "host to host". A log file and two DSN are given on the command line, one for each MySQL server. See the first example in the "SYNOPSIS". Queries are executed and compared on each server as the tool runs. Queries with differences are printed as the tool runs, or when it finishes (see "WHEN QUERIES ARE REPORTED"). Nothing is saved to disk, so this use case requires less hard disk space, but the queries must be executed on both servers if the tool is ran again, even if one of the servers hasn't changed. If there are a lot of queries or executing them takes a long time, and one server doesn't change, you may want to use the second use case.

The second use case is running "reference results to host". Reference results are the complete results from a single MySQL server, saved to disk. In this case, you must first generate the reference results with *--save-results*, then run the tool a second time to compare another MySQL server to the results. See the second example in the "SYNOPSIS". Results are typically generated for the current version of MySQL which doesn't change. This use case can require *a lot* of disk space because the results (i.e. rows) for all queries must be saved, plus other data about the queries. If you plan to do many comparisons against a fixed version of MySQL, this use case is more efficient. Or if you don't have access to both servers at the same time, this use case allows you to "execute now, compare later".

IMPORTANT CONSIDERATIONS

CONSISTENCY

Consistent environments and consistent data are crucial for obtaining an accurate report. **pt-upgrade** should never be ran on a production server or any active server because there is no easy way to ensure a synchronous read for each query. If data is changing on either server while **pt-upgrade** is running, the report could contain more false-positives than legitimate differences. ** **pt-upgrade** assumes that both MySQL servers are static, unchanging (except for any changes made by the tool if ran with ''–no-read-only'').** A read-only workload shouldn't affect the tool, except maybe query times, so read-only slaves could be used.

COMPARED TO

In a host to host comparison, results from the first host establish the norm to which results from the second host are compared. In a reference results to host comparison, the reference results are the norm to which the host is compared. Comparative phrases like "smaller than", "better than", etc. mean compared to the norm.

For example, if the query time for an event is `0.01` on the first host and `0.5` on the second host, that is a significant difference because `0.5` is worse than `0.1`, and so the query will be reported.

READ-ONLY

By default, **pt-upgrade** only executes `SELECT` and `SET` statements. (This does not include 'SELECT...INTO' statements, which do not return rows but dump output to a file or variable.) If you're using recreatable test or development servers and wish to compare write statements too (e.g. `INSERT`, `UPDATE`, `DELETE`), then specify `--no-read-only`. If using a binary log, you must specify `--no-read-only` because binary logs don't contain `SELECT` statements. See `--[no]read-only`.

TRANSACTIONS

The tool does not create its own transactions, but any transactions in the `LOG` are executed as-is. Since logs are serial, transactions shouldn't normally be an issue. If, however, you need to compare queries that are somehow transactionally related (in which case you probably also need to disable `--[no]read-only`), then **pt-upgrade** probably won't do what you need because it's not designed for this purpose.

pt-upgrade runs with `autocommit=1` by default.

THROTTLING

pt-upgrade has no throttling options because the tool should only be ran on dedicated testing or development servers. **Do not run :program:'pt-upgrade' on production servers!** Consequently, the tool is CPU, memory, disk, and network intensive. It executes queries as fast as possible.

QUERY DIFFERENCES

Signficant query differences are determined by comparing these aspects of each query from both hosts:

Row count

> The number of rows returned by the query should be the same. This is reported as "missing rows" under "Row diffs".

Row data

> The row data returned by the query should be the same. All differences are significant: whitespace, float-precision, etc.

Warnings

> The query should either not produce any errors or warnings, or produce the same errors or warnings.

Query time

> A query rarely executes with a constant time, but its execution time should be within the same order of magnitude or smaller.

Query errors

> If a query causes a SQL error on only one host, this is reported as "Query errors". Since the query works on one host, its syntax is probably valid, and the error is due to some condition unique to the other host.

SQL errors

> If a query causes a SQL error on both hosts, this is reported as "SQL errors". The SQL syntax of the query could be invalid.

REPORT

As **pt-upgrade** runs, it prints queries with differences as soon as it can (see "WHEN QUERIES ARE REPORTED"). To prevent the report from becoming too long, queries are not reported individually but grouped by fingerprint into classes. A query fingerprint is the abstracted form of a query, created by removing literal values, normalizing whitespace, etc. So these queries belong to the same class:

```
SELECT c FROM t WHERE id = 1
SELECT c FROM t WHERE id=5
select  c  from  t  where  id  =  9
```

The fingerprint for those queries is:

```
select c from t where id=?
```

Each query class can have up to *--max-class-size* unique queries (1,000 by default). Up to *--max-examples* are reported for each type of difference, per query class. By virtue of being in the same class, an example of one query's difference is usually representative of all queries with the same difference, so it's not necessary to report every example. The total number of queries in a class with a particular difference is indicated in the report.

EXAMPLE

```
#############################################################################
# Logs
#############################################################################

File: /opt/mysql/slow.log
Size: 59700

#############################################################################
# Hosts
#############################################################################

host1:

  DSN:       h=127.1,P=12345
  hostname:  dev1
  MySQL:     MySQL 5.1.68

host2:

  DSN:       h=127.1,P=12348
  hostname:  dev2
  MySQL:     MySQL 5.5.10

############################################################################
# Query class AAD020567F8398EE
############################################################################

Reporting class because it has diffs, but hasn't been reported yet.

Total queries      1
Unique queries     1
Discarded queries  0
```

```
insert into t (id, username) values(?+)

##
## Warning diffs: 1
##

-- 1.

   Code: 1265
  Level: Warning
Message: Data truncated for column 'username' at row 1

vs.

No warning 1265

INSERT INTO t (id, username) VALUES (NULL, 'long_username')

#-----------------------------------------------------------------------
# Stats
#-----------------------------------------------------------------------

failed_queries        0
not_select            0
queries_filtered      0
queries_no_diffs      0
queries_read          1
queries_with_diffs    1
queries_with_errors   0
```

The "Query class <ID>" sections are the most important because they list "QUERY DIFFERENCES". The first part of the section lists the reason why the query class was report, followed by counts of queries in the class, followed by the fingerprint which defines the class.

The rest of the query class section lists the "QUERY DIFFERENCES" that caused the class to be reported. Each type of difference begins with a double hash mark header that lists the type and total number of queries in the class with the difference. Then up to *--max-examples* are listed, numbered "– 1.", "— 2.", etc. Each example lists the difference for the first and second hosts (respective to the "Hosts" section), followed by the first SQL statement that revealed the difference.

WHEN QUERIES ARE REPORTED

A query class is reported as soon as any one of the "QUERY DIFFERENCES" or query errors has *--max-examples*. Else, all queries with differences are reported when the tool finishes.

For example, if two query time differences are found for a query class, it is not reported yet. Once a third query time diffence is found, the query class is reported, including any other differences that may have been found too. Queries for the class will continue to be executed, but the class will not be reported again.

OUTPUT

The "REPORT" is printed to STDOUT as the tool runs. Internal warnings, errors, and *--progress* are printed to STDERR. To keep the two separate, run the tool like:

```
pt-upgrade ... 1>report 2>err &
```

Then `tail -f err` while the tool is running to track its *--progress*.

EXIT STATUS

In general, the tool exits zero if it finishes normally and there were no internal warnings or errors, and no "QUERY DIFFERENCES" were found. Else the tool exits non-zero with one or more of the following codes:

- 1

  There were too many internal errors or warnings; see STDERR. See also `--[no]continue-on-error`.

- 4

  There were "QUERY DIFFERENCES"; see the "REPORT".

- 8

  --run-time expired; the tool did not finish reading the logs or reference results.

Other exit codes indicate that the tool crashed or died unexpectedly. The error that caused this should have printed to STDERR.

To check for a particular exit code, logical `AND` (`&`) the final exit status with the exit code. For example, exit status 5 implies codes 1 and 4 because `5 & 1` is true, and `5 & 4` is true.

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 MySQL.

--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 MySQL. Any other value sets binmode on STDOUT without the utf8 layer, and runs SET NAMES after connecting to MySQL.

--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 after 100 errors, in which case there is probably a bug in the tool or the input is invalid.

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

Create the *--upgrade-table* database and table.

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

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

Default database when connecting to MySQL.

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

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

--[no]disable-query-cache
default: yes

`SET SESSION query_cache_type = OFF` to disable the query cache.

--dry-run
Run but do not execute or compare queries. This is useful for checking command line options, connections to MySQL, and log or reference results parsing.

--filter
type: string

Allow events for which this Perl code returns true.

See the same option in the documentation for pt-query-digest.

--help
Show help and exit.

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

MySQL hostname or IP.

--ignore-warnings
type: Hash

Ignore these MySQL warning codes when comparing warnings.

--log
type: string

Print STDOUT and STDERR to this file when daemonized. This option only takes affect when *--daemonize* is specified. The file is created if it doesn't exist, else output is appended to it.

--max-class-size
type: int; default: 1000

Max number of unique queries in each query class. See "REPORT".

--max-examples
type: int; default: 3

Max number of examples to list for each "QUERY DIFFERENCES". A query class is reported as soon as this many examples for any type of query difference are found.

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

MySQL password for the *--user*.

--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

MySQL port number.

--progress
type: array; default: time,30

Print progress reports to STDERR. The tool prints progress reports while reading logs or reference results, roughly estimating how long until it finishes.

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.

--`[no]read-only`
default: yes

Execute only `SELECT` and `SET` statements. If `--no-read-only` is specified, *all* queries are exeucted: `DROP`, `DELETE`, `UPDATE`, etc. Even when running in default read-only mode, you should use a MySQL user with only `SELECT` privileges to insure against bugs in the tool.

--report
type: Hash; default: hosts, logs, queries, stats

Print these sections of the "REPORT".

--run-time
type: time

How long to run before exiting. By default, the tool runs until it finishes reading the logs or reference results.

--save-results
type: string

Save reference results to this directory. This option works only when one DSN is specified, to generate reference results. When comparing a host to reference results, specify its results directory instead of its DSN. See the second example in the "SYNOPSIS".

Reference results can use *a lot* of disk space.

--set-vars
type: Array

Set the MySQL 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.

--type
type: string; default: slowlog

Type of log files. Valid types are:

```
VALUE    LOG TYPE
=======  ========================================
slowlog  MySQL slow log
genlog   MySQL general log
binlog   MySQL binary log (converted by mysqlbinlog)
rawlog   Custom log with one SQL statement per line
```

--upgrade-table
type: string; default: percona_schema.pt_upgrade

Use this table to clear warnings. To clear all warnings from previous queries, **pt-upgrade** executes `SELECT * FROM --upgrade-table LIMIT 1` on each host before executing each query.

The table must be database-qualified. The database and table are automatically created unless `--no-create-upgrade-table` is specified (see `--[no]create-upgrade-table`). If the table does not already exist, it is created with this definition:

```
CREATE TABLE pt_upgrade (
  id INT NOT NULL PRIMARY KEY
)
```

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

MySQL user if not the current system user.

--version
Show version and exit.

--[no]version-check
default: yes

Check for the latest version of Percona Toolkit, MySQL, and other programs.

This is a standard "check for updates automatically" feature, with two additional features. First, the tool checks the version of other programs on the local system in addition to its own version. For example, it checks the version of every MySQL server it connects to, Perl, and the Perl module DBD::mysql. Second, it checks for and warns about versions with known problems. For example, MySQL 5.5.25 had a critical bug and was re-released as 5.5.25a.

Any updates or known problems are printed to STDOUT before the tool's normal output. This feature should never interfere with the normal operation of the tool.

For more information, visit https://www.percona.com/version-check.

--watch-server
type: string

Parse only events for this IP:port for *--type* tcpdump. All other IP addresses are ignored. If not specified, **pt-upgrade** watches all servers by looking for any IP address using port 3306 or "mysql". 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 **pt-upgrade** cannot auto-detect servers on port 3306 and also be told to watch a server on a non-standard port.

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.

- 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://dev.mysql.com/doc/refman/5.0/en/load-data-local.html 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.

- 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.

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 pt-upgrade ... > FILE 2>&1
```

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

SYSTEM REQUIREMENTS

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

BUGS

For a list of known bugs, see http://www.percona.com/bugs/pt-upgrade.

Please report bugs at https://bugs.launchpad.net/percona-toolkit. Include the following information in your bug report:

- Complete command-line used to run the tool
- Tool *--version*
- MySQL version of all servers involved
- Output from the tool including STDERR
- Input files (log/dump/config files, etc.)

If possible, include debugging output by running the tool with `PTDEBUG`; see "ENVIRONMENT".

DOWNLOADING

Visit http://www.percona.com/software/percona-toolkit/ to download the latest release of Percona Toolkit. Or, get the latest release from the command line:

```
wget percona.com/get/percona-toolkit.tar.gz

wget percona.com/get/percona-toolkit.rpm

wget percona.com/get/percona-toolkit.deb
```

You can also get individual tools from the latest release:

```
wget percona.com/get/TOOL
```

Replace `TOOL` with the name of any tool.

AUTHORS

Daniel Nichter

ABOUT PERCONA TOOLKIT

This tool is part of Percona Toolkit, a collection of advanced command-line tools for MySQL developed by Percona. 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. Visit http://www.percona.com/software/ to learn about other free, open-source software from Percona.

COPYRIGHT, LICENSE, AND WARRANTY

VERSION

pt-upgrade 3.0.4

CHAPTER

THIRTYFOUR

PT-VARIABLE-ADVISOR

NAME

pt-variable-advisor - Analyze MySQL variables and advise on possible problems.

SYNOPSIS

Usage

```
pt-variable-advisor [OPTIONS] [DSN]
```

pt-variable-advisor analyzes variables and advises on possible problems.

Get SHOW VARIABLES from localhost:

```
pt-variable-advisor localhost
```

Get SHOW VARIABLES output saved in vars.txt:

```
pt-variable-advisor --source-of-variables vars.txt
```

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

DESCRIPTION

pt-variable-advisor examines `SHOW VARIABLES` for bad values and settings according to the "RULES" described below. It reports on variables that match the rules, so you can find bad settings in your MySQL server.

At the time of this release, **pt-variable-advisor** only examples `SHOW VARIABLES`, but other input sources are planned like `SHOW STATUS` and `SHOW SLAVE STATUS`.

RULES

These are the rules that **pt-variable-advisor** will apply to SHOW VARIABLES. Each rule has three parts: an ID, a severity, and a description.

The rule's ID is a short, unique name for the rule. It usually relates to the variable that the rule examines. If a variable is examined by several rules, then the rules' IDs are numbered like "-1", "-2", "-N".

The rule's severity is an indication of how important it is that this rule matched a query. We use NOTE, WARN, and CRIT to denote these levels.

The rule's description is a textual, human-readable explanation of what it means when a variable matches this rule. Depending on the verbosity of the report you generate, you will see more of the text in the description. By default, you'll see only the first sentence, which is sort of a terse synopsis of the rule's meaning. At a higher verbosity, you'll see subsequent sentences.

auto_increment

severity: note

Are you trying to write to more than one server in a dual-master or ring replication configuration? This is potentially very dangerous and in most cases is a serious mistake. Most people's reasons for doing this are actually not valid at all.

concurrent_insert

severity: note

Holes (spaces left by deletes) in MyISAM tables might never be reused.

connect_timeout

severity: note

A large value of this setting can create a denial of service vulnerability.

debug

severity: crit

Servers built with debugging capability should not be used in production because of the large performance impact.

delay_key_write

severity: warn

MyISAM index blocks are never flushed until necessary. If there is a server crash, data corruption on MyISAM tables can be much worse than usual.

flush

severity: warn

This option might decrease performance greatly.

flush_time

severity: warn

This option might decrease performance greatly.

have_bdb

severity: note

The BDB engine is deprecated. If you aren't using it, you should disable it with the skip_bdb option.

init_connect

severity: note

The init_connect option is enabled on this server.

init_file

severity: note

The init_file option is enabled on this server.

init_slave

severity: note

The init_slave option is enabled on this server.

innodb_additional_mem_pool_size

severity: warn

This variable generally doesn't need to be larger than 20MB.

innodb_buffer_pool_size

severity: warn

The InnoDB buffer pool size is unconfigured. In a production environment it should always be configured explicitly, and the default 10MB size is not good.

innodb_checksums

severity: warn

InnoDB checksums are disabled. Your data is not protected from hardware corruption or other errors!

innodb_doublewrite

severity: warn

InnoDB doublewrite is disabled. Unless you use a filesystem that protects against partial page writes, your data is not safe!

innodb_fast_shutdown

severity: warn

InnoDB's shutdown behavior is not the default. This can lead to poor performance, or the need to perform crash recovery upon startup.

innodb_flush_log_at_trx_commit-1

severity: warn

InnoDB is not configured in strictly ACID mode. If there is a crash, some transactions can be lost.

innodb_flush_log_at_trx_commit-2

severity: warn

Setting innodb_flush_log_at_trx_commit to 0 has no performance benefits over setting it to 2, and more types of data loss are possible. If you are trying to change it from 1 for performance reasons, you should set it to 2 instead of 0.

innodb_force_recovery

severity: warn

InnoDB is in forced recovery mode! This should be used only temporarily when recovering from data corruption or other bugs, not for normal usage.

innodb_lock_wait_timeout

severity: warn

This option has an unusually long value, which can cause system overload if locks are not being released.

innodb_log_buffer_size

severity: warn

The InnoDB log buffer size generally should not be set larger than 16MB. If you are doing large BLOB operations, InnoDB is not really a good choice of engines anyway.

innodb_log_file_size

severity: warn

The InnoDB log file size is set to its default value, which is not usable on production systems.

innodb_max_dirty_pages_pct

severity: note

The innodb_max_dirty_pages_pct is lower than the default. This can cause overly aggressive flushing and add load to the I/O system.

flush_time

severity: warn

This setting is likely to cause very bad performance every flush_time seconds.

key_buffer_size

severity: warn

The key buffer size is set to its default value, which is not good for most production systems. In a production environment, key_buffer_size should be larger than the default 8MB size.

large_pages

severity: note

Large pages are enabled.

locked_in_memory

severity: note

The server is locked in memory with –memlock.

log_warnings-1

severity: note

Log_warnings is disabled, so unusual events such as statements unsafe for replication and aborted connections will not be logged to the error log.

log_warnings-2

severity: note

Log_warnings must be set greater than 1 to log unusual events such as aborted connections.

low_priority_updates

severity: note

The server is running with non-default lock priority for updates. This could cause update queries to wait unexpectedly for read queries.

max_binlog_size

severity: note

The max_binlog_size is smaller than the default of 1GB.

max_connect_errors

severity: note

max_connect_errors should probably be set as large as your platform allows.

max_connections

severity: warn

If the server ever really has more than a thousand threads running, then the system is likely to spend more time scheduling threads than really doing useful work. This variable's value should be considered in light of your workload.

myisam_repair_threads

severity: note

myisam_repair_threads > 1 enables multi-threaded repair, which is relatively untested and is still listed as beta-quality code in the official documentation.

old_passwords

severity: warn

Old-style passwords are insecure. They are sent in plain text across the wire.

optimizer_prune_level

severity: warn

The optimizer will use an exhaustive search when planning complex queries, which can cause the planning process to take a long time.

port

severity: note

The server is listening on a non-default port.

query_cache_size-1

severity: note

The query cache does not scale to large sizes and can cause unstable performance when larger than 128MB, especially on multi-core machines.

query_cache_size-2

severity: warn

The query cache can cause severe performance problems when it is larger than 256MB, especially on multi-core machines.

read_buffer_size-1

severity: note

The read_buffer_size variable should generally be left at its default unless an expert determines it is necessary to change it.

read_buffer_size-2

severity: warn

The read_buffer_size variable should not be larger than 8MB. It should generally be left at its default unless an expert determines it is necessary to change it. Making it larger than 2MB can hurt performance significantly, and can make the server crash, swap to death, or just become extremely unstable.

read_rnd_buffer_size-1

severity: note

The read_rnd_buffer_size variable should generally be left at its default unless an expert determines it is necessary to change it.

read_rnd_buffer_size-2

severity: warn

The read_rnd_buffer_size variable should not be larger than 4M. It should generally be left at its default unless an expert determines it is necessary to change it.

relay_log_space_limit

severity: warn

Setting relay_log_space_limit can cause replicas to stop fetching binary logs from their master immediately. This could increase the risk that your data will be lost if the master crashes. If the replicas have encountered a limit on relay log space, then it is possible that the latest transactions exist only on the master and no replica has retrieved them.

slave_net_timeout

severity: warn

This variable is set too high. This is too long to wait before noticing that the connection to the master has failed and retrying. This should probably be set to 60 seconds or less. It is also a good idea to use pt-heartbeat to ensure that the connection does not appear to time out when the master is simply idle.

slave_skip_errors

severity: crit

You should not set this option. If replication is having errors, you need to find and resolve the cause of that; it is likely that your slave's data is different from the master. You can find out with pt-table-checksum.

sort_buffer_size-1

severity: note

The sort_buffer_size variable should generally be left at its default unless an expert determines it is necessary to change it.

sort_buffer_size-2

severity: note

The sort_buffer_size variable should generally be left at its default unless an expert determines it is necessary to change it. Making it larger than a few MB can hurt performance significantly, and can make the server crash, swap to death, or just become extremely unstable.

sql_notes

severity: note

This server is configured not to log Note level warnings to the error log.

sync_frm

severity: warn

It is best to set sync_frm so that .frm files are flushed safely to disk in case of a server crash.

tx_isolation-1

severity: note

This server's transaction isolation level is non-default.

tx_isolation-2

severity: warn

Most applications should use the default REPEATABLE-READ transaction isolation level, or in a few cases READ-COMMITTED.

expire_logs_days

severity: warn

Binary logs are enabled, but automatic purging is not enabled. If you do not purge binary logs, your disk will fill up. If you delete binary logs externally to MySQL, you will cause unwanted behaviors. Always ask MySQL to purge obsolete logs, never delete them externally.

innodb_file_io_threads

severity: note

This option is useless except on Windows.

innodb_data_file_path

severity: note

Auto-extending InnoDB files can consume a lot of disk space that is very difficult to reclaim later. Some people prefer to set innodb_file_per_table and allocate a fixed-size file for ibdata1.

innodb_flush_method

severity: note

Most production database servers that use InnoDB should set innodb_flush_method to O_DIRECT to avoid double-buffering, unless the I/O system is very low performance.

innodb_locks_unsafe_for_binlog

severity: warn

This option makes point-in-time recovery from binary logs, and replication, untrustworthy if statement-based logging is used.

innodb_support_xa

severity: warn

MySQL's internal XA transaction support between InnoDB and the binary log is disabled. The binary log might not match InnoDB's state after crash recovery, and replication might drift out of sync due to out-of-order statements in the binary log.

log_bin

severity: warn

Binary logging is disabled, so point-in-time recovery and replication are not possible.

log_output

severity: warn

Directing log output to tables has a high performance impact.

max_relay_log_size

severity: note

A custom max_relay_log_size is defined.

myisam_recover_options

severity: warn

myisam_recover_options should be set to some value such as BACKUP,FORCE to ensure that table corruption is noticed.

storage_engine

severity: note

The server is using a non-standard storage engine as default.

sync_binlog

severity: warn

Binary logging is enabled, but sync_binlog isn't configured so that every transaction is flushed to the binary log for durability.

tmp_table_size

severity: note

The effective minimum size of in-memory implicit temporary tables used internally during query execution is min(tmp_table_size, max_heap_table_size), so max_heap_table_size should be at least as large as tmp_table_size.

old mysql version

severity: warn

These are the recommended minimum version for each major release: 3.23, 4.1.20, 5.0.37, 5.1.30.

end-of-life mysql version

severity: note

Every release older than 5.1 is now officially end-of-life.

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 MySQL.

--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 MySQL. Any other value sets binmode on STDOUT without the utf8 layer, and runs SET NAMES after connecting to MySQL.

--config
type: Array

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

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

--help
Show help and exit.

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

Connect to host.

--ignore-rules
type: hash

Ignore these rule IDs.

Specify a comma-separated list of rule IDs (e.g. LIT.001,RES.002,etc.) to ignore.

--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 MySQL 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.

--source-of-variables
type: string; default: mysql

Read `SHOW VARIABLES` from this source. Possible values are "mysql", "none" or a file name. If "mysql" is specified then you must also specify a DSN on the command line.

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

User for login if not current user.

--verbose
short form: -v; cumulative: yes; default: 1

Increase verbosity of output. At the default level of verbosity, the program prints only the first sentence of each rule's description. At higher levels, the program prints more of the description.

--version
Show version and exit.

--`[no]version-check`
default: yes

Check for the latest version of Percona Toolkit, MySQL, and other programs.

This is a standard "check for updates automatically" feature, with two additional features. First, the tool checks the version of other programs on the local system in addition to its own version. For example, it checks the version of every MySQL server it connects to, Perl, and the Perl module DBD::mysql. Second, it checks for and warns about versions with known problems. For example, MySQL 5.5.25 had a critical bug and was re-released as 5.5.25a.

Any updates or known problems are printed to STDOUT before the tool's normal output. This feature should never interfere with the normal operation of the tool.

For more information, visit https://www.percona.com/version-check.

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: mysql_socket; copy: yes

  Socket file to use for connection.

- u

  dsn: user; copy: yes

  User for login if not current user.

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 pt-variable-advisor ... > FILE 2>&1
```

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

SYSTEM REQUIREMENTS

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

BUGS

For a list of known bugs, see http://www.percona.com/bugs/pt-variable-advisor.

Please report bugs at https://bugs.launchpad.net/percona-toolkit. Include the following information in your bug report:

- Complete command-line used to run the tool
- Tool *--version*
- MySQL version of all servers involved
- Output from the tool including STDERR
- Input files (log/dump/config files, etc.)

If possible, include debugging output by running the tool with `PTDEBUG`; see "ENVIRONMENT".

DOWNLOADING

Visit http://www.percona.com/software/percona-toolkit/ to download the latest release of Percona Toolkit. Or, get the latest release from the command line:

```
wget percona.com/get/percona-toolkit.tar.gz

wget percona.com/get/percona-toolkit.rpm

wget percona.com/get/percona-toolkit.deb
```

You can also get individual tools from the latest release:

```
wget percona.com/get/TOOL
```

Replace `TOOL` with the name of any tool.

AUTHORS

Baron Schwartz and Daniel Nichter

ABOUT PERCONA TOOLKIT

This tool is part of Percona Toolkit, a collection of advanced command-line tools for MySQL developed by Percona. 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. Visit http://www.percona.com/software/ to learn about other free, open-source software from Percona.

COPYRIGHT, LICENSE, AND WARRANTY

VERSION

pt-variable-advisor 3.0.4

CHAPTER

THIRTYFIVE

PT-VISUAL-EXPLAIN

NAME

pt-visual-explain - Format EXPLAIN output as a tree.

SYNOPSIS

Usage

```
pt-visual-explain [OPTIONS] [FILES]
```

pt-visual-explain transforms EXPLAIN output into a tree representation of the query plan. If FILE is given, input is read from the file(s). With no FILE, or when FILE is -, read standard input.

Examples

```
pt-visual-explain <file_containing_explain_output>

pt-visual-explain -c <file_containing_query>

mysql -e "explain select * from mysql.user" | pt-visual-explain
```

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

DESCRIPTION

pt-visual-explain reverse-engineers MySQL's EXPLAIN output into a query execution plan, which it then formats as a left-deep tree – the same way the plan is represented inside MySQL. It is possible to do this by hand, or to read EXPLAIN's output directly, but it requires patience and expertise. Many people find a tree representation more understandable.

You can pipe input into **pt-visual-explain** or specify a filename at the command line, including the magical '-' filename, which will read from standard input. It can do two things with the input: parse it for something that looks like EXPLAIN output, or connect to a MySQL instance and run EXPLAIN on the input.

When parsing its input, **pt-visual-explain** understands three formats: tabular like that shown in the mysql command-line client, vertical like that created by using the G line terminator in the mysql command-line client, and tab separated. It ignores any lines it doesn't know how to parse.

When executing the input, **pt-visual-explain** replaces everything in the input up to the first SELECT keyword with 'EXPLAIN SELECT,' and then executes the result. You must specify *--connect* to execute the input as a query.

Either way, it builds a tree from the result set and prints it to standard output. For the following query,

```
select * from sakila.film_actor join sakila.film using(film_id);
```

pt-visual-explain generates this query plan:

```
JOIN
+- Bookmark lookup
|  +- Table
|  |  table          film_actor
|  |  possible_keys  idx_fk_film_id
|  +- Index lookup
|     key            film_actor->idx_fk_film_id
|     possible_keys  idx_fk_film_id
|     key_len        2
|     ref            sakila.film.film_id
|     rows           2
+- Table scan
   rows           952
   +- Table
      table          film
      possible_keys  PRIMARY
```

The query plan is left-deep, depth-first search, and the tree's root is the output node – the last step in the execution plan. In other words, read it like this:

1

Table scan the 'film' table, which accesses an estimated 952 rows.

2

For each row, find matching rows by doing an index lookup into the film_actor->idx_fk_film_id index with the value from sakila.film.film_id, then a bookmark lookup into the film_actor table.

For more information on how to read EXPLAIN output, please see http://dev.mysql.com/doc/en/explain.html, and this talk titled "MySQL query optimizer internals and upcoming features in v. 5.2": from Timour Katchaounov, one of the MySQL developers: http://goo.gl/VIWvo

MODULES

This program is actually a runnable module, not just an ordinary Perl script. In fact, there are two modules embedded in it. This makes unit testing easy, but it also makes it easy for you to use the parsing and tree-building functionality if you want.

The ExplainParser package accepts a string and parses whatever it thinks looks like EXPLAIN output from it. The synopsis is as follows:

```
require "pt-visual-explain";
my $p    = ExplainParser->new();
my $rows = $p->parse("some text");
# $rows is an arrayref of hashrefs.
```

The ExplainTree package accepts a set of rows and turns it into a tree. For convenience, you can also have it delegate to ExplainParser and parse text for you. Here's the synopsis:

```
require "pt-visual-explain";
my $e      = ExplainTree->new();
my $tree   = $e->parse("some text", \%options);
my $output = $e->pretty_print($tree);
print $tree;
```

ALGORITHM

This section explains the algorithm that converts EXPLAIN into a tree. You may be interested in reading this if you want to understand EXPLAIN more fully, or trying to figure out how this works, but otherwise this section will probably not make your life richer.

The tree can be built by examining the id, select_type, and table columns of each row. Here's what I know about them:

The id column is the sequential number of the select. This does not indicate nesting; it just comes from counting SELECT from the left of the SQL statement. It's like capturing parentheses in a regular expression. A UNION RESULT row doesn't have an id, because it isn't a SELECT. The source code actually refers to UNIONs as a fake_lex, as I recall.

If two adjacent rows have the same id value, they are joined with the standard single-sweep multi-join method.

The select_type column tells a) that a new sub-scope has opened b) what kind of relationship the row has to the previous row c) what kind of operation the row represents.

- SIMPLE means there are no subqueries or unions in the whole query.
- PRIMARY means there are, but this is the outermost SELECT.
- [DEPENDENT] UNION means this result is UNIONed with the previous result (not row; a result might encompass more than one row).
- UNION RESULT terminates a set of UNIONed results.
- [DEPENDENT|UNCACHEABLE] SUBQUERY means a new sub-scope is opening. This is the kind of subquery that happens in a WHERE clause, SELECT list or whatnot; it does not return a so-called "derived table."
- DERIVED is a subquery in the FROM clause.

Tables that are JOINed all have the same select_type. For example, if you JOIN three tables inside a dependent subquery, they'll all say the same thing: DEPENDENT SUBQUERY.

The table column usually specifies the table name or alias, but may also say <derivedN> or <unionN,N...N>. If it says <derivedN>, the row represents an access to the temporary table that holds the result of the subquery whose id is N. If it says <unionN,..N> it's the same thing, but it refers to the results it UNIONs together.

Finally, order matters. If a row's id is less than the one before it, I think that means it is dependent on something other than the one before it. For example,

```
explain select
   (select 1 from sakila.film),
   (select 2 from sakila.film_actor),
   (select 3 from sakila.actor);

| id | select_type | table      |
+----+-------------+------------+
|  1 | PRIMARY     | NULL       |
|  4 | SUBQUERY    | actor      |
|  3 | SUBQUERY    | film_actor |
|  2 | SUBQUERY    | film       |
```

If the results were in order 2-3-4, I think that would mean 3 is a subquery of 2, 4 is a subquery of 3. As it is, this means 4 is a subquery of the nearest previous recent row with a smaller id, which is 1. Likewise for 3 and 2.

This structure is hard to programmatically build into a tree for the same reason it's hard to understand by inspection: there are both forward and backward references. <derivedN> is a forward reference to selectN, while <unionM,N> is a backward reference to selectM and selectN. That makes recursion and other tree-building algorithms hard to get right (NOTE: after implementation, I now see how it would be possible to deal with both forward and backward references, but I have no motivation to change something that works). Consider the following:

```
select * from (
   select 1 from sakila.actor as actor_1
   union
   select 1 from sakila.actor as actor_2
) as der_1
union
select * from (
   select 1 from sakila.actor as actor_3
   union all
   select 1 from sakila.actor as actor_4
) as der_2;

| id   | select_type  | table      |
+------+--------------+------------+
|  1   | PRIMARY      | <derived2> |
|  2   | DERIVED      | actor_1    |
|  3   | UNION        | actor_2    |
| NULL | UNION RESULT | <union2,3> |
|  4   | UNION        | <derived5> |
|  5   | DERIVED      | actor_3    |
|  6   | UNION        | actor_4    |
| NULL | UNION RESULT | <union5,6> |
| NULL | UNION RESULT | <union1,4> |
```

This would be a lot easier to work with if it looked like this (I've bracketed the id on rows I moved):

```
| id   | select_type  | table      |
+------+--------------+------------+
| [1]  | UNION RESULT | <union1,4> |
|  1   | PRIMARY      | <derived2> |
| [2]  | UNION RESULT | <union2,3> |
```

```
|  2   | DERIVED      | actor_1    |
|  3   | UNION        | actor_2    |
|  4   | UNION        | <derived5> |
| [5]  | UNION RESULT | <union5,6> |
|  5   | DERIVED      | actor_3    |
|  6   | UNION        | actor_4    |
```

In fact, why not re-number all the ids, so the PRIMARY row becomes 2, and so on? That would make it even easier to read. Unfortunately that would also have the effect of destroying the meaning of the id column, which I think is important to preserve in the final tree. Also, though it makes it easier to read, it doesn't make it easier to manipulate programmatically; so it's fine to leave them numbered as they are.

The goal of re-ordering is to make it easier to figure out which rows are children of which rows in the execution plan. Given the reordered list and some row whose table is <union...> or <derived>, it is easy to find the beginning of the slice of rows that should be child nodes in the tree: you just look for the first row whose ID is the same as the first number in the table.

The next question is how to find the last row that should be a child node of a UNION or DERIVED. I'll start with DERIVED, because the solution makes UNION easy.

Consider how MySQL numbers the SELECTs sequentially according to their position in the SQL, left-to-right. Since a DERIVED table encloses everything within it in a scope, which becomes a temporary table, there are only two things to think about: its child subqueries and unions (if any), and its next siblings in the scope that encloses it. Its children will all have an id greater than it does, by definition, so any later rows with a smaller id terminate the scope.

Here's an example. The middle derived table here has a subquery and a UNION to make it a little more complex for the example.

```
explain select 1
from (
   select film_id from sakila.film limit 1
) as der_1
join (
   select film_id, actor_id, (select count(*) from sakila.rental) as r
   from sakila.film_actor limit 1
   union all
   select 1, 1, 1 from sakila.film_actor as dummy
) as der_2 using (film_id)
join (
   select actor_id from sakila.actor limit 1
) as der_3 using (actor_id);
```

Here's the output of EXPLAIN:

```
| id   | select_type  | table      |
|  1   | PRIMARY      | <derived2> |
|  1   | PRIMARY      | <derived6> |
|  1   | PRIMARY      | <derived3> |
|  6   | DERIVED      | actor      |
|  3   | DERIVED      | film_actor |
|  4   | SUBQUERY     | rental     |
|  5   | UNION        | dummy      |
| NULL | UNION RESULT | <union3,5> |
|  2   | DERIVED      | film       |
```

The siblings all have id 1, and the middle one I care about is derived3. (Notice MySQL doesn't execute them in the order I defined them, which is fine). Now notice that MySQL prints out the rows in the opposite order I defined the subqueries: 6, 3, 2. It always seems to do this, and there might be other methods of finding the scope boundaries

including looking for the lower boundary of the next largest sibling, but this is a good enough heuristic. I am forced to rely on it for non-DERIVED subqueries, so I rely on it here too. Therefore, I decide that everything greater than or equal to 3 belongs to the DERIVED scope.

The rule for UNION is simple: they consume the entire enclosing scope, and to find the component parts of each one, you find each part's beginning as referred to in the <unionN,...> definition, and its end is either just before the next one, or if it's the last part, the end is the end of the scope.

This is only simple because UNION consumes the entire scope, which is either the entire statement, or the scope of a DERIVED table. This is because a UNION cannot be a sibling of another UNION or a table, DERIVED or not. (Try writing such a statement if you don't see it intuitively). Therefore, you can just find the enclosing scope's boundaries, and the rest is easy. Notice in the example above, the UNION is over <union3,5>, which includes the row with id 4 – it includes every row between 3 and 5.

Finally, there are non-derived subqueries to deal with as well. In this case I can't look at siblings to find the end of the scope as I did for DERIVED. I have to trust that MySQL executes depth-first. Here's an example:

```
explain
select actor_id,
(
   select count(film_id)
   + (select count(*) from sakila.film)
   from sakila.film join sakila.film_actor using(film_id)
   where exists(
      select * from sakila.actor
      where sakila.actor.actor_id = sakila.film_actor.actor_id
   )
)
from sakila.actor;

| id | select_type        | table      |
|  1 | PRIMARY            | actor      |
|  2 | SUBQUERY           | film       |
|  2 | SUBQUERY           | film_actor |
|  4 | DEPENDENT SUBQUERY | actor      |
|  3 | SUBQUERY           | film       |
```

In order, the tree should be built like this:

- See row 1.
- See row 2. It's a higher id than 1, so it's a subquery, along with every other row whose id is greater than 2.
- Inside this scope, see 2 and 2 and JOIN them. See 4. It's a higher id than 2, so it's again a subquery; recurse. After that, see 3, which is also higher; recurse.

But the only reason the nested subquery didn't include select 3 is because select 4 came first. In other words, if EXPLAIN looked like this,

```
| id | select_type        | table      |
|  1 | PRIMARY            | actor      |
|  2 | SUBQUERY           | film       |
|  2 | SUBQUERY           | film_actor |
|  3 | SUBQUERY           | film       |
|  4 | DEPENDENT SUBQUERY | actor      |
```

I would be forced to assume upon seeing select 3 that select 4 is a subquery of it, rather than just being the next sibling in the enclosing scope. If this is ever wrong, then the algorithm is wrong, and I don't see what could be done about it.

UNION is a little more complicated than just "the entire scope is a UNION," because the UNION might itself be inside

an enclosing scope that's only indicated by the first item inside the UNION. There are only three kinds of enclosing scopes: UNION, DERIVED, and SUBQUERY. A UNION can't enclose a UNION, and a DERIVED has its own "scope markers," but a SUBQUERY can wholly enclose a UNION, like this strange example on the empty table t1:

```
explain select * from t1 where not exists(
   (select t11.i from t1 t11) union (select t12.i from t1 t12));

|   id | select_type  | table      | Extra                          |
+------+--------------+------------+--------------------------------+
|    1 | PRIMARY      | t1         | const row not found            |
|    2 | SUBQUERY     | NULL       | No tables used                 |
|    3 | SUBQUERY     | NULL       | no matching row in const table |
|    4 | UNION        | t12        | const row not found            |
| NULL | UNION RESULT | <union2,4> |                                |
```

The UNION's backward references might make it look like the UNION encloses the subquery, but studying the query makes it clear this isn't the case. So when a UNION's first row says SUBQUERY, it is this special case.

By the way, I don't fully understand this query plan; there are 4 numbered SELECT in the plan, but only 3 in the query. The parens around the UNIONs are meaningful. Removing them will make the EXPLAIN different. Please tell me how and why this works if you know.

Armed with this knowledge, it's possible to use recursion to turn the parent-child relationship between all the rows into a tree representing the execution plan.

MySQL prints the rows in execution order, even the forward and backward references. At any given scope, the rows are processed as a left-deep tree. MySQL does not do "bushy" execution plans. It begins with a table, finds a matching row in the next table, and continues till the last table, when it emits a row. When it runs out, it backtracks till it can find the next row and repeats. There are subtleties of course, but this is the basic plan. This is why MySQL transforms all RIGHT OUTER JOINs into LEFT OUTER JOINs and cannot do FULL OUTER JOIN.

This means in any given scope, say

```
| id   | select_type  | table      |
|  1   | SIMPLE       | tbl1       |
|  1   | SIMPLE       | tbl2       |
|  1   | SIMPLE       | tbl3       |
```

The execution plan looks like a depth-first traversal of this tree:

```
      JOIN
     /    \
   JOIN  tbl3
  /    \
tbl1   tbl2
```

The JOIN might not be a JOIN. It might be a subquery, for example. This comes from the type column of EXPLAIN. The documentation says this is a "join type," but I think "access type" is more accurate, because it's "how MySQL accesses rows."

pt-visual-explain decorates the tree significantly more than just turning rows into nodes. Each node may get a series of transformations that turn it into a subtree of more than one node. For example, an index scan not marked with 'Using index' must do a bookmark lookup into the table rows; that is a three-node subtree. However, after the above node-ordering and scoping stuff, the rest of the process is pretty simple.

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 MySQL.

--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 MySQL. Any other value sets binmode on STDOUT without the utf8 layer, and runs SET NAMES after connecting to MySQL.

--clustered-pk
Assume that PRIMARY KEY index accesses don't need to do a bookmark lookup to retrieve rows. This is the case for InnoDB.

--config
type: Array

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

--connect
Treat input as a query, and obtain EXPLAIN output by connecting to a MySQL instance and running EXPLAIN on the query. When this option is given, **pt-visual-explain** uses the other connection-specific options such as *--user* to connect to the MySQL instance. If you have a .my.cnf file, it will read it, so you may not need to specify any connection-specific options.

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

Connect to this database.

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

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

--format
type: string; default: tree

Set output format.

The default is a terse pretty-printed tree. The valid values are:

```
Value  Meaning
=====  ================================================
tree   Pretty-printed terse tree.
dump   Data::Dumper output (see Data::Dumper for more).
```

--help
Show help and exit.

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

Connect to host.

--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 MySQL 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.

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

User for login if not current user.

--version
Show version and exit.

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: mysql_socket; copy: yes

  Socket file to use for connection.

- u

  dsn: user; copy: yes

  User for login if not current user.

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 pt-visual-explain ... > FILE 2>&1
```

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

SYSTEM REQUIREMENTS

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

BUGS

For a list of known bugs, see http://www.percona.com/bugs/pt-visual-explain.

Please report bugs at https://bugs.launchpad.net/percona-toolkit. Include the following information in your bug report:

- Complete command-line used to run the tool
- Tool *--version*

- MySQL version of all servers involved
- Output from the tool including STDERR
- Input files (log/dump/config files, etc.)

If possible, include debugging output by running the tool with `PTDEBUG`; see "ENVIRONMENT".

DOWNLOADING

Visit http://www.percona.com/software/percona-toolkit/ to download the latest release of Percona Toolkit. Or, get the latest release from the command line:

```
wget percona.com/get/percona-toolkit.tar.gz

wget percona.com/get/percona-toolkit.rpm

wget percona.com/get/percona-toolkit.deb
```

You can also get individual tools from the latest release:

```
wget percona.com/get/TOOL
```

Replace `TOOL` with the name of any tool.

AUTHORS

Baron Schwartz

ABOUT PERCONA TOOLKIT

This tool is part of Percona Toolkit, a collection of advanced command-line tools for MySQL developed by Percona. 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. Visit http://www.percona.com/software/ to learn about other free, open-source software from Percona.

COPYRIGHT, LICENSE, AND WARRANTY

VERSION

pt-visual-explain 3.0.4

Part III

Configuration

CHAPTER

THIRTYSIX

CONFIGURATION FILES

Percona Toolkit tools can read options from configuration files. The configuration file syntax is simple and direct, and bears some resemblances to the MySQL command-line client tools. The configuration files all follow the same conventions.

Internally, what actually happens is that the lines are read from the file and then added as command-line options and arguments to the tool, so just think of the configuration files as a way to write your command lines.

SYNTAX

The syntax of the configuration files is as follows:

*

Whitespace followed by a hash sign (#) signifies that the rest of the line is a comment. This is deleted. For example:

*

Whitespace is stripped from the beginning and end of all lines.

*

Empty lines are ignored.

*

Each line is permitted to be in either of the following formats:

```
option
option=value
```

Do not prefix the option with `--`. Do not quote the values, even if it has spaces; value are literal. Whitespace around the equals sign is deleted during processing.

*

Only long options are recognized.

*

A line containing only two hyphens signals the end of option parsing. Any further lines are interpreted as additional arguments (not options) to the program.

EXAMPLE

This config file for pt-stalk,

```
# Config for pt-stalk
variable=Threads_connected
cycles=2  # trigger if problem seen twice in a row
--
--user daniel
```

is equivalent to this command line:

```
pt-stalk --variable Threads_connected --cycles 2 -- --user daniel
```

Options after `--` are passed literally to mysql and mysqladmin.

READ ORDER

The tools read several configuration files in order:

1. The global Percona Toolkit configuration file, */etc/percona-toolkit/percona-toolkit.conf*. All tools read this file, so you should only add options to it that you want to apply to all tools.
2. The global tool-specific configuration file, */etc/percona-toolkit/TOOL.conf*, where `TOOL` is a tool name like `pt-query-digest`. This file is named after the specific tool you're using, so you can add options that apply only to that tool.
3. The user's own Percona Toolkit configuration file, *$HOME/.percona-toolkit.conf*. All tools read this file, so you should only add options to it that you want to apply to all tools.
4. The user's tool-specific configuration file, *$HOME/.TOOL.conf*, where `TOOL` is a tool name like `pt-query-digest`. This file is named after the specific tool you're using, so you can add options that apply only to that tool.

SPECIFYING

There is a special `--config` option, which lets you specify which configuration files Percona Toolkit should read. You specify a comma-separated list of files. However, its behavior is not like other command-line options. It must be given **first** on the command line, before any other options. If you try to specify it anywhere else, it will cause an error. Also, you cannot specify `--config=/path/to/file`; you must specify the option and the path to the file separated by whitespace *without an equal sign* between them, like:

```
--config /path/to/file
```

If you don't want any configuration files at all, specify `--config ''` to provide an empty list of files.

CHAPTER

THIRTYSEVEN

DSN (DATA SOURCE NAME) SPECIFICATIONS

Percona Toolkit tools use DSNs to specify how to create a DBD connection to a MySQL server. A DSN is a comma-separated string of `key=value` parts, like:

```
h=host1,P=3306,u=bob
```

The standard key parts are shown below, but some tools add additional key parts. See each tool's documentation for details.

Some tools do not use DSNs but still connect to MySQL using options like `--host`, `--user`, and `--password`. Such tools uses these options to create a DSN automatically, behind the scenes.

Other tools uses both DSNs and options like the ones above. The options provide defaults for all DSNs that do not specify the option's corresponding key part. For example, if DSN `h=host1` and option `--port=12345` are specified, then the tool automatically adds `P=12345` to DSN.

ESCAPING VALUES

DSNs are usually specified on the command line, so shell quoting and escaping must be taken into account. Special characters, like asterisk (`\*`), need to be quoted and/or escaped properly to be passed as literal characters in DSN values.

Since DSN parts are separated by commas, literal commas in DSN values must be escaped with a single backslash (`\`). And since a backslash is the escape character for most shells, two backslashes are required to pass a literal backslash. For example, if the username is literally `my,name`, it must be specified as `my\\,name` on most shells. This applies to DSNs and DSN-related options like `--user`.

KEY PARTS

Many of the tools add more parts to DSNs for special purposes, and sometimes override parts to make them do something slightly different. However, all the tools support at least the following:

A

Default character set for the connection (`SET NAMES`).

Enables character set settings in Perl and MySQL. 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 MySQL. Other values set binmode on STDOUT without the utf8 layer and run `SET NAMES` after connecting to MySQL.

Unfortunately, there is no way from within Perl itself to specify the client library's character set. `SET NAMES` only affects the server; if the client library's settings don't match, there could be problems. You

can use the defaults file to specify the client library's character set, however. See the description of the F part below.

D

Default database to use when connecting. Tools may `USE` a different databases while running.

F

Defaults file for the MySQL client library (the C client library used by DBD::mysql, *not Percona Toolkit itself*). All tools all read the `[client]` section within the defaults file. If you omit this, the standard defaults files will be read in the usual order. "Standard" varies from system to system, because the filenames to read are compiled into the client library. On Debian systems, for example, it's usually `/etc/mysql/my.cnf` then `~/.my.cnf`. If you place the following in `~/.my.cnf`, you won't have to specify your MySQL username and password on the command line:

```
[client]
user=your_user_name
pass=secret
```

Omitting the F part is usually the right thing to do. As long as you have configured your `~/.my.cnf` correctly, that will result in tools connecting automatically without needing a username or password.

You can also specify a default character set in the defaults file. Unlike the "A" part described above, this will actually instruct the client library (DBD::mysql) to change the character set it uses internally, which cannot be accomplished any other way.

h

MySQL hostname or IP address to connect to.

L

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://dev.mysql.com/doc/refman/5.0/en/load-data-local.html and http://search.cpan.org/~capttofu/DBD-mysql/lib/DBD/mysql.pm. This option does exactly that.

p

MySQL password to use when connecting.

P

Port number to use for the connection. Note that the usual special-case behaviors apply: if you specify `localhost` as your hostname on Unix systems, the connection actually uses a socket file, not a TCP/IP connection, and thus ignores the port.

S

MySQL socket file to use for the connection (on Unix systems).

u

MySQL username to use when connecting, if not current system user.

BAREWORD

Many of the tools will let you specify a DSN as a single word, without any `key=value` syntax. This is called a 'bareword'. How this is handled is tool-specific, but it is usually interpreted as the "h" part. The tool's `--help` output will tell you the behavior for that tool.

PROPAGATION

Many tools will let you propagate values from one DSN to the next, so you don't have to specify all the parts for each DSN. For example, if you want to specify a username and password for each DSN, you can connect to three hosts as follows:

```
h=host1,u=fred,p=wilma host2 host3
```

This is tool-specific.

CHAPTER

THIRTYEIGHT

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 pt-table-checksum ... > FILE 2>&1
```

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

CHAPTER

THIRTYNINE

SYSTEM REQUIREMENTS

Most tools require:

* Perl v5.8 or newer

* Bash v3 or newer

* Core Perl modules like Time::HiRes

Tools that connect to MySQL require:

* Perl modules DBI and DBD::mysql

* MySQL 5.0 or newer

Percona Toolkit officially supports and is tested on many popular Linux distributions and MySQL 5.0 through 5.6; see http://goo.gl/srHm7 for the list of supported platforms and versions.

Part IV

Miscellaneous

CHAPTER

FORTY

BUGS

Please report bugs at https://bugs.launchpad.net/percona-toolkit. Include the following information in your bug report:

* Complete command-line used to run the tool

* Tool `--version`

* MySQL version of all servers involved

* Output from the tool including STDERR

* Input files (log/dump/config files, etc.)

If possible, include debugging output by running the tool with `PTDEBUG`; see "ENVIRONMENT".

CHAPTER

FORTYONE

AUTHORS

Baron Schwartz

Baron created Maatkit, from which Percona Toolkit was forked. Many of the tools and modules were originally written by Baron.

Daniel Nichter

Daniel has been the project's lead developer since 2008 until 2016.

Frank Cizmich

Frank was a full-time Percona Toolkit developer employed by Percona until 2016.

Carlos Salguero

Carlos has been the project's lead developer since 2016. He is hired by Percona.

Others

Many people have contributed code over the years. See each tool's "AUTHORS" section for details.

CHAPTER

FORTYTWO

COPYRIGHT, LICENSE, AND WARRANTY

CHAPTER

FORTYTHREE

VERSION

Percona Toolkit v3.0.4 released 2017-08-02

CHAPTER

FORTYFOUR

RELEASE NOTES

v3.0.4 released 2017-08-02

Percona Toolkit 3.0.4 includes the following changes:

New Features

- PT-90: Added collection of information about prepared statements by `pt-stalk` when Performance Schema is enabled. For more information, see #1642750.
- PT-91: Added the `--preserve-triggers` option for `pt-online-schema-change` to support `AFTER` triggers.
- PT-138: Added `--output-format` option for `pt-mongodb-summary` to choose between JSON format and the default plain text.
- PT-141: Added the `--output-format=csv` parameter for `pt-archiver` to archive rows in CSV format.
- PT-142: Added the `--only-same-schema-fks` option for `pt-online-schema-change` to check foreigns keys only on tables with the same schema as the original table. This should speed up the tool's execution, but keep in mind that if you have foreign keys referencing tables in other schemas, they won't be detected. For more information, see #1690122.
- PT-153: Added the `--check-unique-key-change` option for `pt-online-schema-change` to abort if the specified statement for `--alter` is trying to add a unique index. This is supposed to avoid adding duplicate keys that might lead to silently losing data.
- PT-173: Added the `--truncate-replicate-table` option for `pt-table-checksum` to ensure stale data is removed.

Bug fixes

- PT-136: Fixed `pt-table-checksum` to support tables that have columns with different collations or charsets. For more information, see #1674266.
- PT-143: Fixed primary key handling by `pt-archiver`. For more information, see #1691630.
- PT-144: Limited constraint name in the new table when running `pt-online-schema-change`. For more information, see #1491674.
- PT-146: Fixed the `--no-check-binlog-format` option for `pt-table-checksum` to work as expected.
- PT-148: Fixed the use of uninitialized value in `printf()` for `pt-online-schema-change`. For more information, see #1693614.
- PT-151: Fixed `pt-table-sync` to prevent field type `point` to be taken as decimal.
- PT-154: Reverted PT-116 to remove the `--use-insert-ignore` option from `pt-online-schema-change`.

- PT-161: Fixed the `--skip-check-slave-lag` feature for `pt-table-checksum` to safely check for undefined values.
- PT-178: Fixed regression in `--check-slave-lag` option for `pt-online-schema-change`.
- PT-180: Fixed regression in `--skip-check-slave-lag` option for `pt-online-schema-change`.
- PT-181: Fixed syntax error in `pt-online-schema-change`.

Other Improvements

- PT-162: Updated list of tables ignored by `pt-table-checksum`.

v3.0.3 released 2017-05-18

Percona Toolkit 3.0.3 includes the following changes:

New Features

- Added the `--skip-check-slave-lag` option for `pt-table-checksum`, `pt-online-schema-change`, and `pt-archiver`.

  This option can be used to specify list of servers where to skip checking for slave lag.
- 1642754: Added support for collecting replication slave information in `pt-stalk`.
- PT-111: Added support for collecting information about variables from Performance Schema in `pt-stalk`. For more information, see 1642753.
- PT-116: Added the `--[no]use-insert-ignore` option for `pt-online-schema-change` to force or prevent using `IGNORE` on `INSERT` statements. For more information, see 1545129.

Bug Fixes

- PT-115: Fixed `OptionParser` to accept repeatable DSNs.
- PT-126: Fixed `pt-online-schema-change` to correctly parse comments. For more information, see 1592072.
- PT-128: Fixed `pt-stalk` to include memory usage information. For more information, see 1510809.
- PT-130: Fixed `pt-mext` to work with non-empty RSA public key. For more information, see 1587404.
- PT-132: Fixed `pt-online-schema-change` to enable `--no-drop-new-table` when `--no-swap-tables` and `--no-drop-triggers` are used.

v3.0.2 released 2017-03-27

Percona Toolkit 3.0.2 includes the following changes:

New Features

- PT-73: Added support for SSL connections to `pt-mongodb-summary` and `pt-mongodb-query-digest`
- 1642751: Enabled gathering of information about locks and transactions by `pt-stalk` using Performance Schema if it is enabled (Thanks Agustin Gallego)

Bug Fixes

- PT-74: Fixed gathering of security settings when running `pt-mongodb-summary` on a mongod instance that is specified as the host

- PT-75: Changed the default sort order in `pt-mongodb-query-digest` output to descending
- PT-76: Added support of `&` and `#` symbols in passwords for `pt-mysql-summary`
- PT-77: Updated `Makefile` to support new MongoDB tools
- PT-89: Fixed `pt-stalk` to run `top` more than once to collect useful CPU usage
- PT-93: Fixed `pt-mongodb-query-digest` to make query ID match query key (Thanks Kamil Dziedzic)
- PT-94: Fixed `pt-online-schema-change` to not make duplicate rows in `_t_new` when updating primary key. Also see 1646713.
- PT-101: Fixed `pt-table-checksum` to correctly use the `--slave-user` and `--slave-password` options. Also see 1651002.
- PT-105: Fixed `pt-table-checksum` to continue running if a database is dropped in the process

v3.0.1 released 2017-02-20

Percona Toolkit 3.0.1 GA includes the following changes:

- Added requirement to run `pt-mongodb-summary` as a user with the `clusterAdmin` or `root` built-in roles.

v3.0 released 2017-02-06

Percona Toolkit 3.0.0 RC includes the following changes:

New Features

- Added `pt-mongodb-summary` tool
- Added `pt-mongodb-query-digest` tool

Bug fixes

- 1402776: Updated `MySQLProtocolParser` to fix error when parsing `tcpdump` capture with `pt-query-digest`
- 1632522: Fixed failure of `pt-online-schema-change` when altering a table with a self-referencing foreign key (Thanks Amiel Marqeta)
- 1654668: Fixed failure of `pt-summary` on Red Hat and derivatives (Thanks Marcelo Altmann)

v2.2.20 released 2016-12-09

Percona Toolkit 2.2.20 includes the following changes:

New Features

- 1636068: New `--pause-file` option has been implemented for `pt-online-schema-change`. When used `pt-online-schema-change` will pause while the specified file exists.
- 1638293 and 1642364: `pt-online-schema-change` now supports adding and removing the `DATA DIRECTORY` to a new table with the `--data-dir` and `--remove-data-dir` options.

- 1642994: Following schemas/tables have been added to the default ignore list: `mysql.gtid_execution`, `sys.sys_config`, `mysql.proc`, `mysql.inventory`, `mysql.plugin`, `percona.*` (including checksums, dsns table), `test.*`, and `percona_schema.*`.
- 1643940: `pt-summary` now provides information about Transparent huge pages.
- 1604834: New `--preserve-embedded-numbers` option has been implemented for `pt-query-digest` which can be used to preserve numbers in database/table names when fingerprinting queries.

Bug Fixes

- 1613915: `pt-online-schema-change` could miss the data due to the way ENUM values are sorted.
- 1625005: `pt-online-schema-change` didn't apply underscores to foreign keys individually.
- 1566556: `pt-show-grants` didn't work correctly with *MariaDB* 10 (*Daniël van Eeden*).
- 1634900: `pt-upgrade` would fail when log contained `SELECT...INTO` queries.
- 1639052: `pt-table-checksum` now automatically excludes checking schemas named `percona` and `percona_schema` which aren't consistent across the replication hierarchy.
- 1635734: `pt-slave-restart --config` did not recognize = as a separator.
- 1362942: `pt-slave-restart` would fail on *MariaDB* 10.0.13.

Changelog

- Fixed bug 1362942: pt-slave-restart fails on MariaDB 10.0.13 (gtid_mode confusion)
- Fixed bug 1566556: pt-show-grants fails against MariaDB10+
- Feature 1604834: pt-query-digest numbers in table or column names converted to question marks (–preserve-embedded-numbers)
- Fixed bug 1613915: pt-online-schema-change misses data. Fixed sort order for ENUM fields
- Fixed bug 1625005: pt-online-schema-change doesn't apply underscores to foreign keys individually
- Fixed bug 1634900: pt-upgrade fails with SELECT INTO
- Fixed bug 1635734: pt-slave-restart –config does not recognize = as separator
- Feature 1636068: Added pause to NibbleIterator
- Feature 1638293: –data-dir parameter in order to create the table on a different partition
- Feature 1639052: with pt-table-checksum automatically exclude checking schemas named percona, percona_schema
- Feature 1642364: pt-online-schema-change Added –remove-data-dir feature
- Feature 1643914: Fixed several typos in the doc (Thanks Dario Minnucci)
- Feature 1643940: Add Transparent huge pages info to pt-summary
- Feature 1643941: Add Memory management library to pt-mysql-summary

v2.2.19 released 2016-08-16

Percona Toolkit 2.2.19 includes the following changes:

New Features

- 1221372: `pt-online-schema-change` now aborts with an error if the server is a slave, because this can break data consistency in case of row-based replication. If you are sure that the slave will not use row-based replication, you can disable this check using the `--force-slave-run` option.
- 1485195: `pt-table-checksum` now forces replica table character set to UTF-8.
- 1517155: Added `--create-table-engine` option to `pt-heartbeat`, which can be used to set a storage engine for the `heartbeat` table different from the database default engine.
- 1595678: Added `--slave-user` and `--slave-password` options to `pt-online-schema-change`
- 1595912: Added `--slave-user` and `--slave-password` options to `pt-table-sync` and `pt-table-checksum`
- 1610385: `pt-online-schema-change` now re-checks the list of slaves in the DSN table. This enables changing the contents of the table while the tool is running.

Bug fixes

- 1581752: Fixed `pt-query-digest` date and time parsing from MySQL 5.7 slow query log.
- 1592166: Fixed memory leak when `pt-kill` kills a query
- 1592608: Fixed overflow of `CONCAT_WS` when `pt-table-checksum` or `pt-table-sync` checksums large BLOB, TEXT, or BINARY columns.
- 1593265: Fixed `pt-archiver` deleting rows that were not archived.
- 1610386: Fixed `pt-slave-restart` handling of GTID ranges where the left-side integer is larger than 9
- 1610387: Removed extra word 'default' from the `--verbose` help for `pt-slave-restart`
- 1610388: Fixed `pt-table-sync` not quoting enum values properly. They are now recognized as CHAR fields.

Changelog

- Feature 1610385: Recheck the list of slaves while OSC runs (Thanks Daniël van Eeden & Mikhail Izioumtchenko)
- Fixed bug 1221372: pt-osc should error if server is a slave in row based replication
- Fixed bug 1485195: pt-table-checksum should force replica table charset to utf8 Edit (Thanks Jaime Crespo)
- Fixed bug 1517155: Added –create-table-engine param to pt-heartbeat
- Fixed bug 1581752: SlowLogParser is able to handle dates in RFC339 format for MySQL 5.7 (Thanks Nickolay Ihalainen)
- Fixed bug 1592166: pt-kill leaks memory
- Fixed bug 1592166: pt-kill leaks memory each time it kills a query
- Fixed bug 1592608: Large BLOB/TEXT/BINARY Produces NULL Checksum (Thanks Jervin Real)
- Fixed bug 1593265: Fixed pt-archiver deletes wrong rows #103 (Thanks Tibor Korocz & David Ducos)
- Fixed bug 1595678: Added –slave-user and –slave-password to pt-online-schema-change & pt-table-sync
- Fixed bug 1610386: Handle GTID ranges where the left-side integer is larger than 9 (Thanks @sodabrew)
- Fixed bug 1610387: Remove extra word 'default' from the –verbose help (Thanks @sodabrew)

- Fixed bug 1610388: add enum column type to is_char check so that values are properly quoted (Thanks Daniel Kinon)

v2.2.18 released 2016-06-24

Percona Toolkit 2.2.18 has been released. This release includes the following new features and bug fixes.

New features:

- 1537416: `pt-stalk` now sorts the output of transactions by id
- 1553340: Added "Shared" memory info to `pt-summary`
- PT-24: Added the `--no-vertical-format` option for `pt-query-digest`, allowing compatibility with non-standard MySQL clients that don't support the `\G` directive at the end of a statement

Bug fixes:

- 1402776: Fixed error when parsing `tcpdump` capture with `pt-query-digest`
- 1521880: Improved `pt-online-schema-change` plugin documentation
- 1547225: Clarified the description of the `--attribute-value-limit` option for `pt-query-digest`
- 1569564: Fixed all PERL-based tools to return a zero exit status when run with the `--version` option
- 1576036: Fixed error that sometimes prevented to choose the primary key as index, when using the `-where` option for `pt-table-checksum`
- 1585412: Fixed the inability of `pt-query-digest` to parse the general log generated by MySQL (and Percona Server) 5.7 instance
- PT-36: Clarified the description of the `--verbose` option for `pt-slave-restart`

Changelog

- Feature 1537416 : pt-stalk now sorts the output of transactions by id
- Feature 1553340 : Added "Shared" memory info to pt-summary
- Feature PT-24 : Added the –no-vertical-format option for pt-query-digest, allowing compatibility with non-standard MySQL clients that don't support the G directive at the end of a statement
- Fixed bug 1402776: Fixed error when parsing tcpdump capture with pt-query-digest
- Fixed bug 1521880: Improved pt-online-schema-change plugin documentation
- Fixed bug 1547225: Clarified the description of the –attribute-value-limit option for pt-query-digest
- Fixed bug 1569564: Fixed all PERL-based tools to return a zero exit status when run with the –version option
- Fixed bug 1576036: Fixed error that sometimes prevented to choose the primary key as index, when using the -where option for pt-table-checksum
- Fixed bug 1585412: Fixed the inability of pt-query-digest to parse the general log generated by MySQL (and Percona Server) 5.7 instance
- Fixed bug PT-36 : Clarified the description of the –verbose option for pt-slave-restart

v2.2.17 released 2016-03-07

Percona Toolkit 2.2.17 has been released. This release contains 1 new feature and 15 bug fixes.

New Features:

- Percona Toolkit 2.2.17 has implemented general compatibility with MySQL 5.7 tools, documentation and test suite

Bug Fixes:

- Bug 1523685: `pt-online-schema-change` invalid recursion method where comma was interpreted as the separation of two DSN methods has been fixed.
- Bugs 1480719 and 1536305: The current version of Perl on supported distributions has implemented stricter checks for arguments provided to `sprintf`. This could cause warnings when `pt-query-digest` and `pt-table-checksum` were being run.
- Bug 1498128: `pt-online-schema-change` would fail with an error if the table being altered has foreign key constraints where some start with an underscore and some don't.
- Bug 1336734: `pt-online-schema-change` has implemented new `--null-to-non-null` flag which can be used to convert `NULL` columns to `NOT NULL`.
- Bug 1362942: `pt-slave-restart` would fail to run on **|MariaDB|** 10.0.13 due to a different implementation of `GTID`.
- Bug 1389041: `pt-table-checksum` had a high likelihood to skip a table when row count was around `chunk-size` * `chunk-size-limit`. To address this issue a new `--slave-skip-tolerance` option has been implemented.
- Bug 1506748: `pt-online-schema-change` could not set the `SQL_MODE` by using the `--set-vars` option, preventing some use case schema changes that require it.
- Bug 1523730: `pt-show-grants` didn't sort the column-level privileges.
- Bug 1526105: `pt-online-schema-change` would fail if used with `--no-drop-old-table` option after ten times. The issue would arise because there was an accumulation of tables that have already have had their names extended, the code would retry ten times to append an underscore, each time finding an old table with that number of underscores appended.
- Bug 1529411: `pt-mysql-summary` was displaying incorrect information about Fast Server Restarts for Percona Server 5.6.
- PT-30: `pt-stalk` shell `collect` module was confusing the new mysql variable `binlog_error_action` with the `log_error` variable.

Changelog

- Feature : General compatibility with MySQL 5.7 tools, docs and test suite
- Fixed bug 1529411: pt-mysql-summary displays incorrect info about Fast Server Restarts for Percona Server 5.6
- Fixed bug 1506748: pt-online-schema-change cannot set sql_mode using –set-vars
- Fixed bug 1336734: pt-online-schema-change added –null-to-non-null option to allow NULLable columns to be converted to NOT NULL
- Fixed bug 1498128: pt-online-schema-change doesn't apply underscores to foreign keys individually
- Fixed bug 1523685: pt-online-schema Invalid recursion method: t=dsns

- Fixed bug 1526105: pt-online-schema-change fails when using –no-drop-old-table after 10 times
- Fixed bug 1536305: pt-query-digest : Redundant argument in sprintf
- Fixed bug PT-27 : pt-query-digest doc bug with –since and too many colons
- Fixed bug PT-28 : pt-query-digest: Make documentation of –attribute-value-limit option more clear
- Fixed bug 1435370: pt-show-grants fails against MySQL-5.7.6
- Fixed bug 1523730: pt-show-grants doesn't sort column-level privileges
- Fixed bug 1362942: pt-slave-restart fails on MariaDB 10.0.13 (gtid_mode confusion)
- Fixed bug PT-30 : pt-stalk: new var binlog_error_action causes bug in collect module
- Fixed bug 1389041: pt-table-checksum has high likelyhood to skip a table when row count is around chunk-size * chunk-size-limit
- Fixed bug 1480719: pt-table-checksum redundant argument in printf

v2.2.16 released 2015-11-09

Percona Toolkit 2.2.16 has been released. This release contains 3 new features and 2 bug fixes.

New Features:

- 1491261: When using MySQL 5.6 or later, and `innodb_stats_persistent` option is enabled (by default, it is enabled), then `pt-online-schema-change` will now run with the `--analyze-before-swap` option. This ensures that queries continue to use correct execution path, instead of switching to full table scan, which could cause possible downtime. If you do not want `pt-online-schema-change` to run `ANALYZE` on new tables before the swap, you can disable this behavior using the `--no-analyze-before-swap` option.
- 1402051: `pt-online-schema-change` will now wait forever for slaves to be available and not be lagging. This ensures that the tool does not abort during faults and connection problems on slaves.
- 1452895: `pt-archiver` now issues 'keepalive' queries during and after bulk insert/delete process that takes a long time. This keeps the connection alive even if the `innodb_kill_idle_transaction` variable is set to a low value.

Bug Fixes:

- 1488685: The `--filter` option for `pt-kill` now works correctly.
- 1494082: The `pt-stalk` tool no longer uses the `-warn` option when running `find`, because the option is not supported on FreeBSD.

Changelog

- Fixed bug 1452895: pt-archiver dies with "MySQL server has gone away" when innodb_kill_idle_transaction set to low value and bulk insert/delete process takes too long time
- Fixed bug 1488685: pt-kill option –filter does not work
- Feature 1402051: pt-online-schema-change should reconnect to slaves
- Fixed bug 1491261: pt-online-schema-change, MySQL 5.6, and InnoDB optimizer stats can cause downtime
- Fixed bug 1494082: pt-stalk find -warn option is not portable

- Feature 1389041: Document that pt-table-checksum has high likelihood to skip a table when row count is around chunk-size * chunk-size-limit

v2.2.15 released 2015-08-28

New Features

- Added `--max-flow-ctl` option with a value set in percent. When a Percona XtraDB Cluster node is very loaded, it sends flow control signals to the other nodes to stop sending transactions in order to catch up. When the average value of time spent in this state (in percent) exceeds the maximum provided in the option, the tool pauses until it falls below again.

  Default is no flow control checking.

  This feature was requested in the following bugs: 1413101 and 1413137.

- Added the `--sleep` option for `pt-online-schema-change` to avoid performance problems. The option accepts float values in seconds.

  This feature was requested in the following bug: 1413140.

- Implemented ability to specify `--check-slave-lag` multiple times. The following example enables lag checks for two slaves:

```
pt-archiver --no-delete --where '1=1' --source h=oltp_server,D=test,t=tbl --dest␣
↪h=olap_server --check-slave-lag h=slave1 --check-slave-lag h=slave2 --limit␣
↪1000 --commit-each
```

  This feature was requested in the following bug: 14452911.

- Added the `--rds` option to `pt-kill`, which makes the tool use Amazon RDS procedure calls instead of the standard MySQL `kill` command.

  This feature was requested in the following bug: 1470127.

Bugs Fixed

- 1042727: `pt-table-checksum` doesn't reconnect the slave $dbh

  Before, the tool would die if any slave connection was lost. Now the tool waits forever for slaves.

- 1056507: `pt-archiver --check-slave-lag` agressiveness

  The tool now checks replication lag every 100 rows instead of every row, which significantly improves efficiency.

- 1215587: Adding underscores to constraints when using `pt-online-schema-change` can create issues with constraint name length

  Before, multiple schema changes lead to underscores stacking up on the name of the constraint until it reached the 64 character limit. Now there is a limit of two underscores in the prefix, then the tool alternately removes or adds one underscore, attempting to make the name unique.

- 1277049: `pt-online-schema-change` can't connect with comma in password

  For all tools, documented that commas in passwords provided on the command line must be escaped.

- 1441928: Unlimited chunk size when using `pt-online-schema-change` with `--chunk-size-limit=0` inhibits checksumming of single-nibble tables

  When comparing table size with the slave table, the tool now ignores `--chunk-size-limit` if it is set to zero to avoid multiplying by zero.

- 1443763: Update documentation and/or implentation of `pt-archiver --check-interval`

  Fixed the documentation for `--check-interval` to reflect its correct behavior.

- 1449226: `pt-archiver` dies with "MySQL server has gone away" when `--innodb_kill_idle_transaction` is set to a low value and `--check-slave-lag` is enabled

  The tool now sends a dummy SQL query to avoid timing out.

- 1446928: `pt-online-schema-change` not reporting meaningful errors

  The tool now produces meaningful errors based on text from MySQL errors.

- 1450499: ReadKeyMini causes `pt-online-schema-change` session to lock under some circumstances

  Removed ReadKeyMini, because it is no longer necessary.

- 1452914: `--purge` and `--no-delete` are mutually exclusive, but still allowed to be specified together by `pt-archiver`

  The tool now issues an error when `--purge` and `--no-delete` are specified together

- 1455486: `pt-mysql-summary` is missing the `--ask-pass` option

  Added the `--ask-pass` option to the tool

- 1457573: `pt-sift` fails to download `pt-diskstats` `pt-pmp` `pt-mext` `pt-align`

  Added the `-L` option to `curl` and changed download address to use HTTPS.

- 1462904: `pt-duplicate-key-checker` doesn't support triple quote in column name

  Updated TableParser module to handle literal backticks.

- 1488600: `pt-stalk` doesn't check TokuDB status

  Implemented status collection similar to how it is performed for InnoDB.

- 1488611: various testing bugs related to newer perl versions

  Fixed test failures related to new Perl versions.

v2.2.14 released 2015-04-14

Percona Toolkit 2.2.14 has been released. This release contains two new features and seventeen bug fixes.

New Features:

- pt-slave-find can now resolve the IP address and show the slave's hostname. This can be done with the new `--resolve-address` option.
- pt-table-sync can now ignore the tables whose names match specific Perl regex with the `--ignore-tables-regex` option.

Bugs Fixed:

- Fixed bug 925781: Inserting non-BMP characters into a column with utf8 charset would cause the `Incorrect string value` error when running the pt-table-checksum.
- Fixed bug 1368244: pt-online-schema-change `--alter-foreign-keys-method=drop-swap` was not atomic and thus it could be interrupted. Fixed by disabling common interrupt signals during the critical drop-rename phase.
- Fixed bug 1381280: pt-table-checksum was failing on `BINARY` field in Primary Key. Fixed by implementing new `--binary-index` flag to optionally create checksum table using BLOB data type.

- Fixed bug 1421405: Running pt-upgrade against a log with many identical (or similar) queries was producing repeated sections with the same fingerprint.
- Fixed bug 1402730: pt-duplicate-key-checker was not checking for duplicate keys when `--verbose` option was set.
- Fixed bug 1406390: A race condition was causing pt-heartbeat to crash with sleep argument error.
- Fixed bug 1417558: pt-stalk when used along with `--collect-strace` didn't write the strace output to the expected destination file.
- Fixed bug 1421025: Missing dependency for `perl-TermReadKey` RPM package was causing toolkit commands to fail when they were run with `--ask-pass` option.
- Fixed bug 1421781: pt-upgrade would fail when log contained `SELECT...INTO` queries. Fixed by ignoring/skipping those queries.
- Fixed bug 1425478: pt-stalk was removing non-empty files that were starting with an empty line.
- Fixed bug 1419098: Fixed bad formatting in the pt-table-checksum documentation.

Changelog

- Fixed bug 1402730 pt-duplicate-key-checker seems useless with MySQL 5.6
- Fixed bug 1415646 pt-duplicate-key-checker documentation does not explain how Size Duplicate Indexes is calculated
- Fixed bug 1406390 pt-heartbeat crashes with sleep argument error
- Fixed bug 1368244 pt-online-schema-change –alter-foreign-keys-method=drop-swap is not atomic
- FIxed bug 1417864 pt-online-schema-change documentation, the interpretation of –tries create_triggers:5:0.5,drop_triggers:5:0.5 is wrong
- Fixed bug 1404313 pt-query-digest: specifying a file that doesn't exist as log causes the tool to wait for STDIN instead of giving an error
- Feature 1418446 pt-slave-find resolve IP addresses option
- Fixed bug 1417558 pt-stalk with –collect-strace output doesn't go to an YYYY_MM_DD_HH_mm_ss-strace file
- Fixed bug 1425478 pt-stalk removes non-empty files that start with empty line
- Fixed bug 925781 pt-table-checksum checksum error when default-character-set = utf8
- Fixed bug 1381280 pt-table-checksum fails on BINARY field in PK
- Feature 1439842 pt-table-sync lacks –ignore-tables-regex option
- Fixed bug 1401399 pt-table-sync fails to close one db handle
- Fixed bug 1442277 pt-table-sync-ignores system databases but doc doesn't clarify this
- Fixed bug 1421781 pt-upgrade fails on SELECT ... INTO queries
- Fixed bug 1421405 pt-upgrade fails to aggregate queries based on fingerprint
- Fixed bug 1439348 pt-upgrade erroneously reports number of diffs
- Fixed bug 1421025 rpm missing dependency on perl-TermReadKey for –ask-pass

v2.2.13 released 2015-01-26

Percona Toolkit 2.2.13 has been released. This release contains one new feature and twelve bug fixes.

New Features:

- pt-kill now supports new `--query-id` option. This option can be used to print a query fingerprint hash after killing a query to enable the cross-referencing with the pt-query-digest output. This option can be used along with `--print` option as well.

Bugs Fixed:

- Fixed bug 1019479: pt-table-checksum now works with `ONLY_FULL_GROUP_BY` sql_mode.
- Fixed bug 1394934: running pt-table-checksum in debug mode would cause an error.
- Fixed bug 1396868: regression introduced in Percona Toolkit 2.2.12 caused pt-online-schema-change not to honor `--ask-pass` option.
- Fixed bug 1399789: pt-table-checksum would fail to find Percona XtraDB Cluster nodes when variable `wsrep_node_incoming_address` was set to `AUTO`.
- Fixed bug 1408375: Percona Toolkit was vulnerable to MITM attack which could allow exfiltration of MySQL configuration information via `--version-check` option. This vulnerability was logged as *CVE 2015-1027 <http://www.cve.mitre.org/cgi-bin/cvename.cgi?name=2015-1027>_*
- Fixed bug 1321297: pt-table-checksum was reporting differences on timestamp columns with replication from 5.5 to 5.6 server version, although the data was identical.
- Fixed bug 1388870: pt-table-checksum was showing differences if the master and slave were in different time zone.
- Fixed bug 1402668: pt-mysql-summary would exit if Percona XtraDB Cluster was in `Donor/Desynced` state.
- Fixed bug 1266869: pt-stalk would fail to start if `$HOME` environment variable was not set.

Changelog

- Feature 1391240: pt-kill added query fingerprint hash to output
- Fixed bug 1402668: pt-mysql-summary fails on cluster in Donor/Desynced status
- Fixed bug 1396870: pt-online-schema-change CTRL+C leaves terminal in inconsistent state
- Fixed bug 1396868: pt-online-schema-change –ask-pass option error
- Fixed bug 1266869: pt-stalk fails to start if $HOME environment variable is not set
- Fixed bug 1019479: pt-table-checksum does not work with sql_mode ONLY_FULL_GROUP_BY
- Fixed bug 1394934: pt-table-checksum error in debug mode
- Fixed bug 1321297: pt-table-checksum reports diffs on timestamp columns in 5.5 vs 5.6
- Fixed bug 1399789: pt-table-checksum fails to find pxc nodes when wsrep_node_incoming_address is set to AUTO
- Fixed bug 1388870: pt-table-checksum has some errors with different time zones
- Fixed bug 1408375: vulnerable to MITM attack which would allow exfiltration of MySQL configuration information via –version-check
- Fixed bug 1404298: missing MySQL5.7 test files for pt-table-checksum

- Fixed bug 1403900: added sandbox and fixed sakila test db for 5.7

v2.2.12 released 2014-11-14

Percona Toolkit 2.2.12 has been released. This release contains one new feature and seven bug fixes.

New Features:

- pt-stalk now gathers `dmesg` output from up to 60 seconds before the triggering event.

Bugs Fixed:

- Fixed bug 1376561: pt-archiver was not able to archive all the rows when a table had a hash partition. Fixed by implementing support for tables which have primary or unique indexes.
- Fixed bug 1217466: pt-table-checksum would refuses to run on Percona XtraDB Cluster if `server_id` was the same on all nodes. Fixed by using the `wsrep_node_incoming_address` as a unique identifier for cluster nodes, instead of relying on `server_id`.
- Fixed bug 1269695: pt-online-schema-change documentation now contains more information about limitations on why it isn't running `ALTER TABLE` for a table which has only a non-unique index.
- Fixed bug 1328686: Running pt-hearbeat with –check-read-only option would cause an error when running on server with `read_only` option. Tool now waits for server `read_only` status to be disabled before starting to run.
- Fixed bug 1373937: pt-table-checksum now supports `none` as valid `--recursion-method` when using with Percona XtraDB Cluster.
- Fixed bug 1377888: Documentation was stating that pt-query-digest is able to parse a raw binary log file, while it can only parse a file which was decoded with `mysqlbinlog` tool before. Fixed by improving the documentation and adding a check for binary file and providing a relevant error message.

Changelog

- Fixed bug 1376561: pt-archiver is not able to archive all the rows when a table has a hash partition
- Fixed bug 1328686: pt-heartbeat check-read-only option does not prevent creates or inserts
- Fixed bug 1269695: pt-online-schema-change does not allow ALTER for a table without a non-unique, while manual does not explain this
- Fixed bug 1217466: pt-table-checksum refuses to run on PXC if server_id is the same on all nodes
- Fixed bug 1373937: pt-table-checksum requires recursion when working with and XtraDB Cluster node
- Fixed bug 1377888: pt-query-digest manual for –type binlog is ambiguous
- Fixed bug 1349086: pt-stalk should also gather dmesg output
- Fixed bug 1361293: Some scripts fail when no-version-check option is put in global config file

v2.2.11 released 2014-09-26

Percona Toolkit 2.2.11 has been released. This release contains seven bug fixes.

Bugs Fixed:

- Fixed bug 1262456: pt-query-digest didn't report host details when host was using skip-name-resolve option. Fixed by using the IP of the host instead of it's name, when the hostname is missing.
- Fixed bug 1264580: pt-mysql-summary was incorrectly parsing key/value pairs in the wsrep_provider_options option, which resulted in incomplete my.cnf information.
- Fixed bug 1318985: pt-stalk is now using `SQL_NO_CACHE` when executing queries for locks and transactions. Previously this could lead to situations where most of the queries that were `waiting on query cache mutex` were the pt-stalk queries (INNODB_TRX).
- Fixed bug 1348679: When using `--  -p` option to enter the password for pt-stalk it would ask user to re-enter the password every time tool connects to the server to retrieve the information. New option `--ask-pass` has been introduced that can be used to specify the password only once.
- Fixed bug 1368379: A parsing error caused pt-summary (specifically the `report_system_info` module) to choke on the "Memory Device" parameter named "Configured Clock Speed" when using dmidecode to report memory slot information.

Changelog

- Fixed bug 1262456: pt-query-digest doesn't report host details
- Fixed bug 1264580: pt-mysql-summary incorrectly tries to parse key/value pairs in wsrep_provider_options resulting in incomplete my.cnf information
- Fixed bug 1318985: pt-stalk should use SQL_NO_CACHE
- Fixed bug 1348679: pt-stalk handles mysql user password in awkward way
- Fixed bug 1365085: Various issues with tests
- Fixed bug 1368379: pt-summary problem parsing dmidecode output on some machines
- Fixed bug 1303388: Typo in pt-variable-advisor

v2.2.10 released 2014-08-06

Percona Toolkit 2.2.10 has been released. This release contains six bug fixes.

Bugs Fixed:

- Fixed bug 1287253: pt-table-checksum would exit with error if it would encounter deadlock when doing checksum. This was fixed by retrying the command in case of deadlock error.
- Fixed bug 1311654: When used with Percona XtraDB Cluster, pt-table-checksum could show incorrect result if –resume option was used. This was fixed by adding a new `--replicate-check-retries` command line parameter. If you are having resume problems you can now set `--replicate-check-retries` N , where N is the number of times to retry a discrepant checksum (default = 1 , no retries). Setting a value of `3` is enough to completely eliminate spurious differences.
- Fixed bug 1299387: pt-query-digest didn't work correctly do to a changed logging format when field `Thread_id` has been renamed to `Id`. Fixed by implementing support for the new format.
- Fixed bug 1340728: in some cases, where the index was of type "hash" , pt-online-schema-change would refuse to run because MySQL reported it would not use an index for the select. This check should have been able to be skipped using –nocheck-plan option, but it wasn't. `--nocheck-plan` now ignores the chosen index correctly.

- Fixed bug 1253872: When running pt-table-checksum or pt-online-schema on a server that is unused, setting the 20% max load would fail due to tools rounding the value down. This has been fixed by rounding the value up.
- Fixed bug 1340364: Due to incompatibility of dash and bash syntax some shell tools were showing error when queried for version.

Changelog

- Fixed bug 1287253: pt-table-checksum deadlock
- Fixed bug 1299387: 5.6 slow query log Thead_id becomes Id
- Fixed bug 1311654: pt-table-checksum + PXC inconsistent results upon –resume
- Fixed bug 1340728: pt-online-schema-change doesn't work with HASH indexes
- Fixed bug 1253872: pt-table-checksum max load 20% rounds down
- Fixed bug 1340364: some shell tools output error when queried for –version

v2.2.9 released 2014-07-08

Percona Toolkit 2.2.9 has been released. This release contains five bug fixes.

Bugs Fixed:

- Fixed bug 1335960: pt-query-digest could not parse the binlogs from MySQL 5.6 because the binlog format was changed.
- Fixed bug 1315130: pt-online-schema-change did not find child tables as expected. It could incorrectly locate tables which reference a table with the same name in a different schema and could miss tables referencing the altered table if they were in a different schema.
- Fixed bug 1335322: pt-stalk would fail when variable or threshold was non-integer.
- Fixed bug 1258135: pt-deadlock-logger was inserting older deadlocks into the `deadlock` table even if it was already there creating unnecessary noise. For example, if the deadlock happened 1 year ago, and MySQL keeps it in the memory and pt-deadlock-logger would `INSERT` it into `percona.deadlocks` table every minute all the time until server was restarted. This was fixed by comparing with the last deadlock fingerprint before issuing the `INSERT` query.
- Fixed bug 1329422: pt-online-schema-change foreign-keys-method=none can break FK constraints in a way that is hard to recover from. Allthough this method of handling foreign key constraints is provided so that the database administrator can disable the tool's built-in functionality if desired, a warning and confirmation request when using alter-foreign-keys-method "none" has been added to warn users when using this option.

Changelog

- Fixed bug 1258135: pt-deadlock-logger introduces a noise to MySQL
- Fixed bug 1329422: pt-online-schema-change foreign-keys-method=none breaks constraints
- Fixed bug 1315130: pt-online-schema-change not properly detecting foreign keys
- Fixed bug 1335960: pt-query-digest cannot parse binlogs from 5.6
- Fixed bug 1335322: pt-stalk fails when variable or threshold is non-integer

v2.2.8 released 2014-06-04

Percona Toolkit 2.2.8 has been released. This release has two new features and six bug fixes.

New Features:

- pt-agent has been replaced by percona-agent. More information on percona-agent can be found in the Introducing the 3-Minute MySQL Monitor blogpost.
- pt-slave-restart now supports MySQL 5.6 global transaction IDs.
- pt-table-checkum now has new –plugin option which is similar to pt-online-schema-change –plugin

Bugs Fixed:

- Fixed bug 1254233: pt-mysql-summary was showing blank InnoDB section for 5.6 because it was using `have_innodb` variable which was removed in MySQL 5.6.
- Fixed bug 965553: pt-query-digest didn't fingerprint true/false literals correctly.
- Fixed bug 1286250: pt-online-schema-change was requesting password twice.
- Fixed bug 1295667: pt-deadlock-logger was logging incorrect timestamp because tool wasn't aware of the time-zones.
- Fixed bug 1304062: when multiple tables were specified with pt-table-checksum –ignore-tables, only one of them would be ignored.
- Fixed bug : pt-show-grant –ask-pass option was asking for password in `STDOUT` instead of `STDERR` where it could be seen.

Percona Toolkit packages can be downloaded from http://www.percona.com/downloads/percona-toolkit/ or the Percona Software Repositories (http://www.percona.com/software/repositories/).

Changelog

- Removed pt-agent
- Added pt-slave-restart GTID support
- Added pt-table-checksum –plugin
- Fixed bug 1304062: –ignore-tables does not work correctly
- Fixed bug 1295667: pt-deadlock-logger logs incorrect ts
- Fixed bug 1254233: pt-mysql-summary blank InnoDB section for 5.6
- Fixed bug 1286250: pt-online-schema-change requests password twice
- Fixed bug 965553: pt-query-digest dosn't fingerprint true/false literals correctly
- Fixed bug 290911: pt-show-grant –ask-pass prints "Enter password" to STDOUT

v2.2.7 released 2014-02-20

Percona Toolkit 2.2.7 has been released. This release has only one bug fix.

- Fixed bug 1279502: –version-check behaves like spyware

Although never used, –version-check had the ability to get any local program's version. This fix removed that ability.

Percona Toolkit packages can be downloaded from http://www.percona.com/downloads/percona-toolkit/ or the Percona Software Repositories (http://www.percona.com/software/repositories/).

v2.2.6 released 2013-12-18

Percona Toolkit 2.2.6 has been released. This release has 16 bug fixes and a few new features. One bug fix is very important, so 2.2 users are strongly encouraged to upgrade:

- Fixed bug 1223458: pt-table-sync deletes child table rows

Buried in the pt-table-sync docs is this warning:

> Also be careful with tables that have foreign key constraints with C<ON DELETE> or C<ON UPDATE> definitions because these might cause unintended changes on the child tables.

Until recently, either no one had this problem, or no one reported it, or no one realized that pt-table-sync caused it. In the worst case, pt-table-sync could delete all rows in child tables, which is quite surprising and bad. As of 2.2.6, pt-table-sync has option –[no]check-child-tables which is on by default. In cases were this "bug" can happen, pt-table-sync prints a warning and skips the table. Read the option's docs to learn more.

Another good bug fix is:

- Fixed bug 1217013: pt-duplicate-key-checker misses exact duplicate unique indexes

After saying "pt-duplicate-key-checker hasn't had a bug in years" at enough conferences, users proved us wrong–thanks! The tool is better now.

- Fixed bug 1195628: pt-online-schema-change gets stuck looking for its own _new table

This was poor feedback from the tool more than a bug. There was a point in the tool where it waited forever for slaves to catch up, but it did this silently. Now the tool reports –progress while it's waiting and it reports which slaves, if any, it found and intends to check. In short: its feedback delivers a better user experience.

Finally, this bug (more like a feature request/change) might be a backwards-incompatible change:

- Fixed bug 1214685: pt-mysql-summary schema dump prompt can't be disabled

The change is that pt-mysql-summary no longer prompts to dump and summarize schemas. To do this, you must specify –databases or, a new option, –all-databases. Several users said this behavior was better, so we made the change even though some might consider it a backwards-incompatible change.

Percona Toolkit packages can be downloaded from http://www.percona.com/downloads/percona-toolkit/ or the Percona Software Repositories (http://www.percona.com/software/repositories/).

Changelog

- Added pt-query-digest support for Percona Server slow log rate limiting
- Added pt-agent –ping
- Added pt-mysql-summary –all-databases
- Added pt-stalk –sleep-collect
- Added pt-table-sync –[no]check-child-tables
- Fixed bug 1249150: PTDEBUG prints some info to STDOUT
- Fixed bug 1248363: pt-agent requires restart after changing MySQL options

- Fixed bug 1248778: pt-agent –install on PXC is not documented
- Fixed bug 1250973: pt-agent –install doesn't check for previous install
- Fixed bug 1250968: pt-agent –install suggest MySQL user isn't quoted
- Fixed bug 1251004: pt-agent –install error about slave is confusing
- Fixed bug 1251726: pt-agent –uninstall fails if agent is running
- Fixed bug 1248785: pt-agent docs don't list privs required for its MySQL user
- Fixed bug 1215016: pt-deadlock-logger docs use pt-fk-error-logger
- Fixed bug 1201443: pt-duplicate-key-checker error when EXPLAIN key_len=0
- Fixed bug 1217013: pt-duplicate-key-checker misses exact duplicate unique indexes
- Fixed bug 1214685: pt-mysql-summary schema dump prompt can't be disabled
- Fixed bug 1195628: pt-online-schema-change gets stuck looking for its own _new table
- Fixed bug 1249149: pt-query-digest stats prints to STDOUT instead of STDERR
- Fixed bug 1071979: pt-stak error parsing df with NFS
- Fixed bug 1223458: pt-table-sync deletes child table rows

v2.2.5 released 2013-10-16

Percona Toolkit 2.2.5 has been released. This release has four new features and a number of bugfixes.

Query_time histogram has been added to the pt-query-digest JSON output, not the actual chart but the values necessary to render the chart later, so the values for each bucket.

As of pt-table-checksum 2.2.5, skipped chunks cause a non-zero exit status. An exit status of zero or 32 is equivalent to a zero exit status with skipped chunks in previous versions of the tool.

New –no-drop-triggers option has been implemented for pt-online-schema-change in case users want to rename the tables manually, when the load is low.

New –new-table-name option has been added to pt-online-schema-change which can be used to specify the temporary table name.

- Fixed bug #1199589: pt-archiver would delete the data even with the –dry-run option.
- Fixed bug #821692: pt-query-digest didn't distill LOAD DATA correctly.
- Fixed bug #984053: pt-query-digest didn't distill INSERT/REPLACE without INTO correctly.
- Fixed bug #1206677: pt-agent docs were referencing wrong web address.
- Fixed bug #1210537: pt-table-checksum –recursion-method=cluster would crash if no nodes were found.

Percona Toolkit packages can be downloaded from http://www.percona.com/downloads/percona-toolkit/ or the Percona Software Repositories (http://www.percona.com/software/repositories

Changelog

- Added Query_time histogram bucket counts to pt-query-digest JSON output
- Added pt-online-schema-change –[no]drop-triggers option
- Fixed bug #1199589: pt-archiver deletes data despite –dry-run

- Fixed bug #944051: pt-table-checksum has ambiguous exit status
- Fixed bug #1209436: pt-kill –log-dsn may not work on Perl 5.8
- Fixed bug #1210537: pt-table-checksum –recursion-method=cluster crashes if no nodes are found
- Fixed bug #1215608: pt-online-schema-change new table suffix is hard-coded
- Fixed bug #1229861: pt-table-sync quotes float values, can't sync
- Fixed bug #821692: pt-query-digest doesn't distill LOAD DATA correctly
- Fixed bug #984053: pt-query-digest doesn't distill INSERT/REPLACE without INTO correctly
- Fixed bug #1206728: pt-deadlock-logger 2.2 requires DSN on command line
- Fixed bug #1226721: pt-agent on CentOS 5 fails to send data
- Fixed bug #821690: pt-query-digest doesn't distill IF EXISTS correctly
- Fixed bug #1206677: pt-agent docs reference clodu.percona.com

v2.2.4 released 2013-07-18

Percona Toolkit 2.2.4 has been released. This release two new features and a number of bugfixes.

pt-query-digest –output json includes query examples as of v2.2.3. Some people might not want this because it exposes real data. New option, –output json-anon, has been implemented. This option will provide the same data without query examples. It's "anonymous" in the sense that there's no identifying data; nothing more than schema and table structs can be inferred from fingerprints.

When using drop swap with pt-online-schema-change there is some production impact. This impact can be measured because tool outputs the current timestamp on lines for operations that may take awhile.

- Fixed bug #1163735: pt-table-checksum fails if explicit_defaults_for_timestamp is enabled in 5.6

pt-table-checksum would fail if variable explicit_defaults_for_timestamp was enabled in MySQL 5.6.

- Fixed bug #1182856: Zero values causes "Invalid –set-vars value: var=0"

Trying to assign 0 to any variable by using –set-vars option would cause "Invalid –set-vars value" message.

- Fixed bug #1188264: pt-online-schema-change error copying rows: Undefined subroutine &pt_online_schema_change::get
- Fixed the typo in the pt-online-schema-change code that could lead to a tool crash when copying the rows.
- Fixed bug #1199591: pt-table-checksum doesn't use non-unique index with highest cardinality

pt-table-checksum was using the first non-unique index instead of the one with the highest cardinality due to a sorting bug.

Percona Toolkit packages can be downloaded from http://www.percona.com/downloads/percona-toolkit/ or the Percona Software Repositories (http://www.percona.com/software/repositories

Changelog

- Added pt-query-digest anonymous JSON output
- Added pt-online-schema-change timestamp output
- Fixed bug #1136559: pt-table-checksum: Deep recursion on subroutine "SchemaIterator::_iterate_dbh"

- Fixed bug #1163735: pt-table-checksum fails if explicit_defaults_for_timestamp is enabled in 5.6
- Fixed bug #1182856: Zero values causes "Invalid –set-vars value: var=0"
- Fixed bug #1188264: pt-online-schema-change error copying rows: Undefined subroutine &pt_online_schema_change::get
- Fixed bug #1195034: pt-deadlock-logger error: Use of uninitialized value $ts in pattern match (m//)
- Fixed bug #1199591: pt-table-checksum doesn't use non-unique index with highest cardinality
- Fixed bug #1168434: pt-upgrade reports differences on NULL
- Fixed bug #1172317: pt-sift does not work if pt-stalk did not collect due to a full disk
- Fixed bug #1176010: pt-query-digest doesn't group db and *db* together
- Fixed bug #1137556: pt-heartbeat docs don't account for –utc
- Fixed bug #1168106: pt-variable-advisor has the wrong default value for innodb_max_dirty_pages_pct in 5.5 and 5.6
- Fixed bug #1168110: pt-variable-advisor shows key_buffer_size in 5.6 as unconfigured (even though it is)
- Fixed bug #1171968: pt-query-digest docs don't mention –type=rawlog
- Fixed bug #1174956: pt-query-digest and pt-fingerprint don't strip some multi-line comments

v2.2.3 released 2013-06-17

Percona Toolkit 2.2.3 has been released which has only two changes: pt-agent and a bug fix for pt-online-schema-change. pt-agent is not a command line tool but a client-side agent for Percona Cloud Tools. Visit https://cloud.percona.com for more information. The pt-online-schema-change bug fix is bug 1188002: pt-online-schema-change causes "ERROR 1146 (42S02): "Table 'db._t_new' doesn't exist". This happens when the tool's triggers cannot be dropped.

Percona Toolkit packages can be downloaded from http://www.percona.com/downloads/percona-toolkit/ or the Percona Software Repositories (http://www.percona.com/software/repositories/).

Changelog

- Added new tool: pt-agent
- Fixed bug 1188002: pt-online-schema-change causes "ERROR 1146 (42S02): Table 'db._t_new' doesn't exist"

v2.2.2 released 2013-04-24

Percona Toolkit 2.2.2 has been released. This is the second release of the 2.2 series and aims to fix bugs in the previous release and provide usability enhacements to the toolkit.

Users may note the revival of the –show-all option in pt-query-digest. This had been removed in 2.2.1, but resulted in too much output in certain cases.

A new –recursion-method was added to pt-table-checksum: cluster. This method attempts to auto-discover cluster nodes, alleviating the need to specify cluster node DSNs in a DSN table (–recursion-method=dsn).

The following highlights some of the more interesting and "hot" bugs in this release:

- Bug #1127450: pt-archiver –bulk-insert may corrupt data

pt-archiver –bulk-insert didn't work with –charset UTF-8. This revealed a case where the tool could corrupt data by double-encoding. This is now fixed, but remains relatively dangerous if using DBD::mysql 3.0007 which does not handle UTF-8 properly.

- Bug #1163372: pt-heartbeat –utc –check always returns 0

Unfortunately, the relatively new –utc option for pt-heart was still broken because "[MySQL] interprets date as a value in the current time zone and converts it to an internal value in UTC." Now the tool works correctly with –utc by specifying "SET time_zone='+0:00"', and older versions of the tool can be made to work by specifying –set-vars "time_zone='+0:00"'.

- Bug #821502: Some tools don't have –help or –version

pt-align, pt-mext, pt-pmp and pt-sift now have both options.

This is another solid bug fix release, and all users are encouraged to upgrade.

Percona Toolkit packages can be downloaded from http://www.percona.com/downloads/percona-toolkit/ or the Percona Software Repositories (http://www.percona.com/software/repositories/).

Changelog

- Added –show-all to pt-query-digest
- Added –recursion-method=cluster to pt-table-checksum
- Fixed bug 1127450: pt-archiver –bulk-insert may corrupt data
- Fixed bug 1163372: pt-heartbeat –utc –check always returns 0
- Fixed bug 1156901: pt-query-digest –processlist reports duplicate queries for replication thread
- Fixed bug 1160338: pt-query-digest 2.2 prints unwanted debug info on tcpdump parsing errors
- Fixed bug 1160918: pt-query-digest 2.2 prints too many string values
- Fixed bug 1156867: pt-stalk prints the wrong variable name in verbose mode when –function is used
- Fixed bug 1081733: pt-stalk plugins can't access the real –prefix
- Fixed bug 1099845: pt-table-checksum pxc same_node function incorrectly uses wsrep_sst_receive_address
- Fixed bug 821502: Some tools don't have –help or –version
- Fixed bug 947893: Some tools use @@hostname without */!50038/*
- Fixed bug 1082406: An explicitly set wsrep_node_incoming_address may make SHOW STATUS LIKE 'wsrep_incoming_addresses' return a portless address

v2.2.1 released 2013-03-14

Percona Toolkit 2.2.1 has been released. This is the first release in the new 2.2 series which supersedes the 2.1 series and renders the 2.0 series obsolete. We plan to do one more bug fix release for 2.1 (2.1.10), but otherwise all new development and fixes and will now focus on 2.2.

Percona Toolkit 2.2 has been several months in the making, and it turned out very well, with many more new features, changes, and improvements than originally anticipated. Here are the highlights:

- Official support for MySQL 5.6

We started beta support for MySQL 5.6 in 2.1.8 when 5.6 was still beta. Now that 5.6 is GA, so is our support for it. Check out the Percona Toolkit supported platforms and versions: http://www.percona.com/mysql-support/policies/percona-toolkit-supported-platforms-and-versions

When you upgrade to MySQL 5.6, be sure to upgrade to Percona Toolkit 2.2, too.

- Official support for Percona XtraDB Cluster (PXC)

We also started beta support for Percona XtraDB Cluster in 2.1.8, but now that support is official in 2.2 because we have had many months to work with PXC and figure out which tools work with it and how. There's still one noticeable omission: pt-table-sync. It's still unclear if or how one would sync a cluster that, in theory, doesn't become out-of-sync. As Percona XtraDB Cluster develops, Percona Toolkit will continue to evolve to support it.

- pt-online-schema-change (pt-osc) is much more resilient

pt-online-schema-change 2.1 has been a great success, and people have been using it for evermore difficult and challenging tasks. Consequently, we needed to make it "try harder", even though it already tried pretty hard to keep working despite recoverable errors and such. Whereas pt-osc 2.1 only retries certain operations, pt-osc 2.2 retries every critical operation, and its tries and wait time between tries for all operations are configurable. Also, we removed –lock-wait-timeout which set innodb_lock_wait_timeout because that now conflicts, or is at least confused with, lock_wait_timeout (introduced in MySQL 5.5) for metadata locks. Now –set-vars is used to set both of these (or any) system variables. For a quick intro to metadata locks and how they may affect you, see Ovais's article: http://www.mysqlperformanceblog.com/2013/02/01/implications-of-metadata-locking-changes-in-mysql-5-5/

What does this all mean? In short: pt-online-schema-change 2.2 is far more resilient out of the box. It's also aware of metadata locks now, whereas 2.1 was not really aware of them. And it's highly configurable, so you can make the tool try _very_ hard to keep working.

- pt-upgrade is brand-new

pt-upgrade was written once long ago, thrown into the world, and then never heard from again... until now. Now that we have four base versions of MySQL (5.0, 5.1, 5.5, and 5.6), plus at least four major forks (Percona Server, MariaDB, Percona XtraDB Cluster, and MariaDB Galera Cluster), upgrades are fashionable, so to speak. Problem is: "original" pt-upgrade was too noisy and too complex. pt-upgrade 2.2 is far simpler and far easier to use. It's basically what you expect from such a tool.

Moreover, it has a really helpful new feature: "reference results", i.e. saved results from running queries on a server. Granted, this can take *a lot* of disk space, but it allows you to "run now, compare later."

If you're thinking about upgrading, give pt-upgrade a try. It also reads every type of log now (slow, general, binary, and tcpdump), so you shouldn't have a problem finding queries to run and compare.

- pt-query-digest is simpler

pt-query-digest 2.2 has fewer options now. Basically, we re-focused it on its primary objective: analyzing MySQL query logs. So the ability to parse memcached, Postgres, Apache, and other logs was removed. We also removed several options that probably nobody ever used, and changed/renamed other options to be more logical. The result is a simpler, more focused tool, i.e. less overwhelming.

Also, pt-query-digest 2.2 can save results in JSON format (–output=json). This feature is still in development while we determine the optimal JSON structure.

- Version check is on by default

Way back in 2.1.4, released September/October 2012, we introduced a feature called "version check" into most tools: http://percona.com/version-check It's like a lot of software that automatically checks for updates, but it's also more: it's a free service from Percona that advises when certain programs (Percona Toolkit tools, MySQL, Perl, etc.) are either out of date or are known bad versions. For example, there are two versions of the DBD::mysql Perl module that have problems. And there are certain versions of MySQL that have critical bugs. Version check will warn you about these if your system is running them.

What's new in 2.2 is that, whereas this feature (specifically, the option in tools: –version-check) was off by default, now it's on by default. If the IO::Socket::SSL Perl module is installed (easily available through your package manager), it will use a secure (https) connection over the web, else it will use a standard (http) connection.

Check out http://percona.com/version-check for more information.

- pt-query-advisor, pt-tcp-model, pt-trend, and pt-log-player are gone

We removed pt-query-advisor, pt-tcp-model, pt-trend, and pt-log-player. Granted, no tool is ever really gone: if you need one of these tools, get it from 2.1. pt-log-player is now superseded by Percona Playback (http://www.percona.com/doc/percona-playback/). pt-query-advisor was removed so that we can focus our efforts on its online counterpart instead: https://tools.percona.com/query-advisor. The other tools were special projects that were not widely used.

- pt-stalk and pt-mysql-summary have built-in MySQL options

No more "pt-stalk – -h db1 -u me". pt-stalk 2.2 and pt-mysql-summary 2.2 have all the standard MySQL options built-in, like other tools: –user, –host, –port, –password, –socket, –defaults-file. So now the command line is what you expect: pt-stalk -h dhb1 -u me.

- pt-stalk –no-stalk is no longer magical

Originally, pt-stalk –no-stalk was meant to simulate pt-collect, i.e. collect once and exit. To do that, the tool magically set some options and clobbered others, resulting in no way to do repeated collections at intervals. Now –no-stalk means only that: don't stalk, just collect, respecting –interval and –iterations as usual. So to collect once and exit: pt-stalk –no-stalk –iterations 1.

- pt-fk-error-logger and pt-deadlock-logger are standardized

Similar to the pt-stalk –no-stalk changes, pt-fk-error-logger and pt-deadlock-logger received mini overhauls in 2.2 to make their run-related options (–run-time, –interval, –iterations) standard. If you hadn't noticed, one tool would run forever by default, while the other would run once and exit. And each treated their run-related options a little differently. This magic is gone now: both tools run forever by default, so specify –iterations or –run-time to limit how long they run.

There were other miscellaneous bug fixes, too. See https://launchpad.net/percona-toolkit/+milestone/2.2.1 for the full list.

As the first release in a new series, 2.2 features are not yet finalized. In other words, we may change things like the pt-query-digest –output json format in future releases after receiving real-world feedback.

Percona Toolkit 2.2 is an exciting release with many helpful new features. Users are encouraged to begin upgrading, particularly given that, except for the forthcoming 2.1.10 release, no more work will be done on 2.1 (unless you're a Percona customer with a support contract or other agreement).

If you upgrade from 2.1 to 2.2, be sure to re-read tools' documentation to see what has changed because much as changed for certain tools.

Percona Toolkit packages can be downloaded from http://www.percona.com/downloads/percona-toolkit/ or the Percona Software Repositories (http://www.percona.com/software/repositories/).

Changelog

- Official support for MySQL 5.6
- Official support for Percona XtraDB Cluster
- Redesigned pt-query-digest
- Redesigned pt-upgrade

- Redesigned pt-fk-error-logger
- Redesigned pt-deadlock-logger
- Changed –set-vars in all tools
- Renamed –retries to –tries in pt-online-schema-change
- Added –check-read-only to pt-heartbeat
- Added MySQL options to pt-mysql-summary
- Added MySQL options to pt-stalk
- Removed –lock-wait-timeout from pt-online-schema-change (use –set-vars)
- Removed –lock-wait-timeout from pt-table-checksum (use –set-vars)
- Removed pt-query-advisor
- Removed pt-tcp-model
- Removed pt-trend
- Removed pt-log-player
- Enabled –version-check by default in all tools
- Fixed bug 1008796: Several tools don't have –database
- Fixed bug 1087319: Quoter::serialize_list() doesn't handle multiple NULL values
- Fixed bug 1086018: pt-config-diff needs to parse wsrep_provider_options
- Fixed bug 1056838: pt-fk-error-logger –run-time works differently than pt-deadlock-logger –run-time
- Fixed bug 1093016: pt-online-schema-change doesn't retry RENAME TABLE
- Fixed bug 1113301: pt-online-schema-change blocks on metadata locks
- Fixed bug 1125665: pt-stalk –no-stalk silently clobbers other options, acts magically
- Fixed bug 1019648: pt-stalk truncates InnoDB status if there are too many transactions
- Fixed bug 1087804: pt-table-checksum doesn't warn if no slaves are found

v2.1.9 released 2013-02-14

Percona Toolkit 2.1.9 has been released. This release primarily aims to restore backwards-compatibility with pt-heartbeat 2.1.7 and older, but it also has important bug fixes for other tools.

- Fixed bug 1103221: pt-heartbeat 2.1.8 doesn't use precision/sub-second timestamps
- Fixed bug 1099665: pt-heartbeat 2.1.8 reports big time drift with UTC_TIMESTAMP

The previous release switched the time authority from Perl to MySQL, and from local time to UTC. Unfortunately, these changes caused a loss of precision and, if mixing versions of pt-heartbeat, made the tool report a huge amount of replication lag. This release makes the tool compatible with pt-heartbeat 2.1.7 and older again, but the UTC behavior introduced in 2.1.8 is now only available by specifying the new –utc option.

- Fixed bug 918056: pt-table-sync false-positive error "Cannot nibble table because MySQL chose no index instead of the PRIMARY index"

This is an important bug fix for pt-table-sync: certain chunks from pt-table-checksum resulted in an impossible WHERE, causing the false-positive "Cannot nibble" error, if those chunks had diffs.

- Fixed bug 1099836: pt-online-schema-change fails with "Duplicate entry" on MariaDB

MariaDB 5.5.28 (https://kb.askmonty.org/en/mariadb-5528-changelog/) fixed a bug: "Added warnings for duplicate key errors when using INSERT IGNORE". However, standard MySQL does not warn in this case, despite the docs saying that it should. Since pt-online-schema-change has always intended to ignore duplicate entry errors by using "INSERT IGNORE", it now handles the MariaDB case by also ignoring duplicate entry errors in the code.

- Fixed bug 1103672: pt-online-schema-change makes bad DELETE trigger if PK is re-created with new columns

pt-online-schema-change 2.1.9 handles another case of changing the primary key. However, since changing the primary key is tricky, the tool stops if –alter contains "DROP PRIMARY KEY", and you have to specify –no-check-alter to acknowledge this case.

- Fixed bug 1099933: pt-stalk is too verbose, fills up log

Previously, pt-stalk printed a line for every check. Since the tool is designed to be a long-running daemon, this could result in huge log files with "matched=no" lines. The tool has a new –verbose option which makes it quieter by default.

All users should upgrade, but in particular, users of versions 2.1.7 and older are strongly recommended to skip 2.1.8 and go directly to 2.1.9.

Users of pt-heartbeat in 2.1.8 who prefer the UTC behavior should keep in mind that they will have to use the –utc option after upgrading.

Percona Toolkit packages can be downloaded from http://www.percona.com/downloads/percona-toolkit/ or the Percona Software Repositories (http://www.percona.com/software/repositories/).

Changelog

- Fixed bug 1103221: pt-heartbeat 2.1.8 doesn't use precision/sub-second timestamps
- Fixed bug 1099665: pt-heartbeat 2.1.8 reports big time drift with UTC_TIMESTAMP
- Fixed bug 1099836: pt-online-schema-change fails with "Duplicate entry" on MariaDB
- Fixed bug 1103672: pt-online-schema-change makes bad DELETE trigger if PK is re-created with new columns
- Fixed bug 1115333: pt-pmp doesn't list the origin lib for each function
- Fixed bug 823411: pt-query-digest shouldn't print "Error: none" for tcpdump
- Fixed bug 1103045: pt-query-digest fails to parse non-SQL errors
- Fixed bug 1105077: pt-table-checksum: Confusing error message with binlog_format ROW or MIXED on slave
- Fixed bug 918056: pt-table-sync false-positive error "Cannot nibble table because MySQL chose no index instead of the PRIMARY index"
- Fixed bug 1099933: pt-stalk is too verbose, fills up log

v2.1.8 released 2012-12-21

Percona Toolkit 2.1.8 has been released. This release includes 28 bug fixes, beta support for MySQL 5.6, and extensive support for Percona XtraDB Cluster (PXC). Users intending on running the tools on Percona XtraDB Cluster or MySQL 5.6 should upgrade. The following tools have been verified to work on PXC versions 5.5.28 and newer:

- pt-table-chcecksum
- pt-online-schema-change
- pt-archive

- pt-mysql-summary
- pt-heartbeat
- pt-variable-advisor
- pt-config-diff
- pt-deadlock-logger

However, there are limitations when running these tools on PXC; see the Percona XtraDB Cluster section in each tool's documentation for further details. All other tools, with the exception of pt-slave-find, pt-slave-delay and pt-slave-restart, should also work correctly, but in some cases they have not been modified to take advantage of PXC features, so they may behave differently in future releases.

The bug fixes are widely assorted. The following highlights some of the more interesting and "hot" bugs:

- Fixed bug 1082599: pt-query-digest fails to parse timestamp with no query

Slow logs which include timestamps but no query–which can happen if using slow_query_log_timestamp_always in Percona Server–were misparsed, resulting in an erroneous report. Now such no-query events show up in reports as `/* No query */`.

- Fixed bug 1078838: pt-query-digest doesn't parse general log with "Connect user as user"

The "as" was misparsed and the following word would end up reported as the database; pt-query-digest now handles this correctly.

- Fixed bug 1015590: pt-mysql-summary doesn't handle renamed variables in Percona Server 5.5

Some renamed variables had caused the Percona Server section to work unreliably.

- Fixed bug 1074179: pt-table-checksum doesn't ignore tables for –replicate-check-only

When using –replicate-check-only, filter options like –databases and –tables were not applied.

- Fixed bug 886059: pt-heartbeat handles timezones inconsistently

Previously, pt-heartbeat respected the MySQL time zone, but this caused false readings (e.g. very high lag) with slaves running in different time zones. Now pt-heartbeat uses UTC regardless of the server or MySQL time zone.

- Fixed bug 1079341: pt-online-schema-change checks for foreign keys on MyISAM tables

Since MyISAM tables can't have foreign keys, and the tool uses the information_schema to find child tables, this could cause unnecessary load on the server.

2.1.8 continues the trend of solid bug fix releases, and all 2.1 users are encouraged to upgrade.

Percona Toolkit packages can be downloaded from http://www.percona.com/downloads/percona-toolkit/ or the Percona Software Repositories (http://www.percona.com/software/repositories/).

Changelog

- Beta support for MySQL 5.6
- Beta support for Percona XtraDB Cluster
- pt-online-schema-change: If ran on Percona XtraDB Cluster, requires PXC 5.5.28 or newer
- pt-table-checksum: If ran on Percona XtraDB Cluster, requires PXC 5.5.28 or newer
- pt-upgrade: Added –[no]disable-query-cache
- Fixed bug 927955: Bad pod2rst transformation
- Fixed bug 898665: Bad online docs formatting for –[no]vars

- Fixed bug 1022622: pt-config-diff is case-sensitive
- Fixed bug 1007938: pt-config-diff doesn't handle end-of-line comments
- Fixed bug 917770: pt-config-diff Use of uninitialized value in substitution (s///) at line 1996
- Fixed bug 1082104: pt-deadlock-logger doesn't handle usernames with dashes
- Fixed bug 886059: pt-heartbeat handles timezones inconsistently
- Fixed bug 1086259: pt-kill –log-dsn timestamp is wrong
- Fixed bug 1015590: pt-mysql-summary doesn't handle renamed variables in Percona Server 5.5
- Fixed bug 1079341: pt-online-schema-change checks for foreign keys on MyISAM tables
- Fixed bug 823431: pt-query-advisor hangs on big queries
- Fixed bug 996069: pt-query-advisor RES.001 is incorrect
- Fixed bug 933465: pt-query-advisor false positive on RES.001
- Fixed bug 937234: pt-query-advisor issues wrong RES.001
- Fixed bug 1082599: pt-query-digest fails to parse timestamp with no query
- Fixed bug 1078838: pt-query-digest doesn't parse general log with "Connect user as user"
- Fixed bug 957442: pt-query-digest with custom –group-by throws error
- Fixed bug 887638: pt-query-digest prints negative byte offset
- Fixed bug 831525: pt-query-digest help output mangled
- Fixed bug 932614: pt-slave-restart CHANGE MASTER query causes error
- Fixed bug 1046440: pt-stalk purge_samples slows down checks
- Fixed bug 986847: pt-stalk does not report NFS iostat
- Fixed bug 1074179: pt-table-checksum doesn't ignore tables for –replicate-check-only
- Fixed bug 911385: pt-table-checksum v2 fails when –resume + –ignore-database is used
- Fixed bug 1041391: pt-table-checksum debug statement for "Chosen hash func" prints undef
- Fixed bug 1075638: pt-table-checksum Illegal division by zero at line 7950
- Fixed bug 1052475: pt-table-checksum uninitialized value in numeric lt (<) at line 8611
- Fixed bug 1078887: Tools let –set-vars clobber the required SQL mode

v2.1.7 released 2012-11-19

Percona Toolkit 2.1.7 has been released which is a hotfix for two bugs when using pt-table-checksum with Percona XtraDB Cluster:

- Bug 1080384: pt-table-checksum 2.1.6 crashes using PTDEBUG
- Bug 1080385: pt-table-checksum 2.1.6 –check-binlog-format doesn't ignore PXC nodes

If you're using pt-table-checksum with a Percona XtraDB Cluster, you should upgrade. Otherwise, users can wait until the next full release.

Percona Toolkit packages can be downloaded from http://www.percona.com/downloads/percona-toolkit/ or the Percona Software Repositories (http://www.percona.com/software/repositories/).

Changelog

- Fixed bug 1080384: pt-table-checksum 2.1.6 crashes using PTDEBUG
- Fixed bug 1080385: pt-table-checksum 2.1.6 –check-binlog-format doesn't ignore PXC nodes

v2.1.6 released 2012-11-13

Percona Toolkit 2.1.6 has been released. This release includes 33 bug fixes and three new features: pt-online-schema-change now handles renaming columns without losing data, removing one of the tool's limitations. pt-online-schema-change also got two new options: –default-engine and –statistics. Finally, pt-stalk now has a plugin hook interface, available through the –plugin option. The bug fixes are widely assorted. The following highlights some of the more interesting and "hot" bugs:

- Bug 978133: pt-query-digest review table privilege checks don't work

The same checks were removed from pt-table-checksum on 2.1.3 and pt-table-sync on 2.1.4, so this just follows suit.

- Bug 938068: pt-table-checksum doesn't warn if binlog_format=row or mixed on slaves

A particularly important fix, as it may stop pt-table-checksum from breaking replication in these setups.

- Bug 1043438: pt-table-checksum doesn't honor –run-time while checking replication lag

If you run multiple instances of pt-table-checksum on a badly lagged server, actually respecting –run-time stops the instances from divebombing the server when the replica catches up.

- Bug 1062324: pt-online-schema-change DELETE trigger fails when altering primary key

Fixed by choosing a key on the new table for the DELETE trigger.

- Bug 1062563: pt-table-checksum 2.1.4 doesn't detect diffs on Percona XtraDB Cluster nodes

A follow up to the same fix in the previous release, this adds to warnings for cases in which pt-table-checksum may work incorrectly and require some user intervention: One for the case of master -> cluster, and one for cluster1 -> cluster2.

- Bug 821715: LOAD DATA LOCAL INFILE broken in some platforms

This bug has hounded the toolkit for quite some time. In some platforms, trying to use LOAD DATA LOCAL INFILE would fail as if the user didn't have enough privileges to perform the operation. This was a misdiagnoses from MySQL; The actual problem was that the libmysqlclient.so provided by some vendors was compiled in a way that disallowed users from using the statement without some extra work. This fix adds an 'L' option to the DSNs the toolkit uses, tells the the tools to explicitly enables LOAD DATA LOCAL INFILE. This affected two pt-archiver and pt-upgrade, so if you are on an effected OS and need to use those, you can simply tag an L=1 to your DSN and everything should start working.

- Bug 866075: pt-show-grant doesn't support column-level grants

This was actually the 'hottest' bug in the tracker.

This is another solid bug fix release, and all 2.1 users are encouraged to upgrade.

Percona Toolkit packages can be downloaded from http://www.percona.com/downloads/percona-toolkit/ or the Percona Software Repositories (http://www.percona.com/software/repositories/).

Changelog

- pt-online-schema-change: Columns can now be renamed without data loss

- pt-online-schema-change: New –default-engine option
- pt-stalk: Plugin hooks available through the –plugin option to extend the tool's functionality
- Fixed bug 1069951: –version-check default should be explicitly "off"
- Fixed bug 821715: LOAD DATA LOCAL INFILE broken in some platforms
- Fixed bug 995896: Useless use of cat in Daemon.pm
- Fixed bug 1039074: Tools exit 0 on error parsing options, should exit non-zero
- Fixed bug 938068: pt-table-checksum doesn't warn if binlog_format=row or mixed on slaves
- Fixed bug 1009510: pt-table-checksum breaks replication if a slave table is missing or different
- Fixed bug 1043438: pt-table-checksum doesn't honor –run-time while checking replication lag
- Fixed bug 1073532: pt-table-checksum error: Use of uninitialized value in int at line 2778
- Fixed bug 1016131: pt-table-checksum can crash with –columns if none match
- Fixed bug 1039569: pt-table-checksum dies if creating the –replicate table fails
- Fixed bug 1059732: pt-table-checksum doesn't test all hash functions
- Fixed bug 1062563: pt-table-checksum 2.1.4 doesn't detect diffs on Percona XtraDB Cluster nodes
- Fixed bug 1043528: pt-deadlock-logger can't parse db/tbl/index on partitioned tables
- Fixed bug 1062324: pt-online-schema-change DELETE trigger fails when altering primary key
- Fixed bug 1058285: pt-online-schema-change fails if sql_mode explicitly or implicitly uses ANSI_QUOTES
- Fixed bug 1073996: pt-online-schema-change fails with "I need a max_rows argument"
- Fixed bug 1039541: pt-online-schema-change –quiet doesn't disable –progress
- Fixed bug 1045317: pt-online-schema-change doesn't report how many warnings it suppressed
- Fixed bug 1060774: pt-upgrade fails if select column > 64 chars
- Fixed bug 1070916: pt-mysql-summary may report the wrong cnf file
- Fixed bug 903229: pt-mysql-summary incorrectly categorizes databases
- Fixed bug 866075: pt-show-grant doesn't support column-level grants
- Fixed bug 978133: pt-query-digest review table privilege checks don't work
- Fixed bug 956981: pt-query-digest docs for event attributes link to defunct Maatkit wiki
- Fixed bug 1047335: pt-duplicate-key-checker fails when it encounters a crashed table
- Fixed bug 1047701: pt-stalk deletes non-empty files
- Fixed bug 1070434: pt-stalk –no-stalk and –iterations 1 don't wait for the collect
- Fixed bug 1052722: pt-fifo-split is processing n-1 rows initially
- Fixed bug 1013407: pt-find documentation error with mtime and InnoDB
- Fixed bug 1059757: pt-trend output has no header
- Fixed bug 1063933: pt-visual-explain docs link to missing pdf
- Fixed bug 1075773: pt-fk-error-logger crashes if there's no foreign key error
- Fixed bug 1075775: pt-fk-error-logger –dest table example doesn't work

v2.1.5 released 2012-10-08

Percona Toolkit 2.1.5 has been released. This release is less than two weeks after the release of 2.1.4 because we wanted to address these bugs quickly:

- Bug 1062563: pt-table-checksum 2.1.4 doesn't detect diffs on Percona XtraDB Cluster nodes
- Bug 1063912: pt-table-checksum 2.1.4 miscategorizes Percona XtraDB Cluster-based slaves as cluster nodes
- Bug 1064016: pt-table-sync 2.1.4 –version-check may not work with HTTPS/SSL

The first two bugs fix how pt-table-checksum works with Percona XtraDB Cluster (PXC). Although the 2.1.4 release did introduce support for PXC, these bugs prevented pt-table-checksum from working correctly with a cluster.

The third bug is also related to a feature new in 2.1.4: –version-check. The feature uses HTTPS/SSL by default, but some modules in pt-table-sync weren't update which could prevent it from working on older systems. Related, the version check web page mentioned in tools' documentation was also created.

If you're using pt-table-checksum with a Percona XtraDB Cluster, you should definitely upgrade. Otherwise, users can wait until 2.1.6 for another full release.

Percona Toolkit packages can be downloaded from http://www.percona.com/downloads/percona-toolkit/ or the Percona Software Repositories (http://www.percona.com/software/repositories/).

Changelog

- Fixed bug 1062563: pt-table-checksum 2.1.4 doesn't detect diffs on Percona XtraDB Cluster nodes
- Fixed bug 1063912: pt-table-checksum 2.1.4 miscategorizes Percona XtraDB Cluster-based slaves as cluster nodes
- Fixed bug 1064016: pt-table-sync 2.1.4 –version-check may not work with HTTPS/SSL
- Fixed bug 1060423: Missing version-check page

v2.1.4 released 2012-09-20

Percona Toolkit 2.1.4 has been released. This release includes 26 bug fixes and three new features: Making pt-table-checksum work with Percona XtraDB Cluster, adding a –run-time option to pt-table-checksum, and implementing the "Version Check" feature, enabled through the –version-check switch. For further information on –version-check, see http://www.mysqlperformanceblog.com/2012/09/10/introducing-the-version-check-feature-in-percona-toolkit/. The bug fixes are widely assorted. The following highlights some of the more interesting and "hot" bugs:

- Fixed bug 1017626: pt-table-checksum doesn't work with Percona XtraDB Cluster

Note that this requires Percona XtraDB Cluster 5.5.27-23.6 or newer, as the fix depends on this bug https://bugs.launchpad.net/codership-mysql/+bug/1023911 being resolved.

- Fixed bug 1034170: pt-table-checksum –defaults-file isn't used for slaves

Previously, users had no recourse but using –recursion-method in conjunction with a dsn table to sidestep this bug, so this fix is a huge usability gain. This was caused by the toolkit not copying the -F portion of the main dsn.

- Fixed bug 1039184: pt-upgrade error "I need a right_sth argument"

Which were stopping pt-upgrade from working on a MySQL 4.1 host.

- Fixed bug 1036747: pt-table-sync priv checks need to be removed

The same checks were removed in the previous release from pt-table-checksum, so this continues the trend.

- Fixed bug 1038995: pt-stalk –notify-by-email fails

This was a bug in our shell option parsing library, and would potentially affect any option starting with 'no'.

Like 2.1.3, this is another solid bug fix release, and 2.1 users are encouraged to upgrade.

Percona Toolkit packages can be downloaded from http://www.percona.com/downloads/percona-toolkit/ or the Percona Software Repositories (http://www.percona.com/software/repositories/).

Changelog

- pt-table-checksum: Percona XtraDB Cluster support
- pt-table-checksum: Implemented the standard –run-time option
- Implemented the version-check feature in several tools, enabled with the –version-check option
- Fixed bug 856060: Document gdb dependency
- Fixed bug 1041394: Unquoted arguments to tr break the bash tools
- Fixed bug 1035311: pt-diskstats shows wrong device names
- Fixed bug 1036804: pt-duplicate-key-checker error parsing InnoDB table with no PK or unique keys
- Fixed bug 1022658: pt-online-schema-change dropping FK limitation isn't documented
- Fixed bug 1041372: pt-online-schema-changes fails if db+tbl name exceeds 64 characters
- Fixed bug 1029178: pt-query-digest –type tcpdump memory usage keeps increasing
- Fixed bug 1037211: pt-query-digest won't distill LOCK TABLES in lowercase
- Fixed bug 942114: pt-stalk warns about bad "find" usage
- Fixed bug 1035319: pt-stalk df -h throws away needed details
- Fixed bug 1038995: pt-stalk –notify-by-email fails
- Fixed bug 1038995: pt-stalk does not get all InnoDB lock data
- Fixed bug 952722: pt-summary should show information about Fusion-io cards
- Fixed bug 899415: pt-table-checksum doesn't work if slaves use RBR
- Fixed bug 954588: pt-table-checksum –check-slave-lag docs aren't clear
- Fixed bug 1034170: pt-table-checksum –defaults-file isn't used for slaves
- Fixed bug 930693: pt-table-sync and text columns with just whitespace
- Fixed bug 1028710: pt-table-sync base_count fails on n = 1000, base = 10
- Fixed bug 1034717: pt-table-sync division by zero error with varchar primary key
- Fixed bug 1036747: pt-table-sync priv checks need to be removed
- Fixed bug 1039184: pt-upgrade error "I need a right_sth argument"
- Fixed bug 1035260: sh warnings in pt-summary and pt-mysql-summary
- Fixed bug 1038276: ChangeHandler doesn't quote varchar columns with hex-looking values
- Fixed bug 916925: CentOS 5 yum dependency resolution for perl module is wrong
- Fixed bug 1035950: Percona Toolkit RPM should contain a dependency on perl-Time-HiRes

v2.1.3 released 2012-08-03

Percona Toolkit 2.1.3 has been released. This release includes 31 bug fixes and one new feature: pt-kill –log-dsn to log information about killed queries to a table. The bug fixes are widely assorted. The following highlights some of the more interesting and "hot" bugs:

- Fixed bug 916168: pt-table-checksum privilege check fails on MySQL 5.5

pt-table-checksum used to check the user's privileges, but the method was not always reliable, and due to http://bugs.mysql.com/bug.php?id=61846 it became quite unreliable on MySQL 5.5. So the privs check was removed altogether, meaning that the tool may fail later if the user's privileges are insufficient.

- Fixed bug 950294: pt-table-checksum should always create schema and tables with IF NOT EXISTS

In certain cases where the master and replicas have different schemas and/or tables, pt-table-checksum could break replication because the checksums table did not exist on a replica.

- Fixed bug 821703: pt-query-digest –processlist may crash
- Fixed bug 883098: pt-query-digest crashes if processlist has extra columns

Certain distributions of MySQL add extra columns to SHOW PROCESSLIST which caused pt-query-digest –processlist to crash at times.

- Fixed bug 941469: pt-kill doesn't reconnect if its connection is lost

pt-kill is meant to be a long-running daemon, so naturally it's important that it stays connected to MySQL.

- Fixed bug 1004567: pt-heartbeat –update –replace causes duplicate key error

The combination of these pt-heartbeat options could cause replication to break due to a duplicate key error.

- Fixed bug 1022628: pt-online-schema-change error: Use of uninitialized value in numeric lt (<) at line 6519

This bug was related to how –quiet was handled, and it could happen even if –quiet wasn't given on the command line.

All in all, this is solid bug fix release, and 2.1 users are encouraged to upgrade.

Percona Toolkit packages can be downloaded from http://www.percona.com/downloads/percona-toolkit/ or the Percona Software Repositories (http://www.percona.com/software/repositories/).

Changelog

- pt-kill: Implemented –log-dsn to log info about killed queries to a table
- Fixed bug 1016127: Install hint for DBD::mysql is wrong
- Fixed bug 984915: DSNParser does not check success of –set-vars
- Fixed bug 889739: pt-config-diff doesn't diff quoted strings properly
- Fixed bug 969669: pt-duplicate-key-checker –key-types=k doesn't work
- Fixed bug 1004567: pt-heartbeat –update –replace causes duplicate key error
- Fixed bug 1028614: pt-index-usage ignores –database
- Fixed bug 940733: pt-ioprofile leaves behind temp directory
- Fixed bug 941469: pt-kill doesn't reconnect if its connection is lost
- Fixed bug 1016114: pt-online-schema-change docs don't mention default values
- Fixed bug 1020997: pt-online-schema-change fails when table is empty

- Fixed bug 1022628: pt-online-schema-change error: Use of uninitialized value in numeric lt (<) at line 6519
- Fixed bug 937225: pt-query-advisor OUTER JOIN advice in JOI.003 is confusing
- Fixed bug 821703: pt-query-digest –processlist may crash
- Fixed bug 883098: pt-query-digest crashes if processlist has extra columns
- Fixed bug 924950: pt-query-digest –group-by db may crash profile report
- Fixed bug 1022851: pt-sift error: PREFIX: unbound variable
- Fixed bug 969703: pt-sift defaults to '.' instead of '/var/lib/pt-talk'
- Fixed bug 962330: pt-slave-delay incorrectly computes lag if started when slave is already lagging
- Fixed bug 954990: pt-stalk –nostalk does not work
- Fixed bug 977226: pt-summary doesn't detect LSI RAID control
- Fixed bug 1030031: pt-table-checksum reports wrong number of DIFFS
- Fixed bug 916168: pt-table-checksum privilege check fails on MySQL 5.5
- Fixed bug 950294: pt-table-checksum should always create schema and tables with IF NOT EXISTS
- Fixed bug 953141: pt-table-checksum ignores its default and explicit –recursion-method
- Fixed bug 1030975: pt-table-sync crashes if sql_mode includes ANSI_QUOTES
- Fixed bug 869005: pt-table-sync should always set REPEATABLE READ
- Fixed bug 903510: pt-tcp-model crashes in –type=requests mode on empty file
- Fixed bug 934310: pt-tcp-model –quantile docs wrong
- Fixed bug 980318: pt-upgrade results truncated if hostnames are long
- Fixed bug 821696: pt-variable-advisor shows too long of a snippet
- Fixed bug 844880: pt-variable-advisor shows binary logging as both enabled and disabled

v2.1.2 released 2012-06-12

Percona Toolkit 2.1.2 has been released. This is a very important release because it fixes a critical bug in pt-table-sync (bug 1003014) which caused various failures. All users of Percona Toolkit 2.1 should upgrade to this release. There were 47 other bug fixes, several new options, and other changes. The following is a high-level summary of the most important changes.

In addition to the critical bug fix mentioned above, another important pt-table-sync bug was fixed, bug 1002365: –ignore-* options did not work with –replicate. The –lock-and-rename feature of the tool was also disabled unless running MySQL 5.5 or newer because it did not work reliably in earlier versions of MySQL.

Several important pt-table-checksum bugs were fixed. First, a bug caused the tool to ignore the primary key. Second, the tool did not wait for the checksum table to replicate, so it could select from a nonexistent table on a replica and crash. Third, it did not check if all checksum queries were safe and chunk index with more than 3 columns could cause MySQL to scan many more rows than expected.

pt-online-schema-change received many improvements and fixes: it did not retry deadlocks, but now it does; –no-swap-tables caused an error; it did not handle column renames; it did not allow disabling foreign key checks; –dry-run always failed on tables with foreign keys; it used different keys for chunking and triggers; etc. In short: pt-online-schema-change 2.1.2 is superior to 2.1.1.

Two pt-archiver bugs were fixed: bug 979092, –sleep conflicts with bulk operations; and bug 903379, –file doesn't create a file.

–recursion-method=none was implemented in pt-heartbeat, pt-online-schema-change, pt-slave-find, pt-slave-restart, pt-table-checksum, and pt-table-sync. This allows these tools to avoid executing SHOW SLAVE STATUS which requires a privilege not available to Amazon RDS users.

Other bugs were fixed in pt-stalk, pt-variable-advisor, pt-duplicate-key-checker, pt-diskstats, pt-query-digest, pt-sift, pt-kill, pt-summary, and pt-deadlock-logger.

Percona Toolkit 2.1.2 should be backwards-compatible with 2.1.1, so users are strongly encouraged to upgrade.

Percona Toolkit packages can be downloaded from http://www.percona.com/downloads/percona-toolkit/ or the Percona Software Repositories (http://www.percona.com/software/repositories/).

Changelog

- pt-heartbeat: Implemented –recursion-method=none
- pt-index-usage: MySQL 5.5 compatibility fixes
- pt-log-player: MySQL 5.5 compatibility fixes
- pt-online-schema-change: Added –chunk-index-columns
- pt-online-schema-change: Added –[no]check-plan
- pt-online-schema-change: Added –[no]drop-new-table
- pt-online-schema-change: Implemented –recursion-method=none
- pt-query-advisor: Added –report-type for JSON output
- pt-query-digest: Removed –[no]zero-bool
- pt-slave-delay: Added –database
- pt-slave-find: Implemented –recursion-method=none
- pt-slave-restart: Implemented –recursion-method=none
- pt-table-checksum: Added –chunk-index-columns
- pt-table-checksum: Added –[no]check-plan
- pt-table-checksum: Implemented –recursion-method=none
- pt-table-sync: Disabled –lock-and-rename except for MySQL 5.5 and newer
- pt-table-sync: Implemented –recursion-method=none
- Fixed bug 945079: Shell tools TMPDIR may break
- Fixed bug 912902: Some shell tools still use basename
- Fixed bug 987694: There is no –recursion-method=none option
- Fixed bug 886077: Passwords with commas don't work, expose part of password
- Fixed bug 856024: Lintian warnings when building percona-toolkit Debian package
- Fixed bug 903379: pt-archiver –file doesn't create a file
- Fixed bug 979092: pt-archiver –sleep conflicts with bulk operations
- Fixed bug 903443: pt-deadlock-logger crashes on MySQL 5.5
- Fixed bug 941064: pt-deadlock-logger can't clear deadlocks on 5.5

- Fixed bug 952727: pt-diskstats shows incorrect wr_mb_s
- Fixed bug 994176: pt-diskstats –group-by=all –headers=scroll prints a header for every sample
- Fixed bug 894140: pt-duplicate-key-checker sometimes recreates a key it shouldn't
- Fixed bug 923896: pt-kill: uninitialized value causes script to exit
- Fixed bug 1003003: pt-online-schema-change uses different keys for chunking and triggers
- Fixed bug 1003315: pt-online-schema-change –dry-run always fails on table with foreign keys
- Fixed bug 1004551: pt-online-schema-change –no-swap-tables causes error
- Fixed bug 976108: pt-online-schema-change doesn't allow to disable foreign key checks
- Fixed bug 976109: pt-online-schema-change doesn't handle column renames
- Fixed bug 988036: pt-online-schema-change causes deadlocks under heavy write load
- Fixed bug 989227: pt-online-schema-change crashes with PTDEBUG
- Fixed bug 994002: pt-online-schema-change 2.1.1 doesn't choose the PRIMARY KEY
- Fixed bug 994010: pt-online-schema-change 2.1.1 crashes without InnoDB
- Fixed bug 996915: pt-online-schema-change crashes with invalid –max-load and –critical-load
- Fixed bug 998831: pt-online-schema-change – Should have an option to NOT drop tables on failure
- Fixed bug 1002448: pt-online-schema-change: typo for finding usable indexes
- Fixed bug 885382: pt-query-digest –embedded-attributes doesn't check cardinality
- Fixed bug 888114: pt-query-digest report crashes with infinite loop
- Fixed bug 949630: pt-query-digest mentions a Subversion repository
- Fixed bug 844034: pt-show-grants –separate fails with proxy user
- Fixed bug 946707: pt-sift loses STDIN after pt-diskstats
- Fixed bug 994947: pt-stalk doesn't reset cycles_true after collection
- Fixed bug 986151: pt-stalk-has mktemp error
- Fixed bug 993436: pt-summary Memory: Total reports M instead of G
- Fixed bug 1008778: pt-table-checksum doesn't wait for checksum table to replicate
- Fixed bug 1010232: pt-table-checksum doesn't check the size of checksum chunks
- Fixed bug 1011738: pt-table-checksum SKIPPED is zero but chunks were skipped
- Fixed bug 919499: pt-table-checksum fails with binary log error in mysql >= 5.5.18
- Fixed bug 972399: pt-table-checksum docs are not rendered right
- Fixed bug 978432: pt-table-checksum ignoring primary key
- Fixed bug 995274: pt-table-checksum can't use an undefined value as an ARRAY reference at line 2206
- Fixed bug 996110: pt-table-checksum crashes if InnoDB is disabled
- Fixed bug 987393: pt-table-checksum: Empy tables cause "undefined value as an ARRAY" errors
- Fixed bug 1002365: pt-table-sync –ignore-* options don't work with –replicate
- Fixed bug 1003014: pt-table-sync –replicate and –sync-to-master error "index does not exist"
- Fixed bug 823403: pt-table-sync –lock-and-rename doesn't work on 5.1

- Fixed bug 898138: pt-variable-advisor doesn't recognize 5.5.3+ concurrent_insert values

v2.1.1 released 2012-04-03

Percona Toolkit 2.1.1 has been released. This is the first release in the new 2.1 series which supersedes the 2.0 series. We will continue to fix bugs in 2.0, but 2.1 is now the focus of development.

2.1 introduces a lot of new code for:

- pt-online-schema-change (completely redesigned)
- pt-mysql-summary (completely redesigned)
- pt-summary (completely redesigned)
- pt-fingerprint (new tool)
- pt-table-usage (new tool)

There were also several bug fixes.

The redesigned tools are meant to replace their 2.0 counterparts because the 2.1 versions have the same or more functionality and they are simpler and more reliable. pt-online-schema-change was particularly enhanced to be as safe as possible given that the tool is inherently risky.

Percona Toolkit packages can be downloaded from http://www.percona.com/downloads/percona-toolkit/ or the Percona Software Repositories (http://www.percona.com/software/repositories/).

Changelog

- Completely redesigned pt-online-schema-change
- Completely redesigned pt-mysql-summary
- Completely redesigned pt-summary
- Added new tool: pt-table-usage
- Added new tool: pt-fingerprint
- Fixed bug 955860: pt-stalk doesn't run vmstat, iostat, and mpstat for –run-time
- Fixed bug 960513: SHOW TABLE STATUS is used needlessly
- Fixed bug 969726: pt-online-schema-change loses foreign keys
- Fixed bug 846028: pt-online-schema-change does not show progress until completed
- Fixed bug 898695: pt-online-schema-change add useless ORDER BY
- Fixed bug 952727: pt-diskstats shows incorrect wr_mb_s
- Fixed bug 963225: pt-query-digest fails to set history columns for disk tmp tables and disk filesort
- Fixed bug 967451: Char chunking doesn't quote column name
- Fixed bug 972399: pt-table-checksum docs are not rendered right
- Fixed bug 896553: Various documentation spelling fixes
- Fixed bug 949154: pt-variable-advisor advice for relay-log-space-limit
- Fixed bug 953461: pt-upgrade manual broken 'output' section

- Fixed bug 949653: pt-table-checksum docs don't mention risks posed by inconsistent schemas

v2.0.4 released 2012-03-07

Percona Toolkit 2.0.4 has been released. 23 bugs were fixed in this release, and three new features were implemented. First, –filter was added to pt-kill which allows for arbitrary –group-by. Second, pt-online-schema-change now requires that its new –execute option be given, else the tool will just check the tables and exit. This is a safeguard to encourage users to read the documentation, particularly when replication is involved. Third, pt-stalk also received a new option: –[no]stalk. To collect immediately without stalking, specify –no-stalk and the tool will collect once and exit.

This release is completely backwards compatible with previous 2.0 releases. Given the number of bug fixes, it's worth upgrading to 2.0.4.

Changelog

- Added –filter to pt-kill to allow arbitrary –group-by
- Added –[no]stalk to pt-stalk (bug 932331)
- Added –execute to pt-online-schema-change (bug 933232)
- Fixed bug 873598: pt-online-schema-change doesn't like reserved words in column names
- Fixed bug 928966: pt-pmp still uses insecure /tmp
- Fixed bug 933232: pt-online-schema-change can break replication
- Fixed bug 941225: Use of qw(...) as parentheses is deprecated at pt-kill line 3511
- Fixed bug 821694: pt-query-digest doesn't recognize hex InnoDB txn IDs
- Fixed bug 894255: pt-kill shouldn't check if STDIN is a tty when –daemonize is given
- Fixed bug 916999: pt-table-checksum error: DBD::mysql::st execute failed: called with 2 bind variables when 6 are needed
- Fixed bug 926598: DBD::mysql bug causes pt-upgrade to use wrong precision (M) and scale (D)
- Fixed bug 928226: pt-diskstats illegal division by zero
- Fixed bug 928415: Typo in pt-stalk doc: –trigger should be –function
- Fixed bug 930317: pt-archiver doc refers to nonexistent pt-query-profiler
- Fixed bug 930533: pt-sift looking for `*-processlist1;` broken compatibility with pt-stalk
- Fixed bug 932331: pt-stalk cannot collect without stalking
- Fixed bug 932442: pt-table-checksum error when column name has two spaces
- Fixed bug 932883: File Debian bug after each release
- Fixed bug 940503: pt-stalk disk space checks wrong on 32bit platforms
- Fixed bug 944420: –daemonize doesn't always close STDIN
- Fixed bug 945834: pt-sift invokes pt-diskstats with deprecated argument
- Fixed bug 945836: pt-sift prints awk error if there are no stack traces to aggregate
- Fixed bug 945842: pt-sift generates wrong state sum during processlist analysis
- Fixed bug 946438: pt-query-digest should print a better message when an unsupported log format is specified

- Fixed bug 946776: pt-table-checksum ignores –lock-wait-timeout
- Fixed bug 940440: Bad grammar in pt-kill docs

v2.0.3 released 2012-02-03

Percona Toolkit 2.0.3 has been released. The development team was very busy last month making this release significant: two completely redesigned and improved tools, pt-diskstats and pt-stalk, and 20 bug fixes.

Both pt-diskstats and pt-stalk were redesigned and rewritten from the ground up. This allowed us to greatly improve these tools' functionality and increase testing for them. The accuracy and output of pt-diskstats was enhanced, and the tool was rewritten in Perl. pt-collect was removed and its functionality was put into a new, enhanced pt-stalk. pt-stalk is now designed to be a stable, long-running daemon on a variety of common platforms. It is worth re-reading the documentation for each of these tools.

The 20 bug fixes cover a wide range of problems. The most important are fixes to pt-table-checksum, pt-iostats, and pt-kill. Apart from pt-diskstats, pt-stalk, and pt-collect (which was removed), no other tools were changed in backwards-incompatible ways, so it is worth reviewing the full changelog for this release and upgrading if you use any tools which had bug fixes.

Thank you to the many people who reported bugs and submitted patches.

Download the latest release of Percona Toolkit 2.0 from http://www.percona.com/software/percona-toolkit/ or the Percona Software Repositories (http://www.percona.com/docs/wiki/repositories:start).

Changelog

- Completely redesigned pt-diskstats
- Completely redesigned pt-stalk
- Removed pt-collect and put its functionality in pt-stalk
- Fixed bug 871438: Bash tools are insecure
- Fixed bug 897758: Failed to prepare TableSyncChunk plugin: Use of uninitialized value $args{"chunk_range"} in lc at pt-table-sync line 3055
- Fixed bug 919819: pt-kill –execute-command creates zombies
- Fixed bug 925778: pt-ioprofile doesn't run without a file
- Fixed bug 925477: pt-ioprofile docs refer to pt-iostats
- Fixed bug 857091: pt-sift downloads http://percona.com/get/pt-pmp, which does not work
- Fixed bug 857104: pt-sift tries to invoke mext, should be pt-mext
- Fixed bug 872699: pt-diskstats: rd_avkb & wr_avkb derived incorrectly
- Fixed bug 897029: pt-diskstats computes wrong values for md0
- Fixed bug 882918: pt-stalk spams warning if oprofile isn't installed
- Fixed bug 884504: pt-stalk doesn't check pt-collect
- Fixed bug 897483: pt-online-schema-change "uninitialized value" due to update-foreign-keys-method
- Fixed bug 925007: pt-online-schema-change Use of uninitialized value $tables{"old_table"} in concatenation (.) or string at line 4330
- Fixed bug 915598: pt-config-diff ignores –ask-pass option

- Fixed bug 919352: pt-table-checksum changes binlog_format even if already set to statement
- Fixed bug 921700: pt-table-checksum doesn't add –where to chunk size test on replicas
- Fixed bug 921802: pt-table-checksum does not recognize –recursion-method=processlist
- Fixed bug 925855: pt-table-checksum index check is case-sensitive
- Fixed bug 821709: pt-show-grants –revoke and –separate don't work together
- Fixed bug 918247: Some tools use VALUE instead of VALUES

v2.0.2 released 2012-01-05

Percona Toolkit 2.0.2 fixes one critical bug: pt-table-sync –replicate did not work with character values, causing an "Unknown column" error. If using Percona Toolkit 2.0.1, you should upgrade to 2.0.2.

Download the latest release of Percona Toolkit 2.0 from http://www.percona.com/software/percona-toolkit/ or the Percona Software Repositories (http://www.percona.com/docs/wiki/repositories:start).

Changelog

- Fixed bug 911996: pt-table-sync –replicate causes "Unknown column" error

v2.0.1 released 2011-12-30

The Percona Toolkit development team is proud to announce a new major version: 2.0. Beginning with Percona Toolkit 2.0, we are overhauling, redesigning, and improving the major tools. 2.0 tools are therefore not backwards compatible with 1.0 tools, which we still support but will not continue to develop.

New in Percona Toolkit 2.0.1 is a completely redesigned pt-table-checksum. The original pt-table-checksum 1.0 was rather complex, but it worked well for many years. By contrast, the new pt-table-checksum 2.0 is much simpler but also much more efficient and reliable. We spent months rethinking, redesigning, and testing every aspect of the tool. The three most significant changes: pt-table-checksum 2.0 does only –replicate, it has only one chunking algorithm, and its memory usage is stable even with hundreds of thousands of tables and trillions of rows. The tool is now dedicated to verifying MySQL replication integrity, nothing else, which it does extremely well.

In Percona Toolkit 2.0.1 we also fixed various small bugs and forked ioprofile and align (as pt-ioprofile and pt-align) from Aspersa.

If you still need functionalities in the original pt-table-checksum, the latest Percona Toolkit 1.0 release remains available for download. Otherwise, all new development in Percona Toolkit will happen in 2.0.

Download the latest release of Percona Toolkit 2.0 from http://www.percona.com/software/percona-toolkit/ or the Percona Software Repositories (http://www.percona.com/docs/wiki/repositories:start).

Changelog

- Completely redesigned pt-table-checksum
- Fixed bug 856065: pt-trend does not work
- Fixed bug 887688: Prepared statements crash pt-query-digest
- Fixed bug 888286: align not part of percona-toolkit

- Fixed bug 897961: ptc 2.0 replicate-check error does not include hostname
- Fixed bug 898318: ptc 2.0 –resume with –tables does not always work
- Fixed bug 903513: MKDEBUG should be PTDEBUG
- Fixed bug 908256: Percona Toolkit should include pt-ioprofile
- Fixed bug 821717: pt-tcp-model –type=requests crashes
- Fixed bug 844038: pt-online-schema-change documentation example w/drop-tmp-table does not work
- Fixed bug 864205: Remove the query to reset @crc from pt-table-checksum
- Fixed bug 898663: Typo in pt-log-player documentation

v1.0.1 released 2011-09-01

Percona Toolkit 1.0.1 has been released. In July, Baron announced planned changes to Maatkit and Aspersa development;[1] Percona Toolkit is the result. In brief, Percona Toolkit is the combined fork of Maatkit and Aspersa, so although the toolkit is new, the programs are not. That means Percona Toolkit 1.0.1 is mature, stable, and production-ready. In fact, it's even a little more stable because we fixed a few bugs in this release.

Percona Toolkit packages can be downloaded from http://www.percona.com/downloads/percona-toolkit/ or the Percona Software Repositories (http://www.percona.com/docs/wiki/repositories:start).

Although Maatkit and Aspersa development use Google Code, Percona Toolkit uses Launchpad: https://launchpad.net/percona-toolkit

[1] http://www.xaprb.com/blog/2011/07/06/planned-change-in-maatkit-aspersa-development/

Changelog

- Fixed bug 819421: MasterSlave::is_replication_thread() doesn't match all
- Fixed bug 821673: pt-table-checksum doesn't include –where in min max queries
- Fixed bug 821688: pt-table-checksum SELECT MIN MAX for char chunking is wrong
- Fixed bug 838211: pt-collect: line 24: [: : integer expression expected
- Fixed bug 838248: pt-collect creates a "5.1" file

v0.9.5 released 2011-08-04

Percona Toolkit 0.9.5 represents the completed transition from Maatkit and Aspersa. There are no bug fixes or new features, but some features have been removed (like –save-results from pt-query-digest). This release is the starting point for the 1.0 series where new development will happen, and no more changes will be made to the 0.9 series.

Changelog

- Forked, combined, and rebranded Maatkit and Aspersa as Percona Toolkit.

Changelog

- Fixed bug 1279502: –version-check behaves like spyware

Changelog

- Fixed bug 1402776: Improved fix (protocol parser fix): error when parsing tcpdump capture with pt-query-digest
- Fixed bug 1632522: pt-osc: Fails with duplicate key in table for self-referencing (Thanks Amiel Marqeta)
- Fixed bug 1654668: pt-summary exists with an error (Thanks Marcelo Altmann)
- New tool : pt-mongodb-summary
- New tool : pt-mongodb-query-digest

Percona Toolkit 3.0.0 RC includes the following changes:

New Features

- Added `pt-mongodb-summary` tool
- Added `pt-mongodb-query-profiler` tool

Bug fixes

- 1402776: Updated `MySQLProtocolParser` to fix error when parsing `tcpdump` capture with `pt-query-digest`
- 1632522: Fixed failure of `pt-online-schema-change` when altering a table with a self-referencing foreign key (Thanks Marcelo Altmann)
- 1654668: Fixed failure of `pt-summary` on Red Hat and derivatives (Thanks Marcelo Altmann)

INDEX

Symbols

P