

Working on bugs with perf & bpftrace

Marcos Albe
Percona University
Montevideo, September 2026

PERCONA

© 2026 Percona

Brief intro: about me

- Marcos Albe
- Support Engineer & CBO
- Mostly MySQL
- Love performance analysis
- Handled one-too-many MySQL bugs

PERCONA®

© 2026 Percona

Brief intro: agenda

- Our Goals
- The Plan
- A demo

PERCONA®

© 2026 Percona

Brief intro: credits

- My dear friend Valerii Kravchuck: I found his article doing exactly what I wanted to do while looking for perf syntax. My brother from another mother!
- Roberto "Beto" Garcia: for the BCC script and listening to my ramblings about C++ inlines

PERCONA

© 2026 Percona



PERCONA

© 2026 Percona

Our Goals

- Investigate hard-to-reproduce bugs through variables lifecycle
- Without changing/recompiling running binary
- Being lightweight AND safe enough to run on production
- Impress customers :-P

PERCONA®

© 2026 Percona

Our Goals: variable lifecycle

- Many bugs are simply about wrong variable value at a given point in the code.
- Attempt to identify sequence of calls leading to incorrect value
- When assertion is hit or incorrect value detected, print all stack traces for the failing thread id

PERCONA®

© 2026 Percona

Our Goals: leave the binaries alone

- Avoids downtime
- Avoids risk of introducing a bug related to changes in the binary
- Changing the binary is often not possible in production environments
- Avoids change-approval in many environments :-)

PERCONA®

© 2026 Percona

Our Goals: lightweight & safe

- Attaching gdb to a live process, in production, is risky business
- Can seriously slow down things, specially if you add too many breakpoints
- eBPF bytecode is statically checked so it should be safe and avoid stalls
    - It WILL cause some performance degradation.
    - You should measure!

PERCONA®

© 2026 Percona

Our Goals: NOT goals

- Being dead easy. Process complexity is proportional to code complexity.
- Solve every problem. Is better to reproduce problem if you can.

PERCONA®

© 2026 Percona



© 2026 Percona

The Plan

- Find variable assignments in code
- Get probes for those points in the code using `perf probe`
- Write `bpftrace`/BCC script to trace probes and collect stacktraces and variable value

PERCONA®

© 2026 Percona

The Plan: find assignments

- Use VS Code to find occurrences
- Note the following for each assignment
  - function
  - file path
  - line number
  - how to reference the variable (struct?)
- Use `funccount` to narrow your list if needed
  - Requires running `funccount` until problem happens again

PERCONA

© 2026 Percona

The Plan: get probes

- `perf probe --list/--vars` using file, line number, function to run
  - we get relative line number
- `perf prob --dry-run` using function name or relative line number, and variable name
  - we get probe name and CPU register to access the variable

PERCONA®

© 2026 Percona

The Plan: write eBPF contraption

- Simple once you have the probes
  - Write a handler for each probe
  - Keep for each assignment:

    `threads[thd_id][timestamp]={value,stack}`
- Prototype with **`bpftrace`**
- Most times will require BCC/Python (or C)
  - Build fake .h files to access data structs
- Painful if you want to distribute… :-(

PERCONA®

© 2026 Percona



PERCONA

© 2026 Percona



© 2026 Percona

- https://perconadev.atlassian.net/browse/PS-11428
-

PERCONA®

© 2026 Percona

A Demo: the problem

A customer was concerned about this error:

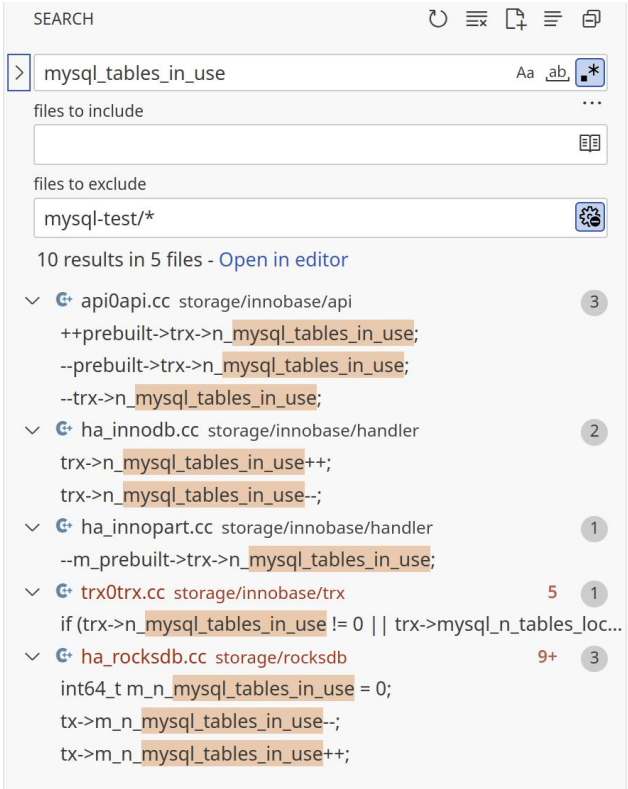
```
2026-03-14T20:16:46.029368-07:00 328248 [ERROR] [MY-013028] [InnoDB]
MySQL is freeing a thd though trx->n_mysql_tables_in_use is 4294967200
and trx->mysql_n_tables_locked is 10084.
TRANSACTION 421919763850984, not started
mysql tables in use 4294967200, locked 10084
```

How do you reproduce that???
The variable is involved in, very much,
EVERYTHING you do with a table.

PERCONA®

© 2026 Percona

A Demo: find the assignments

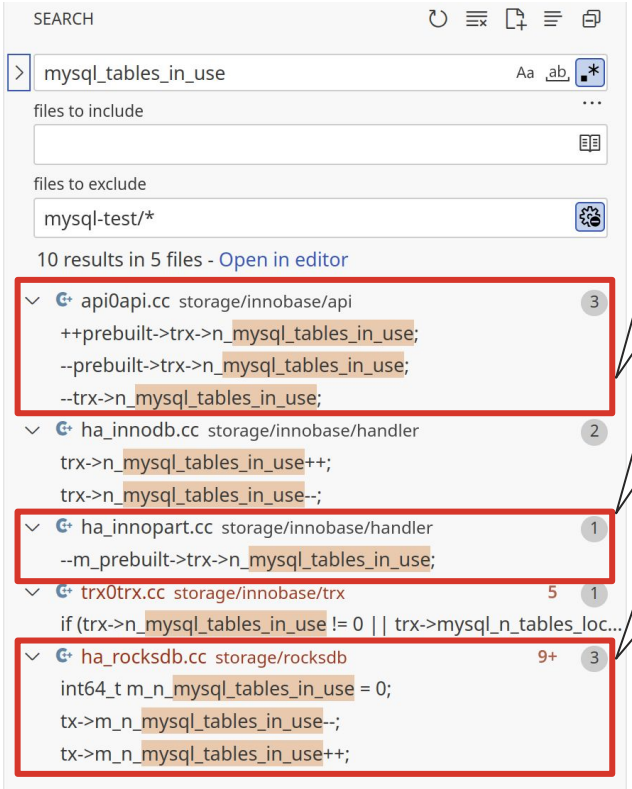


But is only modified in a handful of places...

PERCONA

© 2026 Percona

A Demo: find the assignments



InnoDB access through memcached API is not

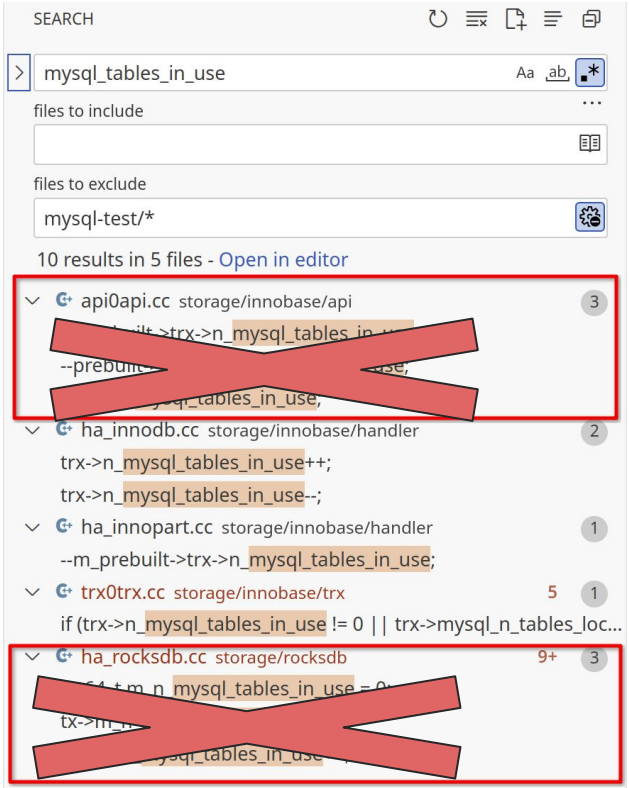
Finally, we don't use partitioned tables either here, so we won't trace this neither.

RocksDB: not in use, so no need to trace it.

PERCONA

© 2026 Percona

A Demo: filter down



InnoDB access through API is not used so no need to track.

Same for RocksDB

PERCONA

© 2026 Percona

A Demo: what will be traced

```
external_lock@storage/innobase/handler/ha_innodb.cc
  19761,12:     trx->n_mysql_tables_in_use++;
  19785,10:   trx->n_mysql_tables_in_use--;

trx_validate_state_before_free@storage/innobase/trx/trx0trx.cc
  578,14:   if (trx->n_mysql_tables_in_use != 0 ...
```

PERCONA®

© 2026 Percona

A Demo: perf probe --line

```
binary=/opt/percona_server/8.4.7/bin/mysqld;
source=/opt/percona_server/src/8.4.7
pp="perf probe --exec=${binary} --source=${source}";

# trx_validate_state_before_free@storage/innobase/trx/trx0trx.cc
$pp --line trx_validate_state_before_free:15; # relative line no.

# external_lock@storage/innobase/handler/ha_innodb.cc
$pp --line storage/innobase/handler/ha_innodb.cc:19747+50 # absolute
$pp --line storage/innobase/handler/ha_innodb.cc:19784+50 # line no.'s
```

PERCONA®

© 2026 Percona

A Demo: perf probe --dry-run --add

```
binary=/opt/percona_server/8.4.7/bin/mysqld;
source=/opt/percona_server/src/8.4.7
pp="sudo perf probe --dry-run -vvv --exec=${binary} --source=${source}";
alias filter="awk '/Probe point found|Writing event/{ print $NF; }'"

# trx_validate_state_before_free@storage/innobase/trx/trx0trx.cc
$pp trx_validate_state_before_free:15 trx |& filter

# external_lock@storage/innobase/handler/ha_innodb.cc
$pp storage/innobase/handler/ha_innodb.cc:19762 trx |& filter
$pp storage/innobase/handler/ha_innodb.cc:19786 trx |& filter
```

PERCONA®

© 2026 Percona

A Demo: the probes and registers

```
# trx_validate_state_before_free@storage/innobase/trx/trx0trx.cc
_ZL30trx_validate_state_before_freeP5trx_t.lto_priv.0+53
trx=%bx:x64

# external_lock@storage/innobase/handler/ha_innodb.cc
_ZN11ha_innobase13external_lockEP3THDi+778
trx=%r14:x64
_ZN11ha_innobase13external_lockEP3THDi+1244
trx=%r14:x64
```

PERCONA®

© 2026 Percona

A Demo: bpftrace prototype

```
sudo bpftrace -p 67394 -e '
uprobe:/opt/percona_server/8.4.7/bin/mysqld:_ZL30trx_validate_state_before_freeP5trx_t.lto_priv.0+53 {
    printf("tid: %lu @ %s: %lu\n", tid, "trx_validate_state_before_free",
           ((struct trx_t *)reg("bx"))->n_mysql_tables_in_use);
}

uprobe:/opt/percona_server/8.4.7/bin/mysqld:_ZN11ha_innobase13external_lockEP3THDi+778 {
    printf("tid: %lu @ %s: %lu\n", tid,
           "ha_innobase::external_lock++ @ 19761 ",
           ((struct trx_t *)reg("r14"))->n_mysql_tables_in_use);
}

uprobe:/opt/percona_server/8.4.7/bin/mysqld:_ZN11ha_innobase13external_lockEP3THDi+1244 {
    printf("tid: %lu @ %s: %lu\n", tid,"ha_innobase::external_lock-- @ 19785 ",
           ((struct trx_t *)reg("r14"))->n_mysql_tables_in_use);
}
'
```

Demo commands and scripts

```
[marcos.albe@support-02|prod bpftrace_demo]$ tail -n +1 *
==> 01--perf_probe_line.sh <==
#!/bin/bash
binary=/opt/percona_server/8.4.7/bin/mysqld;
source=/opt/percona_server/src/8.4.7
pp="perf probe --exec=${binary} --source=${source}";

# trx_validate_state_before_free@storage/innobase/trx/trx0trx.cc
$pp --line trx_validate_state_before_free:15; # relative line no.

# external_lock@storage/innobase/handler/ha_innodb.cc
$pp --line storage/innobase/handler/ha_innodb.cc:19747+50 # absolute
# $pp --line storage/innobase/handler/ha_innodb.cc:19784+50 # line no.'s


==> 02--perf_probe_dry_run_add.sh <==
#!/bin/bash
binary=/opt/percona_server/8.4.7/bin/mysqld;
source=/opt/percona_server/src/8.4.7
pp="sudo perf probe --dry-run -vvv --exec=${binary} --source=${source}";
# alias filter="awk '/Probe point found|Writing event/{ print $NF; }'"
function filter() {  while read data; do { egrep 'Probe point found|Writing event' <<< $data | awk '{print $NF}'; } done; }
```