Linux MySQL Slow Query Tracing with bcc/BPF

04 Oct 2016

My mysqld_qslower tool prints MySQL queries slower than a given threshold, and is run on the MySQL server. By default, it prints queries slower than 1 millisecond:

```
# mysqld_qslower `pgrep -n mysqld`
Tracing MySQL server queries for PID 14371 slower than 1 ms...
TIME(s)        PID          MS QUERY
0.000000       18608   130.751 SELECT * FROM words WHERE word REGEXP '^bre.*n$'
2.921535       18608   130.590 SELECT * FROM words WHERE word REGEXP '^alex.*$'
4.603549       18608    24.164 SELECT COUNT(*) FROM words
9.733847       18608   130.936 SELECT count(*) AS count FROM words WHERE word REGEXP '^bre.*n$'
17.864776      18608   130.298 SELECT * FROM words WHERE word REGEXP '^bre.*n$' ORDER BY word
```

This is a bit like having a custom slow queries log, where the threshold can be picked on the fly.

It is a bcc tool that uses the MySQL USDT probes (user statically defined tracing) that were introduced for DTrace. bcc is a front-end and a collection of tools that use new Linux enhanced BPF tracing capabilities.

USDT support in bcc/BPF is new, and involves allowing BPF code to be attached to USDT probes, eg, from mysqld_qslower:

```
# enable USDT probe from given PID
u = USDT(pid=pid)
u.enable_probe(probe="query__start", fn_name="do_start")
u.enable_probe(probe="query__done", fn_name="do_done")
```

... and then fetching arguments to those USDT probes. This BPF code hashes the timestamp and the query string pointer (from arg1) to the current thread (pid) for later lookup:

```
struct start_t {
    u64 ts;
    char *query;
};
BPF_HASH(start_tmp, u32, struct start_t);

int do_start(struct pt_regs *ctx) {
    u32 pid = bpf_get_current_pid_tgid();
    struct start_t start = {};
    start.ts = bpf_ktime_get_ns();
    bpf_usdt_readarg(1, ctx, &start.query);
    start_tmp.update(&pid, &start);
    return 0;
};
```

The full source to mysqld_qslower is here, and more example output.

The tplist tool from bcc can be used to list USDT probes from a pid or binary. Eg:

```
# tplist -l /usr/local/mysql/bin/mysqld
/usr/local/mysql/bin/mysqld mysql:filesort__start
/usr/local/mysql/bin/mysqld mysql:filesort__done
/usr/local/mysql/bin/mysqld mysql:handler__rdlock__start
/usr/local/mysql/bin/mysqld mysql:handler__rdlock__done
/usr/local/mysql/bin/mysqld mysql:handler__unlock__done
/usr/local/mysql/bin/mysqld mysql:handler__unlock__start
/usr/local/mysql/bin/mysqld mysql:handler__wrlock__start
/usr/local/mysql/bin/mysqld mysql:handler__wrlock__done
/usr/local/mysql/bin/mysqld mysql:insert__row__start
/usr/local/mysql/bin/mysqld mysql:insert__row__done
/usr/local/mysql/bin/mysqld mysql:update__row__start
/usr/local/mysql/bin/mysqld mysql:update__row__done
/usr/local/mysql/bin/mysqld mysql:delete__row__start
/usr/local/mysql/bin/mysqld mysql:delete__row__done
/usr/local/mysql/bin/mysqld mysql:net__write__start
/usr/local/mysql/bin/mysqld mysql:net__write__done
/usr/local/mysql/bin/mysqld mysql:net__read__start
/usr/local/mysql/bin/mysqld mysql:net__read__done
/usr/local/mysql/bin/mysqld mysql:query__exec__start
/usr/local/mysql/bin/mysqld mysql:query__exec__done
/usr/local/mysql/bin/mysqld mysql:query__cache__miss
/usr/local/mysql/bin/mysqld mysql:query__cache__hit
/usr/local/mysql/bin/mysqld mysql:connection__start
/usr/local/mysql/bin/mysqld mysql:connection__done
/usr/local/mysql/bin/mysqld mysql:select__start
/usr/local/mysql/bin/mysqld mysql:select__done
/usr/local/mysql/bin/mysqld mysql:query__parse__start
/usr/local/mysql/bin/mysqld mysql:query__parse__done
/usr/local/mysql/bin/mysqld mysql:command__start
/usr/local/mysql/bin/mysqld mysql:command__done
/usr/local/mysql/bin/mysqld mysql:query__start
/usr/local/mysql/bin/mysqld mysql:query__done
/usr/local/mysql/bin/mysqld mysql:update__start
/usr/local/mysql/bin/mysqld mysql:update__done
/usr/local/mysql/bin/mysqld mysql:multi__update__start
/usr/local/mysql/bin/mysqld mysql:multi__update__done
/usr/local/mysql/bin/mysqld mysql:delete__start
/usr/local/mysql/bin/mysqld mysql:delete__done
/usr/local/mysql/bin/mysqld mysql:multi__delete__start
/usr/local/mysql/bin/mysqld mysql:multi__delete__done
/usr/local/mysql/bin/mysqld mysql:insert__start
/usr/local/mysql/bin/mysqld mysql:insert__done
/usr/local/mysql/bin/mysqld mysql:insert__select__start
/usr/local/mysql/bin/mysqld mysql:insert__select__done
/usr/local/mysql/bin/mysqld mysql:keycache__read__block
/usr/local/mysql/bin/mysqld mysql:keycache__read__miss
/usr/local/mysql/bin/mysqld mysql:keycache__read__done
/usr/local/mysql/bin/mysqld mysql:keycache__read__hit
/usr/local/mysql/bin/mysqld mysql:keycache__read__start
/usr/local/mysql/bin/mysqld mysql:keycache__write__block
/usr/local/mysql/bin/mysqld mysql:keycache__write__done
/usr/local/mysql/bin/mysqld mysql:keycache__write__start
/usr/local/mysql/bin/mysqld mysql:index__read__row__start
/usr/local/mysql/bin/mysqld mysql:index__read__row__done
/usr/local/mysql/bin/mysqld mysql:read__row__start
/usr/local/mysql/bin/mysqld mysql:read__row__done
```

You can also use "readelf -n .../mysqld" to double check. Applications that have these probes typically need to be compiled with --with-dtrace or --enable-dtrace on Linux for them to be included in the binary.

You can comment here, but I can't guarantee your comment will remain here forever: I might switch comment systems at some point (eg, if disqus add advertisements).