如何分析 mysqld crash 的原因

原创 Fander 芬达的学习笔记 2022年06月22日 20:54 广东

今天中午公司某业务系统 mysql crash 了，重启 mysqld 后业务恢复了。业务同事找到我，说需要分析 mysqld crash 的原因。

MySQL 的 err.log 如下:

```
2022-06-22T12:37:36.675314Z 170605886 [Note] Aborted connection 170605886 to db: '' user:
ation packets)
12:37:55 UTC - mysqld got signal 11 ;
This could be because you hit a bug. It is also possible that this binary
or one of the libraries it was linked against is corrupt, improperly built,
or misconfigured. This error can also be caused by malfunctioning hardware.
Attempting to collect some information that could help diagnose the problem.
As this is a crash and something is definitely wrong, the information
collection process might fail.

key_buffer_size=402653184
read_buffer_size=2097152
max_used_connections=6001
max_threads=6000
thread_count=253
connection_count=251
It is possible that mysqld could use up to
key_buffer_size + (read_buffer_size + sort_buffer_size)*max_threads = 61913278 K  bytes o
Hope that's ok; if not, decrease some variables in the equation.


Thread pointer: 0x7f6c8a416140
Attempting backtrace. You can use the following information to find out
where mysqld died. If you see no messages after this, something went
terribly wrong...
stack_bottom = 7f74bd2cee70 thread_stack 0x30000
mysqld(my_print_stacktrace+0x3b)[0xf1102b]
mysqld(handle_fatal_signal+0x461)[0x7c3861]
/lib64/libpthread.so.0(+0xf130)[0x7f77081bf130]
mysqld(_Z16digest_add_tokenP16sql_digest_statejP7YYSTYPE+0x81)[0xca9bd1]
mysqld(_ZN16Lex_input_stream16add_digest_tokenEjP7YYSTYPE+0x1d)[0xcb8b5d]
mysqld(_Z8MYSQLlexP7YYSTYPEP7YYLTYPEP3THD+0x130)[0xcbb1a0]
mysqld(_Z10MYSQLparseP3THD+0x9dc)[0xdaf85c]
mysqld(_Z9parse_sqlP3THDP12Parser_stateP19Object_creation_ctx+0x123)[0xce2c73]
mysqld(_Z11mysql_parseP3THDP12Parser_state+0x1cd)[0xce319d]
mysqld(_Z16dispatch_commandP3THDPK8COM_DATA19enum_server_command+0xa7a)[0xce3e7a]
mysqld(_Z10do_commandP3THD+0x19f)[0xce58bf]
mysqld(handle_connection+0x288)[0xda5438]
mysqld(pfs_spawn_thread+0x1b4)[0x12861a4]
/lib64/libpthread.so.0(+0x7df3)[0x7f77081b7df3]
/lib64/libc.so.6(clone+0x6d)[0x7f7706c753dd]

Trying to get some variables.
...
```

说实话，分析这个东西还真有点难度。

是OOM kill吗？

最常见的 crash 其实是 OOM kill，那么这次是 OOM kill 吗？不是！因为 err.log 里写明了"mysqld got signal 11"，而 OOM kill，其实是"signal 9"，也就是 kill -9，这种情况 MySQL 会直接挂掉，而没有办法保留堆栈信息的。

是服务器扛不住导致的吗？

从堆栈信息中"connection_count=251" 和 内存使用总值不是，从监控上看也不是，监控数值和历史没有很大的区别。

那就是bug咯？

我认为很有可能，一边让业务回顾最近有没有做过什么变更，crash 之前有没有做过什么特殊操作，我一边去分析 bug 去了。

如何分析bug？

我们先来看看堆栈

我们拿到了崩溃位置 0xca9bd1，如何找到与之相对的代码位置呢？找台测试机，获取对应版本的安装包。我们的版本是 5.7.18，下载两个包，一个是二进制安装包，一个是源码包。

解压二进制安装包后，用 gdb 打开 mysqld，大概方法是

```
gdb ${解压后的二进制包}/bin/mysqld
```

在 0xca9bd1 位置打一个断点，大概就会有如下信息:

```
(gdb)b *0xca9bd1
Breakpoint 1 at 0xca9bd1: file ${源码路径}/sql/sql_digest.cc，line: 379
(gdb)
```

我们可以看到，gdb 将崩溃位置的文件名和行号都打印出来，剩下的事情，就可以交给开发工程师，按照这个崩溃堆栈来进行问题排查。

以上步骤参考爱可生开源社区的黄炎的专栏《MySQL一问一实验》里的《第25问：MySQL 崩溃了，打印了一些堆栈信息，怎么读？》，有兴趣的直接关注他们公众号看原文，我没必要做文字的搬运工了。

下面我说一下我是怎么做的？

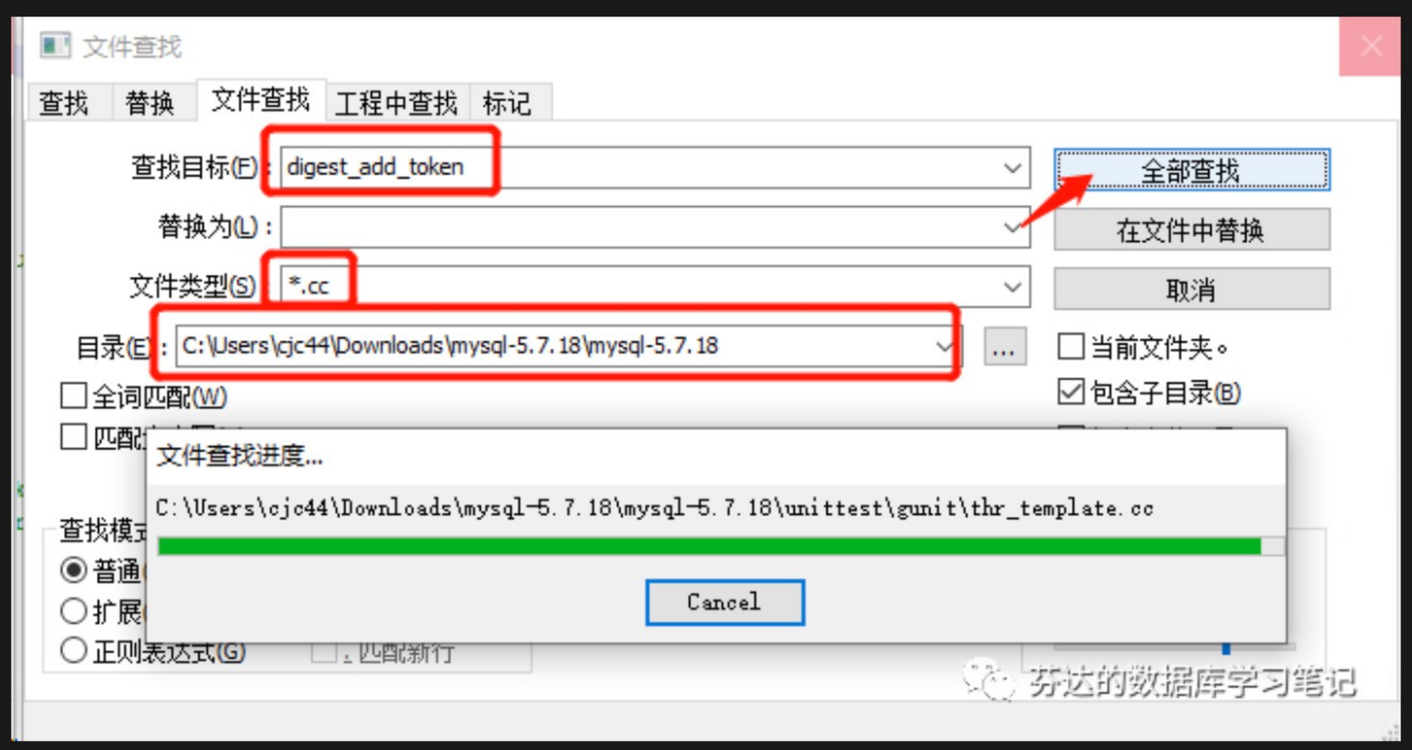
因为我没有源码分析能力，实际上，我并没有用 gdb 定位到具体哪个文件，而我是用 notepad++ 定位的！首先我已经知道了崩溃点在这个位置

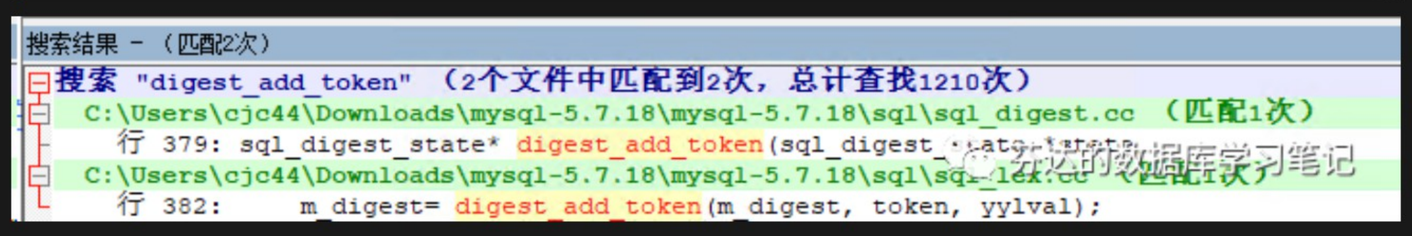
```
mysqld(_Z16digest_add_tokenP16sql_digest_statejP7YYSTYPE+0x81)[0xca9bd1]
```

用 c++filt 解析就是

```
[root@fander ~]# c++filt _Z16digest_add_tokenP16sql_digest_statejP7YYSTYPE
digest_add_token(sql_digest_state*, unsigned int, YYSTYPE*)
```

我们能发现 digest_add_token 是个函数，然后我用 notepad++ 遍历搜索源码文件，如下图:



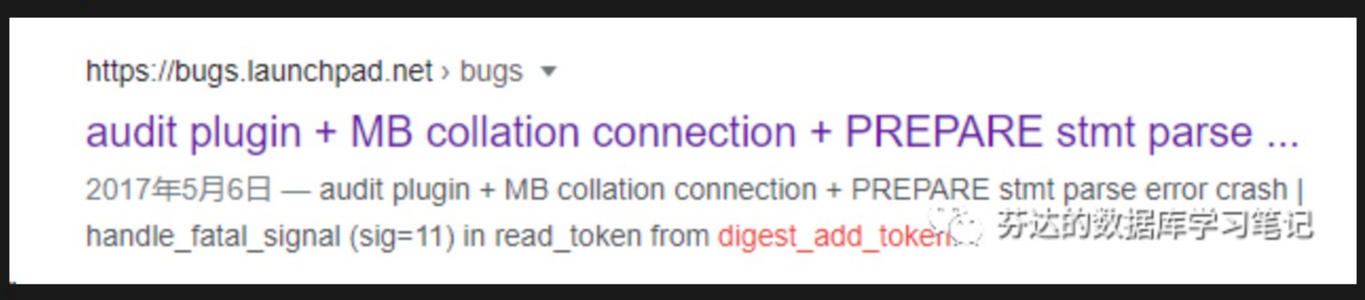


最后只发现两个源码文件涉及这部分，能定位到具体的行。那么我就"双击"直接跳转到定位的地方。

- 看源码函数的注释，还有结合 baidu (实际不是用 baidu，用什么大家也懂)搜索这个函数干嘛的。
- 源码在 sql 目录，能大概知道是 sql 相关的问题。

当然了，我只是从源码能有个大概了解。然后：

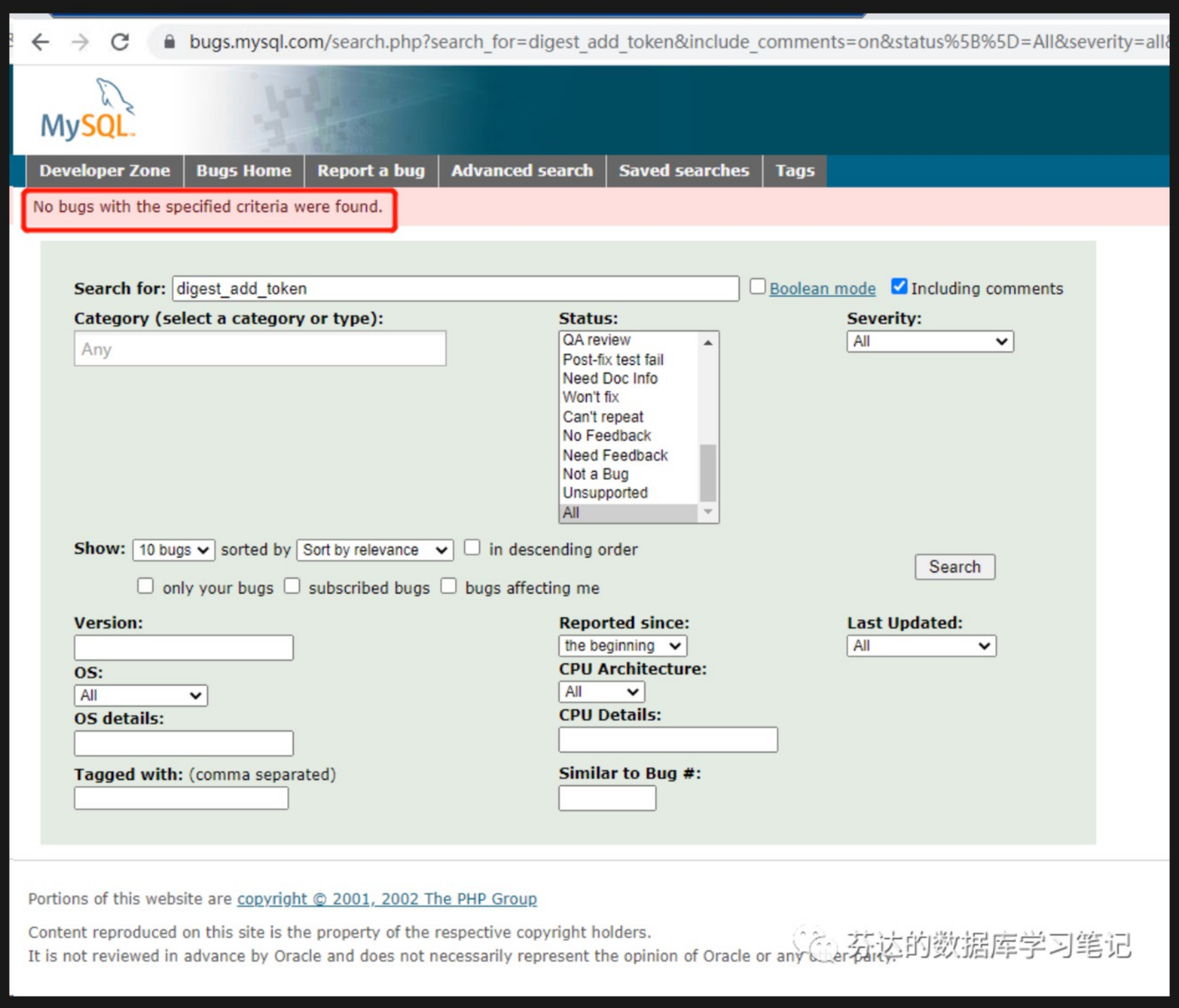
1. 我以 "digest_add_token" 为关键字 baidu 搜索



发现了什么？

- 淘宝内核月报有源码分析过 digest_add_token，这兄弟大概率有 bug 啊
- Percona 官方发现过 digest_add_token 相关 bug，并且在 percona server 5.7.18 版本修复了。尤其是 percona 这个分析，根本和我们遇到的场景一模一样，无论版本号还有错误日志。
- 腾讯 TXSQL 修复过 digest_add_token 相关 bug，说这个函数会导致 overflow 我从 baidu 没有搜到 mysql 官方的相关 bug，于是我直接去 mysql 的 bug report 网站直接搜索了

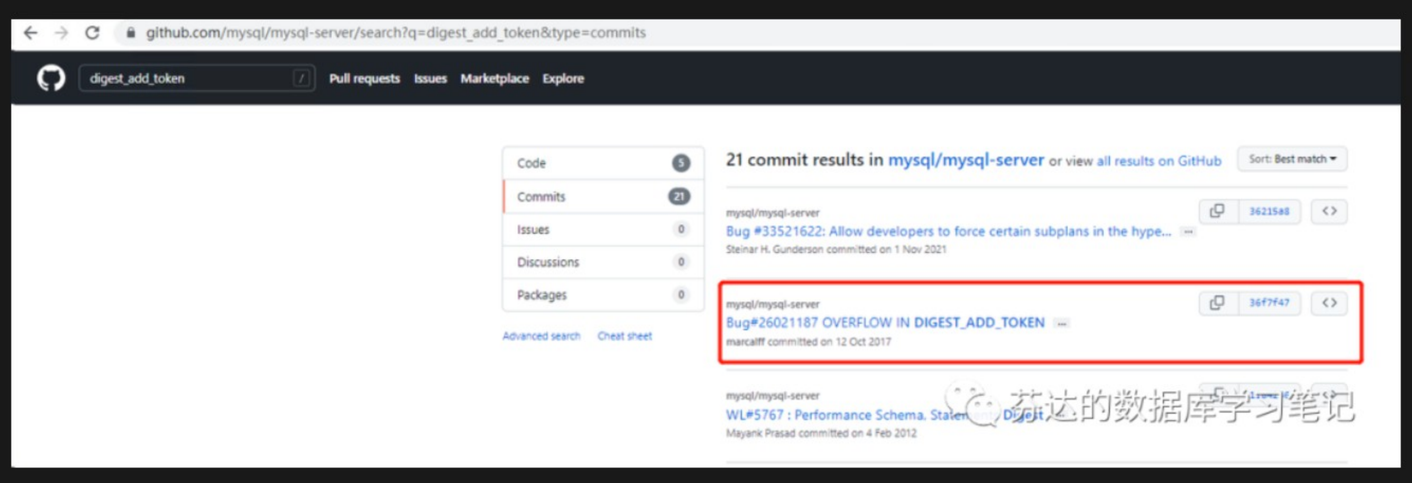
2. 我以 "digest_add_token" 为关键字，在 bugs.mysql.com 搜索。



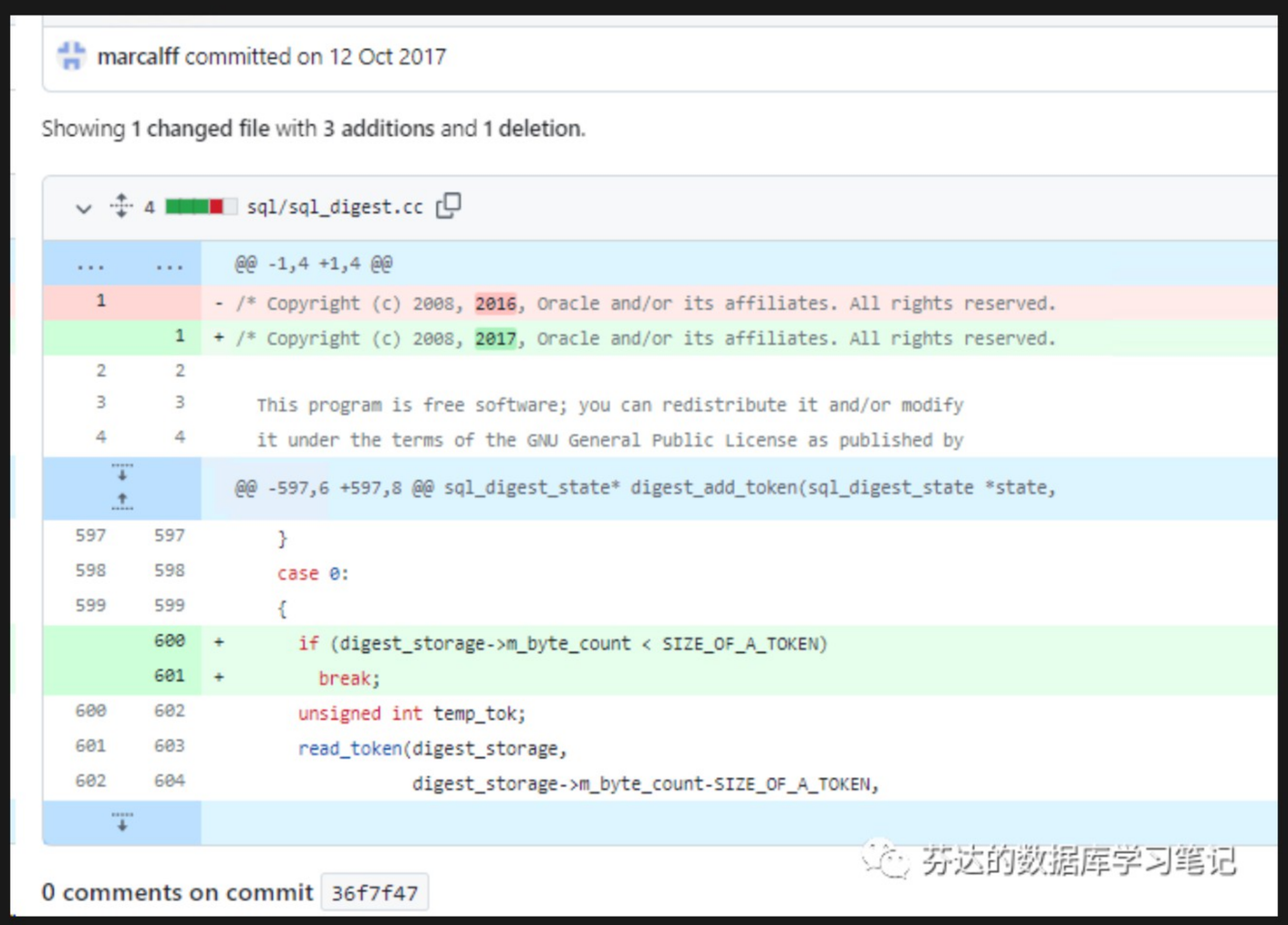
很遗憾，我完全搜不到。

> 当时心想，不厚道啊。。难道 MySQL 的分支兄弟们修复完 bug，没有提交给官方？

3. 我以 "digest_add_token" 为关键字，在 github 开源官方仓库搜索

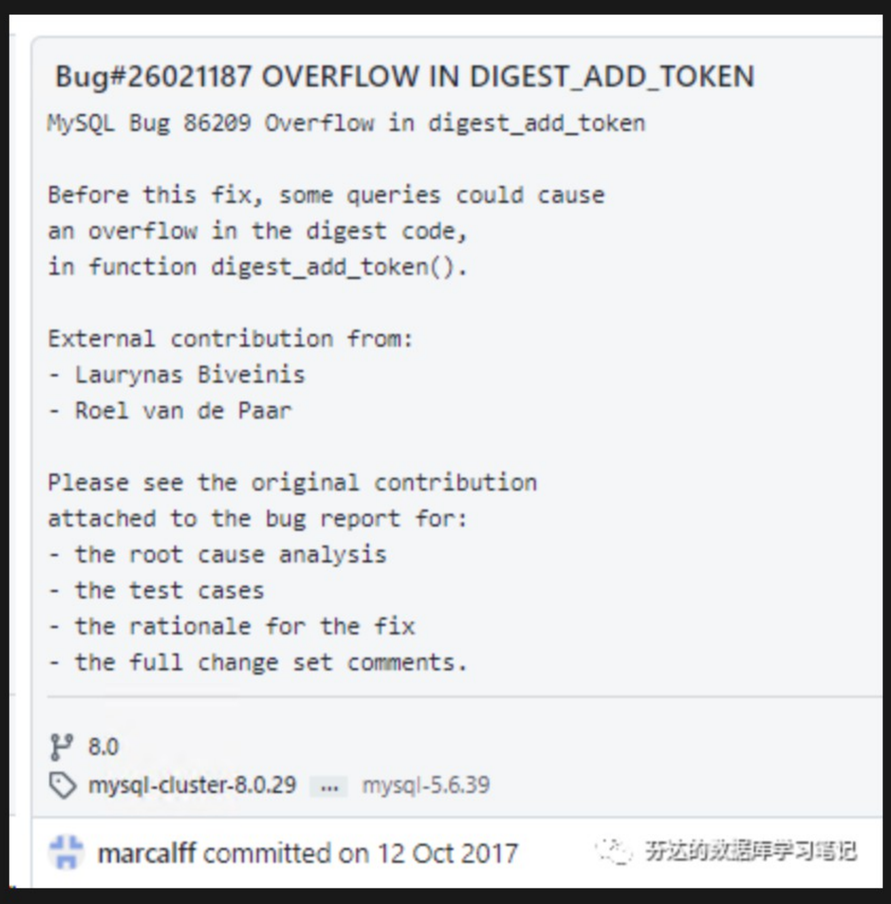


这次我找到了，这个 bug 的描述和我在 percona 网站看的描述是一样的。



官方的修复方法比 percona 晚了 5 个月，但修复的方法是一模一样。

> 实际上是官方不厚道哈。。别人修的 bug，复制粘贴后说自己修复。

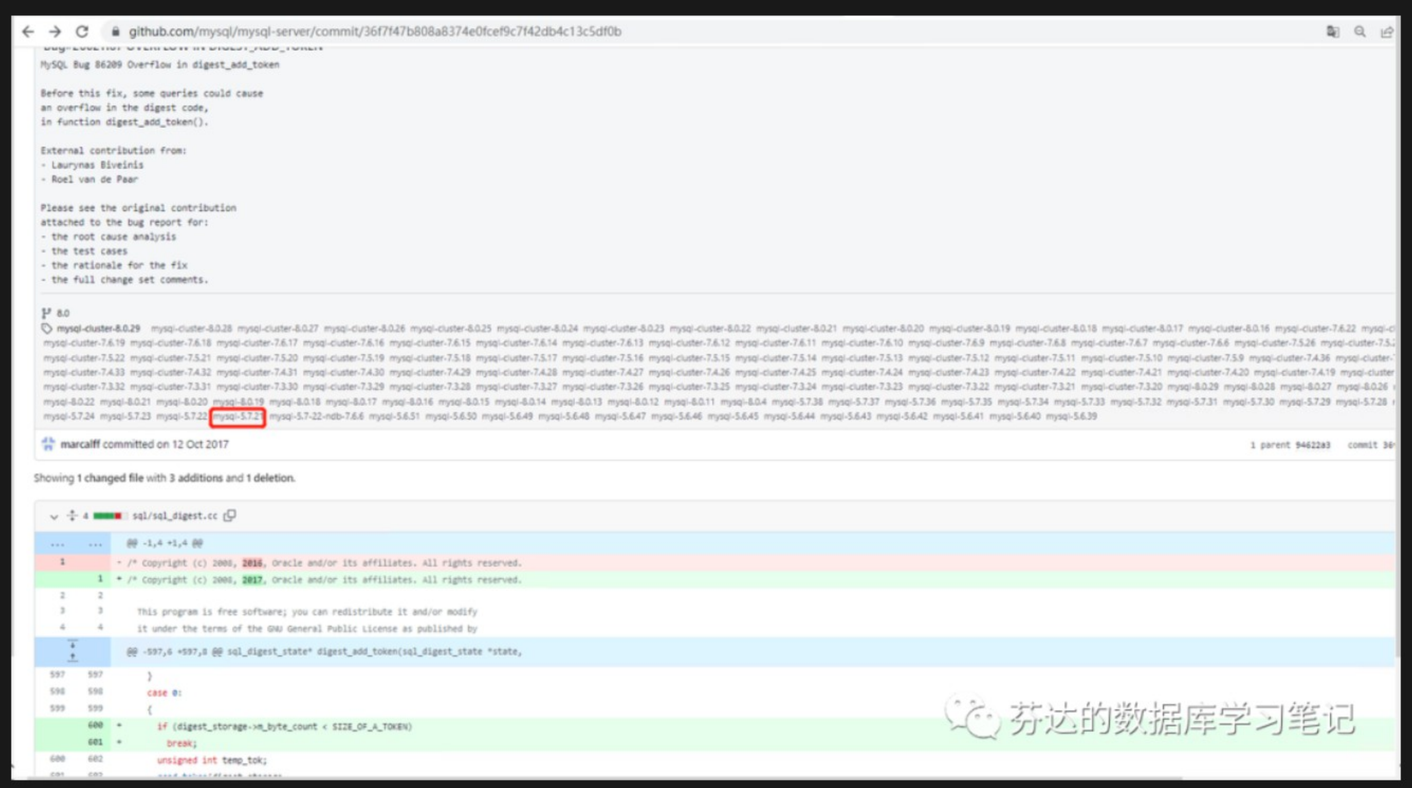


从 bug 的编号看，他是一个内部 bug，"Bug#26021187"，这种长编号的就是内部发现的 bug，在 bug report 网站是搜索不到的，这也是他的 bug 标题虽然有"digest_add_token"关键字，我也搜索不到的原因。但这里又有个短 bug，我的理解是，有可能有外部有人发现和提交过这个 bug (这也正常，percona 早 5 个月就修复了，所以发现那就是更早的事了。)，但官方一直很"狗"，如果他们内部也发现了，他们会把外部的 bug 编号关闭，说和内部 bug 编号重复，导致我们外部人无法查到这个 bug。

如图，我以外部 bug 编号 86209 搜索也搜索不到。

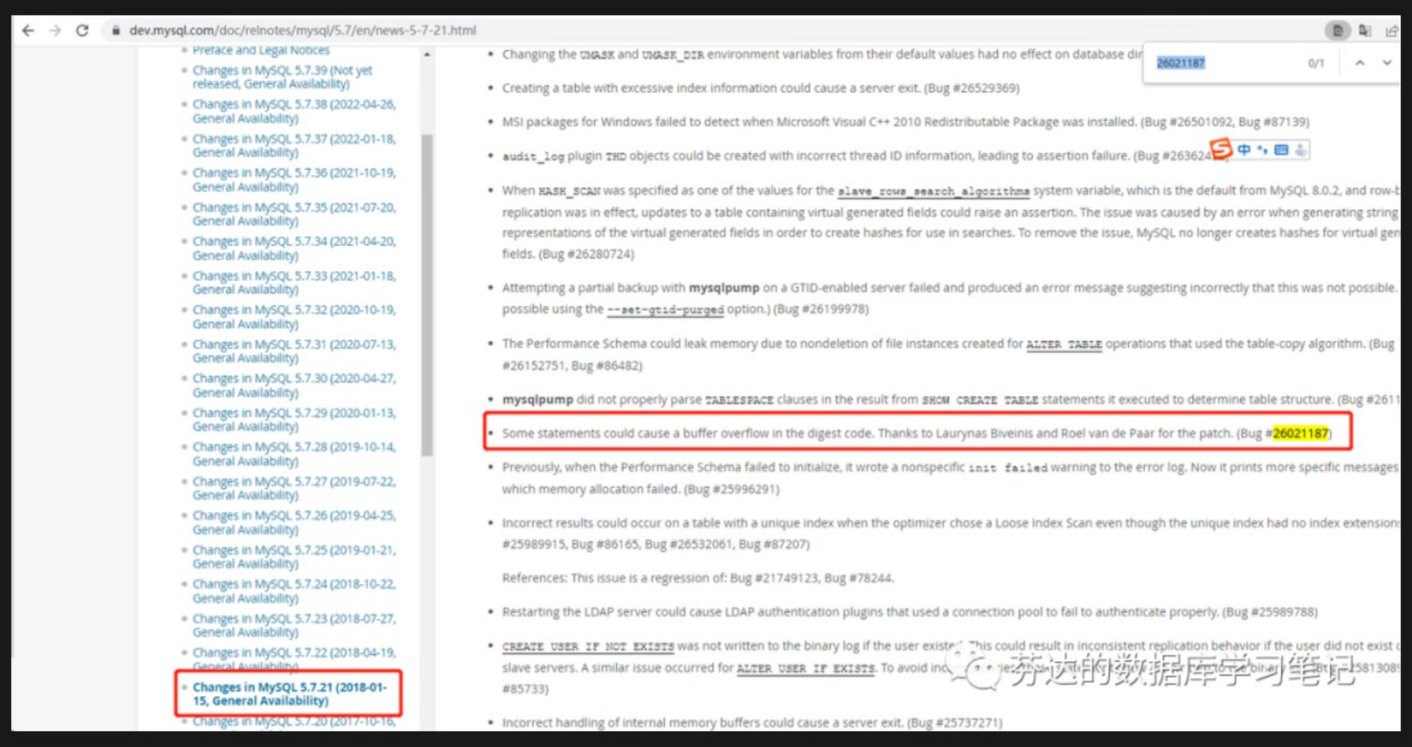
看这个报错，Oracle 是把这个外部 bug 关闭和隐藏了。如果 bug 编号不存在应该下面这个报错。

现在我们知道 oracle 修复了这个 bug 了，那么他是哪个版本修复呢？



还是 github 上的那个页面，这里能看到最早在 5.7.21 修复了，然后我们去 release note 里可以再次确认。

https://dev.mysql.com/doc/relnotes/mysql/5.7/en/news-5-7-21.html



> Some statements could cause a buffer overflow in the digest code. Thanks to Laurynas Biveinis and Roel van de Paar for the patch. (Bug #26021187)

Bug 是如此描述：某些语句可能会导致摘要代码中的缓冲区溢出。

Bug 确实和 sql 有关，我们故障时涉及的具体哪个 sql 我仍然不知道，因为官方很"狗"，不给我看具体 Bug 的描述。

无论如何，我找到了 mysqld crash 的原因，希望大家从我的文章学到知识。

Enjoy MySQL!

mysql 61 # 数据库 38

mysql · 目录

‹ 上一篇
为什么要避免使用“CREATE TABLE AS SELECT”语句

下一篇 ›
全球首发！MySQL 5.7.39、8.0.30、8.0.31 更新预告

修改于2022年06月22日