117 案例实战：千万级数据删除导致的慢查询优化实践（2）

案例实战：千万级数据删除导致的慢查询优化实践（2）

好，今天我们继续讲解这个案例，在当时这个案例的场景发生之后，也就是针对某个表的大量简单的单行数据查询SQL变成慢查询，我们先排查了SQL执行计划以及MySQL服务器负载，发现都没有问题。

此时就必须用上一个SQL调优的利器了，也就是**profiling**工具，这个工具可以对SQL语句的执行耗时进行非常深入和细致的分析，使用这个工具的过程，大致如下所示

首先要打开这个profiling，使用set profiling=1这个命令，接着MySQL就会自动记录查询语句的profiling信息了。

此时如果执行show profiles命令，就会给你列出各种查询语句的profiling信息，这里很关键的一点，就是他会记录下来每个查询语句的query id，所以你要针对你需要分析的query找到对他的query id，我们当时就是针对慢查询的那个SQL语句找到了query id。

然后就可以针对单个查询语句，看一下他的profiling具体信息，使用show profile cpu, block io for query xx，这里的xx是数字，此时就可以看到具体的profile信息了

除了cpu以及block io以外，你还可以指定去看这个SQL语句执行时候的其他各项负载和耗时，具体使用方法，大家自行网上搜索就行了，并不难。

他这里会给你展示出来SQL语句执行时候的各种耗时，比如磁盘IO的耗时，CPU等待耗时，发送数据耗时，拷贝数据到临时表的耗时，等等吧，反正SQL执行过程中的各种耗时都会展示出来的。

这里我们当时仔细检查了一下这个SQL语句的profiling信息，重点发现了一个问题，他的Sending Data的耗时是最高的，几乎使用了1s的时间，占据了SQL执行耗时的99%，这就很坑爹了。

因为其他环节耗时低是可以理解的，毕竟这种简单SQL执行速度真的很快，基本就是10ms级别的，结果跑成了1s，那肯定Sending Data就是罪魁祸首了！

这个Sending Data是在干什么呢?

MySQL的官方释义如下：为一个SELECT语句读取和处理数据行，同时发送数据给客户端的过程，简单来说就是为你的SELECT语句把数据读出来，同时发送给客户端。

可是为什么这个过程会这么慢呢？profiling确实是提供给我们更多的线索了，但是似乎还是没法解决掉问题。但是毕竟我们已经捕获到了第一个比较异常的点了，就是Sending Data的耗时很高！请大家记住这个线索。

有时候针对MySQL这种复杂数据库软件的调优过程，就跟福尔摩斯破案一样，你要通过各种手段和工具去检查MySQL的各种状态，然后把有异常的一些指标记下来，作为一个线索，当你线索足够多的时候，往往就能够汇总大量的线索整理出一个思路了，那也就是一个破案的时刻了！

接着我们又用了一个命令：**show engine innodb status**，看一下innodb存储引擎的一些状态，此时发现了一个奇怪的指标，就是history list length这个指标，他的值特别高，达到了上万这个级别。

这里我们给大家解释一下这个指标，当然如果大家自己在调优的时候发现了类似的情况，不知道一个指标什么意思，直接google一下就可以了，很快就会查到，这里我们直接给大家一个结论了。

大家应该还记得之前我们讲解过的MVCC机制吧？MVCC机制，说穿了就是多个事务在对同一个数据，有人写，有人读，此时可以有多种隔离级别，这个大家应该还记得吧。

至于这个MVCC和隔离级别的实现原理，跟一个Read View机制是有关系的，同时还有一个至关重要的机制，就是数据的undo多版本快照链条。

你必须对一个数据得有一个多版本快照链条，才能实现MVCC和各种隔离级别，这个具体的原理，我们这里不多说了，大家有遗忘的，建议回看之前的文章。

所以当你有大量事务执行的时候，就会构建这种undo多版本快照链条，此时history list length的值就会很高。然后在事务提交之后，会有一个多版本快照链条的自动purge清理机制，只要有清理，那么这个值就会降低。

一般来说，这个值是不应该过于高的，所以我们在这里注意到了第二个线索，history list length值过高！大量的undo多版本链条数据没被清理！推测很可能就是有的事务长时间运行，所以他的多版本快照不能被purge清理，进而导致了这个history list length的值过高！

第二个线索Get！基本可以肯定的一点是，经过两个线索的推测，在大量简单SQL语句变成慢查询的时候，SQL是因为Sending Data环节异常耗时过高，同时此时出现了一些长事务长时间运行，大量的频繁更新数据，导致有大量的undo多版本快照链条，还无法purge清理。

但是这两个线索之间的关系是什么呢？是第二个线索推导出的事务长时间运行现象的发生，进而导致了第一个线索发现的Sending Data耗时过高的问题吗？可是二者之间的关系是什么呢？是不是还得找到更多的线索还行呢？

大家别着急，到此为止，大家就跟看侦探小说一样，福尔摩斯已经找到了一些线索，但是似乎还缺少一些关键线索，把所有线索都串起来，进而去让他形成一个完善的破案推理，真相即将大白了，咱们下次继续讲。

End