链滴

# MySQL 性能分析

作者：zouchanglin

原文链接：https://ld246.com/article/1568720703134

来源网站：链滴

<h2 id="前提知识">前提知识</h2>
Mysql 中有专门负责优化 SELECT 语句的优化器模块，主要功能：通过计算分析系统中收集到的计信息，为客户端请求的 Query 提供他认为最优的执行计划(他认为最优的数据检索方式，但不见得是 DBA 认为是最优的)
当客户端向 MySQL 请求一条 Query，命令解析器模块完成请求分类，区别出是 SELECT 并转发 MySQL Query Optimizer 时， MySQL Query Optimizer 首先会对整条 Query 进行优化，处理掉些常量表达式的预算直接换算成常量值。并对 Query 中的查询条件进行简化和转换，如去掉一些无或显而易见的条件、结构调整等。然后分析 Query 中的 Hint 信息(如果有)，看显示 Hint 信息是否以完全确定该 Query 的执行计划。如果没有 Hint 或 Hint 信息还不足以完全确定执行计划，则会读所涉及对象的统计信息，根据 Query 进行写相应的计算分析，然后再得出最后的执行计划。
<h2 id="MySQL常见性能瓶颈">MySQL 常见性能瓶颈</h2>
CPU：CPU 在饱和的时候一般发生在数据装入内存或从磁盘上读取数据时候
IO：磁盘 I/O 瓶颈发生在装入数据远大于内存容量的时候
服务器硬件的性能瓶颈：top、free、 iostat 和 vmstat 来查看系统的性能状态
<h2 id="Explain-执行计划">Explain-执行计划</h2>
用 EXPLAIN 关键字可以模拟优化器执行 SQL 查询语句，从而知道 MySQL 是如何处理你的 sqL 语句的。分析你的查询语句或是表结构的性能瓶颈！
用法：explain + SQL
<h3 id="explain能干啥-">explain 能干啥？</h3>
表的读取顺序
数据读取操作的操作类型
哪些索引可以使用
哪些索引被实际使用
表之间的引用
每张表有多少行被优化器查询
<h3 id="执行计划包含的信息">执行计划包含的信息</h3>
<img src="https://b3logfile.com/file/2019/09/mysql01-82b618bc.png" alt="mysql01.png">
下面解释一下这些字段是干啥的：
<h4 id="1-id">1、id</h4>
select 查询的序列号，包含一组数字，表示查询中执行 select 子句或操作表的顺序。有三种情：
1、id 相同，执行顺序由上至下
<img src="https://b3logfile.com/file/2019/09/mysql02-c89152f3.png" alt="mysql02.png">
2、id 不同，如果是子查询，id 的序号会递增，id 值越大优先级越高，越先被执行
<img src="https://b3logfile.com/file/2019/09/mysql03-f8ed841d.png" alt="mysql03.png">
PRIMARY 就是主查询，id 越大越先被执行，则毫无疑问在这样的子查询中，肯定最先要查询的就是 table_03，所以这是 MySQL 自己约定的顺序，就和运算符优先级一样！SUBQUERY 代表最外查询。
3、id 相同不同，同时存在
<img src="https://b3logfile.com/file/2019/09/mysql04-5297affc.png" alt="mysql04.png">
id 如果相同，可以认为是一组，从上往下顺序执行；在所有组中，id 值越大，优先级越高，越执行，衍生=DERIVED，所以这里就体现了 explain 的第一个用处，那就是它能看出表的读取和加载序！
<h4 id="2-select-type">2、select_type</h4>
SIMPLE、PRIMARY、SUBQUERY、DERIVED、UNION、UNION RESULT、
查询的类型，主要是用于区别普通查询、联合查询、子查询等的复杂查询
SIMPLE：简单的 select 查询，查询中不包含子查询或者 UNION
PRIMARY：查询中若包含任何复杂的子部分，最外层查询则被标记为 PRIMARY
SUBQUERY：在 SELECT 或 WHERE 列表中包含了子查询
DERIVED：在 FROM 列表中包含的子查询被标记为 DERIVED(衍生)，MySQL 会递归执行这些

查询，把结果放在临时表里。
UNION：若第二个 SELECT 出现在 UNION 之后，则被标记为 UNION；若 UNON 包含在 FRO
子句的子查询中，外层 SELECT 将被标记为: DERIVED
UNION RESULT：从 UNION 表获取结果的 SELECT
<img src="https://b3logfile.com/file/2019/09/mysql05-a7ab6b91.png" alt="mysql05.png
>
所以这样就看到了数据读取操作的操作类型
<h4 id="3-table">3、table</h4>
这个基本不用说，显示这一行的数据是关于哪张表的
<h4 id="4-type">4、type</h4>
type 显示的是访问类型，是较为重要的一个指标，结果值从最好到最坏依次是:
system > const > eq_ref > ref > fulltext > ref_or_null > index_merge &g
;unique_subquery >index _subquery > range > index > ALL
一般来说，得保证查询至少达到 range 级别，最好能达到 ref
<img src="https://b3logfile.com/file/2019/09/mysql06-82ad9b2f.png" alt="mysql06.png
>
这样查询的类型就是 ALL，如果数据达到百万级别的一个全表扫描，那么性能肯定会下降的！</
>
常见的 type 就这些：system > const > eq_ref > ref > range > index > AL

<ul>
<li>
system 表只有一行记录(等于系统表)，这是 const 类型的特例，平时不会出现，这个也可以忽
不计
</li>
<li>
const 表示通过索引一次就找到了，const 用于比较 primary key 或者 unique 索引。因为只匹
一行数据，所以很快。如将主键至于 where 列表中，MySQL 就能将该查询转换为一个常量，比如 w
ere id = 1 就被当成是常量
<img src="https://b3logfile.com/file/2019/09/mysql07-84180922.png" alt="mysql07.png
>
</li>
<li>
eq_ref 唯一性索引，对于每个索引键，表中只有一条记录与之匹配，常见于主键或唯一索引扫

<img src="https://b3logfile.com/file/2019/09/mysql08-9d2fc28c.png" alt="mysql08.png
>
</li>
<li>
ref 非唯一性索引扫描，返回匹配某个单独值的所有行本质上也是一种索引访问，它返回所有匹
某个单独值的行。然而，能会找到多个符合条找和扫描的混合，与 eq_ref 对比，其实可以发现 ref
对非唯一索引进行扫描，其实也就是对数据表的非主键字段建立索引，然后通过这个索引进行扫描，
果可想而知肯定是非唯一性的！
</li>
<li>
range 只检索给定范围的行，使用一个索引来选择行。key 列显示使用了哪个索引，一般就是
你的 where 语句中出现 between、<、>、in 等的查询，这种范围扫描索引比全表扫描要好，
为它只需要开始于索引的某一点，而结束于另一点，不用扫描全部索引。
<img src="https://b3logfile.com/file/2019/09/mysql09-7e460cb9.png" alt="mysql09.png
>
<img src="https://b3logfile.com/file/2019/09/mysql10-8a4de568.png" alt="mysql10.png
>
</li>
<li>

index  Full Index scan，index 与 ALL 区别为 index 类型只遍历索引树。这通常比 ALL 快，因索引文件通常比数据文件小 (也就是说虽然叫 all 和 index 都是读全表,但 index 是从索引中读取的，而 all 是从硬盘中读的
<img src="https://b3logfile.com/file/2019/09/mysql11-6595ae51.png" alt="mysql11.png">
</li>
<li>
all  FullTable scan，将遍历全表以找到匹配的行
</li>
</ul>
<h4 id="5-possible-keys">5、possible_keys</h4>
显示可能应用在这张表中的索引，一个或多个。
查询涉及的字段上若存在索引，则该索引将被列出，但不一定被查询实际使用</stron>
<h4 id="6-key">6、key</h4>
实际使用的索引。如果为 null 则没有使用索引
查询中若使用了覆盖索引，则索引和查询的 select 字段重叠。
所以 possible_keys 和 key 实际上是告诉了使用者，MySQL 理论上会用到哪些索引，实际上会到哪些索引。
<img src="https://b3logfile.com/file/2019/09/mysql12-416251a6.png" alt="mysql12.png">
这就是覆盖索引，也就是如果你要查询的字段顺序正好与索引建立的顺序相等，那么查询类型那接变为 index 查询，而不是 ref 类型！
<h4 id="7-key-len">7、key_len</h4>
表示索引中使用的字节数，可通过该列计算查询中使用的索引的长度。在不损失精确性的情况下长度越短越好
key_len 显示的值为索引最大可能长度，并非实际使用长度，即 key_len 是根据表定义计算而得不是通过表内检索出的
<img src="https://b3logfile.com/file/2019/09/mysql13-365b43a9.png" alt="mysql13.png">
这个其实比较容易理解，精度越高，需要的索引字节数也变长了，很显然，更高精度的查询付出代价就是 key_len 变长了！
<h4 id="8-ref">8、ref</h4>
显示索引那一列被使用了，如果可能的话，是一个常数。那些列或常量被用于查找索引列上的值比如:
<img src="http://b3logfile.com/file/2019/09/mysql04-5297affc.png" alt="">
<img src="https://b3logfile.com/file/2019/09/mysql14-6a73004c.png" alt="mysql14.png">
所以通过这里我们已经知道哪些索引可以使用，哪些索引被实际使用！
<h4 id="9-rows">9、rows</h4>
显示索引那一列被使用了，如果可能的话，是一个常数。那些列或常量被用于查找索引列上的值越少越好！
<img src="https://b3logfile.com/file/2019/09/mysql15-93c0b162.png" alt="mysql15.png">
通过上面的例子进行分析，在没建立索引前，扫描了 641 行，先加载 t2，再加载 t1，查询 t2 的时候是 ALL，也就是逐行扫描，并且本来可以使用主键索引，但是依旧没什么意义，所以还是以逐扫描的方式进行查询，这样会有 640 行结果。接下来对 t2 表建立复合索引，所以可供选择的索引就主见索引、自己新建的复合索引。很明显，新建复合索引之后呢，查询的时候不再采用逐行扫描的方，而是选择了复合索引，行数降到了 142 行，总共 143 行就可以搞定这个查询问题！
<h4 id="10-Extra">10、Extra</h4>
额外的，扩展的。包含不适合在其他列中显示但十分重要的额外信息！
① Using filesort
说明 mysql 会对数据使用一个外部的索引排序，而不是按照表内的索引顺序进行读取。MySQL 无法利用索引完成排序操作成为"文件排序"，这种文件排序是需要尽量避免的。

② Using temporary
使用了临时表保存中间结果，MySQL 在对查询结果排序时使用临时表。常见于排序 order by 分组查询 group by。这个动作更耗费时间，如果说 Using filesort 是九死一生的话那么出现 Using t mporary 是十死无生了
<img src="https://b3logfile.com/file/2019/09/mysql16-5ba8d128.png" alt="mysql16.png">
通过上面的例子我们应该明白，如果建立了索引，那么在进行 group by 的时候应该按照建立索 的顺序使用到索引，否则会造成 Using filesort 和 Using temporary。
③ Using index
表示相应的 select 操作中使用了覆盖索引（Coveing Index）,避免访问了表的数据行，效率不 ！如果同时出现 using where，表明索引被用来执行索引键值的查找；如果没有同时出现 using wher ，表面索引用来读取数据而非执行查找动作。
索引覆盖
覆盖索引( Covering Index)，一说为索引覆盖。
理解方式一：就是 select 的数据列只用从索引中就能够取得，不必读取数据行，MySQL 可以利 索引返回 select 列表中的字段，而不必根据索引再次读取数据文件。换句话说查询列要被所建的索引 盖！
理解方式二：索引是高效找到行的一个方法，但是一般数据库也能使用索引找到一个列的数据， 此它不必读取整个行。毕竟索引叶子节点存储了它们索引的数据；当能通过读取索引就可以得到想要 数据，那就不需要读取行了。一个素引包含了(或覆盖了)满足查询结果的数据就叫做覆盖索引
注意：
如果要使用覆盖索引，一定要注意 select 列表中只取出需要的列，不可 <code>select *</code ，因为如果将所有字段一起做索引会导致索引文件过大，查询性能下降
④ Using where
表明使用了 where 过滤
⑤ Using join buffer
使用了连接缓存，Join 查询动作平凡，数据量大的时候就需要增大 join buffer 的容量
⑥ impossible where
where 子句的值总是 false，不能用来获取任何元组
<img src="https://b3logfile.com/file/2019/09/mysql17-eb210e9b.png" alt="mysql17.png">
比如这样的 MySQL 不能理解的 SQL。
⑦ select tables optimized away
在没有 GROUPBY 子句的情况下，基于索引优化 MIN/MAX 操作或者，对于 MyISAM 存储引 优化 <code>COUNT(*)</code> 操作，不必等到执行阶段再进行计算，查询执行计划生成的阶段 完成优化。
⑧ distinct
优化 distinct，在找到第一匹配的元组后即停止找同样值的工作
<h3 id="小练习">小练习</h3>
<img src="https://b3logfile.com/file/2019/09/mysql18-596305c5.png" alt="mysql18.png">
执行顺序是：
第一行(执行顺序 4)：id 列为 1，表示是 union 里的第一个 select，select type 列的 primary 示该查询为外层查询。tabe 列被标记为 <code><derived3></code>，表示查询结果来自一 衍生表，其中 derived3 中 3 代表该查询衍生自第三个 select 查询，即 id 为 3 的 select【select d1 ame.....】
第二行(执行顺序 2)：id 为 3，是整个查询中第三个 selec 的一部分。因查询包含在 from 中， 以为 derived【 select id, name from t1 where other_column="】
第三行(执行顺序 3)：select 列表中的子查询 select type 为 subquery，为整个查询中的第二个 elect【 select id from t3】
第四行(执行顺序 1)：select type 为 unon，说明第四个 select 是 unon 里的第二个 select，最 执行【 select name, id from t2】
第五行(执行顺序 5)：代表从 union 的临时表中读取行的阶段，table 列的 < unIon1,4> 示用第一个和第四个 select 的结果进行 union 操作。【两个结果 union 操作】