基于 MySql 多表分页查询优化技术①

韩 兵, 王照清, 廖联军

(北方工业大学 计算机学院, 北京 100144)

摘 要: 针对MySql在多表分页查询中由于数据量大导致效率无法满足系统要求, 本文经过深入研究和分析发现,常用的表 union 操作需要耗费大量的时间, 成为影响数据库访问性能提高的关键因素. 为了提升数据库类似操作的查询效率和用户体验, 提出一种新的多表查询算法, 该算法通过精确定位分表的技术避免多表的 union 操作以及使用合理的索引来提高查询效率. 通过大量的实验证实, 优化后的查询算法在实现大数据量的多表 union 分页查询操作中效率有明显的提高.

关键词: MySql; 查询; 优化; 索引

Multi-Table Paging Query Optimization Technology in MySql

HAN Bing, WANG Zhao-Qing, LIAO Lian-Jun

(Computer Institute, North China University of Technology, Beijing 100144, China)

Abstract: Under large amount of data the union operation makes data paging very slow, how to improve the performance of data paging efficiency becomes an import issue for MySql. This paper proposes a new data paging algorithm from analysis of union operation in MySql data paging method's advantages and disadvantages, and the key factors affects the speed of the data paging. By avoiding union operation in multi-table query and using indexes it improves query efficiency and reasonable. The experimental results confirms that the optimized query methods in pagination operations could significantly improve the speed.

Key words: MySql; query; optimization; index

随着互联网技术的迅猛发展, 大数据应用成为软件应用的主流, 随着时间和业务的发展, 数据库中的表会越来越多, 表中的数据量也会越来越大, 在未进行分表的情况下, 数据操作, 增/删/改/查的开销也会越来越大; 最终数据库所能承载的数据量、数据处理能力都将遭遇瓶颈,最终将导致系统在响应时间和吞吐量等关键指标不断下降. 利用数据分表存储技术成为当前解决数据量不断大量增加情况下数据存储和提高查询效率的主要方法. 在这种情况下, 当多表联合查询操作频繁的时候, 多表关联查询在分页显示速度的快慢将直接影响系统应用的性能和网络服务的质量[15].

MySql 作为关系型数据库的代表, 由于具有高性能、易部署、易使用, 存储数据方便等特点, 使其在各种应用系统中被大量使用[12]. 通常情况下,系统在设计时会针对超过 500 万条以上数据量的表进行分表,利用多表联合查询时使用 union 技术进行数据提取, 这种查询方法在分表数量和并发访问用户量较小时, 响应速度较快. 但是当分表数量比较大时, union 操作会变得很慢导致用户无法忍受. 因而对 MySql 分表情况下多表分页查询操作进行改进和优化, 使其能在大量数据查询效率方面得到提升, 获得良好的用户体验成为当前亟待解决的关键问题[7-10].

1 传统多表分页查询技术

1.1 union 多表分页查询

在当前基于数据业务的应用中, 分页查询是典型

① 基金项目:北京市自然科学基金(4131001);北京市属高等学校创新团队建设与教师职业发展计划(IDHT20130502)
收稿时间:2015-12-20;收到修改稿时间:2016-01-18 [doi:10.15888/j.cnki.csa.005292]

的查询操作[3], 常规做法是使用前后页预读机制来提高效率, 但是在大数据量的情况下仍然无法满足用户的要求. 因为 MySql 在查询多表数据并进行分页显示时, 首先统计出各个表中所有符合条件的记录总数, 以计算出总页面用于分页面的显示,然后根据偏移量和页面大小返回需要显示的数据.

通过分析 MySql 执行计划可以看出数据库引擎在分页返回数据时按照以下的方式进行处理: 首先按照查询条件全表扫描各个分表符合条件的记录, 把每个分表符合条件的记录通过 union 合成一张临时表; 然后根据页面参数确定数据的偏移量 offset, 最终找到用户要求显示页面的首记录位置, 并按照页面要求显示记录数量的 PageSize 返回该页面的数据明细. 例如: 假设在一个用户日常上网习惯的个性分析系统中, 待查询数据库中记录用户日常数据的单表约定数据量为 1000 万, 按天分表且建有复合索引(time, userid), 当查询某一用户在三天内的网页浏览记录时, 假定页面偏移量为 0, 每页显示 20 条记录. 那么, 下面的过程就是分表情况下多表查询返回特定页数据的 sql 语句.

(1) 统计出符合条件的总记录数

```
Select count(*) from (
(Select * from t1 where time>'00:00:00' and time<'13:00:00' and
userid=' 03717800736350518')  union
(Select * from t2 where time>'00:00:00' and time<'13:00:00' and
userid=' 03717800736350518')  union
(Select * from t3 where time>'00:00:00' and time<'13:00:00' and
userid=' 03717800736350518')
) tmp
```

(2) 返回指定数量的数据

```
Select *  from (
(Select * from t1 where time>'00:00:00' and time<'13:00:00' and userid=' 03717800736350518') union
(Select * from t2 where time>'00:00:00' and time<'13:00:00' and userid=' 03717800736350518') union
(Select * from t3 where time>'00:00:00' and time<'13:00:00' and userid=' 03717800736350518')
) tmp  limit 0,20
```

Sql 的执行计划如图 1 所示, 统计总记录数时使用 union 方法时数据库引擎先根据条件扫描三张表找出符合条件的记录数, 然后再将三张表符合条件的记录合成一张临时表 tmp, 最后统计出符合条件的记录总数[6]. 返回指定数量的数据时数据库引擎也是先根据条件扫描三张表找出符合条件的记录数, 然后再将三张表符合条件的记录合成一张临时表 tmp, 最终根据 limit 关键字后面的页面偏移量和页面大小返回指定的数据给用户. 这种方法能很容易地实现分表情况下多表分页查询.

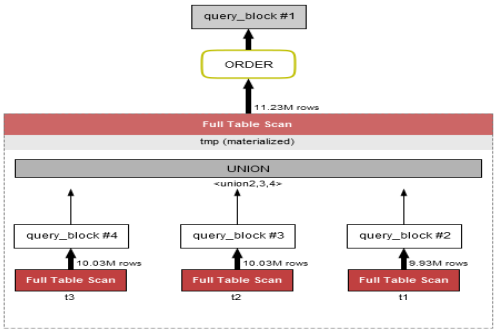

图 1　利用 union 方法的 sql 语句执行计划

1.2 不足之处

在三张表的前提下, 针对不同单表数据量进行查询实验分析, 当数据量增大时 union 操作的耗时分布情况如图 2 所示:

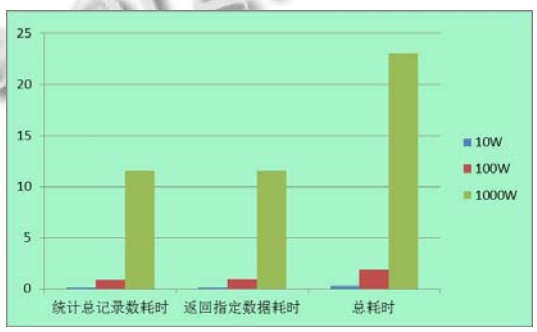

图 2　不同数据量级 union 操作效率

由图 2 可以看出, MySql 常用的 union 分页查询方法, 在单表数据量为 10 万条时查询耗时 0.227s;当单表数据量增加到 100 万时数据查询耗时 1.861s;当单表数据量增加到 1000 万时查询耗时 23.010s. 通过分析图 2 的柱状图可以得出以下结论: 随着分表数据量的成倍增加,union 分页查询方法所耗的时间会成指数级的增

加, 这样严重影响了用户体验,因此需要对 union 分页查询方法加以改进.

1.3 原因分析

通过分析 union 方法分页查询时 sql 语句的执行计划可以看出:MySql 存储引擎首先对三张表进行全表扫描,从而得出符合条件的记录数;然后将各个表返回的符合条件的记录数合成一张临时表[14]. 当分表数据成倍增加时,那么 MySql 存储引擎需要扫描的数据也会成倍的增加, 这将导致查询性能急剧下降.

为了进一步找出 union 方法耗时的原因, 经过对系统查询调用过程分析可以看出:使用 union 时 sql 语句的查询策略是系统对 3 张表进行了 5 次的全表扫描, 一共扫描了近 3000 万条数据,同时使用 I/O 操作创建了 2 张临时表, 一共耗时 11.53 秒. Sql 查询状态如图 3 所示:

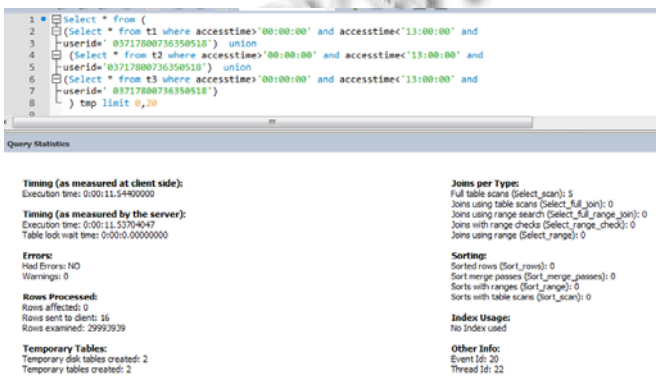

图 3 3000 万数据查询策略状态图

从图 3 可以看出时间消耗的原因在于: 虽然服务器仅向客户端返回了十六条数据(Rows sent to client 16), 却扫描了三张表近三千万行的数据(Rows examined 29993939), 而且参与了相对于内存操作效率极低的磁盘 I/O 操作, 以此来创建 2 张临时表(Temporary disk tables created 2).

此外从图 1 可以看出 t1、t2、t3 虽然建有索引, 但是由于查询并未使用到指定的索引字段, 所以也进行了全表扫描并返回了表的全部数据. 当把上述 sql 语句的查询条件 time 的范围缩小到一定范围时, t1、t2、t3 将会利用索引进行部分扫描, 返回部分数据. 例如把 time 的范围改成 00: 00: 00<time and time <02:00:00, 执行计划如图 4 所示

从上述分析可以看出当查询范围比较大时, 分表查询将不会使用索引而是进行全表扫描, 返回了分表全部的数据而不是过滤后的数据, 这将导致查询操作比较耗时, 而且还占用了大量的内存空间. 此外再从 union 后的临时数据表中返回指定的数据也将消耗大量的时间. 即使每次查询虽然只返回 20 条数据, 也要全部扫描每一张分表, 因此导致了 union 方法的查询效率过于低下.

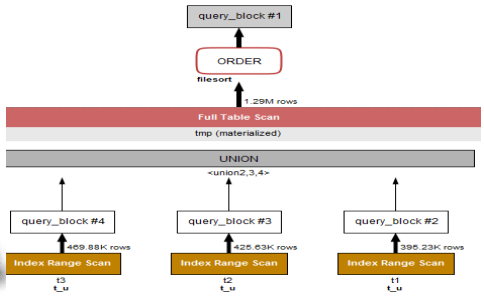

图 4 查询范围缩小后执行计划图

2 多表分页查询算法

通过上述实验数据和性能分析可以得出以下结论: 在使用 union 方法的多表分页查询过程中, 时间主要消耗在了全表扫描和 union 操作上, 如果能找到一种方法避免 union 操作并且减少分表扫描的的个数, 将会在很大程度上提高查询效率. 通过对应用的场景分析发现, 每次只需要根据页面传过来的偏移量返回指定大小的数据量即可. 根据以上分析, 用户所需内容可能只在第一张分表里和其它的分表无关, 因此没有必要扫描其它表, 更没有必要做 union 操作, 此时直接从第一张分表内获取指定偏移量指定大小的数据直接返回给用户即可[1,2].

通过对以上的实验深入研究和分析, 在针对大数据量的分表查询时, 系统会传给指定条件范围以及其它查询条件(如: 本算法按时间分表). 本算法先根据指定的条件确定查询所要涉及的分表, 再根据查询条件算出各个表中符合条件的记录数; 然后将各个分表中符合条件的记录数相加得到符合条件的总记录数, 从而根据页面大小计算记录总页数. 根据用户页面传过来的页数计算偏移量, 通过偏移量和页面的大小以及各个表中的符合条件的记录数做比较, 得出需要返回给用户的数据存在于哪张表中, 然后从该表中返回给用户指定偏移量以及指定数量的记录即可. 本算法通过逻辑判断精确地得出了数据所在的表, 使每次查询只需要查询一张表. 在极端的情况下每个表只有少量的数据需要查询, 多张表才能获取一页的数据, 通

过采用精准的单表查询过滤符合条件的数据返回给用户. 由于当符合条件的数据量比较小时执行计划会合理利用内存操作以获得较高的查询效率. 通过以上分析可以看出, 本算法避免了不必要的多表扫描和union. 下面是三张表的情况下本算法的步骤:

(1) 根据用户的查询时间得出查询所要涉及的表, 假设为 t1、t2、t3.

(2) 根据用户查询条件从各个表中查询出符合条件的记录数, 假设为 count1、count2、count3,并求出三个表中符合条件的记录总数 total=count1+ count2+ count3.

(3) 根据页面传过来的 pagesize 求出总页数 totalpage=total/pagesize.

(4) 根据页面传过来的 pagenum 求出偏移量 offset=(pagenum-1)*pagesize.

(5) 创建集合 rs.

(6) if(offset+pagesize<=count1)
{结果集全部落在第一张表里,查询 t1 将符合条件的数据存入结果集 rs}
else
if(offset<count1&&offset+pagesize>count1&&offset+pagesize<count1+count2)
{结果集部分落在 t1 中部分落在 t2 中,查询 t1,t2 将符合条件的数据存入结果集 rs}
else
if(offset<count1&&offset+pagesize>count1+count2)
{结果集分别落在 t1,t2,t3 中,查询 t1,t2,t3 将符合条件的数据存入结果集 rs}
else if(offset>count1&&offset+pagesize<=count1+count2){结果集落在 t2 中, 查询 t2 将符合条件的数据存入结果集 rs}
else
if(offset>count1&&offset+pagesize>count1+count2)
{结果落在 t2,t3 中,查询 t2,t3 将符合条件的数据存入结果集 rs}
else (offset>=count1+count2)
{结果将落在第三张表中,查询 t3 将符合条件的数据存入结果集 rs}

(7) 返回总页数 totalpage 和数据集 rs.

3 实验验证与分析

为了验证上述算法的可行性, 本文在另一业务系统中针对系统日志为例进行试验, 系统中日志数据按月分表, 假定每次查询最多涉及三个月的记录, 页面最大显示为 20 条记录. 分别测试分表数据量 10 万、100 万、1000 万时, 传统查询多表查询方法与上述多表查询算法所耗时间, 结果如图 6 所示.

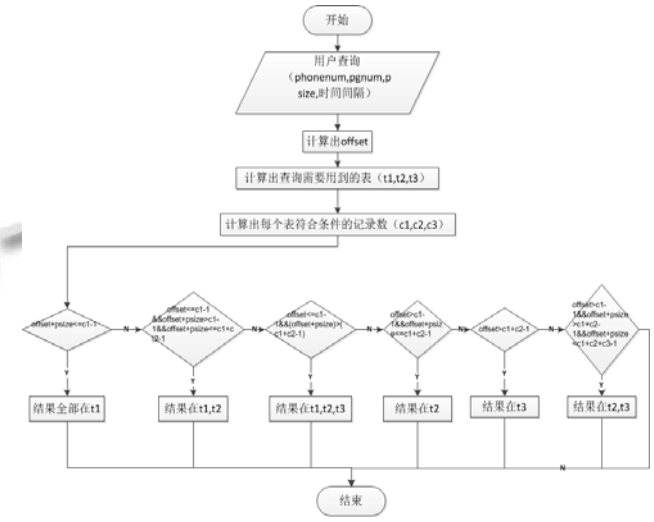

图 5 算法流程图

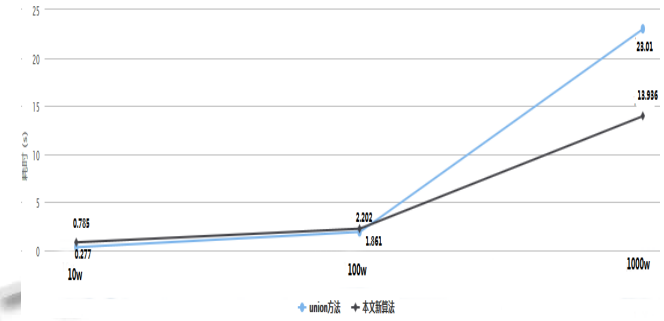

图 6 耗时对比结果

从图 6 可以看出, 当数据量在 100 万以下时 union 方法的查询效率略高于本文的新算法. 当数据量大于 100 万时, 本文新算法的查询效率开始高于 union 方法. 可以看出当数据量较大时本文新算法的效率明显高于 union 方法.

4 索引的使用

在当前 web 应用中, 分页是典型的查询操作, 速度很大程度上取决于索引的使用[5]. 为了更好的提升效率, 可在上述试验系统的日志数据表中合理创建复

合索引. 通过业务分析可知, 当后台管理员查询某段时间内所有人的日志记录时涉及一个字段, 查询某段时间内指定用户的日志记录时涉及两个字段. 因此查询既有可能涉及一个字段, 也有可能涉及两个字段. 根据索引的特点, 这时建立复合索引比较适合. 以 t1 表为例, 应该创建复合索引 key t_u(time, userid). 根据复合索引的左缀原则, 在查询的过程中应该调整 sql 语句, 避免无法使用索引的情况. 下面是复合索引(t_u)在各种查询语句中的使用情况[4].

(1) 使用 order 时可能使用到复合索引的情况

--order by time

--order by time,userid

--order by time desc,userid desc

(2) 使用 where 和 order 可能使用到复合索引的情况

--where time=const order by userid

--where time=const and userid=const roder by time,userid

(3) 使用 order 无法使用复合索引的情况

--order by time asc,userid desc

--where userid=const order by time

--where time=const order by time,other

通过上述分析可以看出复合索引的使用必须按照一定的顺序. 当复合索引的前缀丢失、混合排序的方向不一致、排序中有其它字段时, MySql 数据库并不能使用索引. 此外, 中间字段缺失也将导致复合索引不可用, 例如 key test(a,b,c),where a=const order by c 语句也不能使用索引.

MySql 的查询优化方式简单直接, 在建立索引之后可以很好的提升性能. 正确有效的使用索引也十分重要, 否则有可能导致索引不可用. 部分情况下当一个表中建立的索引比较多时, MySql 数据库引擎会在重复的选择索引时造成一定的开销, 导致查询速度变慢. 所以在某些应用场景下, 可以固定的使用一个索引以达到最好的性能. 管理员可以使用 USEINDEX 关键字指定需要使用的索引, 避免数据库引擎做不必要的尝试, 从而提高查询效率[13].

5 结语

为了提高大数据量多表分页查询效率, 实现 MySql 查询的稳定性和可靠性, 本文通过对传统多表 union 方法执行过程的研究以及对分页查询特点的分析, 找出了 union 方法的不足, 并在此基础上提出了一种新的基于分页的多表查询算法, 通过逻辑判断避免多表扫描及 union 操作, 根据页面偏移量快速定位查询所要涉及的表, 从而有效避免不必要的操作, 提高了查询效率. 实验结果表明, 本文提出的多表分页查询算法行之有效, 算法性能明显优于现有的多表 union 方法. 此外本文还研究了如何有效的使用索引进一步的提高查询效率.

参考文献

1 吴沧舟,兰逸正,张辉.基于 MySql 数据库的优化.电子科技,2013,26(9).

2 谷伟.基于 MySql 的查询优化技术研究.微型电脑应用,2013,30(7).

3 李辉,王瑞波.多条件分页查询优化的设计方法.计算机工程,2010,36(2):51–52,55.

4 Yahoo. Efficient pagination using MySql. Percona Performance Conference, 2009

5 张士军,陆海轮.索引在 MySql 查询优化中的应用.计算机与数字工程,2007,35(1).

6 孙辉.MySql 查询优化的研究和改进[硕士学位论文].武汉:华中科技大学,2007.

7 陈国旗.Web 的财务查询系统的性能优化与实现.中国计量学院学报,2006,17(3).

8 许平格.数据库管理系统中查询优化的设计与实现[硕士学位论文].杭州:浙江大学,2005.

9 何源,戴小鹏,张林峰.数据查询优化算法综述.计算机与现代化,2005,116.

10 周斌.基于 sql 语句优化数据查询.计算机系统应用,2005,14(11):82–85.

11 周东平.关系数据库查询优化的研究与实现[硕士学位论文].南京:南京航空航天大学,2002.

12 Axmark D, Larsson A, Widenius M, et al. MySql5.0 Reference Manual, first ecition. Sweden, 2006: 56–200.

13 李昶,余立人.数据库应用系统性能与数据查询优化,现代计算机,2002,35.

14 杨庚,章韵.关系数据库 sql 语言查询过程分析和优化设计.计算机工程与应用,1999,(11):87–89.

15 任志波,边小帆.B/S 模式下查询方法性能分析.河北大学学报,2001,(1):83–84.