MongoDB 中数据分页优化技术①

王振辉[1], 王振铎[2]

[1](西安翻译学院 工程技术学院, 西安 710105)

[2](西安思源学院 电子信息工程学院, 西安 710038)

摘 要: 针对 MongoDB 在大数据量情况下, 内置的 skip 操作会变的很慢, 数据库的分页查询效率成为影响数据库访问性能提高的重要问题. 从分析 MongoDB 内置的 skip-limit 分页方法的优点和缺点及影响分页查询速度的关键因素入手, 提出一种新的 where-limit 数据分页方法. 通过改变查询文档的规则及使用合理的索引来提高分页效率. 实验结果证实, 优化后的查询方法在实现分页显示的操作中速度有明显的提高.

关键词: MongoDB; 数据分页; 索引; 优化

Data Paging Optimization in MongoDB

WANG Zhen-Hui[1], WANG Zhen-Duo[2]

[1](School of Technology and Engineering, Xian Fanyi University, Xi'an 710105, China)

[2](School of Electronic Information Engineering, Xian Siyuan University, Xi'an 710038, China)

Abstract: Under large amount of data the skip operation made data paging very slow, how to improve the performance of data paging efficiency became an import issue for MongoDB. We came up a where-limit data paging method from analysis of the built-in skip-limit MongoDB data paging method's advantages and disadvantages, and the key factors affecting the speed of the data paging. Changed the rules of query document and used reasonable indexes can improve data paging efficiency. The experimental results confirmed that the optimized query methods in pagination operations could significantly improved the speed.

Key words: MongoDB; data paging; index; optimization

随着互联网技术和电子商务的蓬勃发展, Web 应用成为软件开发的主流. 数据分页技术也是 Web 应用开发中经常使用的数据表示方法, 分页显示速度的快慢将直接影响 Web 应用的性能和网络服务质量. MongoDB 作为 NoSQL 技术的代表, 由于具有高性能、易部署、易使用, 存储数据方便等特点, 使其在云计算和大数据存储环境下, 为 Web2.0 应用提供了高性能和可扩展的数据存储解决方案[1-4]. MongoDB 中使用 skip-limit 方法进行数据分页. 这种数据分页方法在数据量和并发访问用户较小时, 分页响应速度问题不大. 但是当数据量很大时, skip 操作会变的很慢, 所以有必要对 MongoDB 分页技术进行改进和优化, 使其在海量数据查询和最终用户体验方面表现的更好.

1 skip-limit分页方法

1.1 分页原理

Mongodb 查询某页数据时, 按照查询条件使用 find 函数返回结果集, 然后使用基于偏移量的方法使用 skip 函数跳过部分数据从而找到用户要求显示页面的首记录位置. 最后, 使用限制显示记录数量的 limit 函数获取该页面的数据. 例如: 查询 db 数据库中 users 集合中年龄大于 30 的全部员工信息, 假定每页显示 10 条记录. 那么, 下面的公式就是返回特定页数据的公式推导.

第 1 页:

```
db.users.find({age:{ $gt:30}}).skip(0).limit(10);
```

第 2 页:

① 基金项目:陕西省教育厅科研计划项目(4JK2087)

收稿时间:2014-10-02 ;收到修改稿时间:2014-11-28

db.users.find({age:{ $gt:30}}).skip(10).limit(10);

第 n 页:

db.users.find({age:{ $gt:30}}).skip((n-1)*10).limit(10);

通用公式是: 数据库名.集合名.find(查询条件).skip((页码-1)*页记录数).limit(页记录数)

其中 find 函数: 对于当前数据库中的某个集合进行数据查找, 可包含查询条件.

skip 函数: 限制返回记录的起点.

limit 函数: 限制目标记录数量.

使用 find 函数返回符合条件的结果集, 使用 limit 函数限制返回的结果数, 使用 skip 函数跳过指定条数的记录, 所以使用 limit 并结合 skip 能够方便的实现数据分页显示.

1.2 不足之处

Mongodb 的内置分页方法是 skip+limit, 这种方式在小数据量下很不错, 但数据量一大, 页面数据获取速度就非常慢, 通过 explain 查询解析命令得出 nscanned 扫描的记录数接近于目标结果集的记录数, 例如要获取一个 100 万数据的集合里的第 999000-999100 这 100 条数据, 那么 Mongodb 会扫描 999100 条记录, 导致性能急剧下降. 图 1 是通过 Mongodb 性能跟踪工具 explain 得出的比较数据. explain 查询解析命令返回查询的处理情况.

```
db.userinfo.find().sort({logdate:1}).limit(100).explain()
{
        "cursor" : "BtreeCursor logdate_1",
        "isMultiKey" : false,
        "n" : 100,
        "nscannedObjects" : 100,
        "nscanned" : 100,
        "nscannedObjectsAllPlans" : 100,
        "nscannedAllPlans" : 100,
        "scanAndOrder" : false,
        "indexOnly" : false,
        "nYields" : 0,
        "nChunkSkips" : 0,
        "millis" : 0,
db.userinfo.find().sort({logdate:1}).skip(999000).limit(100).explain();
{
        "cursor" : "BtreeCursor logdate_1",
        "isMultiKey" : false,
        "n" : 100,
        "nscannedObjects" : 100,
        "nscanned" : 999100,
        "nscannedObjectsAllPlans" : 100,
        "nscannedAllPlans" : 999100,
        "scanAndOrder" : false,
        "indexOnly" : false,
        "nYields" : 0,
        "nChunkSkips" : 0,
        "millis" : 1043,
```

图 1 不同页数据显示效率对比

图 1 中重要的信息如下:

Cursor 是查询使用的游标,有两个值 BasicCursor 和 BtreeCursor, 前者表示全表查询, 后者表示使用了索引; nscanned 查询所扫描的记录总数; n 查询返回的记录总数;理想的性能状态是 nscanned 接近于 n; millis 查询耗时, 单位毫秒.

从图 1 可以看出 skip 数量不同, 需要扫描 999100 个对象, 得到 100 个结果, 耗时 1043 毫秒, 而首页数据不需要 skip, 扫描的对象个数和结果是一样的, 耗时小于 1ms, 两页数据显示速度差异非常大.

1.3 原因分析

MongoDB 的查询优化方式简单直接, 以查询模式为单位并行执行多种查询计划, 选取速度最快的. 查询模式的分类是按照查询条件和排序涉及的字段及顺序进行划分的, 并不考虑 skip 和 limit 函数. 当执行查询结果分页显示时, 需要使用 skip 函数跳过一部分数据. 例如每页显示 n 条信息, 显示第 m+1 页则需要 skip(n*m).limit(n). 在海量数据下, 当实际要读取的数据在排序的数据集后面时, skip 的数量会很大, 有时可以达到几万、十几万. 遇到这样 skip 数量的查询, 仍使用 skip 函数, 速度会很慢. 当 MongoDB 检测到该查询计划速度变慢时, 会重新执行查询计划选取, 若检测使用的查询也是 skip 很大的, 会导致选取错误的索引, 造成该查询模式正常查询的速度变慢[5,6].

2 where-limit分页方法

针对上述问题, 本文提出 where-limit 分页方法. where-limit 算法核心不是计算数据偏移量, 而是传上一页的数据标记或关键词, 根据 where 条件查询大于这个关键词的 limit 条数据来实现分页, 这种模式必须有一个连续的索引, 才能通过直接指定位置, 查找到要显示页的起始数据标记. 方法是在程序启动时将所有数据关键词读取到数组中, 需要分页的时候通过页码记录和数组下标的对应关系去查询页码首记录的数据标记, 这种分页算法体现了以空间换时间的思想. 如果数据本身没有主键的, 可以用 MongoDB 自带的 ObjectId 来查, 由于基于索引, 速度很快. 在大多数 Web 信息系统都是按时间升序或降序检索的, 所以, 一般在时间列上建立索引. MongoDB 是众多 NoSQL 数据库中与关系型数据库最相近的一种, 特别是在分页技术上和 MySQL 及其相似, 同样使用基于偏移量和 limit 方法, 但由于其不支持 SQL 语法, 在获得小于

或大于查询页首记录关键词值时不能用子查询获得，而只能将这些数据标记暂时存放在内存中. 下面 Web 日志系统中用 where-limit 方法进行数据分页的步骤.

(1)发出查询查找符合条件的每条记录的关键词字段值，并存入数组.

(2)根据数组长度(记录数)和每页显示的记录数计算总页数.

(3)默认显示第 1 页数据.

(4)用户请求显示某页数据.

此步骤(1)和 skip-limit 方法不同之处是后者只需使用 count 函数获得符合条件的记录数即可. 而 where-limit 第一次查询时要进行数组形成工作，速度相比 skip-limit 方法要慢一些，百万数据量大致在 200-500 毫秒. 但其后用户随意请求显示某页数据时，skip-limit 算法随数据偏移量即 skip 的数值不同，每页显示的时间差异很明显. 而 where-limit 算法，会根据数据偏移量从对应值的数组下标元素，找到上一页最后一条记录的关键词，从而构建新的查询条件，避免使用 skip 函数. 下面是用 Java 语言实现的 where-limit 算法核心代码.

```
//查询记录关键词数组
String logdate[]=db.webinfo.find(null,new
                BasicDBObject("logdate", true)).
                sort({logdate:-1}).ToArray();
//获取总条数
long recno = logdate.Length;
//总页数
int PageNum = (int) (recno/pagesize);
//获得该行关键词
String mydate=logdate[(pageno-1)*pagesize];
 //构造新的查询条件，返回记录列表
List datas =db.webinfo.find.(new
BasicDBObject({logdate:{$lt:mydate}})).sort({logdate:-
1}).
      limit(pagesize);
```

3　实验验证与分析

为验证 Where-limit 算法的效果，本文以 Web 日志系统为例，在结果集记录数在 100 万，每页显示 100 条记录的情况下对两种分页方法进行了时间测试(单位为毫秒)，每个查询测 10 次，取平均值，结果如图 2 所示.

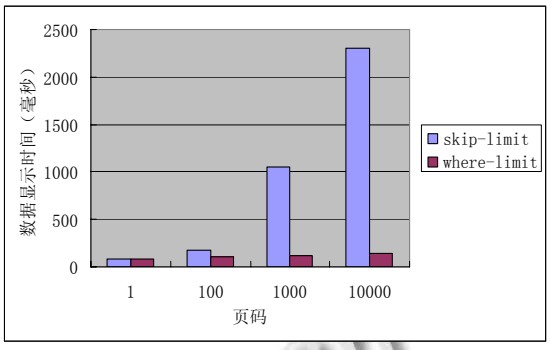

图 2　两种分页方法效率对比图

由实验结果可知，where-limit 方法由于每次只返回特定页面的数据，网络数据传输小，同时不再使用 skip 函数，分页速度快且稳定可靠. 缺点是关键词数组的大小会对服务器内存要求较高，但可以使用分割数组的方法解决. 同时对海量数据而言，关键词数组中常存在重复现象，必要时选择增加时间戳列保证存关键词的唯一性.

4　索引的使用

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[7,8]. MongoDB 的索引机制类似于关系型数据库，关系型数据库上积累的索引创建经验很多都适用 MongoDB. 但 MongoDB 是文档存储模型，自身也有一些创建索引的限制. 要根据查询创建索引，必要是使用复合索引，使用 db.posts.totallndexSize()检查索引大小是否可以放入内存. 当索引大小不能全部放入内存时，查询需要进行磁盘 I/O 速度会变慢很多. 例如机器内存 4GB，但数据库有 3GB 的内存，这样索引就不能全部放入内存. 此时可以增加内存或检查创建的索引是否真正被使用.

MongoDB 的查询优化方式简单直接,在选定了索引之后可以很好地提升性能,但在重复选择索引时会造成一定的开销,导致查询速度变慢. 所以若某些应用场景下查询固定地使用一个索引可以达到最好的性能，管理员可以使用 MongoDB 提供的 hint 函数指定索引，避免数据库无谓的尝试[9].

5　结语

为跳高当前海量数据中数据分页显示效率，实现 MongoDB 分页速度的稳定性和可靠性，本文给出了一

种改变查询文档的规则及使用合理的索引作为切入点的解决方法. 重点研究了数据分页的操作流程, 分析关系数据库分页方法, 提出了基于条件分页的 where-limit 算法, 在 Web 应用中增加分页时间戳列, 采用时间索引, 再进行页面关键数据定位的方法, 快速定位页面数据. 同时, 为了降低首次检索时计算页码等工作的耗时, 可以使用 Ajax 技术改善用户体验, 进一步优化数据分页性能[10,11]. 最后, 在 Web 日志系统中验证了 where-limit 分页方法. 实验结果表明, 本文提出的 where-limit 分页方法行之有效, 算法性能明显优于现有 skip-limit 算法. 在海量数据情况下, 连续索引的创建往往困难, 索引指定属性上的取值常重复存在, 虽然本人使用了分割数组和时间戳列来建立唯一索引, 但对此类普遍问题有待进一步研究.

参考文献

1 王光磊.MongoDB 数据库的应用研究和方案优化.中国科技信息,2011,(20):93–95.
2 潘凡.从 MySQL 到 MongoDB—视觉中国的 NoSQL 之路.程序员,2010,(6):78–80.
3 蔡柳青.基于 MongoDB 的云监控设计与应用[学位论文].北京:北京交通大学,2011.
4 申德荣,于戈,王习特等.支持大数据管理的 NoSQL 系统研究综述.软件学报,2013,24(8):1786–1799.
5 程显峰.MongoDB 权威指南.北京:人民邮电出版社,2011.
6 沈姝.NoSQL 数据库技术及其应用研究[学位论文].南京:南京信息工程大学,2012.
7 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.
8 亓传伟,王新勇.数据库分页优化技术分析与实现.煤炭技术,2012,31(8):184–185.
9 张文盛,郑汉华.基于 MongoDB 构建高性能网站技术研究.吉林师范大学学报(自然科学版),2013,(2):123–127.
10 凌建杭.ASP.NET 下高效通用的无刷新分页实现.电脑编程技巧与维护,2013,(17):57–58.
11 徐跃伟.网上购物系统的实现及性能优化.计算机时代,2012,(1):8–9.