MongoDB 分页技术优化研究

戴传飞[1],马明栋[2]
(1. 南京邮电大学 通信与信息工程学院,江苏 南京 210003;
2. 南京邮电大学 地理与生物信息学院,江苏 南京 210023)

摘 要:随着云计算、互联网等技术的飞速发展,数据量呈现爆炸性增长趋势,半结构和非结构化数据增长迅速,传统的关系型数据库愈来愈不能满足人们的需求。NoSQL(非关系型数据库)的出现,对于解决庞大数据量和高并发等问题提供了非常有效的解决方案。MongoDB 数据库作为 NoSQL 中的一员,以其独特的优势而受人青睐。在关系数据库中,查询作为数据库中最为频繁的操作,查询的效率一直是人们研究的重点。相对应的,在大数据背景下,在非关系数据库中实现数据的快速查询变得愈加重要。文中主要研究 MongoDB 数据库的分页查询技术,针对 MongoDB 系统中内置的 skip_limit 分页技术查询效率低的现象,从分析影响分页查询速度的关键因素入手,提出一种新的分页技术进行优化。实验结果表明,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:大数据;NoSQL;MongoDB;分页
中图分类号:TP31 **文献标识码**:A **文章编号**:1673-629X(2018)06-0097-05
doi:10.3969/j.issn.1673-629X.2018.06.022

Research on Optimization of MongoDB Paging Technology

DAI Chuan-fei[1], MA Ming-dong[2]
(1. School of Telecommunications & Information Engineering, Nanjing University of Posts and Telecommunications, Nanjing 210003, China;
2. School of Geographical and Biological Information, Nanjing University of Posts and Telecommunications, Nanjing 210023, China)

Abstract: With the rapid development of cloud computing, Internet and other technologies, the amounts of data have presented explosive growth trend. Semi-structured and unstructured data grow rapidly, so the traditional relational database cannot meet the public needs. NoSQL (non relational database) provides an effective solution for the problems like huge amount of data and high concurrency. MongoDB database, as a member of the NoSQL, is favored by people with its unique advantages. The query as the most frequent operation in the relational database, its efficiency has been the focus of the research. Correspondingly, in the context of big data, in the non-relational database achieving data rapidly becomes more important. In this paper, we mainly studies the paging technology of MongoDB database. Aiming at the low query efficiency of the skip_limit paging technology in MongoDB system, we propose a new paging technique to optimize by analysis of the key factors affecting the speed of paging technology. The experiments show that the optimized method has a significant improvement in the speed of the paging operation.

Key words: big data; NoSQL; MongoDB; paging

0 引 言

大数据时代,越来越多不同内容和形式的数据涌现出来,半结构化数据和非结构化数据的使用使当前系统应用越来越丰富多彩。面对海量数据的存储,原始的关系型数据库已经显得力不从心[1]。大数据的出现以及云计算的盛行使 NoSQL 这种非结构化数据库越来越受到重视[2]。尽管数据库技术发生了巨大变革,但是查询操作仍然是最频繁的数据库操作,因此研究数据库的查询及其优化变得十分重要。作为 NoSQL 之一的 MongoDB,因其在处理高并发、大数据等

收稿日期:2017-05-02 **修回日期**:2017-09-06 **网络出版时间**:2018-02-08
基金项目:江苏省自然科学基金-青年基金项目(BK20140868)
作者简介:戴传飞(1989-),男,硕士研究生,研究方向为图像处理与图像通信;马明栋,博士,教授,研究方向为地理信息系统平台软件设计与开发等。

领域的优势而备受青睐[3]。文中将主要关注 MongoDB 的查询优化,针对 skip_limit 分页查询技术的不足提出一种新的分页查询算法[4]。

1 MongoDB 简介

MongoDB 是一个基于分布式文件存储的开源数据库系统。在高负载的情况下,添加更多的节点,可以保证服务器性能。MongoDB 旨在为 Web 应用提供可扩展的高性能数据存储解决方案。

MongoDB 将数据存储为一个文档,数据结构由键值对组成,字段值可以包含其他文档,数组及文档数组[5]。MongoDB 文档类似于 JSON 对象的 BSON。BSON 是为效率而设计的,只需要使用很少的空间,同时其编码和解码都是非常快速的。即使在最坏的情况下,BSON 格式也比 JSON 格式在最好的情况下存储效率高。

MongoDB 的主要功能特性如下:

(1)面向集合存储,容易存储对象类型的数据。在 MongoDB 中数据被分组存储在集合中,集合类似 RDBMS 中的表,并且一个集合中可以存储无限多的文档[4]。

(2)模式自由,采用无模式结构存储。

(3)支持完全索引,可以在任意属性上建立索引,包含内部对象。MongoDB 的索引和 RDBMS 的索引基本一样,可以在指定属性、内部对象上创建索引以提高查询速度[6]。

(4)强大的聚合工具。MongoDB 除了提供丰富的查询功能外,还提供强大的聚合工具,如 count、group 等,同时支持使用 MapReduce 完成复杂的聚合任务[7]。

(5)支持复制和数据恢复。MongoDB 支持主从复制机制,可以实现数据备份、故障恢复、读扩展等功能。

(6)使用高效的二进制数据存储,包括大型对象(如视频)[8]。使用二进制格式存储,可以保存任何类型的数据对象。

(7)自动处理分片,以支持云计算层次的扩展。MongoDB 支持集群自动切分数据,对数据进行分片可以使集群存储更多的数据,实现更大的负载,也能保证存储的负载均衡[9]。

(8)支持 Perl、PHP、Java、C#、JavaScript、Ruby、C 和 C++语言的驱动程序[10],开发人员使用任何一种主流开发语言都可以轻松编程。

2 分页查询技术

随着信息技术的高速发展,应用系统中的数据量越来越庞大,高效的查询方法显得尤为重要。一般来讲,不要将查询获得的数据一次性全部显示出来,因为这样会耗费大量的查询时间[11]。当面对海量数据时,将大量数据直接显示出来会给用户浏览带来极大的不便,对系统性能也会造成严重的影响。再者,一次性将大量的数据发往客户端,会增加网络带宽的负担,延缓系统的反应速度,降低系统的性能。面对这些问题,采取分页方式来显示数据就变得很重要。在数据库查询中对数据进行分页显示是十分必要的。

与关系型数据库中的分页方法类似,MongoDB 数据库也可以使用 skip_limit 方法进行数据库分页。针对小型用户系统,使用 skip_limit 分页方法确实可以迅速地查询出指定数据,并且不会对分页响应速度产生太大影响。但是,如果系统中存储的数据量变得非常庞大,skip 操作就会随之变得非常慢,从而严重影响数据库查询效率。引发这种状况的主要原因是:skip 操作需要先找到被略过的数据,然后再将这些数据进行抛弃。目前很多数据库都会将大量的元数据保存在索引当中,其目的是为了减轻 skip 函数的工作量。虽然这种方法非常有用,但是目前的 MongoDB 数据库并不支持这一操作,因此有必要使用新的分页方法来处理海量数据查询。

在当前 Web 应用开发中,数据库分页技术是经常使用的一种数据表示方法,数据分页效率的好坏极大地影响了系统性能,已经被作为数据库评判的重要指标[12]。不仅如此,分页显示速度的快慢会对 Web 应用的性能和网络服务质量产生极大的影响。

3 skip_limit 分页方法

3.1 原 理

使用 skip_limit 方法进行分页查询时,首先会根据查询条件查找出所有的结果集;然后使用 skip 函数跳过指定数量的数据;最后通过 limit 函数来限制显示记录的数量。

在常用的查询语句中,经常会使用 find 函数来查找指定条件的结果集,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。通常在数据库查询中使用分页查询时还会使用 sort 方法,该方法可以对查询记录进行排序,这为数据分页查询提供了极大的便利。这三种方法可以组合使用,对于分页非常有效。

使用 skip_limit 分页方法在 test 数据库的 users 集合中分别查询第 1 页、第 2 页、第 n 页年龄为 20、21、22 的用户数据,有关查询语句如下所示:

第 1 页:

db. users. find ({" age": {" $in": [20, 21,

22]}}). skip(0). limit(10)

第 2 页:

db. users. find ({"age":{"$in":[20, 21, 22]}}). skip(10). limit(10)

第 *n* 页:

db. users. find ({"age":{"$in":[20, 21, 22]}}). skip((n-1) * 10). limit(10)

通用公式可以表示为:

db. 集合名称. find(查询条件). skip((页码-1) * 每页记录数). limit(每页记录数)

上述公式首先使用 find 函数查询当前数据库中指定条件的集合数据,即查找出 users 集合中的所有数据(注意 find 方法中可包含查询条件);然后使用 skip 方法略过指定的文档数,即略去当前页之前的所有数据;最后使用 limit 方法来限制每页需要显示多少条记录,该实验为 10 条。一般在进行分页查询时还会使用 sort 对结果集进行排序。

3.2　不　足

MongoDB 数据库采用内置的 skip+limit 分页方法,对于处理少量的数据,这种分页的确能起到不错的效果,查询速度也能得以提升[13]。但是,当存储的数据量达到百万级别甚至更大时,内置的分页方法就变得难以处理,它会导致页面数据获取速度变得极其缓慢,系统性能因而受到极大的影响[14]。造成该结果的原因是如果查询的数据位于排序的集合后面,此时使用 skip 函数就需要跳过很大的数据量,在这种情况下仍使用 skip 函数就会极大地影响数据查询效率。也就是说在大数据下,skip_limit 的分页查询已经不再适用。

4　where_limit 分页方法

面对海量数据查询,针对 skip_limit 方法存在的问题,为提升数据库的查询效率,文中提出了一种新的分页方法—where_limit。与 skip_limit 方法的原理不同,where_limit 算法不再以数据偏移量为核心,它通过寻找出分页信息中当前所在页的上一页的数据标记或关键词,然后以该关键词为基础进行条件查询,只需使查询语句中的条件参数大于这个关键词,就能决定最后到底需要 limit 到多少条数据,从而实现分页查询。考虑到关键词数组的连续性,需要在 MongoDB 数据库中创建一个连续的索引,因为只有在连续索引中,才能成功找到指定的关键词。

4.1　核心思想

利用 where_limit 分页方法进行数据查询时,首先通过查询条件来获取文档中所有数据的关键词,然后将其读取到相应数组中,然后根据关键词数组的下标来确定需要跳过的记录数,接着使用 limit 方法来限定显示的记录数目,最后使用 sort 方法对结果排序。这种分页算法的核心思想便是牺牲空间来提高查询效率。虽然需要用比较大的存储空间去存放关键词数组,但是如果能极大地提升查询效率,这种分页方法也是非常可行的。

以用户基本信息为例来测试 where_limit 方法是否优于 skip_limit 方法。为进行数据分页查询,需要先在用户信息系统中实现 where_limit 方法,可以将其分为三步:

(1)使用 count 函数确定数据库中总的记录数的长度,然后根据每页需要显示的记录数来计算出数据需要显示的总页数;

(2)根据查询条件获取关键词数组;

(3)通过关键词数组中的关键词进行分页查询。

where_limit 方法要优于 skip_limit 方法,那是因为它避免使用 skip 函数,系统不需要花费大量时间来跳过大量数据。使用 skip_limit 方法进行分页查询时,会优先考虑分页的数据偏移量(skip 函数的参数值),随着偏移量不同,每页查询的时间就会不同:偏移量越大,查询时间也就越长。where_limit 为解决这一问题,使用了查找关键词数组的方法:在查询过程中,用户只需要根据查询条件寻找出关键词数组,然后根据数组中关键词的下标来决定要跳过的记录数,这样就避免使用 skip 函数。算法处理流程如图 1 所示。

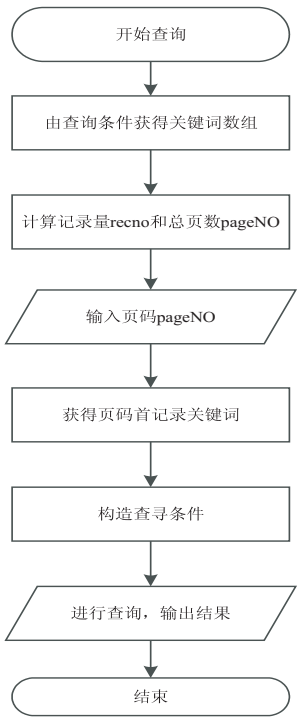

图 1　where_limit 算法处理流程

4.2　验证分析

为验证 where_limit 算法的可行性,通过 Mon-

goDB 自带的 Mongo shell 向数据库名为 test,集合名为 users 的集合中插入 100 万条数据。集合中文档结构如下:

```
{
"_id":ObjectId('584e58e46dadd1120d4f75345'),
"name":"Siri",
"age":35,
"sex":"woman",
"num":1
}
```

users 集中存放有 100 万条记录,分页时限制在每页显示 100 条记录。

使用 skip_limit 方法和 where_limit 方法分别对 1 000页和 8 000 页数据进行查询,可得出通过 where_limit 方法对应的查询条件分别是 num 值为 100 000、800 000,使用 MongoDB 性能分析工具 explain( )进行分析,实验结果如图 2 ~5 所示。

```
>db.users.find().sort({num:1}).skip((1000-1)*100).limit(100).explain()
{    cursor : BtreeCusor num_1 ,
     isMultiKey :false,
     n :100,
     nscannedObjects :100,
     nscanned :100000,
     nscannedObjectsAllplans :100,
     nscannedAllplans :100000,
     scanAndOrder :false,
     indexOnly :false,
     nYields :0,
     nChunkSkips :0,
     millis :65,
     indexBounds :{
          num : [
               [
                    {
                         $minElement :1
                    },
                    {
                         $minElement :1
                    }
               ]
          ]
     },
     server : 7YYNX1TM0HMESPN:27017
}
```

图 2 使用 skip_limit 方法查询第 1 000 页数据

```
>db.users.find({num:{ gt :100000}}).sort({num:1})..limit(100).explain()
{    cursor : BtreeCusor num_1 ,
     isMultiKey :false,
     n :0,
     nscannedObjects :0,
     nscanned :0,
     nscannedObjectsAllplans :0,
     nscannedAllplans :0,
     scanAndOrder :false,
     indexOnly :false,
     nYields :0,
     nChunkSkips :0,
     millis :0,
     indexBounds :{
          num : [
               [
                    {
                         gt :100000
                    },
                    {
                         gt :100000
                    }
               ]
          ]
     },
     server : 7YYNX1TM0HMESPN:27017
}
```

图 3 使用 where_limit 方法查询第 1 000 页数据


```
>db.users.find().sort({num:1}).skip((8000-1)*100).limit(100).explain()
{    cursor : BtreeCusor num_1 ,
     isMultiKey :false,
     n :100,
     nscannedObjects :100,
     nscanned :800000,
     nscannedObjectsAllplans :100,
     nscannedAllplans :800000,
     scanAndOrder :false,
     indexOnly :false,
     nYields :0,
     nChunkSkips :0,
     millis :402,
     indexBounds :{
          num : [
               [
                    {
                         $minElement :1
                    },
                    {
                         $minElement :1
                    }
               ]
          ]
     },
     server : 7YYNX1TM0HMESPN:27017
}
```

图 4 使用 skip_limit 方法查询第 8 000 页数据

```
>db.users.find({num:{ gt :800000}}).sort({num:1})..limit(100).explain()
{    cursor : BtreeCusor num_1 ,
     isMultiKey :false,
     n :0,
     nscannedObjects :0,
     nscanned :0,
     nscannedObjectsAllplans :0,
     nscannedAllplans :0,
     scanAndOrder :false,
     indexOnly :false,
     nYields :0,
     nChunkSkips :0,
     millis :0,
     indexBounds :{
          num : [
               [
                    {
                         gt :800000
                    },
                    {
                         gt :800000
                    }
               ]
          ]
     },
     server : 7YYNX1TM0HMESPN:27017
}
```

图 5 使用 where_limit 方法查询第 8 000 页数据

从以上图中可以发现,查询第 1 000 页、8 000 页数据,skip_limit 方法所需时间为 65 ms、402 ms,而使用 where_limit 方法所需时间接近 0 ms、0 ms。使用 where_limit 方法的查询耗时远小于使用 skip_limit 方法,优化效果明显。

为了提高实验数据的可靠性,使用多台虚拟机进行实验,采用取平均值的方法,对两种分页方法进行对比,测试结果如图 6 所示。

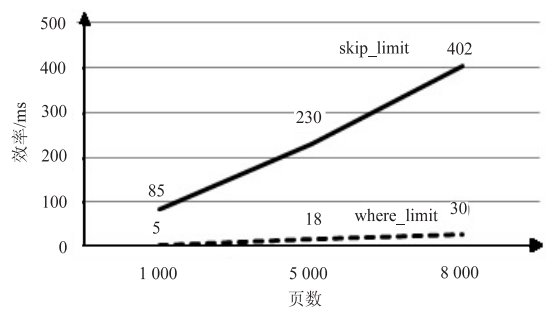

图 6 两种分页方法效率对比

由实验结果可知,skip_limit 方法在 skip 方法跳过

记录数越大时,数据查询所耗费的时间越长,因此查询效率越低;而 where_limit 方法由于每次只返回特定页面的数据,不再使用 skip 函数,分页速度快且稳定可靠,查询效率明显优于 skip_limit 方法。

5　结束语

MongoDB 内置的 skip 操作在小数据的情况下实现分页技术简单,但是在数据量达到百万级别甚至更大时,skip 操作的弊端愈加明显。数据库的分页查询效率成为影响数据库访问性能的重要因素。通过分析 MongoDB 内置的 skip_limit 分页方法的优缺点及影响分页查询速度的关键因素,提出一种新的数据分页方法—where_limit。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。实验结果表明,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。

参考文献:

[1] 程显峰. MongoDB 权威指南[M]. 北京:人民邮电出版社,2011.

[2] 金　鑫. 非结构化数据查询处理与优化[D]. 杭州:浙江大学,2015.

[3] HUANG Yu, LUO Tiejian. NoSQL database: a scalable, availability, high performance storage for big data[M]. [s. l.]: Springer International Publishing, 2014.

[4] 潘　凡. 从 MySQL 到 MongoDB-视觉中国的 NoSQL 之路[J]. 程序员,2010(6):79-81.

[5] STEVIC M P, MILOSAVLJEVIC B, PERISIC B R. Enhancing the management of unstructured data in e-learning systems using MongoDB[J]. Program Electronic Library and Information Systems, 2015, 49(1):30-45.

[6] 吴德宝. 关系与非关系数据库应用对比研究[D]. 抚州:东华理工大学,2015.

[7] 梁云柯. MongoDB 索引机制研究[D]. 重庆:重庆邮电大学,2016.

[8] 沈　姝. NoSQL 数据库技术及其应用研究[D]. 南京:南京信息工程大学,2012.

[9] 祁　兰. 基于 MongoDB 的数据存储与查询优化技术研究[D]. 南京:南京邮电大学,2016.

[10] 吕明育,李小勇. NoSQL 数据库与关系数据库的比较分析[J]. 微型电脑应用,2011,27(10):55-58.

[11] 郭忠南,孟凡荣. 关系数据库性能优化研究[J]. 计算机工程与设计,2006,27(23):4484-4486.

[12] 丁智斌,石浩磊. 关系数据库设计与规范化[J]. 计算机与数字工程,2005,33(2):114-116.

[13] BACH M, WERNER A. Standardization of NoSQL database languages[M]. [s. l.]: Springer International Publishing, 2014.

[14] 王光磊. MongoDB 数据库的应用研究和方案优化[J]. 中国科技信息,2011(20):93-94.

(上接第 96 页)

鲁棒机理研究[J]. 软件学报,2015,26(1):52-61.

[4] 齐美彬,岳周龙,疏　坤,等. 基于广义关联聚类图的分层关联多目标跟踪[J]. 自动化学报,2017,43(1):152-160.

[5] LIU Zongang, DU Xiaoxue, PAN Diansheng. A new tracking algorithm based on mean shift and particle filter[C]//Proceedings of the 2016 IEEE/RSJ international conference on intelligent robots and systems. Washington, DC: USA: IEEE Computer Society, 2016:38-42.

[6] MESHGI K, ISHII S. Expanding histogram of colors with gridding to improve tracking accuracy[C]//Proceedings of the fourteenth IAPR international conference on machine vision applications. [s. l.]: [s. n.], 2015:475-479.

[7] DONG Jiwen, ZHAO Lei, ZHANG Liang. The face tracking research based on cam-shift and improved NMI[C]//Proceedings of the 2013 international conference on computational and information sciences. Washington, DC: USA: IEEE Computer Society, 2013:148-151.

[8] 郭文俊,常桂然. 基于三帧间的移动障碍物背景目标提取算法[J]. 软件,2016,37(2):58-62.

[9] 熊　波,尹周平. 滑动平均和改进权重函数的快速非局部平均图像去噪算法[J]. 中国图象图形学报,2012,17(5):628-634.

[10] PLYER A, BESNERAIS G, CHAMPAGNAT F. Massively parallel Lucas Kanade optical flow for real-time video processing applications[J]. Journal of Real-Time Image Processing, 2016, 11(4):713-730.

[11] LIGORIO G, SABATINI A M. A novel Kalman filter for human motion tracking with an inertial-based dynamic inclinometer[J]. IEEE Transactions on Biomedical Engineering, 2015, 62(8):2033-2043.

[12] 石祥滨,张　健,代　钦,等. 采用显著性分割与目标检测的形变目标跟踪方法[J]. 计算机辅助设计与图形学学报,2016,28(4):645-653.

[13] LI Chenglong, WANG Xiao, ZHANG Lei, et al. Weighted low-rank decomposition for robust grayscale-thermal foreground detection[J]. IEEE Transactions on Circuits & Systems for Video Technology, 2017, 27(4):725-738.

[14] 高跃明,伊腾增,韦孟宇,等. 二维 Otsu 和改进区域生长法的荧光免疫层析试条浓度的定量检测[J]. 传感技术学报,2016,29(9):1356-1360.

[15] 赵毅力,周　屹,徐　丹. 微距摄影的多聚焦图像拍摄和融合[J]. 中国图象图形学报,2015,20(4):544-550.

[16] YADAIAH V, HARAGOPAL V V. Multi-objective optimization of time-cost-quality using Hungarian algorithm[J]. American Journal of Operations Research, 2016, 6(1):31-35.