应用分布式索引提高海量数据查询性能①

窦晓峰, 陈 胜, 王熠航, 麦联叨, 由建宏

(亚信联创 联通事业部, 北京 100086)

摘 要: 在电信领域的精准化营销、即席查询业务中, 存在着大量针对一张宽表或几张宽表(超过 50 字段)的随机查询场景. 传统处理模式(直接查询数据库)在数据量不大(<1000 万)时, 查询响应时间可优化到几秒至数十秒级, 而当数据量到达几千万、上亿甚至十亿记录以上时, 此处理模式无论如何优化或更改索引机制, 都无法满足秒级并发查询要求.新的处理模式通过引入分布式 Solr 索引层解决上述问题.索引层预先对数据库记录建立索引, 查询不再作用于数据库而直接查询索引层, 如此, 可大幅提高查询性能.经过对两种处理模式的对比验证, 在相同环境下, 数据量到达 5000 万, 每秒 20 并发访问的宽表查询场景, 传统处理模式的查询全部超时失败, 而使用分布式索引层的查询可以在 2 秒以内返回, 查询全部成功.

关键词: 精准化营销; 即席查询; 海量数据; 大数据; 查询; Solr 集群; 分布式索引; 分片; B-Tree

Improve Big Data Query Performance by Applying Distributed Indexing

DOU Xiao-Feng, CHEN Sheng, WANG Yi-Hang, MAI Lian-Tao, YOU Jian-Hong

(Department of China Unicom, Asiainfo-Linkage, Beijing 100086, China)

Abstract: In the field of telecommunications precision marketing and ad-hoc query, there are a lot of random queries scenarios on one or more wide-tables (which have more than 50 fields). In the traditional system (the queries are performed on the database directly), the query response time can be optimized less than a few seconds to tens of seconds when the database records size is under 10 million. When the data size reaches tens of millions, hundreds of millions or even more than one billion records, whatever optimization including changing indexing mechanism are unable to meet the second-level concurrency query requirements. In the new query system, we introduce the Solr distributed index layer to solve these problems. The layer will index the database records firstly and queries will access the Solr index layer and not perform on the database directly, therefore, the performance will be improved highly. After a comparison of the two processing patterns in same environment, for the data of 50 million, 20 per concurrent access query scenario, the traditional accessing queries all are timeout; while the other's queries can be returned within 2 seconds and all are success.

Keywords: precision marketing; ad-hoc query; massive data; big data; query; solr cluster; sharding; B-tree

在电信领域的精准化营销中, 需要按照客户相应属性划定客户人群, 如某省"3G 理财 VIP 用户人工回访"客户维系营销, 查询条件包含"是否 3G 用户", "出账收入", "3G 流量", "套餐外流量", "拨打分钟数", "客户状态", "是否合约客户", "是否靓号", "是否银行托收"等属性. 对于大学校园营销, 可能就会选择另外一套属性. 因此, 每次营销所选属性条件都会不同, 即查询是对客户属性的任意组合.

从数据库的层面上来看, 无法预知用户每次查询是哪几种属性的组合及顺序, 也就无法建立相应的联合索引. 而如果在每一个属性字段上都建立索引, 会导致索引数据空间的急剧膨胀. 同时, 对于客户属性值为"是"和"否"这样的的稀疏属性, 即使对该字段建立索引, 索引对数据查询加速效果不大. 因此, 像精

收稿时间:2013-09-07;收到修改稿时间:2013-11-27

准化营销这样的查询业务场景, 基本通过不建索引即全表扫描来实现. 这种处理模式, 在实际的精细化营销应用中, 当数据量不大时, 对用户的查询性能影响尚不显著. 如某省的精细化营销系统, 数据量在 1000 万左右, 查询基本在 5 秒左右返回, 而对于有些即席查询场景, 需要 2 分钟左右才能返回. 当数据量上升到几千万, 上亿甚至十数亿数据规模时, 查询时间则会到达无法忍受的地步.

由上可以看出, 在电信领域的精准化营销[1]和即席查询[2]业务中, 宽表的查询性能问题已经成为一个难题.

1 分布式索引方案

分布式检索引擎 Apache Solr 的出现, 为解决此类问题提供了另外一种处理模式. Solr 支持对数据库的记录建立索引[3]. 数据库记录抽象为文档, 由多个字段组成, 每个字段由一个单词组成. 并使用倒排索引技术[4]来创建索引.

从索引使用方式上, 电信领域的精准化营销业务场景索引方案可以分为简单索引方案和复杂索引方案两种方式, 下面分别予以介绍.

1.1 简单索引方案

简单索引方案如下图 1 所示, 当数据查询请求发送到索引集群服务器时, 索引集群服务器按照查询条件进行查询, 并返回指定的结果. 索引集群服务器此时不访问数据库.

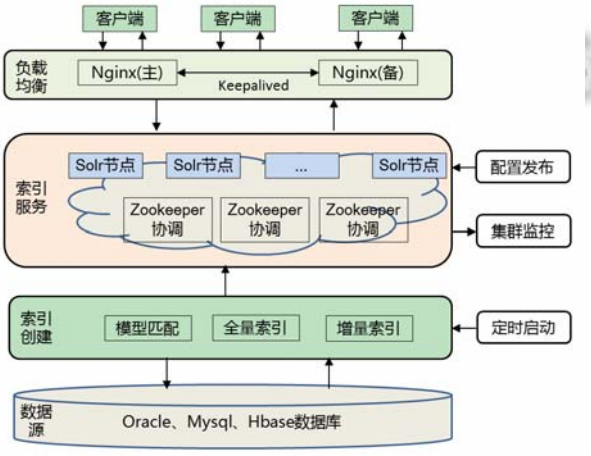

图 1　海量数据查询简单索引方案

由于查询结果直接由索引集群服务器返回, 因此需要索引集群服务器保存要返回的字段及其数据. 当返回结果不固定时(用户可自主选择返回结果的字段集), 就需要存储所有的可能返回字段, 这在数据量很大时会占用很多存储空间, 且影响查询性能.

简单索引方案优点: 查询实现简单, 在查询过程中不需再和数据库交互, 数据库不会成为性能瓶颈, 因而可以支持高并发量. 其缺点就是需要保存所有的返回结果字段, 除了占用存储空间外, 且和数据库所存记录冗余. 另外, 当返回结果字段数目巨大, 且数据海量时, 会严重影响性能.

1.2 复杂索引方案

复杂索引方案在索引创建部分和简单索引方案基本相同, 区别在于索引服务部分, 如图 2 所示.

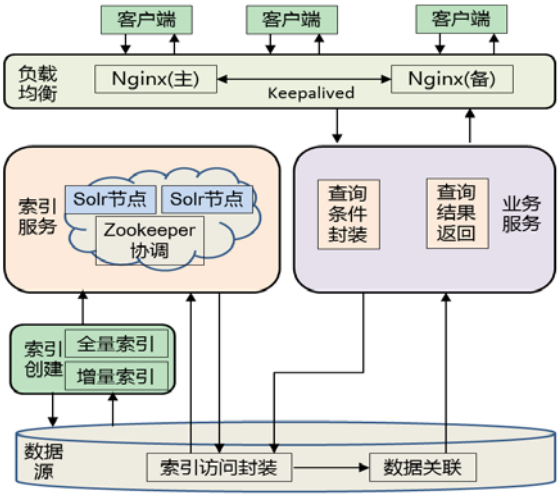

图 2　海量数据查询复杂索引方案

复杂索引方案在索引创建部分和简单索引方案基本相同, 区别在于索引服务部分, 如图 2 所示. 群, 而是将请求发送给数据库服务器, 数据库服务器通过加载索引访问封装模块来访问 Solr 索引集群, Solr 索引集群根据查询条件返回检索结果(一般仅为主键), 数据库根据返回的主键与数据表关联, 然后返回查询结果给查询请求. 复杂索引方案与简单索引方案的最大区别是在索引集群中不再冗余保存查询结果字段.

其优点是在 Solr 集群索引服务器中只存储主键字段, 减少了其它字段的保存, 这些字段由数据库表关联获取. 如此处理可以保证 Solr 服务器占用极少的存储空间, 查询性能高.

此方案的缺点是增加了业务复杂性, 需要开发在数据库层面访问索引集群服务器的模块, 还需要开发数据关联模块. 同时, 由于查询直接作用于数据库层面, 数据库容易成为性能瓶颈.

2 海量数据查询实现

通过上面的两种索引方案分析可以知道, 对于电

信精准化营销这样的海量宽表查询业务场景, 由于其查询返回结果固定, 简单索引方案比较适合. 在简单索引方案中, 需要实现索引创建和索引查询两个功能模块. 索引创建负责数据库记录数据的索引创建功能, 包括全量索引和增量索引. 索引查询负责解析用户所选查询条件, 并对 Solr 索引集群服务器解析查询.

为达到索引创建的高性能, 索引创建模块采用多线程的线程池模式, 线程区分为生产者线程和消费者线程[5], 生产者线程负责从数据库分页读取记录, 消费者线程负责将数据库记录转换为索引文档并提交到索引集群服务器, 生产者线程和消费者线程之间通过非阻塞队列进行数据交换, 并使用计数栓锁协调各线程.

索引创建模块支持全量索引和增量索引两种方式, 如果是全量索引方式, 会先删除原来的索引; 如果是增量索引, 则会按更新时间获取对应的记录.

索引查询主要利用 Solr 提供的 API 实现用户的查询条件转换, 并向 Solr 服务器集群查询获取返回结果.

3 海量数据查询性能测试

海量数据分布式索引查询方案实现后, 需要对其进行性能测试. 测试会同时选择多种传统数据库方式作为对比. 测试用数据表字段为 50 个, 数据量为 5000 万. 传统数据库查询方式分别选择 Mysql Cluster, Mysql InnoDB 及 Infobright 列数据库. 测试条件为: 单用户访问; 10 用户并发访问; 20 用户并发访问; 在并发访问时, 用户所选条件随机选取以避免缓存.

设定上述测试场景后, 分别根据上述条件进行压力测试, 压力测试每次进行 10 分钟, 每种方式测试进行 5 次测试, 分别记录测试结果, 最后使用 5 次测试结果平均值作为最终结果.

4 海量数据查询测试结果

表 1 测试结果

查询方式	数据库索引	单用户	10 用户	20 用户
Mysql Cluster	无	187s	超时	超时
	有	1560s	超时	超时
Mysql InnoDB	无	135s	超时	超时
	有	127s	超时	超时
Infobright	不支持	2.6s	超时	超时
Solr	无	0.18s	2.21s	1.45s

由测试结果可以看出, 在单用户查询模式下, 按照传统模式, 除了列数据库 Infobright 可以在 3 秒内返回结果外, 其它两种传统数据库查询方式时间都在几分钟之上, 无法到达使用标准. 同时当使用并发查询时, 在传统数据库模式下, 所有数据库都无法返回结果, 系统不可用. 而使用 Solr 分布式索引集群查询方式, 在单用户模式, 可以到达百毫秒级的高性能, 在 10 用户并发, 20 用户并发时, 性能虽然比不上单用户, 但也都在 3 秒内返回, 特别是 20 用户并发时, 性能甚至还高于 10 用户并发, 这主要是由于 Solr 集群进行了缓存, 从而使缓存命中提高, 性能得到提高.

另外, 从传统数据库查询模式看, 对于海量宽表业务查询场景, 对数据库字段建立索引或不建立索引区别不大, 甚至更慢.

5 结 语

综上所述, 在电信领域的精准化营销及即席查询这类宽表的自助查询业务场景中, 当数据量变大超过 5000 万时, 传统数据库处理模式根本无法应对, 而通过引进分布式索引方案, 可以很好地解决上述业务场景, 且性能甚至超过非海量数据下的性能.

此外, 除海量数据宽表查询, 对于普通数据库表, 当数据量达到海量时, 也可以使用这种技术来提高查询性能. 因此, 这种分布式索引查询技术有着极大应用空间.

参考文献

1 林桂珠,范鹏飞.电信企业基于 3G 时代的精准营销.重庆邮电大学学报(社会科学版), 2009,21(4):1–5.
2 熊全洪,魏娟,刘武.即席查询研究.现代商贸工业,2008,12:23–26.
3 霍庆,刘培植.使用 Solr 为大数据库搭建搜索引擎.软件,2011,32(6):11–14.
4 郑榕增,林世平.基于 Lucene 的中文倒排索引技术的研究.计算机技术与发展,2010,3.
5 陈益.利用 JAVA 多线程并发机制解决生产者—消费者问题.电脑学习,2010, 2:147–149.