**一种基于Redis TwemProxy索引缓存的HBase查询方案**

李星毅，瞿龙俊

（江苏大学计算机科学与通信学院， 江苏 镇江 212000）

**摘 要：近年来，随着互联网技术的飞速发展，爆炸式增长的数据量已经成为众多应用领域的瓶颈，更多的用户开始寻求高效的数据存储解决方案。HBase是运行在Hadoop分布式云计算平台上的NoSQL解决方案，它为海量数据的存储提供了一个分布式、高扩展性的大数据仓库。HBase采用B+树索引策略，在基于主健查询时较为高效，在对于非主健查询时，则需要进行扫描全表操作，效率十分低下。本文设计并实现了一种提高HBase集群基于非主健查询优化方案，将Redis作为缓存数据库，并引入TwemProxy作为Redis代理，吸收LRU和LFU的优点，提出了一种基于热度的缓存替换算法，降低HBase的访问开销，提高查询性能。实验结果表明：改进后的HBase集群较原生HBase集群在非主键查询效率上，性能提升了30%，并且有良好的可扩展性。**

**关键词：HBase；非主健索引；Redis；社群；缓存替换；**

**中图分类号：TP391.9 文献标志码：A**

**A HBase query optimization scheme based on Redis index buffer**

**Abstract: Modeling group organizations is the foundation for further research on group coordinative behavior. Modeling group organizations is the foundation for further research on group coordinative behavior. Many existing agent organization models concern little about the description of high-level dynamic interactions, and few of them provide decidable reasoning of tasks. We propose a novel group organizational coordinative relationship model named GRNA to describe the organizational coordinative relationship of agent groups. The GRNA model is based on the description logic of the task and it contains two sub models: cognition sub-model and interaction sub-model. The cognition sub-model which provides accurate definitions of terms is built on domain ontology, while the interaction sub-model based on the description logic of tasks reflects the collaboration between the tasks. Through the case study, we show that the GRNA model can describe norms and abilities of groups, provide powerful services of decidable deduction for the achievability of organization tasks, and express high-level task interaction semantics of groups.**

**Keywords: collaboration; description logic of tasks; group；organization**

1. **引言**

随着互联网信息量的爆炸式增长，海量数据的处理技术已经成为了一个严峻的问题，众多的互联网公司，较为突出的如Amazon、Google、Yahoo!等，致力于研究全新的存储技术和数据处理，Google公司在2003年发表了“The Google File System”,论文描述了Google内部使用的一种面向大规模数据密集型应用的、可伸缩的分布式文件系统[1]。2008年,Google公司发表了“BigTable：A Distribute Storage System for Structured Data[2]”，以此为基础，众多海量数据的存储和管理技术不断涌现，其中，HBase就是基于Apach的BigTable的产物，既是BigTable的开源实现，也是Hadoop生态圈的核心产品，HBase为大规模的数据提供实时的随机的读写服务。

传统的HBase数据库是采用类B+树索引，可以进行快速的基于主键的检索操作，在面对基于非主键的查询请求时，HBase采用扫描全表的方法进行检索，查询效率低下，难以满足快速响应的数据查询的应用场景。众多互联网公司选择为HBase建立缓存机制，从而提高HBase基于非主键的查询索引效率。Redis是一个基于内存的，开源的K/V存储引擎，支持多种数据结构，通常被用来作为缓存层使用。然而单机Redis是单线程单进程的，在处理大数据的请求时，会造成其他请求的阻塞，仅仅使用单机Redis是远远无法满足企业的需求。TwemProxy Redis是Twitter公司开发的一个开源Redis集群解决方案，具有高稳定性、安全性、可靠性等方面的优势。本文设计了一种基于Redis TwemProxy数据库的HBase索引缓存的方案，在基于非行键检索操作时，会先进入缓存数据库检索，有效的提升了HBase索引的效率。

1. **缓存索引存储模型**
2. **缓存分布式中间件模型**

相比Redis Cluster，TwemProxy的批量读写吞吐量远高于Redis Cluster[3]，TwemProxy是Redis的代理（Proxy）服务，通过TwemProxy，可以有效的避免单点故障问题，当接收到服务请求时，Proxy会根据路由规则，将这些请求分片(Sharding),并分发给不同的Redis Service，并收集Service响应的结果返回给服务请求者。这种Proxy又称为分布式中间件， TwemProxy有效的解决了单个Redis单进程单线程导致的承载能力不足的缺陷[4]。其集群架构如图2-1所示。

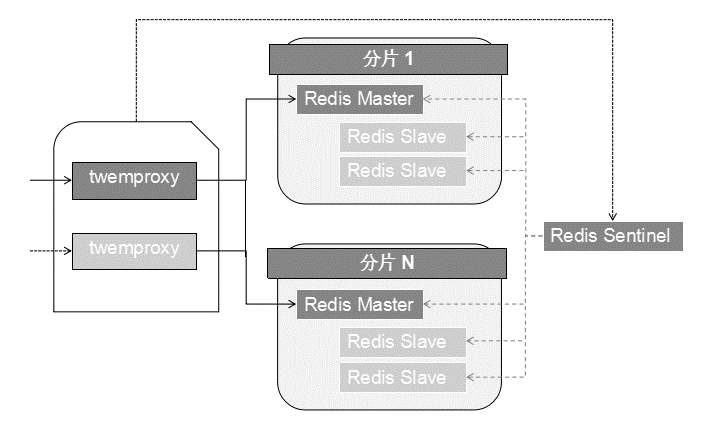


图2-1 TwemProxy集群架构

Keepalived是一个基于VRRP（Virtual Router Redundancy Protocol）协议实现的多机热备需求的解决方案[5]，Keepalived使用VRRP来避免单点故障，支持失败节点直接自动删除。Keepalived保证若集群中有一台或若干台机器失效，其他机器可以继续提供服务[4]。使用Keepalived可以动态的监听多个TwemProxy，若主TwemProxy代理节点故障，Keepalived会自动切换成从TwemProxy代理，保证了服务的高可靠性和高可用性。

1. **缓存索引存储模型**

缓存索引存储模型采用Redis作为HBase基于非主键记录值的缓存数据库，当用户需要进行非主键业务操作时，会首先进入缓存数据库检索对应的记录，如若命中，则将记录值取出并进入HBase数据库查询，若未命中，则直接转发请求到HBase数据库中查询，查询的结果将更新至Redis缓存数据库，缓存索引存储模型架构如图2-2所示。

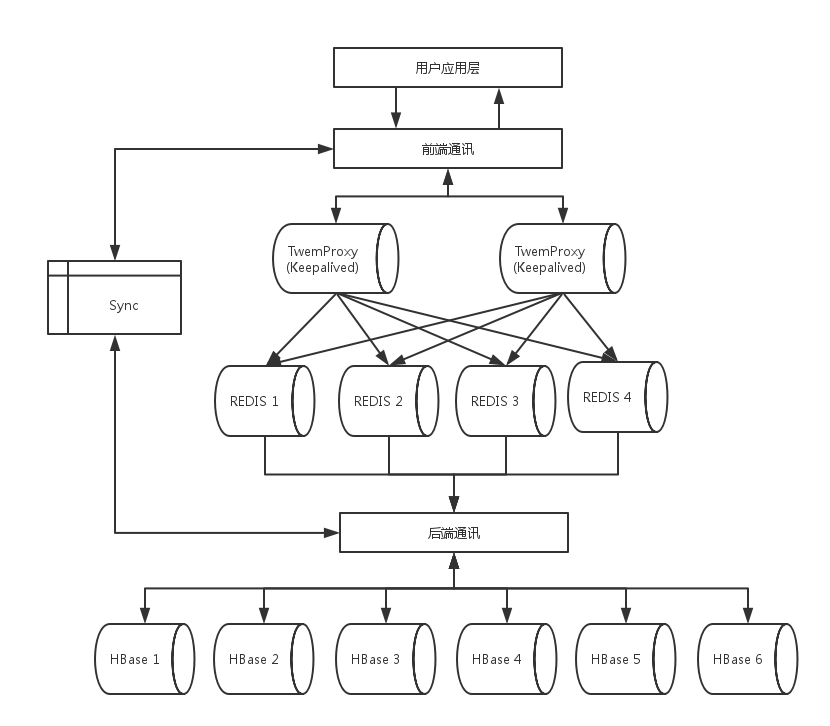


图2-2 缓存索引存储模型架构

前端通讯接收来自用户业务的请求，并将请求发送给TwemProxy代理层，代理层接收来自业务的请求，并根据转发规则，将这些请求分发给相应的Redis缓存数据库中并返回检索值，若命中缓存，即存在结果值，则在后端通讯层将基于非行键的查询转换成基于行键的查询发送到HBase数据库进行检索，若未命中，则直接对HBase进行扫描（Scan）操作，并将结果结果返回给后端通讯层，后端通讯层实时同步给前端并更新Redis缓存数据库，完成一次数据库检索操作。

1. **缓存数据存储结构**

Redis是高性能的键/值存储引擎，支持多种数据类型。Redis的HASH是由字符串的键和字符串的值组成的映射，所以被广泛用来存储对象。本模型采用哈希（HASH）值保存缓存记录，我们定义如下格式的缓存索引表键值：

键：<索引标识符\_记录值>

值：<INDEX:索引集合，FREQUENCY:历史访问热度，COUNT:当前访问记录>

其中，索引标识符为HBase数据库中，非主键属性的列名，由于Redis中KEY采用HASH存储，因此需要合理设置KEY的长度，减少存储空间开销，提升检索效率[6]。记录值表示在HBase中，当前非主键属性的值之一。HBase中非主键的列名具有唯一性，因此使用索引标识符\_记录值作为key可以有效的保证索引缓存数据库中的KEY的唯一。索引集合内存储所有满足当前“索引标识符\_记录值”的记录的主键值。当请求进入索引缓存数据库检索时，会根据“索引标识符\_记录值”进行查找，并将命中的值取出，将基于非行键的查询转换成基于行键的查询，提升HBase查询效率。

本模型采用倒排索引（Inverted Index）思想，倒排索引是“单词-文档矩阵”的一种存储形式，索引表中每一项都包括HBase属性值及该属性值对应的记录地址。图3-1（1）和图3-1（2）分别给出了一个HBase数据表示例以及非主键属性对应的缓存索引表。该示例用来统计不同学生的不同科目成绩。在图3-1（1）中，“id”属性为HBase主键属性，“Name” 、“Subject”、“Score”分别表示“学生姓名”、“考试科目”、“科目成绩”为HBase的非主键属性。

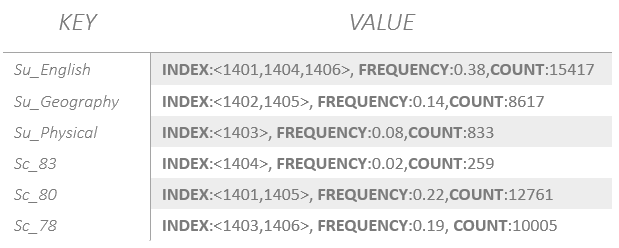


图3-1（2）Redis 索引表

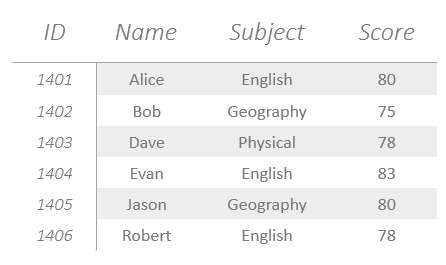


图3-1（1） HBase用户数据表

图3-1 学生科目成绩数据表示例

当用户发起一条业务请求：查询所有选修了“English”科目的学生的姓名，缓存索引存储模型会将“Subject”字段解析为Redis中预设的索引标识符“Su”，并添加需要查询的值的后缀：“Su\_English”,TwemProxy代理会将该KEY转发到相应的Redis进行结果查询，若未命中记录，则返回NULL，若命中记录，则返回对应的VALUE值：INDEX:<1401,1404,1406>, FREQUENCY:0.38,COUNT:15417，即共有四条记录符合选修了“English”科目，其主键分别为1401，1404和1406，后端通讯层解析缓存层查询的结果并转发到HBase数据库查询，将查询结果返回给用户。

由此看出，缓存索引存储模型的查询过程为：首先由前端通讯根据用户请求拼接缓存检索所需的KEY，并将KEY交给TwemProxy代理，代理将请求转发到Redis缓存服务器中进行检索，若命中，则将检索结果发送到后端通讯，后端通讯根据结果交由HBase查询，若未命中，则直接将请求转发到HBase。由于Redis是运行在内存中的数据库，因此能在缓存中命中待查询记录的行键，从而降低磁盘扫描的开销，提升了性能。

1. **缓存数据淘汰策略**

Redis内存池资源是有限的，因此，不断写入缓存数据必然会涉及到缓存淘汰的问题。通过将部分数据存储到缓存中，可以提高对数据的重复访问效率，但缓存容量不可能无限扩充，当缓存容量达到某个阈值时，需要选出合适的缓存数据淘汰出缓存，这就是缓存数据淘汰策略。

目前常见的缓存替换算法共有三种：

1. 先进先出置换（First In Frist Out，FIFO）算法：该算法在进行筛选时，会根据记录在缓存中停留的时间进行淘汰，算法核心思想为：如果一条记录最先进入缓存，则该条记录被访问到的概率越低，越容易被淘汰出缓存。FIFO算法由于根据停留时间作为淘汰依据，缓存命中率较低。
2. 最近最少使用置换（Least Recently Used，LRU）算法：该算法在进行筛选时，会根据历史访问记录进行淘汰，算法核心思想为：如果在缓存中，某一条记录在近期没有被访问过，则该条记录在将来被访问到的概率也就越小。LRU算法优先淘汰在最近访问次数少的记录。LRU算法只考虑到了数据最近一次访问的差异，数据访问频率却没有进行考虑。
3. 最近最不常用置换（Least Frequently Used，LFU）算法：该算法在进行筛选时，会根据缓存记录的被访问次数进行淘汰，算法核心思想为：如果某一条记录在一定时间内被访问的次数越小，则在将来一段时间内被访问的概率也就越低。LFU算法相比LRU算法命中率较高，但LFU复杂度较高，实现难度较大。

本模型在LRU算法基础上进行了改进，提出了一种基于访问频度和累积热度的缓存淘汰策略，其核心思想为：在一段时间内，如果某条记录最近被插入，则该条记录在将来被访问的概率就越大。若某条记录在一段时间内未被访问或者历史访问频率低，则该条记录在将来被访问的概率则越小。

热度积累的基本单位为Redis缓存数据库中的一条记录。当TwemProxy将来自前端通讯的查询请求转发到Redis缓存数据库之后，Redis会进行缓存记录检索，若命中记录，则返回对应的Value值，并将COUNT字段自增操作之后重新写入Redis，字段COUNT表示在一段时间内，当前缓存记录被命中的次数。COUNT值越大，表示当前缓存记录被命中次数越多，当前记录热度越高。

由于内存价格昂贵，决定了Redis缓存数据库容量不可能无限扩充下去，因此，用户需要在使用之前设定一个缓存阈值*execPoint*。在模型运行初期，缓存记录数远远达不到*execPoint*值，此时若有Redis缓存记录写入操作，则不进入缓存淘汰流程。即“访问即写入”操作。

当Redis的容量达到预设的阈值*execPoint*时，立即进入循环的缓存淘汰流程，文献[7]中提出了一种在时间周期内当前缓存热度的一种计算公式：

其中，*Frequency*表示在生命周期*lifeCycle*内,缓存记录被命中的热度。*visitCount*表示当前生命周期内缓存记录被命中的次数，即当前缓存记录的COUNT字段。*α*为衰减系数，用来衡量当前生命周期和历史频率之间的关系。*α*越大，表示当前生命周期内记录被命中的次数对计算热度影响越大，反之，表示缓存记录的历史访问热度对计算当前热度的影响越大。基于热度计算公式，本模型的缓存淘汰算法流程如下：

Step 1：从Redis中读取一条记录Recordn。

Step 2：

1. **实验分析与性能评估**

为了评价基于TwemProxy的Redis索引缓存对HBase非主键查询的性能，针对本文提出的索引缓存模型，主要测试使用HBase自身非行健查询与使用本解决方案性能差异，为了使结果具有普遍性，实验将分为小数据量的查询测试和海量数据的查询测试。

1. **实验设备与环境**

为了验证基于本缓存模型的HBase进行非主键查询效率高于传统HBase数据库，我们共部署了3台计算机用来搭建HBase集群，本次测试所使用的操作环境如下所示。

**硬件环境：**

中央处理器（CPU）：Intel(R) E-1231V3.1 CPU 3.2GHZ

内存（RAM）：2GB

硬盘（Disk）：20GB

**软件环境：**

操作系统（OS）：CentOS 7.0

Hadoop版本：1.0.3

HBase版本：0.94.16

Zookeeper版本：3.4.5

JDK版本: JDK1.70

网络环境：

计算机1：

IP：192.168.56.100

计算机2：

IP：192.168.56.101

计算机3：

IP：192.168.56.102

本次实验使用“全国公安交通车辆违章记录”作为数据集，数据集共50GB数据。

1. **实验结果与分析**

在实验中，共设置了两个对照组，共四次实验记录，分别为：

1. 使用原生HBase集群对小数据集进行基于非主键的查询。
2. 使用本文提出的索引模型作为缓存，对小数据集进行基于非主键的查询。
3. 使用原生HBase集群对海量数据集进行基于非主键的查询。
4. 使用本文提出的索引模型作为缓存，对海量数据集进行基于非主键的查询。

如果没有特殊说明，测试的环境及参数均保持一致。为了模拟用户的使用行为，实验中将进行逐步提高并发的请求数量，分别记录查询出对应记录的具体时间，为了减少误差，每轮实验都将进行三次，并根据三次结果取平均值。

**对照组一：小数据集查询性能测试**

1. 原生HBase集群查询性能测试
2. 使用本文索引模型的HBase集群查询性能测试

**对照组二：海量数据集查询性能测试**

1. 原生HBase集群查询性能测试
2. 使用本文索引模型的HBase集群查询性能测试
3. **结语**

本文针对HBase非主键查询效率较低，不适合于较大规模使用场景的发展现状。本文利用Redis内存数据库的高性能、高稳定性的特性，设计并实现了一种使用基于TwemProxy代理的Redis的索引作为缓存的HBase集群，吸收LRU和LFU缓存淘汰算法的优点，提出了一种改进后的基于热度的缓存淘汰策略，由实验结果可知，改进后的HBase集群性能较原生HBase集群在非主键查询性能提高30%，且随着数据量的增大，改进后的方案优势越大。可以更好的适用于海量数据查询的应用场景。

**参考文献**

[1] Ghemawat S,Gobioff H,Leung S T.The Google File System[C].ACM SIGOPS operating systems review.ACM,2003,37(5):29-43

[2]Chang F,Dean J,Ghemawat S,et al. BigTable:A distributed storage system for structured data[J].ACM Transactions on Computer Systems(TOCS),2008,26(2):4

[3]柳皓亮.王丽.周阳辰.Redis集群性能测试分析[J].微型机与应用,2016,(10):70-71

[4]王心妍.毛莉君.基于Twemproxy的Redis集群解决方案的设计与实现[J].Electronic Test,2016,(06):70-71

[5]钱景辉.廖锂.基于Keepalived的动态浮动IP集群实现[J].化工自动化及仪表,2012,(07):926-928

[6] Salvatore Sanfilippo.Ted Nyman.Pieter Noordhuis.Redis:The Definitive Guide. O'Reilly,2011

[7] 葛微.罗圣美.周文辉等. HiBase:一种基于分层式索引的高效HBase查询技术与系统[J].计算机学报,2016,39(01):140-153