第 17 卷第 8 期　　计算机集成制造系统　　Vol. 17 No. 8
2011 年 8 月　　Computer Integrated Manufacturing Systems　　Aug. 2011

文章编号:1006－5911(2011)08－1827－07

面向云环境的图像高维特征索引框架

陈凤娟[1],丁贵广[2],朱妤晴[2]
(1.清华大学 计算机科学与技术系,北京　100084;2.清华大学 软件学院,北京　100084)

摘　要:针对海量图像数据的高维特征索引和查询方法,设计了一个面向云环境的两阶段图像高维特征索引框架,并基于 MapReduce 机制进行了系统实现。提出了一种基于位置敏感哈希函数的两阶段索引框架,可有效支持高维特征索引的分布式创建;利用 MapReduce 计算机制,设计和实现了分布式索引构建和查询算法,并集成到非结构化数据管理系统中。实验结果表明,该索引框架的查询速度随着数据规模不断增大呈亚线性增长。

关键词:高维特征索引;分布式索引;位置敏感哈希算法;基于内容的图像检索;云计算;数据管理
中图分类号:TP319.4　　**文献标志码**:A

Cloud-oriented image high-dimensional feature index framework

CHEN Feng-juan[1], *DING Gui-guang*[2], *ZHU Yu-qing*[2]
(1. Department of Computer Science and Technology, Tsinghua University, Beijing 100084, China;
2. School of Software, Tsinghua University, Beijing 100084, China)

Abstract: Aiming at high-dimensional feature indexing and searching method of mass images data in the cloud, a two-phase cloud-oriented image high-dimensional feature indexing framework was designed and was implemented based on MapReduce mechanism. A based Image high-dimensional Feature indexing Framework (LIFF) based on Locality Sensitive Hash (LSH) function was proposed to effectively support the distributed creation of massive high-dimensional feature data in the cloud. Distributed indexing and searching algorithm based on MapReduce framework was designed and implemented, which was integrated into laSQL Unstructured Data Management System (LaUD-MS). Experimental results showed that query speed on the LIFF index was sub-linear growth with the data scale increased constantly.

Key words: high-dimensional indexing; distributed indexing; locality sensitive Hash algorithm; content-based image retrieval; cloud computing; data management

0　引言

互联网上的图像等多媒体数据不断增多,而云环境可以支持海量数据的存储及分布式并行处理,并具有高可扩展、高可用等特性。在云环境中,对海量图像数据进行基于内容的检索是一个值得关注的研究方向。清华大学软件学院承担了国家"核高基"科技重大专项,开发了非结构化数据管理系统(LaSQL Unstructured Data Management System, LaUD-MS[1]),用以支持非结构化数据(如文本、图像、音频、视频)的统一管理。LaUD-MS 底层的云平台由分布式文件系统(Hadoop Distributed File System, HDFS)[2] 负责存储原始数据文件,Hbase[3]负责存储特征数据等结构化数据,MapReduce[4]提供分布式计算机制,Katta[5]负责存储文本索引数据并提供分布式关键字检索功能。此外,

收稿日期:2011-04-10;**修订日期**:2011-07-01。Received 10 Apr. 2011; accepted 01 July 2011.
基金项目:国家自然科学基金资助项目(61073005);核高基专项资助项目(2010ZX01042-002-002-01)。**Foundation items**: Project supported by the National Natural Science Foundation, China(No. 61073005), and the National Science and Technology Major Project, China(No. 2010ZX01042-002-002-01).

LaUD-MS 定义了类 SQL 的 LaSQL 操作语言，以支持对外交互。

本文结合 LaUD-MS，面向云环境中的海量图像数据查询应用需求，在深入研究位置敏感哈希函数(Locality Sensitive Hash，LSH)[6]特性的基础上，设计了一个适合大规模数据的分布式高维特征索引器，最后通过实验分析了其性能。本文的主要贡献：①设计了一种基于 LSH 的高维特征索引框架(LSH-based Image Feature indexing Framework，LIFF)，用于创建和维护海量图像数据的高维特征索引；②利用 MapReduce 计算机制，设计并实现了分布式索引创建和查询算法，并集成到非结构化数据管理系统(LaUD-MS)中。

1 相关工作

1.1 图像高维特征索引

基于内容的图像检索(Content-Based Image Retrieval，CBIR)首先需要提取高维特征，然后对高维特征建立索引来加快查询速度。不同的图像特征向量维数不一(从几维到几百维)，要求高维索引结构具有较好的扩展性，即随着维数的增加，索引能够保持较好的性能。高维特征的索引技术大致可以分成基于树的索引技术、基于近似的索引技术和基于降维的索引技术三大类。这些高维特征索引技术存在以下不足：①大多数基于树的索引结构扩展性差(当维数超过 10 维时性能急剧下降，甚至差于顺序查找，即维数灾难[7]问题)；②大多数基于树的索引机制在划分数据空间时，对数据分布做了一定的假设(如均匀分布)，通常与数据的真实分布相差甚远；③大多数高维索引结构，其空间和时间复杂度较高。

基于降维的索引技术将高维特征数据映射为仅有若干维的低维空间数据，再用降维后向量的距离来近似原始高维空间的精确距离。这是解决维度灾难问题的一个有效途径，LSH 是一种典型的降维算法，相关的数学理论[8-9]证明了其正确性。

1.2 位置敏感哈希函数

LSH 具有位置敏感性：原始空间中的相近点在经过哈希映射后，能以一定的概率保证在映射空间相近。因为原始数据空间中距离相近的点将被映射到同一个桶中，在查询时可以只搜索与查询点在同一个桶中的部分点，而不需要搜索数据集中的全部点。其优势在于，对任意纬度的高维数据的支持和查询的亚线性速度；不足之处在于，查询得到的是近似结果，可能漏掉了部分点，也可能加入了干扰点。形式化定义如下：

定义 1 位置敏感哈希函数 LSH。对于属于原始度量空间(S,D)的任意点 p 和 q，有从原始空间 S 到映射空间 U 的函数族 $H=\{S\to U\mid h\}$。对距离函数 $D(p,q)$及距离阈值 R 和概率值 $1>p_1>p_2>0$，满足以下两个条件：

$$D(p,q)\leqslant R\to Pro[h(p)=h(q)]\geqslant p_1;\quad(1)$$

$$D(p,q)>(1+\varepsilon)R\to Pro[h(p)=h(q)]<p_1。\quad(2)$$

则称哈希函数 h 是关于(R,ε,p_1,p_2)敏感的。

式(1)指 p 和 q 两点之间的距离小于等于 R 时，p 和 q 碰撞的概率至少为 p_1；式(2)指 p 和 q 两点之间的距离大于 R 时，p 和 q 碰撞的概率最多为 p_2。

本文使用位采样方法和稳定分布方法两种 LSH 构造方法。

(1)位采样方法

位采样方法(bit sampling[8])适用于定义了汉明距离(hamming distance)的 d 维二值空间$\{0,1\}^d$。哈希函数族就是 d 维坐标的一维投影，即

$$HF=\{h:\{0,1\}^d\to\{0,1\}\mid h(\boldsymbol{X})=X_i,\ i=1,2,\cdots,d\}。\quad(3)$$

式中：X_i 为向量 $\boldsymbol{X}$ 的第 i 维坐标。当 i 为$[1,d]$的独立均一分布时，从 HF 中均匀地随机选出哈希函数 h，即对 d 维坐标进行均匀采样。

(2)稳定分布方法

稳定分布方法(pStable Distribution[9])针对 p 范式度量空间距离(当 $p=2$ 时为欧氏距离)

$$D_{p\text{-}norm}(v,q)=\sqrt[p]{\sum_{i=1}^{d}(v_i-q_i)^p},\ p\in(0,2]\quad(4)$$

定义位置敏感哈希函数为

$$h_{a,b}(v)=floor\left(\frac{\boldsymbol{a}\cdot v+b}{w}\right)。\quad(5)$$

式中：a 为 norm$[0,1/R]$正态分布 d 维向量，b 为$[0,w]$均一分布的取值，w 为实数。

其基本思想是：随机选取一条稳定分布的向量 a，并将其划分为固定长度(长度为 w)的小段，将原始数据空间的两个点投影到这条向量上，再加上随机偏移量 b 之后，看其是否落入同一个小段内。由于稳定分布向量的特性，原始数据空间上邻近的点经哈希映射后在很大概率上保持邻近。

1.3　云环境中的图像检索系统

MapReduce[10]是 Google 提出的一个用于大规模数据并行运算的编程模型,其基本思想是将计算移动到数据所在地,利用分布式存储环境下的数据本地性优势,减少网络传输数据量。MapReduce 的优点是简化了大规模并行计算机制,由框架解决底层通讯、调度、错误控制等问题,使开发者能专注于数据处理上。其缺点在于:①每一个 map 和 reduce 任务的启动开销都比较大;②每一个任务都需要重新读取、分发数据,不适合需要频繁进行的工作。MapReduce 是处理海量图像检索问题的一种有效手段。下面是已有的两个系统。

CHANG 等[11]提出利用 MapReduce 并行化基于内容的图像检索的模型。其思想是简单直观的:将所有图像特征分布到 M 个节点上进行 map,计算每个图像与查询图像之间的距离,输出〈md5(imgID)-distance〉的键值对;再将键值对分配到 R 个节点上进行 reduce,即在每个节点上进行 KNN(top-k nearest neighbor)查询;最后再汇总 KNN 查询的结果。

LIU 等[12]提出了一种可应用于大规模图像集的聚类方法:先利用 MapReduce 在大图像集中快速查找近似 KNN,再将其结果应用到聚类上。其中查找近似 KNN 算法的核心包括:①并行建树,先在数据集中选取一部分作为样本,然后对样本建立决策树(metric tree),接着通过 MapReduce 方法将剩下的数据全部映射到决策树的各个叶子节点上,最后在每个叶子节点上建立溢出树(spill tree);②并行查找,给出批量查询,对于每个查询,计算其 KNN 可能存在的子树,然后将查询分配到包含相应子树的机器上,每个机器为分配到的查询计算本地的 KNN 结果,返回〈query-result〉(查询—结果集)键值对,最后针对每个查询,合并其结果集,并排序输出前 K 个作为查询结果。

2　基于位置敏感哈希函数的高维特征索引框架

本文结合 LaUD-MS 的图像检索需求,设计了一个 LIFF,可以在亚线性的时间复杂度下,在保证较高准确率的同时支持图像的高维特征检索,并且在结构上易于分布化。

2.1　基于位置敏感哈希函数的高维特征索引框架结构

LIFF 包括两个阶段(如图 1,其中实线为建立索引过程,虚线为查询过程):

(1)LSH 映射阶段　设计 L 个不同的位置敏感哈希族 LSHFamily,每个 LSHFamily 由 k 个独立的 LSH 函数运算构成,其中 LSH 函数将一个图像的高维特征值映射为一个整数值。通过第一阶段的 LSH 映射(总计 $L\times k$ 个 LSH 函数),可以得到 L 个 k 维映射值。

(2)二次哈希阶段　对 k 维映射值进行二次哈希,得到一维索引值。二次哈希的算法采用通用散列函数(如 MD5、大素数取模运算等),一方面可以把一个任意维的数据哈希成较短长度的一维数据,降低了存储和查找的复杂度;另一方面通用散列函数的冲突概率低,可以保证第一阶段产生的不同的 k 维映射值仍然被映射到不同索引值上。每个索引值构成一个桶,将该图像在 Hbase 的图像表(ImgTable)中的行地址链接到这个桶上,便形成了一个倒排表。

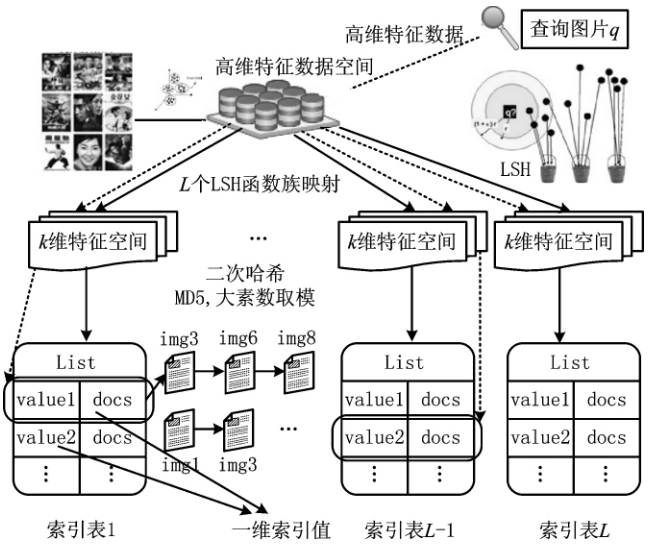

图1　LIFF索引框架

在 LaUD-MS 系统中,图像文件存入 HDFS 中,图像在 HDFS 中的地址和提取的高维特征存储到 Hbase 的图像表(ImgTable)中。基于 LIFF 框架的高维特征索引机制设计如下:

(1)创建　将高维特征数据映射为 L 个一维索引值;将索引值存入 Hbase 的 L 张索引表(IdxTable)中,相同索引值的图像在 Hbase 图像表中的行地址以逗号分隔(SrcRowList)存储在一行中。

(2)检索　首先计算查询图像的索引值;然后在 L 张索引表中查找相同的桶,获得 SrcRowList 列

表;将列表汇总去重后,获得候选集(CandidateSet);按照行地址从图像表中取出其原始特征数据,计算查询图像的高维特征与候选集图像的高维特征的精确距离;最后根据查询条件(如距离阈值 R)进行筛选,得到查询结果。查询的时间复杂度为 $O(L+\#\text{CandidateSet})$。

(3)管理索引表 Hbase 存储持久化的索引表(IdxTable)记录了索引值 IdxValue(作为 RowKey)、图像行地址列表 SrcRowList。对于每张图像表的一种特征都需建立 L 张索引表,因此随着表的增多和特征的不断丰富,索引表会越来越多。为了便于管理和统计索引表,在 Hbase 中建立了一张索引元数据表(Idx__Meta__Table),用来记录索引的表名 IdxTableName(作为 RowKey)、被索引的表名 SrcTableName、被索引的列名 SrcColName、算法类型 AlgType、索引表的行数 RecordNum。

2.2 索引创建算法

本文实现了 LIFF 的两种算法,即 IDX_BitSampling 和 IDX_PStable。

2.2.1 IDX_BitSampling 算法

IDX_BitSampling 算法首先根据式(6)和式(7)将高维特征二值化,再采用位采样 LSH 函数族(见 1.2 节)进行映射,最后采用 MD5 进行二次哈希,最终将一个高维特征映射为 L 个一维索引值。

$$\#PositiveBit[Dim] = floor(v_d \times ScaleFactor \times MaxBit) \bmod MaxBit = \{0,1,2,\cdots,MaxBit\}, d = 0,1,\cdots,Dim; \quad (6)$$

$$binaryVector[i] = \begin{cases} 1, & (i - d \times Dim) \leqslant \#PositiveBit[Dim]; \\ 0, & (i - d \times Dim) > \#PositiveBit[Dim]; \end{cases} \quad i = 0,1,\cdots,Dim \times MaxBit - 1。 \quad (7)$$

算法流程如下:

算法 1 IDX_BitSampling。

输入:参数文件 lsh.conf,一个图像高维特征 v。

输出:L 个索引值。

```
BEGIN
1. LoadConf(L,k,Dim,MaxBit,ScaleFactor)  //读取参数
2. GenerateUniformRand(Dim,MaxBit,rand[L][k])  //生成均一分布的随机向量 rand
3. GetBinaryVector(binaryVector[Dim * MaxBit])  //二值化特征数据 v
4. ComputeLSHHash(rand[L][k],binaryVector[Dim * MaxBit],hashValue[L][k])//LSH 映射
5. ComputeMD5Hash(hashValue[L][k],idxValue[L])//二次哈希
END
```

2.2.2 IDX_PStable 算法

IDX-PStable 算法首先采用稳定分布 LSH 函数族(见 1.2 节)进行映射,再根据式(8)和式(9)的大素数取模算法进行二次哈希,最终将一个高维特征映射为 L 个一维索引值。

$$h_{\text{mod}}(hv_1,hv_2,\cdots,hv_k) = (\sum_{i=1}^{k} rand_i \times hv_i) \bmod PRIME = (\sum_{i=1}^{k} h_{\text{mod}}(hv_i)) \bmod PRIME。 \quad (8)$$

式中:$rand_i$ 为[0~2^{29}]之间均匀分布的随机整数值,大素数 $PRIME = 2^{32} - 5$。

$$h_{\text{mod}}(hv_i) = (low[rand_i \times hv_i] + 5 \times high[rand_i \times hv_i]) \bmod PRIME = \begin{cases} \alpha, & \alpha < (2^{32} - 5); \\ \alpha - (2^{32} - 5), & \alpha \geqslant (2^{32} - 5); \end{cases} \quad \alpha = low[rand_i \times hv_i] + 5 \times high[rand_i \times hv_i] < 2 \times (2^{32} - 5)。 \quad (9)$$

式中:$low[rand_i \times hv_i]$ 为 $rand_i \times hv_i$(共 64 位)的低 32 位,$high[rand_i \times hv_i]$ 为 $rand_i \times hv_i$(共 64 位)的高 32 位。

算法流程如下:

算法 2 IDX_PStable。

输入:参数文件 lsh.conf,一个图像高维特征 v。

输出:L 个索引值。

```
BEGIN
1. LoadConf(R,L,k,W,Dim,SuccPro)  //读取参数
2. InitLSHFamily(L,K,a,b,LSHFamily[L][k],controlRand)  //初始化 LSH 哈希函数族
3. ComputeLSHHash(v,LSHFamily[L][k],hashValue[L][k])  //根据式(5),LSH 映射
4. ComputePrimeModHash(hashValue[L][k],idxValue[L])//根据下面公式,二次哈希
END
```

3 基于 MapReduce 的分布式高维特征索引

当图像表的数据量越来越大时,集中式的索引创建和检索都会因内存限制变得越来越慢,甚至不可行。本文利用 LaUD-MS 平台提供的 MapReduce 计算框架,将基于 LIFF 的索引创建和检索过程分布化和并行化,以适应海量数据的索引需求。

3.1 基于 MapReduce 的索引创建算法

当图像表的数据量变得很大时,可利用基于 MapReduce 的分布式索引创建。下面的 MRIdxCreate 算

法中每个 map 任务的计算时间(1 次 Hbase 读表操作和 L 个索引计算)相对于 map 的启动时间来说不大。每个 reduce 任务的计算时间主要包括组织索引散列表的时间和将其写入 Hbase 索引表的时间,后者的时间消耗是主要的。算法流程如下:

算法 3　MRIdxCreate。

输入:参数文件 lsh. conf,图像特征表 ImgTable,任务参数 jobConf。

输出:L 张索引表。

```
BEGIN
1. InitMapReduce (ImgTable, jobConf)//初始化 MapReduce 任务
        //对图像表 ImgTable 中的一行数据启动一个 map 任务
        //设置 reduce 任务的个数为 L,不输出任何东西
2. map<Key1,Value1,Key2,Value2>
   {
     InitLSHFamily(jobConf,LSHFamily[L][k]);
     //从 jobConf 中获得算法参数,初始化 LSH 哈希函数族
     ReadFromTable(Key1,Value1,srcFeature);
     //解析图像表的每一行内容即 Value1
     //得到特征数据 srcFeature 和行地址 srcRow(即 RowKey)
     ComputeIdxValue(LSHFamily[L][k],idxValue[L]);
     //利用 LSHFamily 计算出 L 个索引值
      Output (Key2, Value2, idxTableName_i, idxValue[i],
srcRow);
     //输出 L 个<Key2,Value2>=<idxTableName_i,idxValue
[i]+srcRow>
   }
3. reduce<Key3,Value3,Key4,Value4>
   {
     Combine(dist,List<HdfsAddr>);
     //将 map 的输出按 idxTableName_i 进行合并
     //即<Key3,Value3>=<idxTableName_i,List[idxValue[i]
+srcRow]>
      OrganizeHashTable(List[idxValue+srcRow],hashtable
<idxValue,srcRowList>);
     //将 List[idxValue+hbaseAddr]按照不同的 idxValue
     //组织成 hashtable<idxValue,List[srcRow]>
     WriteIdxTable(hashtable,idxTableName_i);
     //将 hashtable 写入对应的索引表 idxTableName_i 中
   }
END
```

3.2　基于 MapReduce 的图像查询算法

针对一次查询,当图像数据集变得很大时,L 张索引表的总数据量也将变得很大,无法一次性装入内存中,可利用 MapReduce 机制对 L 张索引表启动 map 任务并行化地查找,在 reduce 中汇总候选集利用精确距离和查询条件进行筛选。MRIdxSearch 算法中共有 2 次 HDFS 文件操作(上传. hv 文件和下载 r-0000 文件)及(L + # CandidateSet)次读 Hbase 表操作,是主要耗时之处。算法流程如下:

算法 4　MRIdxSearch。

输入:查询图像特征 queryData,图像特征表 ImgTable,L 张索引表,任务参数 jobConf。

输出:查询结果集 resultSet。

```
BEGIN
1. InitLSHFamily(jobConf,LSHFamily[L][k])//从 jobConf 中获得参数,初始化 LSHFamily
2. ComputeIdxValue(LSHFamily[L][k],idxValue[L])//利用 LSHFamily 计算出 L 个索引值
3. WriteIdxValue(idxValue[L], hvFile, inputPath)//将<#i,idxValue[i]>值写成 L 行到. hv 文件中,上传到 HDFS 中作为 inputPath
4. InitMapReduce(ImgTable,jobConf)//初始化,map 任务为 L 个,reduce 任务为 0 个
5. map<Key1,Value1,Key2,Value2>//map 任务的个数为 L
   {
ReadFromInputPath(Value1,idxValue,i);//解析 inputPath 文件内容(Value1),得到 idxValue 和所在索引表的编号#i
GetSrcRowList (# i, idxValue, srcRowList);//在索引表 IdxTable_#i 中查找键值为 idxValue 的行,得到文档地址列表 srcRowList
GetSrcFeatureAndAddr (srcRowList, srcFeatures, hdfsAddrs);//解析 srcRowList 得到多个文档地址,从图像表中获得特征数据 srcFeatures 和 hdfsAddrs
ComputeExactDistance(queryData, srcFeatures, R, dist);//计算 queryData 和每个原始特征数据的精确距离 dist,根据距离阈值 R 删除 dist>R 的项
        SortAndOutput(Key2,Value2,dist,hdfsAddr);//按 dist 升序输出<dist,hdfsAddr>
   }
6. reduce<Key3, Value3, Key4, Value4>//reduce 任务的个数为 1
   {
        Combine(dist,List<HdfsAddr>);//将 L 个 map 的输出按 dist 进行合并
        SortAndOutput(Key2,Value2,dist,hdfsAddr,outputPath);//按 dist 升序输出<Key4,Value4>=<dist,hdfsAddr>,写入 HDFS 的 outputPath/r-0000 文件中
   }
7. GetResultSet(outputPath, resultSet)//从 HDFS 的 outputPath/r-0000 文件中解析查询结果集 resultSet
END
```

4　实验结果分析

4.1　实验环境和评价指标

实验使用的硬件环境为 11 台由百兆局域网连

接的个人电脑。其中,每台电脑的基本配置为 CPU 主频 3 GHz,内存 4 G,硬盘 160 G。每台电脑上都安装了 Ubuntu 操作系统(v8.04)和 Java 运行环境(v6.023)。操作系统上都配置有相同的非根用户,作为运行用户。电脑节点间使用 SSH(secure shell)协议进行通讯。11 台机器部署了 LaUD-MS 系统,另有一台电脑运行索引程序。

实验考察以下三个评价指标:①查询时间(QueryTime),指从接收查询点到返回查询结果集的时间;②查准率(precision),指检索结果中与查询图像相似的图像数和图像库中所有与查询图像相似的图像数的比值;③查全率(recall),指检索结果中与查询图像相似的图像数与检索结果中图像数的比值。

4.2 索引创建实验

4.2.1 索引的存储空间代价

为测试索引的空间代价,对 1 000,10 000 和 100 000 幅图像分别用 IDX_PStable 和 IDX_BitSampling 算法创建索引表,如表 1 所示。

表 1 两种索引表的数据量与原始数据量的比值 %

图像数	IDX_BitSampling		IDX_PStable	
	均值	总量	均值	总量
1 000	19.63	196.30	20.90	209.01
10 000	6.80	68.04	9.43	94.31
100 000	17.31	173.12	13.23	132.32

索引表中一行代表一个映射桶。桶数量太少意味着 LIFF 索引的区分度不高,很多数据被映射到了同一个桶内,对于查询命中的桶获得的候选集合也大,直接影响了查询时间;而当桶的数量过多时,意味着哈希映射的聚合性不高,将过多的点过滤掉了,直接影响了查准率和查全率;若每张索引表中的数据量适中,既有一定的区分度又能保证一定的聚合性,则查询时间和查准率、查全率能达到平衡。同时 L 张表索引数据总量不应过多,否则意味着空间复杂度太高。实验结果表明,当每张索引表的数据量约为图像表数据量的 20%时,查询效果较好。

4.2.2 MapReduce 方式创建索引的时间代价

由表 2 可知,MapReduce 方式创建索引表的时间随着数据集呈亚线性增长,IDX_PStable 创建索引的速度比 IDX_BitSampling 方法快。

表 2 MapReduce 方式索引创建时间 ms

图像数	IDX_BitSampling	IDX_PStable
1 000	64 570	45 322
10 000	283 922	224 190
100 000	1 384 494	463 164

4.3 查询实验

通过三个不同数据量实验,对比 Force(顺序查找)、IDX_BitSampling 和 IDX_PStable 三种查询方法在 MapReduce 执行时的查询时间、查全率和查准率,如表 3 所示。

MapReduce 方式下的顺序查找算法(即 Force)方法,是实验的对比基线。采用的是在 map 阶段分布所有图像并计算欧式距离的方法。其 map 的计算时间(包括一次 Hbase 读表操作和一个高维向量的精确距离计算)相对于 map 的启动时间来说不大。由于针对图像表的每一行数据启动一个 map 任务,当图像表的数据量大时,map 任务的数量将变得很可观,反而导致性能下降。

表 3 MapReduce 分布式执行时三种查询方法的性能对比

查询方法	数据量=10 000			数据量=100 000		
	查询时间/ms	查全率/%	查准率/%	查询时间/ms	查全率/%	查准率/%
Force	32 148.92	31.99	6.52	127 594.9	24.39	4.52
IDX_BitSampling	41 578.43	19.70	7.92	126 572.7	14.40	8.92
IDX_PStable	33 034.22	22.27	7.20	80 499.0	17.27	8.30

如表 3 所示,当数据量为 10 000 时,Force 方法最优。IDX_BitSampling 比 Force 慢了约 30%,比 IDX_PStable 慢了约 3%。分析其原因在于:Force 方法将原始图像进行分布(启动 10 000 个 map 任务),而两种索引方法对索引表进行分布(启动 10 个 map 任务),并行度比 Force 低。当数据量为 100 000 时,IDX_PStable 的表现最优。IDX_PStable 比 Force 快了约 37%;而 IDX_BitSampling 的查询时间接近于 Force 方法。

Force 方法将图像进行分布,当图像数量进一

步增大时，启动的 map 任务数量将变得很多。可以预见，当数据集变得很大时，Force 的性能将变差；而此时索引的分布式查询方法将会体现出优势，因为它的 map 任务相对较少，且可以通过参数 L 进行控制。

4.4　实验小结

实验表明，IDX_PStable 索引方法优于 IDX_BitSampling 索引方法和顺序查找方法，在时间和效果上达到了较好的折中。基于 LIFF 框架的索引机制的优点在于：①查询时间复杂度低，随数据集呈亚线性增长 $O(L+\sharp \text{CandidateSet})$；②准确性较高，即返回的结果和线性查找的结果接近，查全率上接近、查准率上超过于顺序查找方法；③LIFF 索引结构易于并行化，MapReduce 分布式执行下的实验结果表明了分布式索引查询算法正确性，同时索引可随数据集的增大而拓展，在检索海量数据时表现出优势。LIFF 索引机制的缺点在于，索引创建的时间比较长，生成的索引总数据量比较大。

5　结束语

本文面向云环境中海量图像查询需求，设计了一个基于 LSH 的索引框架 LIFF，用以创建易于更新和分布化所提出的高维特征索引；结合非结构化数据管理系统，实现了基于 MapReduce 的分布式的图像索引和查询算法，并通过实验分析了索引性能。本文所提出的高维特征索引机制是一个初步尝试，尚有不少需进一步开展的工作，如通过自适应的参数设置来进一步提高索引算法的性能；融合多种高维特征（包括高层语义特征），拓展 LIFF 框架以适应融合特征，来达到更好的查询效果。

参考文献：

[1] ZHU Yuqing, DU Naiqiao, TIAN Hao, et al. LaUD-MS: an extensible system for unstructured data management[C]// Proceedings of the 12th International AsiaPacific Web Conference on USD Workshop. Washington, D. C., USA: IEEE, 2010: 987-994.

[2] Apache. Apache open source project HDFS[EB/OL]. [2011-03-03]. http://hadoop.apache.org/hdfs/.

[3] Apache. Apache open source project Hbase[EB/OL]. [2011-03-03]. http://hadoop.apache.org/hbase/.

[4] Apache. Apache open source project MapReduce[EB/OL]. [2011-03-03]. http://hadoop.apache.org/mapreduce/.

[5] Apache. Apache open source project Katta[EB/OL]. [2011-03-03]. http://hadoop.apache.org/katta/.

[6] HAGHANI P. Distributed similarity search in high dimensions using locality sensitive hashing[C]//Proceedings of International Conference on Extending Database Technology. New York, N. Y., USA: ACM, 2009: 744-755.

[7] CLAKSON K. An algorithm for approximate closest point queries[C]//Proceedings of the 10th Annual ACM Symposium on Computational Geometry. New York, N. Y., USA: ACM, 1994: 160-164.

[8] INDYK P, MOTWANI R. Approximate nearest neighbor: towards removing the curse of dimensionality[C]//Proceedings of the Symposium on Theory of Computing. New York, N. Y., USA: ACM, 1998: 604-613.

[9] DATAR M, IMMORLICA N, INDYK P, et al. Locality-sensitive hashing scheme based on p-stable distributions[C]//Proceedings of the ACM Symposium on Computational Geometry. New York, N. Y., USA: ACM, 2004: 253-262.

[10] DEAN J, GHEMAWAT S. MapReduce: simplified data processing on large clusters[C]//Proceedings of the 6th Symposium on Operating System Design and Implementation. Berkeley, Cal., USA: USENIX Association, 2004: 137-150.

[11] CHANG Fengcheng, HUANG H C. A programming model for distributed content-based image retrieval[C]//Proceedings of the 3rd International Conference on Intelligent Information Hiding and Multimedia Signal Processing. Washington, D. C., USA: IEEE, 2007: 210-213.

[12] LIU Ting, ROSENBER C, ROWLEY H. Clustering billions of images with large scale nearest neighbor search[C]//Proceedings of the 8th IEEE Workshop on Applications of Computer Vision. Washington, D. C., USA: IEEE, 2007: 28.

作者简介：

陈凤娟（1986－），女，福建龙岩人，硕士研究生，研究方向：非结构化数据管理，E-mail：chenfj04@gmail.com；

丁贵广（1976－），男，辽宁昌图人，副教授，研究方向：多媒体信息检索与管理；

朱妤晴（1982－），女，广东深圳人，博士后，研究方向：非结构化数据管理。