面向批量处理的大数据检索过滤模型研究

李兆兴 马自堂
(解放军信息工程大学密码工程学院 郑州 450000)

摘 要 大数据作为新的战略资源,在信息领域发挥着重要作用。大数据的检索规模往往达到十亿甚至百亿级,导致传统的查询机制效率低下成为常态。因此,提高大数据的查询效率、降低查询负担成为大数据研究的重要方面。为此提出了一种面向批量处理的大数据检索过滤模型IMFM,介绍了其核心思想及工作原理,论证了IMFM对于多维查询的支持,并给出了IMFM的部署策略。在大数据索引结构中的适当位置部署该模型,在检索请求通过节点时对检索请求进行快速过滤,避免无关请求对节点下方索引结构的操作,从而降低检索对性能的消耗。实验证明,在大数据批量处理环境下,该模型可以有效缩短大数据一维和多维查询的路径长度,提高检索效率,大幅减轻大数据存储和处理平台的负担。

关键词 大数据,检索,过滤,索引结构,多维查询
中图法分类号 TP391 **文献标识码** A **DOI** 10.11896/j.issn.1002-137X.2015.9.035

Research on Big Data Retrieve Filter Model for Batch Processing

LI Zhao-xing MA Zi-tang
(School of Cypher Engineering, PLA Information Engineering University, Zhengzhou 450000, China)

Abstract As a new strategic resource, big data plays an important role in the field of information. The scale of big data retrieval often reaches billions or even ten billions, resulting in that traditional query mechanism's low efficiency becomes regular. Therefore, improving the efficiency of big data query and reducing the burden of querying big data have become an important aspect of big data research. In order to speed up the querying of big data as well as reduce the burden, we proposed a big data retrieval filtering model IMFM of batch-oriented processing, demonstrated its support for multi-dimensional queries, and gave out the IMFM's deployment strategy. By deploying the model in the appropriate position of the index structure, IMFM can filter the search requests that pass through the node quickly to avoid that the lower node is searched, so as to reduce the consumption of retrieval performance. Experiments show that, in the batch-oriented processing of big data environment, IMFM can effectively reduce the path length of both single and multi-dimensional data queries, improve the efficiency of retrieval and reduce the workload of big data storage and processing platform significantly.

Keywords Big data, Retrieve, Filter, Index architecture, Multi-dimensional query

1 引言

随着分布式计算、云计算等概念的广泛运用,大数据[1]在越来越多的领域中得到了应用和重视。公认的大数据特点主要包括数据量(Volumes)大、数据类别(Variety)众多、处理速度(Velocity)快[2]等,这也导致无法在单一计算机甚至单一集群上对完整的大数据集合进行处理。此时,No-SQL型分布式数据库出现并得到了广泛使用,但更大的数据库容量也对大数据查询效率提出了更高的要求。另外,大数据价值的体现离不开大规模数据分析,即实现对大量数据快速、高效、及时地分析与计算,从而得出数据间潜在的规律、关系和内在逻辑,帮助用户找出复杂要素间的联系,对用户感兴趣的内容进行预测,而这也对大数据的查询提出了更高吞吐率的要求。由此可见,大数据的高效查询是体现大数据价值的基本保障。

1.1 数据检索

随着计算机在科研等领域的运用,数据查询的研究也不断进行。20世纪60年代,为帮助用户方便地找到所需数据,相关专家提出了信息检索技术。信息检索是将信息按照某种规则进行组织,进而根据用户需要找到信息的一系列过程,在计算机普及后,信息检索通常通过对资料建立的索引进行操作来实现。传统的信息检索技术主要包括基于树形结构的多层次索引和基于hash函数的哈希表索引等。但在大数据环境下,两类传统的信息索引技术都遇到了新的问题:一方面,树形结构索引会遇到节点规模过大和树形深度过大的问题,导致索引效率过低;而哈希表则会遇到索引规模过大难以维护的问题;另一方面,两类索引在分布式环境下都无法满足大数据对数据吞吐率的更高要求,因此两类方式都难以直接应用于大数据场景。

到稿日期:2014-05-15 返修日期:2014-08-02 本文受国防预研课题基金资助。
李兆兴(1990—),男,硕士生,主要研究方向为大数据,E-mail:hoopsli@qq.com;马自堂(1963—),男,教授,主要研究方向为云计算、体系结构设计与优化、信息安全等。

针对云计算环境和大数据应用场景，新的信息检索研究思路包括基于元数据的信息检索[3,4]和面向分布式系统的信息检索[4]等。

Alilla Soner Balkir 等提出了一种面向数据挖掘请求的基于 MapReduce 模型的数据检索技术[3]，通过采用全局最优化方法以整个语料库来计算本地特征，通过每一步的操作更新数据库中的参数。在实现中，他们采用了 Bloom Filter 模型和内存缓存技术，并通过 HBase 集群降低通信成本。实验证明，该方法可以有效缩短延迟时间，并降低对存储空间的需求。

吴广君等[5]则针对当前 No-SQL 数据库不支持多列查询的缺点，提出了一种海量结构化数据存储检索系统。该系统采用列存储结构，以集中分布式 B+Tree 索引和局部索引相结合的方式提高检索效率。经实验验证，该系统适用于流记录数据等海量结构化数据的应用场合。

可以看出，当前简单的数据检索方式已无法满足大数据查询的需求，另一个数据查询的思路是信息过滤技术。

1.2 信息过滤

20 世纪 70 年代，相关学者提出了信息过滤技术。类似于信息检索，信息过滤是对信息进行过滤操作，保留用户所需信息并排除不符合用户需求信息的一系列过程。过滤的程度与过滤结构的层次和组织有关，通常过滤的结构越复杂，其过滤的精度越高。现有的信息过滤技术主要应用于数据挖掘等场景，以用户需求与喜好为过滤的依据和基础，通过多层次的判定与过滤，完成信息查询的过程。但在大数据环境下，传统的信息过滤陷入了过滤规则过于庞大导致的维护困难和过滤效率大幅降低等问题。

近期的信息过滤研究工作包括对过滤算法的研究[8]，以及对获取用户知识、兴趣和行为模式的研究等[13]。

Yan T 等[9]以传播模型为切入点，研究了斯坦福大学的开源信息过滤工具(SIFT)，通过采用著名的信息检索模型提供了对全文的过滤操作，并提出了面向用户兴趣模型和服务器通信的 SIFT 方法。通过对 USENET 新闻的传播实验，考察了该方法对内存的需求，证明了该方法在信息与用户群规模的支持方面有一定突破。

Wang Meng-Fan 等[10]提出了一种 P2P 环境下基于 Counting Bloom Filter 的多关键字查询算法。通过将数据进行 Bloom Filter 编码实现了对大规模数据的压缩，并减小了通信开销。通过在 PEERSIM 上的实验，验证了该方法在多关键词查询时能够以损失部分精度为代价提高召回率并缩短查询延迟，在节点变动频繁的 P2P 中，该模型显著提高了查询效率。

1.3 信息检索请求过滤

信息检索与信息过滤的目的类似，在结构上也较为相似，抽象的信息检索与信息过滤的一般流程分别如图 1 和图 2 所示。

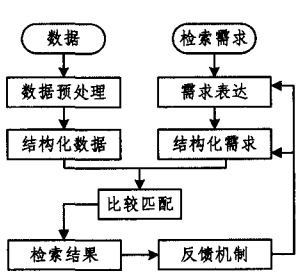

图 1 一般的信息检索流程

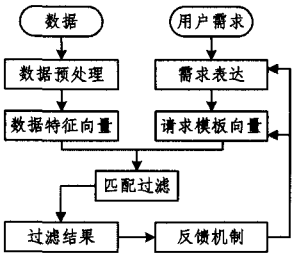

图 2 一般的信息过滤流程

由此可见，抽象后，信息过滤与信息检索的一般流程高度相似，均包括对原始数据(包含数据库中的数据和需求数据)的预处理，并将其转换为满足一定规则的数据，进而将处理后的数据进行匹配，再将匹配结果进行处理后反馈给用户。这就为信息检索和信息过滤技术的协同使用提供了便利。

显然，单一的信息过滤或信息检索均能在一定场景下满足信息查询的需求，但在大数据环境下，依靠单一的信息查询技术则难以满足大数据的新要求，信息检索、信息过滤和大数据查询需求的主要特征如表 1 所列。

表 1 信息过滤、检索和大数据查询

	信息过滤	信息检索	大数据查询需求
数据源	动态数据	静态结构化数据	大数据
目标	将无关信息过滤去除	通过检索得到相关信息	从大数据中查询所求信息
匹配规则来源	用户设计	数据累积	灵活
个性化程度	高	低	高
典型应用	推荐系统	搜索引擎	数据挖掘

由此可见，以传统方式查询数据变得非常吃力，即便通过一些优化方法提高了信息过滤或信息检索效率，其使用范围往往也十分有限。在这种情况下，分析了大数据查询相关特性，结合信息过滤和信息检索两种方法，提出了一种大数据检索请求过滤模型，并给出了模型在索引树结构上的部署策略。该模型考虑了检索操作对于系统负荷的消耗，在检索结构上引入大数据的检索请求过滤机制，通过对检索请求的过滤避免匹配无关需求对于大数据存储和查询系统资源的消耗，减轻系统负担；同时通过过滤模型的特性实现了对热点元素的直接定位，从而降低了检索的时间开销。实验证明，在大数据环境下，该模型可以有效降低系统负荷，在用户检索请求不能全部与数据库中元素相匹配时可以降低检索的时间开销。

可见，上述方法中还存在如下问题：

(1)未对系统负荷做出优化

Esper 公布的数据表明：CPU 是影响数据处理性能的关键资源。在大数据环境下，如果不对激增的庞大规模运算进行优化，直接由计算机进行暴力运算，很容易导致大数据处理系统不堪重负，影响数据处理速度。

(2)未能兼顾数据精度与速度

数据查询的精度与速度是数据查询最关键的两项指标。但上述方法或通过对数据的压缩编码来减少通信开销，实现压缩索引空间的目的，从而损失了部分精度；或基于 MapReduce 编程模型，通过集中与分布式索引相结合的方式保证查询精度，并在一定程度上提高查询速度，但由于 MapReduce 模型[12]每次运行中都需要将中间结果写入磁盘，导致实际上查询效率非常低下。

针对上述问题，本文提供了一种结合数据检索与数据过滤的大数据查询办法，结合数据检索与数据过滤的特点，面向

大数据批量处理[12]需求，提出了一种弱化了精度需求的多规则信息过滤模型 IMFM，其通过高速的数据过滤操作排除不符合规则的查询请求，降低系统负担；依靠数据检索保证查询精度，从而在保证数据查询精度的前提下提高数据查询速度。

2 检索请求过滤模型

本节主要介绍面向批量处理的大数据检索请求过滤模型。首先设计一种多规则信息过滤模型 IMFM，提出并论述了 IMFM 的体系结构及引入 IMFM 的检索流程。

2.1 一种多规则信息过滤模型 IMFM

以往的研究往往把数据过滤作为一套完整的数据查询方法，要求精确屏蔽无用信息，通常通过将用户查询请求和数据映射到同一个属性域中进行对比实现[13]。典型的信息过滤往往呈现多层次结构，从而保证对无用信息的精确和完全过滤。但对精度的要求也会增大时间和空间的开销，从而导致应用在不要求过高精度场合中的效率较低。

本文提出的大数据检索请求过滤模型中，信息过滤模块的作用是筛选无用检索请求，目的是减轻系统负荷，但传统的信息过滤机制的运行将为系统带来额外负担。因此，本文提出了一种规则弱化了的高效信息过滤模型 IMFM，它不要求过滤掉所有用户不感兴趣的信息，而是以损失部分精度为代价尽可能高效地过滤掉更多用户不感兴趣的信息。

2.1.1 基本思想

定义 1(数据集合的规模，Scale) 对于数据集合 S，数据集合 S 的规模[14]是它所包含的元素个数：$Scale=sizeof(s)$。

IMFM 的基本思想是通过一定数量的规则由用户需求驱动的映射函数 $map:V\leftarrow A$ 将集合 $A=\{a_1,a_2,a_3,\cdots,a_n\}$ 表示为一个 y 维数组 $V=\{v_1,v_2,v_3,\cdots,v_y\}$，初始阶段 V 中的每个元素均为 0，对于每一个集合 A 中的元素 a_i，利用映射函数 map 映射到数组 V 中的 x 个位置，并将相应位置上的元素＋1，在删除元素时则对相应位置上元素－1，从而完成对动态集合 A 的压缩映射表示。在单一规则环境下，判定元素 α 与集合 A 的隶属关系时，只需使用映射函数 map 作用于元素 α 并查看 $map:V\leftarrow\alpha$ 是否存在 0 元素，若不存在 0 元素则认为 $\alpha\in A$；否则，反之。由于该模型能够通过映射实现对数据的压缩，且能够将多规则场景的过滤请求转化为在各规则下目标集合的求交运算，且计算的时间复杂度不随数据集规模的增大而发生变化，因此该模型的优势在于规则自由度高、对数据的压缩表示、多规则过滤和常数时间开销。这里，n 是一个常量，这就要求能对该模型能够处理的集合尺寸进行预先判定。IMFM 的基本思想如图 3 所示。

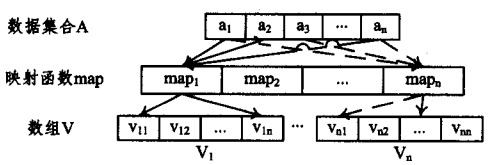

图 3 IMFM 的基本思想

显然，由于 IMFM 通过有限函数的映射操作并将结果与规则逐位比较完成过滤，IMFM 运行时所需时间等同于进行有限次函数运算的时间，是一个常量，相对于进行检索所需时间是一个极小值，即其时间复杂度为 $O(n)$。

现介绍多维查询过滤机制：

在分布式数据库中的数据通常具有多维属性[15]，如视频数据的规模、时长、压缩率、帧率等。本文通过 IMFM 将数据 a_i 映射到一维数组，记作：

$$a_i=\{v_{i1},v_{i2},v_{i3},\cdots,v_{ix}\}$$

其中，$i\in(1,y)$。

为使 IMFM 支持多维查询，本文的解决思路是将维度为 $j\in(1,\infty)$ 的多维查询 Q_{dj} 转换为多个并行维度上的查询求交运算，并建立多个并行数组 $V_1,V_2,\cdots,V_j$ 与多维查询的维度一一对应。若令对 Q_{di} 的数组 V_i 的过滤结果为 R_i，则多维查询的最终结果 $R_m=\bigcap\sum_{i=1}^{j}R_i$，从而实现了在不丢失各维信息的前提下对多维查询的并行处理，保证了处理结果的完备性[16]。为降低系统负荷，当任意过滤子进程出现不匹配情况时，过滤操作即中止，认为数据集合中不存在完全匹配的元素。

对于集合 $A=\{a_1,a_2,a_3,\cdots,a_n\}$ 的 IMFM m 维映射和过滤规则匹配算法表示如下：

```
Function Formula. Create( )
For(i=1;i<m,i++)
  Create Array Vi=(vi1,vi2,vi3,…,viy);
Define mapi(V,e)
for(r=1,r<x;r++) random r∈(1,n)
vim←map(ei)
For(j=1;j<y;j++);vij==0;
For(l=1,l<n;l++)
Map(Vi,ei);
End
Function Formula. Filter( )
Define Array Q=(q1,q2,q3,…,qz)
For(t=1,t<z,t++)
Map(Q,QueryRequest x)
For every qt=1;
If vit!=0;
return QueryRequest x∈A
else return 0;
```

下面对 IMFM 中涉及的若干参数进行讨论：不失一般性，令 map 函数的值域空间维度为 y，每个元素被映射到 x 维上，数据集合 $S\{a_1,a_2,\cdots,a_n\}$ 的规模为 n，一定时间内查询请求集合 $Q\{q_1,q_2,\cdots,q_m\}$ 的规模为 m，其中成功检索到相应目标的检索请求集合为 $Q_S\{q_{s1},q_{s2},\cdots,q_{sa}\}$，在数据集合 S 中不存在对应目标元素的检索请求集合为 $Q_u\{q_{u1},q_{u2},\cdots,q_{ub}\}$，通过 IMFM 的检索请求集合为 $Q_p\{q_{p1},q_{p2},\cdots,q_{pc}\}$，未能通过 IMFM 的检索请求集合为 $Q_f\{q_{f1},q_{f2},\cdots,q_{fd}\}$。

定义 2(假通过率，FalsePassRate(FPR)) 由于映射函数的不确定性，在两种情况发生时该模型可能会作出错误判断：(1)两个不同的元素被映射到数据 V 中的相同位置；(2)某元素被映射到的维度全部被数据集合所覆盖。在此称此类模型发生错误判断，将不属于某集合的元素与集合的隶属关系进行错误判断的概率称为假通过率。IMFM 的假通过率为无关结果通过过滤器的概率，即

$$FPR=\frac{sizeof(I_p-I_s)}{sizeof(I_p)}$$

由于各个数组 V_i 中均可能出现多个元素映射结果相同的情况，而该情况则会导致过滤结果冗余，因此本文将 IMFM

模型应用于层次索引的一定层次中，通过层次索引结构保证查询精度。

2.1.2 部署策略

大数据检索请求过滤模型的部署策略如下：对于大数据集合 $S\{a_1,a_2,\cdots,a_n\}$，每个 $a_i(i=1,2,\cdots,n)$ 对应大数据索引结构中的一个叶节点 $L_i(i=1,2,\cdots,n)$。选择索引结构上适当深度的节点作为大数据集合 S 的 IMFM 部署节点（可由预测大数据集合 S 的规模计算得到，在数据进行大的变动时可以进行重新计算），并称此类节点为 FilterNode。

索引的叶节点 $L_i(i=1,2,\cdots,n)$ 都在 FilterNode 上有备份，为提高并行计算能力、容错性和可用性，一个大数据集合 S 有多个互相平行的 FilterNode。由于大数据的分布式存储通常有多个备份，因此叶节点的局部失效不会影响大数据检索请求过滤模型的可靠性和可用性。

对于某一个 FilterNode，选择下层节点组 $U_i(i=1,2,\cdots,r)$ 作为其上索引叶节点的存储节点，并称此类下层节点为 PathNode，则此类节点所能提供的索引空间大小应满足 $\sum_{i=1}^{r}|U_i|\geqslant\sum_{i=1}^{n}|L_i|$（此处 $|U_i|$ 和 $|L_i|$ 分别表示各自对应的空间）。

IMFM 的关键参数包括：多维查询的维度 k，数组维度 y，目标元素集合的规模 n，每个元素被映射到的维度 x，随机请求通过 IMFM 的概率 p。针对不同场景，参数的选择标准均有所区别，首先找出参数间的联系：

$$p=(1-(1-\frac{x}{y})^{n-1})^{x^k}$$

计算过程如下：

先令 $k=1$，将多维查询简化为一维查询。此时可将该问题转换为如下的数理统计问题：每次从规模为 y 的集合中随机抽取 x 个元素并标记，重复 n 次后，求第 $n+1$ 次随机抽取时存在未被标记元素的概率。此问题经递推后，可得出如下关系式：

$$p=1-C_y^x*C_{y-x}^x*C_{y-2x}^x*\cdots*C_{y-nx}^x=$$

$$\frac{y*\cdots*(y-x+1)*y\cdots*(y-2x+1)*\cdots y*\cdots*(y-nx+1)}{x!*x!*\cdots x!}$$

$$=\frac{y^n*(y-x+1)^{n-1}*\cdots(y-(n-1)x+1)^2*(y-nx+1)}{x!^n}$$

$$=(1-(1-\frac{x}{y})^{n-1})^x$$

当 $k\neq1$ 时，对应多维查询情况，多维查询的最终结果 $R_m=\bigcap\sum_{i=1}^{j}R_i$，此时有

$$p=(1-(1-\frac{x}{y})^{n-1})^{x^k}$$

计算过程完毕。

根据数据集合 S 规模能否精确预测，IMFM 的部署可以分为 PredictableDeploy（PD）策略和 UnpredictableDeploy（UD）策略。

· PD 策略

当大数据集合 S 存储集群中计算机数量和存储空间规模相对稳定时，由于大数据通常进行分块存储（以 Hadoop[17] 为例，默认数据块（block）的大小为 64MB），大数据集合 S 的规模通常是可以进行较为准确的预判的，此时可以执行 PredictableDeploy（PD）策略：预先确定索引层次结构及上层节点的位置，并在该层节点上部署大数据检索请求过滤模型。这种方式可以在索引结构建立时就在最佳位置部署该模型，减少了模型的迁移，但该策略的适用环境存在一定的局限性。

对于此类情况，可直接根据对 IMFM 的过滤速度、假通过率等参数的要求及索引结构的层次位置情况进行计算得出最优位置并加以部署，在此不做论述。

· UD 策略

当大数据集合 S 存储集群中计算机数量或存储空间规模变动较为频繁时，无法对 S 的规模做出较为准确的预判，此时可以执行 UD 策略：在索引结构达到一定规模时选择当前的上层节点位置并部署大数据检索请求过滤模型，当索引结构发生较大变化时重新评价索引结构并判断上层节点位置，将大数据检索请求过滤模型迁移至新的最佳位置并再初始化。UD 策略需要考虑 IMFM 的位置最优解和迁移过程中模型的可靠性、可用性和时间开销。

模型需要进行迁移时可分为以下两种情况进行讨论。

（1）模型下方节点数目过少，元素规模小于阈值

模型下方节点数目减少至阈值以下时，直接进行检索操作效率较高，过滤操作成为检索时的额外负担，可考虑将类似模型节点合并。伪码如下：

```
Model. Combine( )
For every Model_i ∈ RooNode
While sizeof(LeafNode_i)<min sizeof(LeafNode)
Get V_i=(v_i1, v_i2, v_i3, ···, v_iy)
For every j!=I;
If there is another Model_j ∈ RooNode
sizeof(LeafNode_j)<min sizeof(LeafNode)
and sizeof(LeafNode_i)+sizeof(LeafNode_j)<max sizeof(LeafNode)
new Array V_n=(v_n1, v_n2, v_n3, ···, v_ny)
for(g=1,g<y,g++)
v_ng=v_ig=v_jg
New ModelNode n;
End;
```

（2）模型下方节点数目过多，过滤精度低于阈值

当模型下方数据规模不断增大导致节点数目过多时，模型过滤精度下降，当精度低于阈值时，模型需进行重新部署。

一个思路是对模型进行分裂操作，从而增大模型所能容纳的最大元素个数，但模型分裂时将导致至少一半元素执行函数映射和规则删减操作，将对系统造成较大负担，在批量处理时将导致系统负担过重、存储和检索效率过低。因此在该情况下，本文的策略是增大节点上方节点的扇出，新建一个同层次节点并部署 IMFM。伪码如下：

```
Model. New( )
For every Model_i ∈ ParentNode_i
While max sizeof(LeafNode)< sizeof(LeafNode_i)
New ModelNode n
New Model_n
Link n to the ParentNode_i;
Create Array V_n=(v_n1, v_n2, v_n3, ···, v_ny);
for(j=1,j<x;j++)
v_nj==0;
End;
```

2.1.3 体系结构

大数据检索请求过滤模型的框架如图 4 所示，模型主要

由检索请求接口、检索读取接口、过滤结果接口和检索请求管理器、过滤规则管理器、部署位置管理器和过滤处理器组成。

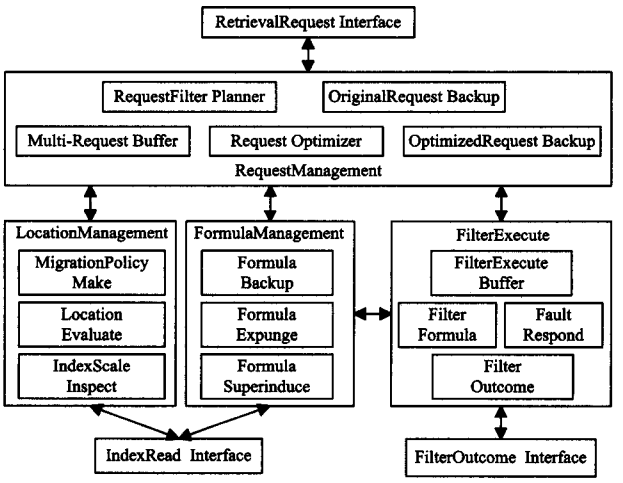

图 4 体系结构

(1)检索请求管理器

检索请求管理器用于管理从检索请求接口获取的检索请求,在获取检索请求后,先通过检索请求备份和规划模块分别进行检索请求的备份和请求处理的规划,通过请求优化模块对待过滤请求进行优化处理,并在已优化请求备份模块中备份;如果同时获取的检索请求过多,则将请求队列放入请求缓冲区中,以保证检索请求队列的连续可靠。

(2)过滤规则管理器

过滤规则管理器是大数据检索请求过滤模型的重要部分,负责对模型的过滤规则进行维护、备份和增减。它通过索引读取接口获取索引结构的变化,由此进行过滤规则的增添和删减;同时通过过滤规则备份模块对过滤规则进行维护并保证做出改动时过滤规则的完整性,从而保证大数据检索请求过滤模型的正常有效运行。

(3)过滤处理器

过滤处理器是大数据检索请求过滤模型的核心,对检索请求的过滤操作即在过滤处理器中进行。它从检索请求管理器中获取检索请求,从过滤规则管理器中获取过滤规则,并以该规则对它所获取的检索请求进行过滤处理。当故障出现时,故障响应模块对其进行响应处理:若过滤规则出现故障,则直接对过滤输出模块发送检索请求通过过滤的指令;当检索请求出现错误时,则由过滤处理器重新向检索请求管理器索取检索请求。最终过滤结果存储在过滤输出模块中,并通过过滤输出接口输出过滤结果。

(4)模型位置管理器

位置管理器的主要功能是通过检索读取接口获取检索结构信息,从而对大数据检索请求过滤模型的位置进行评估,为模型提供部署策略,并在必要时制订模型迁移策略。它确保了大数据检索请求过滤模型始终处于最佳运行位置。

(5)负载检测模块

负载检测模块直接连接的输入端是查询管理模块,输出端是过滤输出接口,是针对热点请求的快速处理通道。负载检测模块通过对热点数据节点的统计检测,并在模块中加入热点数据存储位置、物理地址等独有信息,建立模块与数据节点间的直接联系。在此类请求出现时,直接由该模块转向目标节点,从而解放路径节点压力,避免数据倾斜状况的发生,使索引结构整体负载均衡。

2.2 部署 IMFM 的索引树检索流程

在引入 IMFM 的索引系统中进行检索请求的流程如图 5 所示。检索请求首先被发送到 RootNode,然后从 RootNode 开始逐级向下进行,发送给各个上层节点,并在此类节点上执行检索请求的过滤工作,若某检索请求通过了过滤,则继续向该上层节点的子节点执行,至根节点结束,并返回遍历结果;若检索请求未能通过某上层节点的过滤,则认为该请求所对应的元素不存在于该节点的子节点范围内,在该节点上的检索任务到此终止。最终将所有结果汇总后返回该检索请求的发起者,一次检索过程完成。

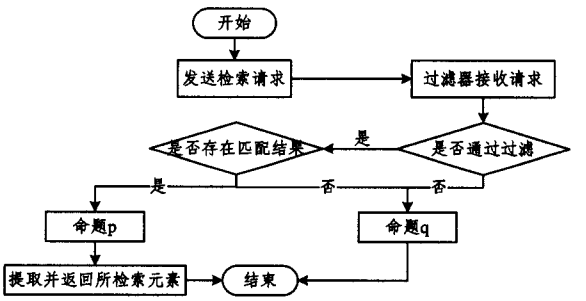

图 5 引入过滤模型的大数据索引流程

由此可知,该模型通过增加一次过滤处理的方式将检索请求划分为两类:α,即通过过滤器的请求;β,即被过滤器过滤掉的请求。α 类请求在通过过滤器后由服务器向索引数据库发出遍历请求,以执行遍历操作(流程同图 4),β 类检索请求在过滤器完成过滤后由服务器直接返回 $x \notin S$,即数据集合 S 中不存在用户检索元素,从而使索引数据库和服务器从 β 类检索请求中解放,降低此类请求对系统性能和带宽的消耗。

3 实验和性能分析

本节通过借助 MatLab 和 Hadoop 平台进行仿真实验评估数据检索性能。在考察查询效率时,选取路径长度(path length)、索引构建时间和查询时间作为指标,路径长度在本文中被定义为查询请求转发经过的节点跳数[18](hops)。本节通过设置数据集合规模、检索请求过滤、检索请求有效率和多维查询维数等参数发生的变化,来比较本文提出的大数据检索请求过滤和传统树形索引结构的查询路径长度。

在搭建 Hadoop 平台时,选取 2 台浪潮 NF5240M3 服务器(CPU:Xeon E5-2407,内存:4GB DDR3,硬盘:1TB 7200 转)作为 NameNode,4 台 Lenovo B4360(CPU:赛扬 G1620,内存:2GB DDR3 1333MHz,硬盘:500G 7200 转)作为 DataNode,彼此以 100M 带宽互联组成硬件环境。Hadoop 平台选取 2013.10.1 发布的 2.2.0 版本,在 Ubuntu12.04 系统上搭建,为在实验室中模拟大量数据集环境,将 Hadoop 数据块大小由默认的 64MB 修改为 64kB。

3.1 关键实验参数定义及选取

首先对性能分析时引入的若干参数做出如下定义。

定义 3 查询路径是指一次查询操作在索引结构中经过节点构成的路径的总长,其在数值上等于经过节点数量,若令查询操作经过的节点集合为 $N=\{N_1,N_2,\cdots,N_q\}$,则有 $QP=sizeof(N)$。Average Valid Query Path(AVQP)(平均有效查询路径)是指从用户发送检索请求到从数据集合中返回所检索元素的值的平均路径,当查询请求集合为 $Q=\{q_1,q_2,\cdots,$

$q_m\}$时，平均有效查询路径为

$$AVQP=\frac{(sizeof(Q_S)+sizeof(Q_u))*sizeof(N)}{sizeof(Q_S)}$$

引入大数据检索请求过滤模型时，令检索请求集合 Q 中成功检索到相应目标的请求所占比例为 f_q，则

$$f_q=\frac{sizeof(Q_S)}{sizeof(Q)} \tag{1}$$

若令经过过滤后的检索请求中有效请求所占比例为 f_a，则 f_a 满足的约束如下：

$$f_a=\frac{\frac{sizeof(Q_S)}{sizeof(Q)}}{\frac{sizeof(Q_S)}{sizeof(Q)}+(FPR)*(1-\frac{sizeof(Q_S)}{sizeof(Q)})} \tag{2}$$

而

$$FPR=\frac{sizeof(Q_p-Q_S)}{sizeof(Q_p)} \tag{3}$$

对于单次检索请求，则有：

①引入模型前，平均有效查询路径

$$AVQP_0=\frac{sizeof(Q_S)\sum_{i=1}^{m}QPL_i}{sizeof(Q_S)+sizeof(Q_u)} \tag{4}$$

②引入模型后，最终有效平均查询路径则包含过滤器完成过滤等效路径和通过过滤器的查询请求完成查询的平均路径，有效平均查询路径应为

$$\begin{aligned}AVQP_1&=\frac{1}{f_a}+o(n)\\&=\frac{sizeof(Q_S)\sum_{i=1}^{m}QPL_i}{sizeof(Q_S)+sizeof(Q_u)}*[\frac{sizeof(Q_S)}{sizeof(Q)}+\\&(FPR)*(1-\frac{sizeof(Q_S)}{sizeof(Q)})]+o(n)\end{aligned} \tag{5}$$

3.2 实验设计

3.2.1 基于 MatLab 的实验

实验 1 比较 IMFM 在索引位置的不同深度时的查询效率。随机生成一维查询任务 $Q1$，根据大数据存储系统的实际情况，使数据规模与节点个数同步变化。其中检索请求有效率设置为 80.0%，数据集合的规模设置为 2^{30}，检索请求集合 I 的规模设置为 2^{20}，IMFM 值域空间维度设置为 2^{20}，每个元素被映射到的维度设置为 1。

如表 2 所列，IMFM 的效果随着所处索引结构深度的变化而显著变化，这是由于 IMFM “管辖”的数据集合规模随着 IMFM 所处深度的变化增大或减小，这将导致检索时路径长度的变化；但另一方面，IMFM “管辖”的数据集合发生变化使得过滤的假通过率随之变化，影响过滤精度。

表 2 QPL 与 h 和 x 的关系

x/h	1	5	10	15	20	25	30
1	2.8147	0.3518	0.0055	0.0002	0.0006	0.0176	0.5629
2	0.0549	0.1947	0.054	0.0019	0.0055	0.1759	5.6295

此外，随着每个元素被映射到的维度 x 的变化，IMFM 最佳位置深度逐渐减小。这是由于随着 x 的变化，IMFM 在维持相等过滤假通过率时所能容纳的元素个数呈正相关。为此，在本节实验中，x 的默认值设置为 1。

实验 2 数据规模 s 与索引结构规模的关系。比较 IMFM 最优部署前后树形结构索引在数据规模变化时的数据查询效率(见式(4)、式(5))，其中 IMFM 函数目标维度 y 设置为 2^{20}，查询请求规模设置为 2^{10}，元数据维度设置为 4，多维查询选择率 k 设置为 1%，模拟在数据规模发生变化时查询路径长度的变化。实验结果如图 6 所示，其中横轴为数据规模 $Scale$ 的对数 $log2(Scale)$，纵轴为查询路径长度。

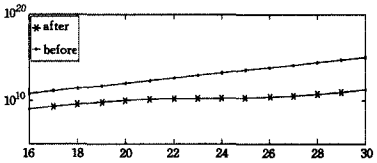

图 6 log2($Scale$)/AVQP 测试

由此可见，部署 IMFM 后，进行查询时查询路径随数据规模的增大变化相对缓慢，且始终低于直接对索引结构进行查询，当数据规模增大时，查询路径减小的幅度随之增大。

实验 3 比较 IMFM 部署前后在多维查询的维数变化时的查询路径长度。数据集合规模 $Scale$ 设置为 2^{30}。实验结果如图 7 所示。其中横轴为多维查询的维度，纵轴为查询路径长度。

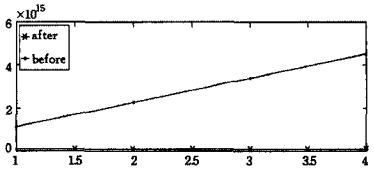

图 7 查询维度/AVQP 测试

由此可见，部署 IMFM 后，进行多维查询时 AVQP 基本不会随着查询维度的变化而改变，这是因为在查询维度变化时，IMFM 的过滤通过率随之呈指数降低。相似地，当元数据维度发生变化时，IMFM 部署前后的查询路径呈相似规律，在此不再进行论述。

3.2.2 基于 Hadoop 的实验

实验 4 比较针对不同维度的数据集，将部署 IMFM 前后的分布式 B-Tree[19,20] 索引结构与 HBase[21-23] 的哈希表索引以及传统的 B-Tree 索引结构相比较，得出的实验结果如图 8 所示。其中纵轴为索引构建时间，单位为 ms。

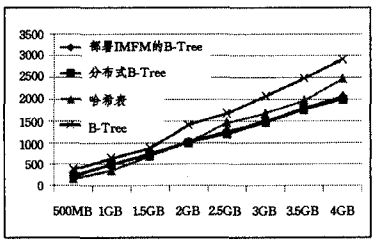

图 8 索引建立时间/数据集合规模

由此可见，部署 IMFM 前后，分布式 B-Tree 索引结构的构建时间并未发生明显变化，仍优于传统的 B-Tree 索引结构，在数据规模较大时优于 HBase 的哈希表索引。这是因为在进行索引创建时，制约速度的主要因素是磁盘的寻址开销，而 IMFM 所带来的 CPU 额外开销被控制在较小的区间内，并未给索引构建时间带来明显影响。在数据规模较小时，哈希表索引层次简单，易于建立，而数据规模较大时，哈希表的建立和维护都为计算机的计算性能带来了较大负担，效率开始低于分布式 B-Tree 索引结构。

在同一维度情况下，对数据规模发生变化时进行的实验与实验 4 结果一致，部署了 IMFM 的索引结构构建时间仍然

近似等于分布式 B-Tree 索引结构的构建时间，且优于传统的 B-Tree 结构索引，在此不再赘述。

实验 5　在数据集合规模分别取 500GB、1TB、1.5TB、2TB、2.5TB、3TB、3.5TB、4TB，查询集合规模取 100，查询维数设置为 1 的情况下，观察 IMFM 部署前后索引的平均单次查询速度。得出的实验结果如图 9 所示。其中纵轴为平均查询时间，单位为 ms。

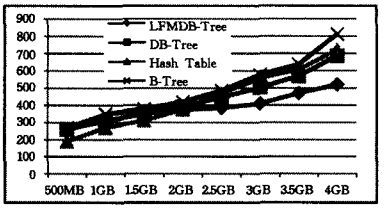

图 9　平均查询时间/数据集合规模

由此可见，在查询请求规模较小、系统无批处理负担时，部署 IMFM 后索引结构的单次查询速度与分布式 B-Tree 基本一致，在数据集合规模较大时略高于哈希表和 B-Tree。

实验 6　比较在查询请求集合规模不同的情况下，数据规模控制为 2TB 时，IMFM 部署前后索引的平均查询速度。得出的实验结果如图 10 所示。其中纵轴为平均查询时间，单位为 ms。

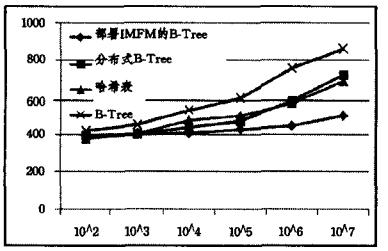

图 10　平均查询时间/查询请求集合

由此可见，部署 IMFM 后，在查询请求集合规模增大时，数据查询的平均速度优于部署前，且随着查询请求集合规模的进一步增大，部署 IMFM 所带来的效率提高幅度更大。这是由于部署 IMFM 后索引结构节点负载减小，所能承受的批处理查询请求规模更大，查询效率更高。与 HBase 的哈希表索引相比时，部署 IMFM 的分布式二叉树索引能够在数据规模较大时提供明显的效率增幅。

结束语　本文提出了大数据检索请求过滤模型，通过对检索请求的过滤操作，显著规避了检索请求对无关索引区域的遍历操作，降低了在索引规模过大时频繁在索引上进行检索操作对系统造成的不必要负担，并能够在一定程度上缩短检索所需时间，提高检索效率。本文还设计并简要分析了大数据检索请求过滤模型的部署、初始化和改动策略，使得大数据检索请求过滤模型具有广泛的可用性和较高的稳定性；并通过实验室环境中进行基于 Bloom Filter 和 Hadoop 平台的一种实现，证明了大数据检索请求过滤模型适合在大数据系统中使用。

今后的研究将集中于 IMFM 在不同需求环境中的最优化部署策略以及更适合部署 IMFM 的索引结构等，以使模型实现效果更优。

参 考 文 献

[1] Manyika J, Chui M, Brown B, et al. Big data: The next frontier for innovation, competition and productivity [R]. McKinsey Global Institute Report, 2011

[2] Howe D, Costanzo M, Fey P. Big data: The future of biocuration [J]. Nature, 2008, 455: 47-50

[3] Balkir A S, Foster I, Rzhetsky A. A Distributed Look Up Architecture for Text mining Applications using MapReduce [C]// Conference on High Performance Computing Networking, Storage and Analysis, SC 2011. 2011

[4] Andrew W, Shao Ming-long, Bisson T, et al. Spyglass: Fast scalable metadata search for large-scale storage systems [C]// USENIX. 2010

[5] 吴广君，王树鹏，陈明，等. 海量结构化数据存储检索系统[J]. 计算机研究与发展，2012，45(1)：1-6
Wu Guang-jun, Wang Shu-peng, Chen Ming, et al. Massive Structured Data Oriented Storage and Retrieve System [J]. Journal of Computer Research and Development, 2012, 45(1): 1-6

[6] Hua Yu, Jiang Hong, Zhu Yi-feng, et al. SmartStore: a new metadata organization paradigm with semantic-awareness for next-generation file systems [J]. ACM, Portland Oregon, USA, 2009, 32(1): 1-12

[7] Belkin N J, Croft B B. Information filtering and information retrieval: two sides of the same coin? [J]. Communications of the ACM, 1992, 35: 29-38

[8] Lieberman H, Van Dyke N W, Vivacqua A S. Let's browse: A collaborative web browsing agent [C]// Proceedings of the 1999 International Conference on Intelligent User Interfaces (IUI'99). 1999: 65-68

[9] Yan T, Garcia-Molina H. SIFT-A tool for wide-area information dissemination [C]// Proceedings in 1995 USENIX Technical Conference. 1995: 177-186

[10] Wang Meng-fan, Zhang Da-fang, Tian Xiao-mei. Multi-keyword search for P2P based on Counting Bloom Filter [C]// 2011 International Conference on Networking and Information Technology (IPCSIT). 2011

[11] 覃雄派，王会举，杜小勇，等. 数据分析-RDBMS 与 MapReduce 的竞争与共生[J]. 软件学报，2012，23(1)：32-45
Qin Xiong-pai, Wang Hui-ju, Du Xiao-yong, et al. Big Data Analysis-Competition and Symbiosis of RDBMS and MapReduce [J]. Journal of Software, 2012, 23(1): 32-45

[12] 亓开元，赵卓峰，房俊，等. 针对高速数据流的大规模数据实时处理方法[J]. 计算机学报，2012，35(3)：477-490
Qi Kai-yuan, Zhao Zhuo-feng, Fang Jun, et al. Real-Time Processing for High Speed Stream over Large Scale Data [J]. Chinese Journal of Computers, 2012, 35(3): 477-490

[13] Boyd D, Crawford K. Critical Questions for Big Data [J]. Information, Communication & Society, 2012, 15(5): 662-679

[14] Brinkmann B H, Bower M R, Stengel K A. Large-scale electrophysiology: acquisition, compression, encryption, and storage of big data [J]. Journal of Neuroscience, 2009, 180(1): 185-192

[15] Li Ai-guo, Zhang Chi, Zhang Jiu-long, et al. A Balanced Multiway Search Tree for Multi-Dimension Searching [J]. Applied Mechanics and Materials, 2010, 44-47: 3574-3578

[16] 王珊，王会举，覃雄派，等. 架构大数据：挑战、现状与展望[J]. 计算机学报，2011，34(10)：1741-1752

Wang Shan, Wang Hui-ju, Qin Xiong-pai, et al. Architecting Big Data: Challenges, Studies and Forecasts[J]. Chinese Journal of Computers, 2011, 34(10): 1741-1752

[17] Karun K A, Chitharanjan K. Locality Sensitive Hashing based Incremental Clustering for Creating Affinity Groups in Hadoop-HDFS-An Infrastructure Extension[C]// 2013 International Conference on Circuits, Power and Computing Technologies(ICCPCT'2013). 2013

[18] Abouzeid A, Bajda-Pawlikowski K, Abadi D J, et al. HadoopDB: An architec-tural hybrid of MapReduce and DBMS technologies for analytical workloads[C]// Proceedings of the 35th International Conference on Very Large Data Bases(VLDB'09). Lyon, France, 2009: 733-743

[19] Fusco E G, Pelc A. Distributed tree comparison with nodes of limited memory[J]. Networks, 2012, 60(4): 235-244

[20] Dehne F, Kong Q, Rau-Chaplin A. A distributed tree data structure for real time OLAP on cloud architectures[C]// 2013 IEEE International Conference on Big Data. 2013: 499-505

[21] Taylor R C. An overview of the Hadoop/MapReduce/HBase framework and its current applications in bioinformatics[C]// Proceeding of the 11th Annual Bioinformatics Open Source Conference(BOSC). 2010

[22] Sun Jian-ling, Jin Qiang. Scalable RDF store based on HBase and MapReduce[C]// 2010 3rd International Conference on Advanced Computer Theory and Engineering (ICACTE). 2010: 633-636

[23] Nishimura S, Das S, Agrawal D, et al. MD-HBase: A Scalable Multi-dimensional Data Infrastructure for Location Aware Services[C]// 2011 12th IEEE International Conference on Mobile Data Management(MDM). 2011: 7-16

(上接第 153 页)

种新的基于分形加权维数的方法来识别主用户信号和 PUE 攻击用户信号。由于用户的指纹特征通常隐藏在包络信息中，因此文中通过对接收到的用户信号进行脉内包络信号的提取，进而研究其分形特征，给出了切实可行的识别方案。

实验环节通过 MATLAB 对理论算法进行验证，证实了主用户和 PUE 攻击用户的辐射源指纹特征寄生调制在包络信息中，并且复合型加权分形维数对用户的指纹特征的识别具有可靠性。因此，本文对 PUE 攻击检测问题的研究将为未来认知网络提供更好的安全保障，具有深远的研究意义，是认知网络安全的重要内容。

参 考 文 献

[1] 肖天梅. 认知无线电 PUE 攻击下次用户性能分析[J]. 通信技术, 2013, 46(4): 22-27
Xiao Tian-mei. the performance analysis of Secondary user by Cognitive radio PUE attack [J]. Communications Technology, 2013, 46(4): 22-27

[2] 逄德明, 胡罡, 徐明. 基于能量指纹匹配的无线认知网络仿冒主用户攻击检测[J]. 计算机科学, 2011, 38(3): 28-33
Pang De-ming, Hu Gang, Xu Ming. The Cognitive wireless network counterfeit primary user attack detection of the energy fingerprints matching[J]. Computer Science, 2011, 38(3): 28-33

[3] 薛楠, 周贤伟, 辛晓瑜, 等. 一种解决认知无线电网络模仿主用户攻击问题的方案[J]. 计算机科学, 2009, 36(8): 45-48
Xue Nan, Zhou Xian-wei, Xin Xiao-yu, et al. A solution of cognitive radio network imitating the primary user attack[J]. Computer Science, 2009, 36(8): 45-48

[4] 赵陆文, 缪志敏, 周志杰. 基于 SVDD 的认知无线电网络仿冒主用户检测技术[J]. 信号处理, 2010, 26(7): 974-979
Zhao Lu-wen, Miao Zhi-min, Zhou Zhi-jie. The cognitive radio network counterfeit primary user detection technology based on SVDD [J]. The Signal Processing, 2010, 26(7): 974-979

[5] 任黎丽. 辐射源指纹识别与细微特征提取方法研究[D]. 哈尔滨: 哈尔滨工程大学, 2012
Ren Li-li. The study of Radiation source fingerprint identification and microscopic feature extraction method [D]. Harbin: Harbin Engineering University, 2012

[6] 徐书华. 基于信号指纹的通信辐射源个体识别技术研究[D]. 武汉: 华中科技大学, 2007
Xu Shu-hua. The study of Communication emitter individual identification technology based on the signal fingerprint [D]. Wuhan: Huazhong University of Science and Technology, 2007

[7] 余志斌. 基于脉内特征的雷达辐射源信号识别研究 [D]. 成都: 西南交通大学, 2010
Yu Zhi-bin. The Study of radar emitter signal recognition based on the Intra-pulse characteristics[D]. Chengdu: Southwest Jiaotong University, 2010

[8] 林震鹊, 姜秋喜, 黄建冲. 雷达脉冲信号上升/下降沿测量影响因素[J]. 四川兵工学报, 2010(1): 117-120
Lin Zhen-que, Jiang Qiu-xi, Huang Jian-chong. The measurement factors of the radar pulse signal increase/decrease edge [J]. Journal of Sichuan Armaments Factories, 2010(1): 117-120

[9] 郑文秀. MSK 信号的参数估计[J]. 电路与系统学报, 2011, 16(2): 23-27
Zheng Wen-xiu. MSK signal parameter estimation[J]. Journal of Circuits and Systems, 2011, 16(2): 23-27

[10] 张葛祥. 雷达辐射源信号智能识别方法研究[D]. 成都: 西南交通大学, 2005
Zhang Ge-xiang. The study of Radar emitter signal intelligent identification method[D]. Chengdu: Southwest Jiaotong University, 2005

[11] 唐智灵. 通信辐射源非线性个体识别方法研究[D]. 西安: 西安电子科技大学, 2013
Kang Zhi-ling. The study of Nonlinear individual identification method about communication source[D]. Xi'an: Xi'an University of Electronic Science and Technology, 2013

[12] 周斌. 信号细微特征提取及识别技术研究[D]. 哈尔滨: 哈尔滨工业大学, 2011
Zhou Bin. The Study of signal fine feature extraction and recognition technology[D]. Harbin: Harbin Institute of Technology, 2011

[13] 王超, 刘涛, 杜利平. 一种新的认知无线电主用户信号识别方法[J]. 电波科学学报, 2009, 24(6): 1119-1123
Wang Chao, Liu Tao, Du Li-ping. A new method for cognitive radio users signal recognition[J]. Waves Science Journals, 2009, 24(6): 1119-1123