DHT 网络中数据索引和发布策略研究

翟建设[1,2],孙 鹏[1],吴 青[2,3]
ZHAI Jian-she[1,2],SUN Peng[1],WU Qing[2,3]

1.解放军理工大学 气象学院,南京 211101
2.国防科技大学 信息系统与管理学院,长沙 410073
3.南京审计学院 信息科学学院,南京 210029
1.Institute of Meteorology,PLA University of Science and Technology,Nanjing 211101,China
2.Institute of Information System and Management,National University of Defense Technology,Changsha 410073,China
3.School of Information and Science,Nanjing Audit University,Nanjing 210029,China

ZHAI Jian-she,SUN Peng,WU Qing.Research on data information indexing and searching algorithm based on DHT. Computer Engineering and Applications,2008,44(31):159-163.

Abstract: In order to solve the contradictions between the complex inquiry and DHT network,in Structured P2P network,a new algorithm of data indexing and searching based on multi-keyword in DHT network is proposed,and give detailed analysis and further optimizing considerations are also given.It is proved to be an effective method.

Key words: data information indexing;data information searching;Distributed Hash Table(DHT)

摘 要:在结构化 P2P 网络中,针对分布式散列表与复杂查询之间的矛盾,提出了一个在分布式散列表网络中基于多关键字的数据信息索引和查找算法,对该算法进行了分析和优化,为解决分布式散列表网络与复杂查询之间的矛盾提供了一种有效方法。

关键词:数据信息索引;数据信息查找;分布式散列表
DOI:10.3778/j.issn.1002-8331.2008.31.046 **文章编号**:1002-8331(2008)31-0159-05 **文献标识码**:A **中图分类号**:TP391

结构化 P2P 网络的资源搜索技术主要采用分布式散列表[1](Distributed Hash Table,DHT)技术来组织网络中的结点。DHT 是一个由广域范围大量结点共同维护的巨大散列表。散列表被分割成不连续的块，每个结点被分配给一个属于自己的散列块,并成为这个散列块的管理者。DHT 类结构能够自适应结点的动态加入/退出,有着良好的可扩展性、鲁棒性、结点 ID 分配的均匀性和自组织能力。最经典的案例是 Tapestry[2]、Pastry[3]、Chord[4]和 CAN[5]。

基于 DHT 的 P2P 网络采用相容散列函数根据精确关键词进行对象的定位与发现,而复杂查询往往给出的不是对资源的精确描述,而是资源信息的笼统描述,因此,DHT 提供的精确关键词映射的特性阻碍了 DHT 在复杂查询方面的应用。

提出了基于多关键字的数据索引和查询算法,并对该算法进行了定量分析和优化，提供了一种解决 DHT 网络的精确定位与复杂查询之间矛盾的方法。

1 相关工作

对于 DHT 网络中的资源信息的发布和搜索问题，已经有了一些相关的研究成果:

(1)M.Balazinska 等人提出了一个资源查询框架——INS/Twine[6],该框架是建立在 DHT 网络之上的,用于解决网络中的资源描述和查找问题,但是该框架将同一资源存放在系统的多个节点上,这样使得整个系统需要维护的资源数目较大,同时不便于数据信息的更新。

(2)GARCES-ERICE 等人提出了一种基于用户查询信息的数据索引方式[7],通过使用 XPath 语言来对数据资源的 XML 表示的查询进行表示，并将最详细的查询与数据资源来进行匹配,按照一定规则通过 DHT 将不同的查询进行关联,以达到数据信息发布和查找的功能，但是该方法并没有考虑到所有的数据查询形式，用户在进行查询过程中可能找不到所有的资源信息。

(3)M.Harren 等人提出了一种通过使用传统的关系型数据库操作语言(如选择,映射,分组等操作)来实现对资源进行详细描述和查找[8],采用这种方法不适合构建轻量级的 P2P 数据共享应用,必须要与相应的数据库配合使用。

2 基于多关键字的数据索引与查找算法

提出的基于多关键字的数据索引和查找算法的核心思想是:通过对数据资源信息的元数据描述的拆分,使其描述转换为含有一个或者多个关键字的关键字组,然后通过一定的机制

基金项目:江苏省高校自然科学基金指导性项目(No.06KJD520091)。
作者简介:翟建设(1962-),男,副教授,博士,主要研究方向:计算机支持的协同工作、信息系统;孙鹏(1979-),男,硕士,主要研究方向:分布式数据管理;吴青(1963-),女,副教授,主要研究方向:管理信息系统。
收稿日期:2007-12-03 **修回日期**:2008-02-18

对关键字进行索引,完成数据信息的索引操作;而查找是数据索引的一个逆过程,即通过一个或多个关键字来完成对数据资源信息的查找操作。

2.1 数据信息字符型描述方式

数据资源通过 XML 语言进行元数据描述,为数据用户提供详细的数据信息。为了便于进行数据信息的索引和发布,将数据资源信息的 XML 描述形式拆分成为若干个关键字和值的键值对,并将这些键值对按照一定的规则来进行组织。具体拆分规则如下:

规则 1 对于 XML 中元素和值,使用[元素名/值]的形式进行表示。

规则 2 对于 XML 中的复合元素,使用[复合元素名/[元素名 1/值][元素名 2/值]…[元素名 n/值]]的形式进行表示。

规则 3 对于复合元素下的复合元素以规则二给出的形式进行嵌套。

在这里举一个较为简单的气象数据描述的例子,如图 1 所示。

```
<WeatherDataDescription>
    <Title>AAA</Title>
    <Type>格点报</Type>
    <ElementInfo>
        <ElementName>温度<ElementName>
    </ElementInfo>
    <Temporal>
        <SingleDate>
            2006-07-30T12:00:00
        </SingleDate>
    </Temporal>
    <Spatial>
        全球
    </Spatial>
</WeatherDataDescription>
                    Data
```

图 1 数据资源 Data 的 XML 描述

对图 1 中数据资源 Data 描述按上述规则进行描述,其结果如图 2 所示。

```
d=[Title/AAA][Type/格点报][Element/
  [ElementName/温度]][Temporal/
  [SingleDate/2006-07-30T12:00:00]]
  [Spatial/全球]
```

图 2 使用描述规则描述的数据资源信息 Data

2.2 相关定义

下面给出算法的相关定义:

定义 1 对一个数据资源 f,如果存在一个描述可以唯一的确定 f,那么称该描述为 f 的完全描述,记为 d_{all}。可知,一个资源的完全描述能够唯一确定一个数据资源。图 2 中 d 就是数据资源 Data 的一个完全描述。

定义 2 对一个数据资源 f,如果一个描述只是对 f 完全描述 d_{all} 中部分信息进行描述,那么就称该描述为 f 的部分描述,记为 d_{part}。由于部分描述所描述的是一个数据资源的个别属性,而这些属性别的数据资源也可能拥有,因此,部分描述表示的资源可能不只一个,而是数据资源的一个集合。对于图 1 给出的 Data 的完全描述 d 的部分描述如图 3 所示。

```
d1=[Title/AAA][Type/格点报]
d2=[Title/AAA][Type/格点报][Element/[ElementName/温度]]
d3=[Element/[ElementName/温度]][Temporal
   /[SingleDate/2006-07-30T12:00:00]][Spatial/全球]
d4=[Temporal/[SingleDate/2006-07-30T12:00:00]][Spatial/全球]
```

图 3 Data1 的部分描述

定义 3 对于两个描述 d_1 和 d_2,如果描述 d_1 所表示的数据资源集合 R_1 和 d_2 所表示的数据资源集合 R_2 有以下关系:

$R_1 \supseteq R_2$

那么称 d_1 包含 d_2,或者 d_2 包含于 d_1,记为 $d_1 \supseteq d_2$。

如果有 $d_1 \supseteq d_2$,对于 d_1 所表示的资源集合 R_1,d_2 所表示的资源集合 R_2,如果存在一个数据资源 f,有 $f \in R_1 \Rightarrow f \in R_2$,那么,$d_2$ 比 d_1 对 f 的描述更为详细。

以图 1 给出的 Data 的完全描述 d 和其部分描述 d_1、d_2、d_3、d_4 为例,按照定义 3 可得以下关系:

$d_1 \supseteq d, d_2 \supseteq d, d_3 \supseteq d, d_4 \supseteq d, d_1 \supseteq d_2, d_4 \supseteq d_3$

2.3 数据资源的索引

对一个数据资源 f,由定义 1 可知其完全描述 d_{all} 可以对 f 进行唯一的映射。因此,可将 f 存放在其完全描述散列后所对应的节点上。然而,用户在进行资源查询时大都用数据资源的部分描述进行查询,此时若仅依据使用完全描述建立的数据索引就难以查询到所需的数据资源。为此,提出了数据索引策略 1,其描述如下。

步骤 1 设 $HashD()$ 为分布式哈希函数,对于一个数据资源 f,设 d_{all} 是其完全描述,d_{all} 中含有 n 个关键字,求出 d_{all} 所对应的散列值 k_f:

$k_f = HashD(d_{all})$

将 f 存放在与 k_f 值相对应的节点上。

步骤 2 产生一组对于完全描述 d_{all} 的部分描述,这些部分描述只包含 d_{all} 中的 n 个关键字中的 $n-1$ 个关键字,并形成一个集合,记为 D,于是有:

$D=\{d_1, d_2, \cdots, d_m | d_i \supseteq d_{all}, i=1,2,\cdots,m\}$

步骤 3 对于 D 中的每一个部分描述 d_i,分别计算出它的散列值 k_i,

$k_i = HashD(d_i)$

对于每一个部分描述 d_i 将元组 (d_i, d_{all}) 存储在网络中由 k_i 表示的节点之上[7],其存储索引形式为 $((d_i, d_{all}), d_{all}, q)$,其中 d_{all} 表示这个索引的最终目标是寻找一个以 d_{all} 标识的数据资源 f,其中 q 表示在进行数据搜索时有 q 个该数据资源检索路径需要使用到该元组,初始值 $q=1$。当对当前元组进行存储时,要判断元组是否已经存在,如果存在那么 q 要加 1。

步骤 4 采用对完全描述 d_{all} 的处理方式来处理部分描述集合 D 中的每一个部分描述 d_i,假设部分描述 d_i 中所含关键字个数为 m,那么选择其中的 $m-1$ 个关键字,以这样的方式再形成一个部分描述集合,重复步骤 3 的操作,直到集合中所有部分描述的关键字个数为 1。

由以上过程可以看出,数据信息进行索引的时候,实际上是将索引信息分成不同的信息段,将这些信息段作为数据查找的路由信息存放到每个节点,当进行数据资源的查找时,就按照索引的信息进行路由,最终找到数据资源。

对于那些使用率比较高的热门数据资源,本文定义了索引策略 2。

步骤 1　对于热门数据资源 f,其完全描述为 d_{all},产生其所有的部分描述并组成一个集合 D。

步骤 2　对于部分描述集合 D 中的每一个查询 d_i,通过 $HashD()$函数计算出它的散列值 k_i,

$k_i=HashD(d_i)$

对于每一个部分描述 d_i 将元组$((d_i,d_{dl}),d_{all},1)$存储在网络中由 k_i 表示的节点之上。

对于热门数据资源的确定,按照该资源被共享的次数来进行衡量,当某个数据资源达到一定的共享次数,那么就将该资源确定为热门资源,对于次数的具体制定,要根据网络的规模而制定。

2.4　数据资源的查询

数据资源的查询实际上是数据资源索引的逆过程。当使用一个部分描述 d_0 对数据资源 f 进行查询时,用户首先通过 $HashD$ 函数对该 d_0 进行索引,$k_0=HashD(d_0)$,如果 d_0 是数据的完全描述,那么就可以直接通过 k_0 找到在网络中拥有数据资源 f 的点。如果 d_0 不是数据资源 f 的完全描述,那么散列值 k_0 指向的节点返回的应该是一个元组的数据信息路由列表 L,如图 4 所示。

$((d_0,d_1),d_{all},1)$
$((d_0,d_3),d_{all},1)$
$((d_0,d_5),d_{all},2)$
$((d_0,d_8),d_{all},1)$
$((d_0,d_{11}),d_{all},1)$

图 4　某一个节点上存储的数据检索列表 L

然后在该元组列表 L 中选择一个或几个元组得到下一个数据描述的定位节点。具体的选择步骤如下:

步骤 1　检查元组列表中各元组中下一跳的部分描述是否是资源 f 的完全描述,如果是就用完全描述定位到下一个节点,该节点就是数据资源 f 的节点,如果不是进行下面的操作。

步骤 2　检查拥有同一个目标的索引元组中的下一个资源描述的包含关系(见定义 3),将有包含关系的部分描述进行分组。

步骤 3　对于每组部分查询按照包含关系进行排序,选择出最为详细的一个部分描述。

步骤 4　对各组最详细的部分描述中所含的关键字个数进行比较,选择关键字个数多的部分描述定位到查找的下一个节点,如果各组最详细的部分描述所包含的关键字个数都相同,那么就将这几个部分描述同时作为路由的下一跳。

定位到了下一个节点后,重复步骤 1~步骤 4 的操作直到找到存放资源 f 的节点。具体流程如图 5 所示。

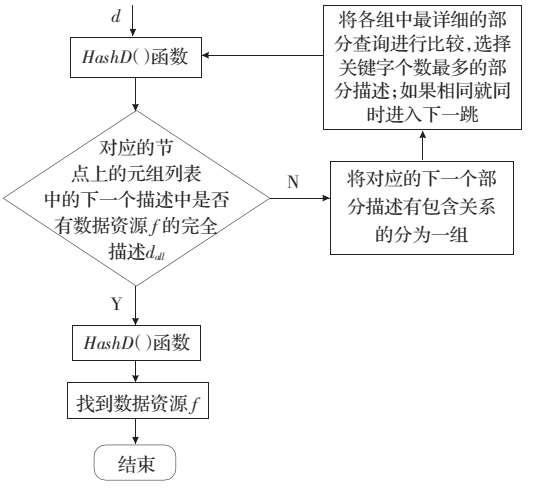

图 5　查找数据资源 f 的流程图

3　算法性能分析

将从整个网络需要维护的索引的代价和资源查询的效率两个方面对算法进行讨论。

3.1　系统维护索引的代价

系统维护索引的代价集中表现在对于某一个资源进行索引时,所产生的元组对的个数,元组个数越多,那么说明系统需要维护某一数据资源的索引所花费的代价就越大。

若一个数据资源 f 的完全描述中包含的关键字个数为 n,按照索引策略 1 对 f 进行索引,在逻辑上会形成了一个深度为 n 的树,称为资源描述逻辑树。该树的每个节点表示 f 部分描述,是 f 关键字的集合。每个节点上的部分描述中的关键字要比与它的子节点所包含的关键字多一个,如图 6 所示。树的根节点是完全描述,具有 n 个关键字,而叶节点都是只含有一个关键字的部分描述。第 m 层中每个元素含有的关键字的个数为:$n-(m-1)$。

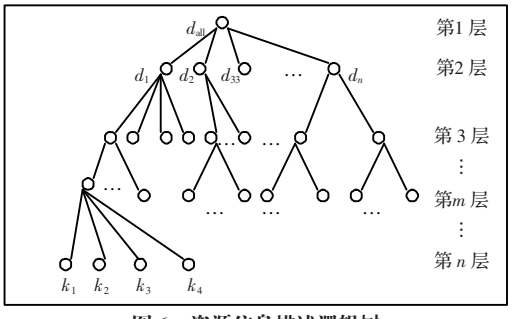

图 6　资源信息描述逻辑树

第 m 层中互不相同的节点(即 f 的部分描述)的个数为:

$$C_n^{n-(m-1)}=C_n^{n-m+1} \tag{1}$$

显然,第 m 层和第 $m-1$ 层之间组成的互不相同的元组个数为:

$$C_n^{n-m+1}C_{n-(n-(m-1))}^{1}=C_n^{n-m+1}C_{m-1}^{1} \tag{2}$$

那么,整个 DHT 网络需要维护的元组个数(即索引总数)$N_1(n)$为:

$$N_1(n)=\sum_{m=2}^{n}C_n^{n-m+1}C_{m-1}^{1}=\sum_{m=2}^{n}\frac{n!}{(n-m+1)!\ (m-1)!}(m-1)=$$

$$\sum_{m=2}^{n}C_n^{m-2}(n-m+2) \tag{3}$$

f 的关键字个数 n 与 $N_1(n)$的关系如图 7 所示。

按照索引策略 2 对数据资源 f 进行索引后,整个 DHT 网络需要维护的索引个数 $N_2(n)$为:

$$N_2(n)=C_n^1+C_n^2+\cdots+C_n^{n-1}=\sum_{i=1}^{n-1}C_n^i=\sum_{i=1}^{n-1}\frac{n!}{i!\ (n-i)!} \tag{4}$$

那么该资源的关键字个数 n 与 $N_2(n)$的关系如图 8 所示。

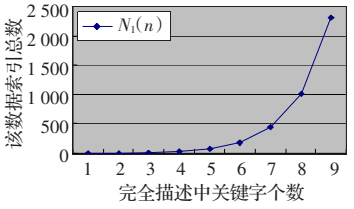

图 7　策略 1 中关键字个数与索引总数关系图

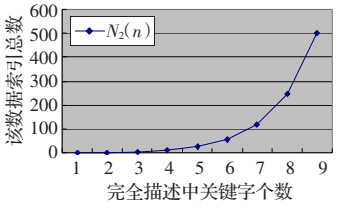

图 8　策略 2 中关键字个数与索引总数关系图

3.2　数据资源查询的效率

资源查询效率与资源检索时所需要的跳数有关。若一个数据资源 f 的完全描述中有 n 个关键字，如果使用索引策略一对 f 进行索引的话，检索路径长度(即跳数)最大为 n，最少为 1。这与用户在进行数据查询时给出的数据描述的详细程度有关，越详细，查询到数据资源的跳数越少。

如果采用策略 1 对数据资源进行索引，那么在进行数据查询时最多会有 n 跳。假设用户在进行数据查询时所使用的关键字个数为 x，查询的跳数为 y，那么 n、x、y 之间有如下的关系：

$$\begin{cases} y=n-x \\ n>x \end{cases} \tag{5}$$

图 9 是查询数据资源 f 时索引路径长度、用户查询关键字的个数以及数据完全描述中关键字个数三者之间的关系图。由图 9 可知，当完全描述中的关键字个数一定时，用户查询使用的关键字个数与资源 f 最大索引路径长度成反比；当用户查询使用的关键字个数一定时，完全描述中关键字个数与资源 f 的最大索引路径成正比。

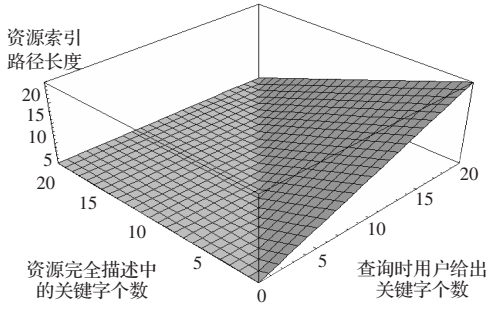

图 9　资源完全描述中的关键字个数、查询时用户给出关键字个数及资源索引路径长度关系

如果采用策略 2 对数据进行索引，那么数据资源 f 的索引路径长度为 1，此时，查询 f 的效率是最高的。

4　算法优化

在进行性能分析的过程中可以看出提出的数据信息索引策略使得系统对资源索引信息的维护代价非常高，因此需要对数据索引策略进行改进。

4.1　改进方法 1：规定形成部分描述的规则

其基本思想是考察用户在对数据进行查询时最关心的关键字，在将数据的完全描述进行拆分时，只拆分包含这些关键字的部分描述。

假设对于某一个数据资源 f，其完全描述中的关键字个数为 n，用户在进行数据查询时，最为关心的关键字个数为 k，($k<n$)，那么对数据资源 f 进行索引后，要求得到数据资源的描述树中的节点(数据的部分描述)中必须要含有用户最为关心的 k 个关键字中的一个或多个。

4.1.1　对索引策略 1 的改进分析

图 6 中表示的资源信息描述逻辑树中的第 m 层中满足这一约束条件的节点个数为：

$$C_n^{n-m+1}-C_{n-k}^{n-m+1}\text{，且 } m\geqslant k+1 \tag{6}$$

显然，第 m 层和第 $m-1$ 层之间组成的互不相同的元组个数为：

$$(C_n^{n-m+1}-C_{n-k}^{n-m+1})C_{m-1}^1\text{，且 } m\geqslant k+1 \tag{7}$$

此时，DHT 网络需要进行维护的索引个数 $N_1(n,k)$ 为：

$$N_1(n,k)=\sum_{m=1}^{n}(C_n^{n-m+1}-C_{n-k}^{n-m+1})C_{m-1}^1 \tag{8}$$

考虑到 $m\geqslant k+1$，那么：

$$N_1(n,k)=\sum_{m=1}^{n}(C_n^{n-m+1}-C_{n-k}^{n-m+1})C_{m-1}^1=$$
$$\sum_{m=1}^{n}C_n^{n-m+1}C_{m-1}^1-\sum_{m=k+1}^{n}C_{n-k}^{n-m+1}C_{m-1}^1 \tag{9}$$

根据公式(9)，将改进前、后的索引策略 1 进行对比，结果如图 10 所示。$N_1(n)$、$N_1(n,k)$(k=1,2,3,4;n=1,2,…,7)分别表示改进前、后策略 1 的数据索引总数。

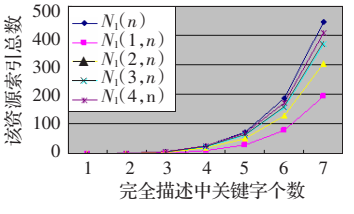

图 10　改进前、后策略 1 的数据索引总数对比

4.1.2　对索引策略 2 的改进分析

按照索引策略 2 对数据资源 f 进行索引时，去除掉数据资源 f 的所有部分查询中不包含 k 个关键字中一个或多个的部分查询，那么此时整个 DHT 网络需要维护的索引个数 $N_2(k,n)$ 为：

$$N_2(k,n)=\sum_{i=1}^{n-1}C_n^i-\sum_{i=1}^{n-k}C_{n-k}^i=$$
$$\sum_{i=1}^{n-1}\frac{n!}{i!\ (n-i)!}-\sum_{i=1}^{n-k}\frac{(n-k)!}{i!\ (n-k-i)!} \tag{10}$$

同样，将改进后的策略 2 与策略 2 进行对比，结果如图 11 所示。$N_2(n)$、$N_2(k,n)$(k=1,2,3,4;n=1,2,…,7)分别表示改进前、后策略 2 的数据索引个数。

由图 10、图 11 可以看出，改进方法 1 对多关键字的 DHT 数据索引在一定程度上能够减少系统对某一数据资源的信息索引总数。当 k=1 时，效果最好，在策略 1 和策略 2 中可以减少 50%，随着 k 值增大，索引总数减少量改进的效果相应减弱。

4.2　改进方法 2：对关键字进行分类

在用户进行数据查询的时候，有可能某几个关键字在查询

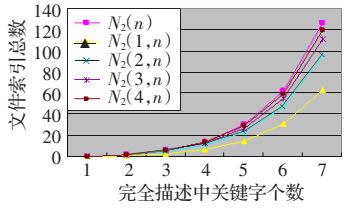

图 11　改进前、后策略 2 的数据索引总数对比

过程中总是一起出现的,或者说某几个关键字一起出现的概率比较大。在进行数据索引的过程中,将同时出现概率较大的关键字进行分类,分类后的关键字组成关键字组,那么在进行数据索引的过程中,规定这些关键字组是不可拆分的,即作为一个整体参与到资源索引中,实际上是将完全描述中关键字个数 n减少, 当 n 的个数减少时那么系统中需要维护索引的总数将会大大减少。

5　结论

提出的基于 DHT 多关键字查询策略实现了数据信息资源的索引和查找,它具有以下优点:(1)数据查找路径 d 小于等于完全描述中关键字个数 n,当某个资源被确定为热门资源时,那么其搜索路径长度仅为1;(2)本策略将数据的完全描述拆分的非常详细,因此使得查询具有较高的命中率。

由于直接使用该策略导致维护数据索引的代价较大,又进一步提出了两个改进的方法。改进的思想中是减少资源部分描述个数,降低数据资源的索引维护代价。

参考文献:

[1] 邹福泰,马范援.基于分布式哈希表的对等系统关键技术研究[D].上海:上海交通大学,2004.

[2] Tapestry[EB/OL].(2008-09-21).http://tapestry.apache.org.

[3] Rowstron A,Druschel P.Pastry:scalable,decentralized object location and routing for large-scale Peer-to-Peer systems[C]//Proc of the 18th IFIP/ACM International Conference on Distributed Systems Platforms(Middleware).Heidelberg,Germany,2001.

[4] Stoica I,Morris R,Karger D,et al.Chord:a scalable Peer-to-Peer lookup service for Internet application[C]//ACM SIGCOMM,San Diego,California,USA,2003.

[5] Ratnasamy S,Francis P,Handley M,et al.A scalable content-Addressable network[C]//SIGCOMM'01,San Diego,California,USA,2001.

[6] Balazinska M,Balakrishnan H,Karger D.INS/Twine:a scalable Peer-to-Peer architecture for intentional resource discovery[C]//Proceedings of the 1st International Conference on Pervasive Computing,2002.

[7] Garces-Erice L,Felber P A,Biersack E W,et al.Data indexing in Peer-to-Peer DHT networks[C]//Proceedings of the 24th International Conference on Distributed Computing Systems (ICDCS 2004),Tokyo,Japan,2004:200-208.

[8] Harren M,Hellerstein J,Huebsch R,et al.Complex queries in dht-based Peer-to-Peer networks[C]//Proceedings of IPTPS02,Cambridge,USA,2002.

(上接 155 页)

6.2　客观测试-SNR 对比

设 $f_{i,j}$ 为原始信号,$f_{i,j}'$为量化后的信号,信噪比的计算公式为:

$$SNR=10\lg\frac{\|f_{i,j}\|^2}{\|f_{i,j}'-f_{i,j}\|^2}$$

SNR 值越大,则表明该量化器的性能越好。

表 3　客观测试-SNR 对比

序列	G729(SNR)	新 VQ(SNR)	差值(VQ-G729)
es01	9.166	8.935	-0.231
es02	7.152	7.000	-0.152
es03	8.190	8.156	-0.034
sc01	7.898	8.071	0.173
sc02	6.605	6.819	0.214
sc03	6.005	6.267	0.262
si01	7.802	7.664	-0.138
si02	11.403	11.303	-0.100
si03	7.009	7.203	0.194
sm01	12.834	12.561	-0.273
sm02	12.094	12.146	0.052
sm03	6.428	6.531	0.103
average	8.549	8.555	0.006

从上述的主观听力测试及客观测试可以看出,该量化方法的效果与 G729.EV 相当。

7　全文总结

目前格型矢量量化被广泛地应用于现代的音视频压缩编码中。格型矢量量化器的性能主要取决于两个方面:恰当的选取格型结构以及基于该格型结构扩展方法的设计。提出的基于偶数格的格型量化方法,利用偶数格的特点能够实现基础码本的低存储空间及快速索引查找。在扩展码本的设计中使用了球型扩展, 使得码本的覆盖范围更大, 并且量化更精确。与 G729EV 的对比实验结果也表明,该方法的量化效果是令人满意的。

参考文献:

[1] Agrell E,Eriksson T.Lattice-based quantization[R].Sweden:Chalmers University of Technology,1996.

[2] Jean-Marie M,Pierre L,Antonini M.Low-complexity indexing method for Z_n and D_n lattice quantizers [J].IEEE Transactions on Communications,1998,46(12):390-402.

[3] Rault P,Guillemot C.Index algorithms for Z_n,A_n,D_n,and D_n^{++} lattice vector quantizers[J].IEEE Trans Multimedia,2001,3(4):395-404.

[4] ITU-T G.729.EV.G.729 based Embedded Variable bit-rate (G.729EV) coder[S],2006.