DHT网络中的多维复杂查询处理方法研究

徐 强[1] 孙乐昌[1] 刘京菊[1] 赵 亭[2] 蔡 铭[3]
(中国人民解放军电子工程学院网络工程系 合肥 230037)[1]
(中国人民解放军61541部队 北京 100194)[2] (中国人民解放军73677部队 南京 210016)[3]

摘 要 DHT网络中的高级查询处理是关系其应用领域拓展的重要问题,也是学术界与工业界共同关注的研究热点。基于Kademlia协议提出一种DHT网络中的多维复杂查询处理方法,其索引结构考虑了用户的查询偏好,使同类数据的存储位置具有相关性,同时巧妙地利用了Kademlia路由表的特点与更新方法,避免了索引维护过产生额外的通信开销,并且通过多点存储、沿路缓存改善了系统的容错性与负载均衡性。分析和实验表明,该方法以O(log*N*)的路由跳数复杂度和较低的开销实现了资源的多维复杂查询。

关键词 多维查询,复杂查询,分布式哈希表,对等网络
中图法分类号 TP393 **文献标识码** A

Multi-dimensional Complex Query Processing over DHT

XU Qiang[1] SUN Le-chang[1] LIU Jing-ju[1] ZHAO Ting[2] CAI Ming[3]
(Department of Network Engineering, Electronic Engineering Institute of PLA, Hefei 230037, China)[1]
(No. 61541 Troops of PLA, Beijing 100194, China)[2] (No. 73677 Troops of PLA, Nanjing 210016, China)[3]

Abstract Advanced query processing is a critical problem for the application of DHT networks. It has attracted much attention from both academic and industrial community. This paper presented a technique for multi-dimensional complex query processing based on Kademlia. It takes user's preference into consideration so that homogeneous data is relevantly indexed. Furthermore, the index maintenance brings no extra communication cost by piggybacking on routing table recovery, besides the advantages in resilience and load balance. The analysis and simulation results show that it implements multi-dimensional complex query processing with O(log*N*) query length and low cost.

Keywords Multi-dimensional query, Complex query, Distributed hash table, Peer-to-peer

1 引言

目前基于分布式哈希表(Distributed Hash Table)的对等网络(DHT网络)中相对成熟的资源查询技术只有面向关键字的精确查询,这严重限制了其应用领域。而如何支持各种高级查询,仍然是一个开放性的难题[1]。

根据涉及属性数目的不同,DHT网络中的高级查询可分为单维查询、多维查询和高维查询等;根据操作方式的不同,则可分为精确查询、区间查询和语义查询等。上述两种分类方式交叉作用,又将高级查询细分为更多种类。本文研究的多维复杂查询就是其中的一种适用面广但处理难度较大的高级查询,它具有以下特点:(1)查询涉及的属性数目及属性组合不确定,这种不确定性增加了查询处理的难度;(2)查询同时涉及分类(Categorical)和数值(Numeric)两种属性并兼容对异构属性的混合查询操作,即对分类属性的精确查询和对数值属性的区间查询。

现有研究提出的DHT网络中的多维查询处理方法主要可以归纳为以下两类:

(1)利用单维索引处理多维查询。此类方法又包括两种不同的技术路线:第一种,使用空间填充曲线(Space Filling Curve),如Hilbert曲线[2]或Z曲线[3],将多维数据变换到单维空间,然后通过建立单维索引来处理多维查询;另一种则为多维数据的每一维分别建立单维索引并通过多次查询和数据过滤来处理多维查询[4,5]。无论遵循哪种技术路线,此类方法的性能都会随着数据维度增加而急剧下降,并且均无法处理多维复杂查询。

(2)通过改造传统数据管理领域的多维索引结构,使它们能够适应分布式存储环境和不同DHT网络的覆盖空间,从而直接构建多维索引来处理多维查询[6-9]。此类方法虽然在查询效率方面具有一定优势,但必须对属性的数目或类型加以限制,因而也不支持多维复杂查询。

GChord[10,11]是一种基于Chord[12]协议设计的多维复杂查询处理框架,但它存在以下缺陷:首先,GChord忽视了用户

到稿日期:2010-10-11 返修日期:2010-12-14 本文受国家自然科学基金(60972161),军队国防科技项目,解放军电子工程学院博士生创新基金资助。

徐 强(1982—),男,博士生,主要研究方向为对等网络、网络安全,E-mail:yfnm126@126.com;孙乐昌(1951—),男,教授,博士生导师,主要研究方向为分布式系统、网络安全、信息安全。

在查询时的倾向性，将分类属性值与数值属性值无区分地进行混洗编码来构造资源索引键，破坏了同类数据在覆盖空间中存储位置的相关性，削弱了分类属性在数据筛选时应有的优势；其次，GChord 忽视了覆盖空间中节点分布的稀疏性，为了维护索引结构各节点间需频繁交互，索引的维护代价较高；最后，GChord 在抵御抖动(Churn)和负载均衡方面也存在不足。

为此，本文基于 Kademlia 协议提出一种 DHT 网络中的多维复杂查询处理方法，其主要思想是对分类属性值与数值属性值分别采用不同的编码策略，使同类数据的存储位置具有相关性，并利用用户的查询偏好提高了查询处理效率；根据 Kademlia 路由表的“一位差异”特性和“捎带更新”[13]方法，设计了与之相适应的索引结构以及无额外通信开销的索引维护方法；通过多点存储和沿路缓存改善了系统的容错性和负载均衡性。

2 基本定义与说明

下面给出资源以及查询的形式化描述。

定义 1(数据模式) 数据模式指资源的描述规范，表示为一个 n 元组$\langle A_1,\cdots,A_i,\cdots,A_n\rangle$，其中，$A_i$ 代表描述资源某一特征的属性，可以是分类属性或数值属性。

定义 2(资源) 根据已定义的数据模式，资源 a 描述为形如$\langle a_1,\cdots,a_i,\cdots,a_n\rangle$的 n 元组，其中，a_i 为 a 在属性 A_i 上的取值。

定义 3(单维查询) 单维查询 q 是形如 $P(A_i,c^q)$的二元谓词，P 是定义在属性 A_i 上的某种操作，c^q 为某一特定取值。若 A_i 为分类属性，则 P 只能定义为“=”操作；若 A_i 为数值属性，则 P 可以定义为“$=,<,>,\leqslant,\geqslant$” 5 种操作之一。单维查询的结果是所有令谓词 $P(A_i,c^q)$为真的资源的集合，记为 $R^q=\{a\mid \forall a,P(a_i,c^q)=\text{true}\}$。

定义 4(多维查询、多维复杂查询) 多维查询 Q 可以表示为若干单维查询的析取式 $q_1\wedge\cdots\wedge q_j\wedge\cdots\wedge q_m$，而多维查询的结果 R^Q 则为这些单维查询结果的交集 $R^{q_1}\cap\cdots\cap R^{q_j}\cap\cdots R^{q_m}$。如果这些单维查询的数目不定，并且同时涉及分类属性和数值属性，则这样的多维查询被称为多维复杂查询。

本文中，网络的构建与演化遵循 Kademlia 协议。作为互联网中应用最广泛的 DHT 网络协议之一，它为每个节点分配唯一标识(NodeID)。NodeID 生成算法一般采用 160bit 的 SHA-1 哈希函数。在 160bit 的覆盖空间中，网络的逻辑拓扑可以表示为一棵非完全二叉树，而节点为树中由 NodeID 唯一确定的叶子。Kademlia 最显著的特征是使用节点标识的“异或”(XOR，$\otimes$)运算度量节点的逻辑距离。该距离保证了路由过程的收敛性，是节点及资源查询的基础。Kademlia 使用的路由表称为 k 桶表(k-buckets)，表中为每个序号 $i\in[0,160)$保存一个最多包含 K 项的称为 k 桶(k-bucket)的链表，其中记录着逻辑距离在范围$[2^i,2^{i+1})$内的邻居，即其 NodeID 中第 i 位与本地节点相反的邻居。

3 多维数据索引

数据索引是高效查询的基础。DHT 网络中数据索引的目标是将逻辑相关的数据存储在覆盖空间中相近的节点上。目前主要存在下列两种构建数据索引的方式[7]：(1)不改变原 DHT 网络的覆盖空间，而是在上层叠加与覆盖空间相适应的索引结构，如 GChord；(2)根据特定索引结构改造或重新构造 DHT 网络的覆盖空间，如 SONAR[8]。因为第一种方式具有易部署、代价低的优点，所以本文使用它设计了与 Kademlia 覆盖空间相适应的索引结构及维护方法。

3.1 资源标识生成方法

为有效利用 Kademlia 路由表的“一位差异”特性，本文通过借鉴 GChord 中的方法，使用同样具有“一位差异”性质的格雷码(Gray Code[14])作为资源标识的编码体系，实现了相关数据的临近存储。

在由异构属性描述的多维数据集中，通常分类属性能够快速将整个数据集划分为若干互不相交的子集。因此，用户在提交查询时也倾向于首先使用分类属性将查询目标锁定在某个较小的范围内，然后使用数值属性细粒度地筛选数据。基于上述分析，本文提出了一种分段构造资源标识的方法，将资源标识分为高低相接的两段：资源标识中由分类属性值的编码构成的部分称为分类标识，它占据资源标识的高位段；资源标识中由数值属性值的编码构成的部分称为数值标识，它占据资源标识的低位段。

对于分类属性，它的取值空间为有限集，那么首先对有限集中的元素依次进行普通二进制编码。编码长度可以由系统预先定义，也可以在选定分类标识的长度后按照分类属性的数目平均分配。然后，将属性值的二进制原码转换[14]为等长的格雷码。最后，使用资源中所有分类属性值的格雷码构造分类标识，方法为按照用户对分类属性的查询偏好，从高到低依次排列各个分类属性值的格雷码。

对于数值属性，首先判断属性值域是否有界。若有界，则将值域直接划分为若干区间；若无界，则使用 Sigmoid 函数或反正切函数将原值域映射为某个新的有界值域，并将新值域划分为若干区间。然后，对划分好的区间依次进行普通二进制编码，编码长度一般按照数值属性的数目平均分配数值标识长度(数值标识的长度等于标识总长与分类标识长度的差值)，并且同样将二进制原码转换为对应的格雷码。最后，对各个数值属性值所在区间的格雷码进行编码混洗(各个编码依次间隔排列)，得到资源的数值标识。

3.2 索引结构与维护

资源标识将作为该资源的索引键(Key)并以〈Key，Value〉对的形式存储于 NodeID 与 Key 逻辑距离最近的 $\lambda(1<\lambda\leqslant K)$个节点中。资源多点存储的目的是避免单点失效，增强网络在抖动条件下的容错能力。λ 取值可由用户根据网络的抖动状况灵活定制。资源的 Key 是由各属性值的格雷码构成的，因而逻辑相关的资源，其 Key 也相似，并且 Key 还唯一确定了资源在覆盖空间中的存储位置，因此逻辑相关的资源能够被临近存储。

定义 5(相邻索引键) 对于任意索引键或节点标识，定义所有与它具有“一位差异”的索引键为该索引键或该节点的相邻索引键。

定义 6(锚节点) 本地节点已知的逻辑距离某个相邻索引键最近的节点定义为该相邻索引键的锚节点。

因为在 Kademlia 中 NodeID 为 160bit，所以任意节点共有 160 个相邻索引键。节点根据差异位在 NodeID 中的位置对这些相邻索引键逐一编号，并维护一份记录相邻索引键及

其锚节点的列表。列表项形如〈i,Key,Pointer〉。其中,$i\in[0,160)$为列表项的序号,代表相邻索引键 Key 与节点 NodeID 的第 i 位相反;Pointer 为锚节点指针,指向第 i 个 k 桶中逻辑距离 Key 最近的邻居。在节点更新第 i 个 k 桶时,如果发现距离第 i 个相邻索引键更近的节点时就修改锚节点指针 Pointer。因为 Kademlia 使用了独特的"捎带更新"路由表的方法,节点能够利用任何消息中的信息更新 k 桶,因此锚节点表也能够被"捎带更新",从而避免了 GChord 中为维护索引结构而进行的节点间频繁的信息交换。因此,锚节点表的维护过程不产生任何额外的通信开销,索引的维护代价较 GChord 显著减小。

通常网络节点总数 $N<<2^{160}$,即覆盖空间中节点呈稀疏分布,那么每个节点将负责存储某一范围内的所有资源,而不仅是那些 Key 与自身 NodeID 相同的资源。因此,锚节点表中序号较小的表项的 Pointer 指针将指向本地节点自身。

定义 7(索引区) 锚节点表中那些 Pointer 指针指向本地节点的相邻索引键的公共前缀定义为本地节点的索引区。

图 1 展示了覆盖空间中节点的分布情况及各个节点的索引区,如节点 A 的索引区为 111^*,节点 C 的索引区为 10^*。虽然节点 D 和 E 拥有相同的索引区 01^*,但因为资源同时存储在 λ 个节点上,这样的索引区重叠不影响资源的发布和查询。节点的索引区将随着节点动态加入或离开而改变。若新节点 G 加入网络,其 NodeID 为 0101,则节点 D 和 E 在感知 G 入网后,随着 k 桶更新而更新自己的锚节点表和索引区,并将不属于新索引区的资源移交给 G 存储。因为 Kademlia 的路由表具有"小世界"特性,节点能够及时感知所有逻辑距离较近的邻居的变化,所以节点的索引区也能够被及时、准确地更新。

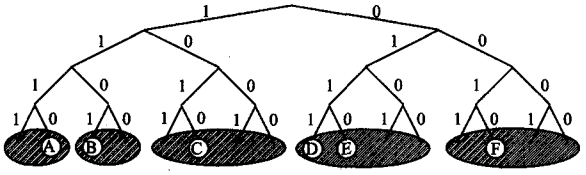

图 1 节点索引区示意图

本文设计的多维数据索引结构具备下列性质:

性质 1 具有相同分类属性值的资源被存储在覆盖空间的同一棵子树中。

性质 2 对于任意两个资源,如果它们在某一个属性的取值紧邻,而其他属性取值相同,则它们的索引键互为相邻索引键。

性质 1 说明索引结构保持了同类资源在存储位置上的相关性。性质 2 说明索引结构保持了每个属性值的局部性。这 2 个性质对于提高查询处理效率,降低查询处理开销具有重要作用。

4 多维复杂查询处理

4.1 查询处理算法

设 Q 是任意多维复杂查询,可表示为分别定义在属性 $A_1',\cdots,A_j',\cdots,A_m'$ 上的 m 个单维查询的析取式 $q_1\wedge\cdots\wedge q_j\wedge\cdots\wedge q_m$。查询处理时,首先按照 3.1 节中的方法得到 A_j' 上所有满足查询 q_j 的属性值的格雷码。这些属性值构成的集合称为 A_j' 上的候选集。若 A_j' 为分类属性,则候选集中只存在唯一的属性值;若 A_j' 为数值属性,则可能存在多个取值区间同时满足查询 q_j,因而候选集中可能包含多个属性值。按照上述方法获得属性 $A_1',\cdots,A_j',\cdots,A_m'$ 上的候选集之后,下一步将根据 3.1 节中的方法构造所有满足多维复杂查询 Q 的索引键,对于 Q 中未涉及的属性,一律使用通配符"x"占位。

例 1 设已定义的数据模式为〈A_1,A_2,A_3,A_4〉,其中,属性 A_1,A_2 为分类属性,A_3,A_4 为数值属性,且所有属性的编码长度均为 2bit。Q 是涉及 A_1 和 A_3 的多维复杂查询,并且 A_1 和 A_3 上符合查询的属性值的候选集分别为{11}和{01,11}。图 2 展示了所有可能满足查询 Q 的索引键的生成过程。按照数据模式定义的属性顺序,逐一从各个属性的候选集中选取可能的属性值,并得到所有的属性值组合。对于每个组合,按照 3.1 节中资源标识的构造方法能够产生一个满足多维复杂查询 Q 的索引键。

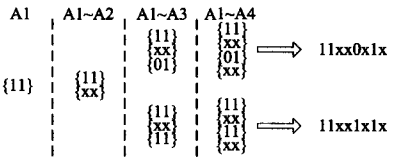

图 2 满足 Q 的索引键的构造过程

含有通配符的索引键并不能直接用于资源查询,必须使用可能的取值替换通配符后得到具体的索引键。通配符替换的过程将生成一棵多叉树,其中包含了所有满足 Q 的具体的索引键。查询源节点只要从根节点开始,沿着这棵多叉树进行"中序遍历",即可获得多维复杂查询的结果。

图 3 显示了索引键 11xx0x1x 的通配符替换过程生成的多叉树。从图中可知,多叉树中父子节点都具有"一位差异",它们互为相邻索引键。因此,在锚节点表的辅助下能够快速对整个多叉树进行中序遍历。

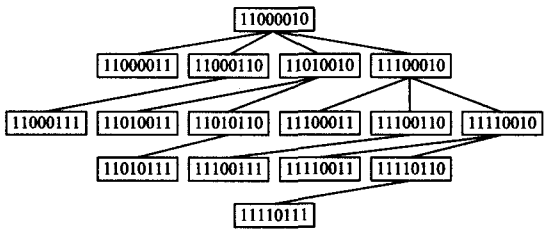

图 3 通配符替换过程生成的多叉树

若查询源节点为 n,构造的含有通配符的索引键为 Key,则多维复杂查询算法描述如下。

算法 1 多维复杂查询处理算法

```
//初始化
Current_Key=Zero_x(Key);
flag=160;
//多维复杂查询调用
Multi_Query(Key,Current_Key,flag)
(1) if Is_Local_Index(Current_Key)
(2)    ReturnResult();
(3)    n. Local_ Query(Key,Current_Key,flag);
(4) else
(5)    u=Node_Lookup(Current_Key);
(6)    u. Local_ Query(Key,Current_Key,flag);
(7) return
```

```
//遍历多叉树
Local_ Query(Key,Current_Key,flag)
(1) For i=1 to flag
(2) if Is_x(Key,i)
(3)    Temp=One_x(Current_Key,i);
(4)    if Is_Local_Index(Temp)
(5)       ReturnResult();
(6)    else
(7)       Anchor=Nearest_NighborIndex(Temp). Pointer;
(8)       Anchor. Multi_Query(Key,Temp,i);
(9) return
```

整个查询处理算法包括多维复杂查询调用和遍历多叉树两个过程。首先,节点将 Key 中的通配符全部替换为 0,以获得多叉树的根节点。然后,检查目标索引键是否属于自己的索引区。如果是,则调用多叉树遍历过程;如果不是,则调用 Node_Lookup 操作,查找逻辑距离目标索引键最近的节点,并由它发起多叉树遍历过程。在遍历多叉树时,通过使用逻辑距离目标索引键最近的相邻索引键的锚节点递归地进行"中序遍历"。

定理 1 若 Key_1 和 Key_2 是满足多维复杂查询 Q 的含有通配符的索引键,则 Key_1 和 Key_2 进行通配符替换所产生的多叉树相互独立。

证明:反证法。假设两个多叉树不独立,那么它们必然拥有公共节点,即由 Key_1 替换通配符得到的索引键 Key_1' 和由 Key_2 替换通配符得到的索引键 Key_2' 相同。根据 Key_1 和 Key_2 的生成方法可知,它们的非通配符位中至少有一位不同,则 Key_1' 和 Key_2' 也至少有一位不同。矛盾由此推出,定理 1 得证。

定理 1 表明,虽然同一个多维复杂查询可能产生多个含有通配符的索引键,因而查询处理需要进行多次多叉树遍历过程,但由于多叉树相互独立,因此不存在相同索引键被重复查询的情况。

4.2 算法分析

定理 2 多维复杂查询处理算法是收敛且完备的。

证明:首先证明算法的收敛性。理想情况下,如果每跳查询的目标索引键都在本地节点的索引区内,则查询请求完全由相邻索引键的锚节点转发,那么,查询过程将在有限次转发后结束;如果查询的目标索引键不在本地节点的索引区内,则查询转发依赖于 Node_Lookup 操作,获得逻辑距离目标索引键最近的节点。因为 Node_Lookup 是收敛的,所以有限次调用 Node_Lookup 操作仍然收敛。根据上述分析,多维复杂查询算法具备收敛性。

由算法 1 可知,资源查询的本质是对其存储节点的查询,因此算法在完备性方面与节点查询操作 Node_Lookup 等效。

综上所述,多维复杂查询处理算法是收敛且完备的。

定理 3 多维复杂查询处理算法的路由跳数复杂度为 $O(\log N)$,其中,N 为网络中的节点总数。

证明:设算法输入的索引键中包含的通配符数目为 M,则通配符替换过程生成的多叉树共有 M 层。

理想情况下,查询过程将完全沿着多叉树中的边中序遍历树中的节点,则算法的路由跳数复杂度为 $O(M)$。

在最坏情况下,每一次查询转发都将调用 Node_Lookup 操作。鉴于 Node_Lookup 的路由跳数复杂度为 $O(\log N)$,则算法的路由跳数复杂度为 $O(M\log N)$。因为 M 为常数,且当网络规模较大时 $M<<\log N$,所以多维复杂查询处理算法的路由跳数复杂度为 $O(\log N)$。

4.3 负载均衡

虽然将资源同时存储于多个节点能够在一定程度上避免单点过载,但对于网络中的热门资源仍然可能出现多个节点同时过载的情况。为此,本文提出了一种沿路缓存的负载均衡机制。

定义 8(节点负载率) 若 C_{max} 代表单位时间内节点能够处理的查询数目的上限,B_{cur} 为节点实际处理的查询数目,则节点的负载率 $R=B_{cur}/C_{max}$。当 R 大于某门限值 R_{thr} 时,称节点处于过载状态。

设查询源节点 n_s 的请求经过 m 跳节点 $n_1,\cdots,n_i\cdots,n_m$ 转发到达目的索引键的存储节点 n_d,如果此时 n_d 处于过载状态,它不仅将结果返回源节点,还沿着转发路径反向传播缓存请求。对于转发路径中的节点 n_i,如果它处于非过载状态,则将结果中的资源缓存,并继续向上转发缓存请求;如果它已处于过载状态,则仅向上转发缓存请求。每个缓存的资源均设置有效期,如果有效期内未收到新的缓存请求,则该资源将被自动删除。当 n_i 收到新的查询请求时,它不仅检查目标索引键是否属于自己的索引区,还需检查本地是否缓存了目标索引键对应的资源。如果存在,则直接将结果返回查询源节点。n_d 处于过载状态的时间越长,它在网络中缓存的资源就越多,许多查询在途中就被命中,而不需转发到 n_d,这样它的负载也随之下降。由于资源缓存时选择了那些处于非过载状态的节点,网络中的负载分布趋于均衡。

5 仿真实验

通过仿真实验测试了本文提出的多维复杂查询处理方法的有效性。实验工具选用麻省理工学院的开源仿真器 P2PSim[15],它是一款基于离散事件的多线程 P2P 通用模拟器。通过派生 P2PSim 提供的 Kademlia 类得到支持多维复杂查询的新协议类 MQKad,并分别指定 King 数据集[16] 和 E-2EGraph 模型作为拓扑结构数据和拓扑结构数据模型。

网络节点总数为 3000,资源数目为 60000,每个资源由 4 个分类属性和 4 个数值属性描述。分类属性值编码长度为 10bit,数值属性值编码长度为 30bit,且各个属性的取值呈均匀分布。MQKad 中的 k 桶容积参数 $K=10$,多点存储参数 $\lambda=3$。实验中分别选取路由跳数及访问节点数作为查询处理算法的性能指标。路由跳数越少,说明算法的效率越高。而访问节点数越少,则说明算法的开销越小。

首先测试了查询属性数目变化时多维复杂查询处理算法的性能。随机选取特定数目的查询属性及属性值,但是数值属性的区间查询不大于该属性值域的 10%。计算每 100 次查询中性能指标的平均值,并重复实验 10 轮。实验结果如图 4 所示,可见查询属性的数目越多,初始的查询索引键中包含的通配符越少,处理算法的性能越好。

在实际查询中用户并非随机选择属性,而是具有明显的倾向性。通常用户首先使用分类属性将目标首先锁定在某个较小的范围内,而后使用数值属性进行细粒度的数据筛选,并且用户对各分类属性的使用频率也不尽相同。

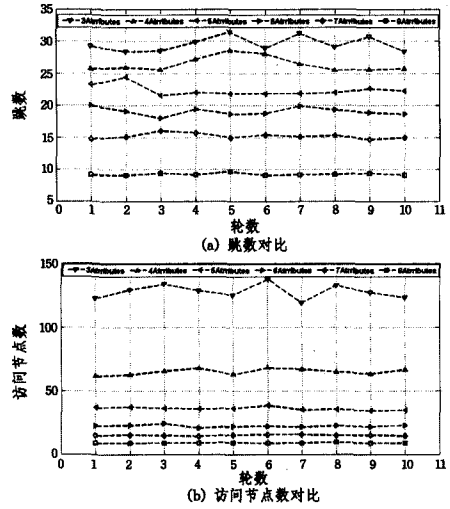

图 4 属性数目改变时查询处理算法的性能对比

为了与 GChord 进行比较，实验进一步测试了本文提出的属性值编码方式较 GChord 中的方式在处理用户倾向性查询时具有的性能优势。作为 MQKad 的对比，基于 Kademlia 协议派生出使用本文查询处理算法但采用 GChord 中属性值编码方式的新协议类 GKad。实验中查询属性的数目固定，包括两个分类属性和两个数值属性。4 个分类属性的选取概率分别为 0.4，0.3，0.2 和 0.1，而 4 个数值属性为等概率选取。随机生成查询属性的属性值，并且数值属性的区间查询仍不大于该属性值域的 10%。计算每 100 次查询中性能指标的平均值，并重复实验 10 轮。实验结果如图 5 所示，可见 MQKad 的查询路由跳数和访问节点数均少于 GKad。从实验结果可知，本文提出的编码方式将具有相同分类属性值的资源存储在覆盖空间的同一棵子树中，有效利用了分类属性在数据筛选过程中的特有优势，在提高查询处理效率的同时减少了查询处理开销。

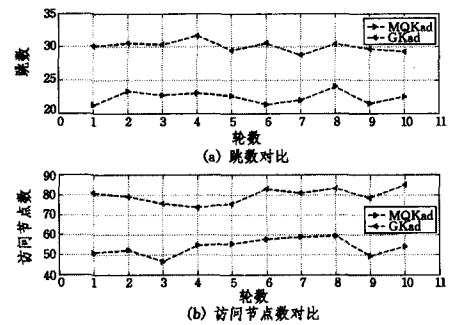

图 5 不同编码方式下查询处理算法的性能对比

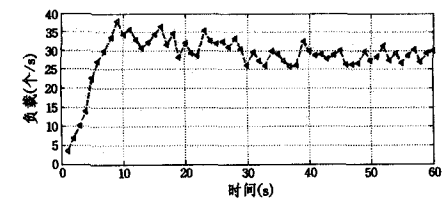

图 6 “热点”负载变化情况

为验证负载均衡方法的有效性，测试了网络中查询“热点”负载的变化情况。所有节点在每 30s 内随机地发起对特定索引键的查询，节点每秒处理的查询数目的上限 $C_{max}=30$，节点过载门限 $R_{thr}=1$，缓存资源的有效期为 60s。图 6 记录了距离查询索引键最近的节点在前 60s 内负载的变化情况。从图中可知，节点负载在 30 左右波动，且时间越久波动幅度越小。一旦负载超过 30，就会很快调整到 30 以下。

结束语 本文提出了一种 DHT 网络中的多维复杂查询处理方法，它具有以下优点：(1)实现了同类数据在覆盖空间中的存储位置的相关性，充分利用了分类属性在数据筛选时的特有优势；(2)将索引维护与路由表维护有机统一，极大地减少了系统的通信开销；(3)提高了系统抵御抖动的能力及节点间负载均衡性。实验结果表明，本文方法具有较高的查询处理效率和较低的开销。本文未考虑查询的相似性及发起相似查询的节点的分布特性，针对这些问题优化查询处理算法，将是下一步研究的重点。

参 考 文 献

[1] Doudane S, Agoulmine N. Enhancing the P2P protocols to support advanced multi-keyword queries [J]. Lecture Notes in Computer Science, 2006, 3976: 630-641

[2] Shen D R, Shao Y C, Nie T Z, et al. HilbertChord: a P2P framework for service resources management [J]. Lecture Notes in Computer Science, 2008, 5036: 331-342

[3] Shu Y, Ooi C B, Tan K. Supporting multi-dimensional range queries in peer-to-peer systems[C]// Proc. of the 5th IEEE International Conference on Peer-to Peer Computing. Washington, 2005: 173-180

[4] 刘德辉，周宁，尹刚，等. QFMA：一种支持负载均衡的多属性资源定位方法 [J]. 计算机学报，2008，31(8)：1376-1382

[5] Cai M, Frank M, Chen J, et al. MAAN: a multi-attribute addressable network for grid information servies [J]. Journal of Grid Computing, 2004, 2(1): 3-14

[6] Matteo V, Christophe D, Ernst B. A Walkable Kademlia Network for Virtual Worlds[C]// Proc. of the 28th IEEE International Conference on Computer Communications Workshops. Piscataway, 2009

[7] Tang Y Z, Xu J L, Zhou S G, et al. m-LIGHT: Indexing Multi-dimensional Data over DHTs[C]// Proc. of the 29th IEEE International Conference on Distributed Computing Systems. Montreal, 2009: 191-198

[8] Schutt T, Schintke F, Reinefeld A. Range queries on structured overlay networks [J]. Computer Communications, 2008, 31: 280-291

[9] 张蓉，钱卫宁，周傲英. 一种支持多维数据范围查询的对等计算索引框架 [J]. 计算机研究与发展，2009，46(4)：529-540

[10] Zhou M Q, Zhang R, Qian Q N, et al. Gchord: indexing for multi- attribute query in P2P system with low maintenance cost [J]. Lecture Notes in Computer Science, 2010, 4443: 55-66

[11] 周傲英，周敏奇，钱卫宁，等. 大规模分布式系统中的多属性查询处理 [J]. 计算机学报，2008，31(9)：1563-1572

[12] Stoica I, Morris R, Karger D, et al. Chord: A Scalable peer-to-peer lookup service for internet application[C]// Proc. of ACM SIGCOMM. New York, 2001: 149-160

[13] Liu Z Y, Yuan R F, Li Z H, et al. Survive under high churn in structured P2P systems: evaluation and strategy [J]. Lecture Notes in Computer Science, 2006, 3394(1): 404-411

[14] Gray F. Pulse Code Communication [P]. United States Patent 263205, 1953

[15] P2PSim Project [CP/OL]. http://pdos.csail.mit.edu/p2psim/, 2010-05

[16] Index of King Data [CP/OL]. http://pdos.csail.mit.edu/p2psim/kingdata/, 2010-05