

# 大规模分布式系统中的多属性查询处理

周傲英<sup>1),2)</sup> 周敏奇<sup>2)</sup> 钱卫宁<sup>1)</sup> 张 蓉<sup>2)</sup>

<sup>1)</sup>(华东师范大学海量计算研究所 上海 200062)

<sup>2)</sup>(复旦大学计算机科学与工程系 上海 200433)

**摘 要** 大规模分布式系统中的复杂查询处理是将对等计算技术运用于关键应用中的重要问题,是学术界与工业界所共同关注的研究问题.文中介绍了一种高效、可伸缩的通用的基于类 Chord 协议的多属性查询处理技术 GChord.它既支持匹配查询也支持范围查询.和现有其它技术相比,对于任何数据元组,GChord 只需要对其编码和索引一次,且能将查询处理的代价限制在一个很小的范围内.因此,它能在索引维护代价和查询效率之间达到平衡.GChord 还提供优化技术以进一步提升性能.实验证实了 GChord 具有较高的查询处理效率以及较低的索引维护代价.

**关键词** 多属性查询处理;重叠网络;分布式系统

中图法分类号 TP311

## Complex Query Processing in Large-Scale Distributed System

ZHOU Ao-Ying<sup>1),2)</sup> ZHOU Min-Qi<sup>2)</sup> QIAN Wei-Ning<sup>1)</sup> ZHANG Rong<sup>2)</sup>

<sup>1)</sup>(Institute of Massive Computing, East China Normal University, Shanghai 200062)

<sup>2)</sup>(Department of Computer Science and Engineering, Fudan University, Shanghai 200433)

**Abstract** Complex query processing in large-scale distributed systems is an important problem in bringing peer-to-peer techniques into applications. It has attracted much attention in both academic and industrial community. This paper presents a generalized Chord-like technique, GChord, for evaluating queries with multi-attributes with scalability and efficiency. GChord supports not only exact match queries but also range queries. It has advantages over existing methods in that each tuple is only encoded and indexed once, while the query efficiency is guaranteed. Thus, index maintenance cost and search efficiency are balanced. Additional optimization techniques further improve the performance of GChord. Extensive experiments are conducted to validate the efficiency of the proposed method.

**Keywords** multi-attribute query processing; overlay network; distributed system

## 1 引 言

随着数据采集技术的发展,基于海量数据的关键应用越来越多.传统的集中式和分布式数据管理

技术<sup>[1-2]</sup>已经不能满足这些应用中对系统稳定性和处理性能的需要<sup>[3]</sup>.与此同时,由于网络技术的发展,出现了很多基于大规模分布式系统(large-scale distributed system)的应用,例如针对文件共享的对等网络<sup>[4]</sup>、科学计算系统<sup>[5]</sup>、即时消息传递系统<sup>[6]</sup>以

及搜索引擎<sup>[7-9]</sup>.

虽然大规模分布式系统已经被证明具有良好的可伸缩性和可靠性,但是,如何在其中像普通的关系数据库那样管理多属性数据仍然是一个难题.该问题近年来吸引了来自数据库和网络等领域的研究人员的关注<sup>[10-13]</sup>.大规模分布式环境中多属性数据管理具有以下 3 个特点:(1)数据是复杂的,它可能既包含种属型(categorical)属性又包括数值型(numerical)属性.很多现有技术或者只能索引其中一种属性(如 Chord<sup>[14]</sup>只能索引种属型数据,而 P-Ring<sup>[15]</sup>只能索引数值型数据).(2)查询是复杂的,且多数时候是即时(on-demand)的.一个查询可能包含对于任意一个属性的限制,也可能是对某些属性限制的组.一些现有技术只能处理对于所有属性都有限制的查询(如 BATON<sup>[10]</sup>和 VBI-tree<sup>[11]</sup>).(3)其对于性能的要求.对于不同复杂程度的查询,系统的索引代价和查询处理代价都应该稳定而高效.一些现有技术,如 MAAN<sup>[12]</sup>和 Mercury<sup>[13]</sup>仅对单属性索引技术进行简单的复用,虽然能够有效处理 2~3 个属性上的查询,但在处理更多属性的数据时性能下降明显.因此,多属性数据查询(Multi-Attribute Query, MAQ)在大规模分布式系统中仍然是一个重要的研究问题.

本文提出一种多属性数据查询处理技术 GChord (Generalized Chord).它具有以下 3 个特点:

(1)它能够处理包含种属型和数值型数据的多属性数据,且具有较小的索引维护代价.在网络环境下,带宽受到限制,而网络延迟也通常比本地 I/O 访问要大. GChord 利用了网络连接的“一位差异”(one-bit difference)特性,为多属性数据构建了一个通用的编码方式,使得任意属性值相邻的数据尽量被映射到网络中的相邻节点,从而减少了数据索引的代价.

(2)依靠“一位差异”特性, GChord 能够快速定位所要查询数据在网络中的位置,从而具有较好的查询处理性能.在具体实现时,一个 MAQ 首先被转换成一系列对于节点的访问请求,这些需要被访问的节点被组织成一棵或多棵多播树.查询请求沿着多播树传播,以获取所有需要的数据.

(3) GChord 提供了一个大规模分布式系统中的 MAQ 查询处理框架,在这一基本框架下,可方便地实现索引和查询优化策略.我们研究了索引的缓存与多播树的聚类问题,以进一步根据负载,优化查询处理的效率.

大量的模拟实验表明,与相同目标的 Mercury 系统相比, GChord 在处理 MAQ 时,在索引维护代价和查询处理效率这两个方面具有优势,当查询所涉及属性较多、而查询范围较大时,优势尤为明显.

本文第 2 节介绍 MAQ 问题的形式化定义;第 3 节详细介绍 GChord 中的数据编码与索引技术;第 4 节介绍 GChord 的基本查询处理技术;第 5 节介绍在 GChord 框架下的索引与查询优化;第 6 节介绍 GChord 的仿真实验结果;第 7 节和第 8 节介绍相关工作并小结全文.

## 2 问题定义

一个多属性数据元组(tuple)是一个集合: $t\{\langle attr_i, v_i \rangle\}, i=1, 2, \dots, N$ , 其中  $attr_i \in A$  是属性名,而  $v_i$  是属性值.属性可以是种属型(categorical)的,也可以是数据型(numerical)的.  $A_i$  是  $attr_i$  的值域,即  $v_i \in A_i$ .我们假设  $A_i$  有界且  $\{A_i\}$  已知,即数据的所有属性的值域已知.

一个多属性查询(Multi-Attribute Query, MAQ)是一个形如  $\langle attr, op, v \rangle$  谓词的集合:  $q\{\langle attr_i, op_i, v_i \rangle\}, i=1, 2, \dots, m, m \leq N$ , 其中,  $attr_i$  是属性名;  $op$  是比较符,对于种属型数据,  $op$  是  $=$ ,而对于数值型数据,  $op$  是  $<, \geq, =, >, \leq$  其中之一.一个查询中可以有任意多个谓词,每个谓词可以针对任意一个属性.查询的结果应是所有符合所有谓词的元组,即

$$R(q\{\langle attr_i^q, op_i, v_i^q \rangle\}) = \{t\{\langle attr_i^t, v_i^t \rangle\} |$$

$$\forall attr_i^q, attr_i^q = attr_i^t, v_i^q op_i v_i^t, t \in DB\},$$

其中,  $DB$  是当前系统中所有元组的集合.

**例 1.** 图 1 展示的是一个有 5 个属性的数据库,共有 6 个元组.数据库中既有数值型属性也有种属型属性. Q1~Q5 是 5 个 MAQ 查询.各个查询所包含的谓词个数不同;每个谓词所限定的可能是某个属性的一个点,或者是数值型属性上的一段区间;各个查询所涉及的属性也各不相同.

本文研究具有类 Chord 性质的重叠网络(overlay network)上的 MAQ 查询处理问题. Chord<sup>[14]</sup>作为最早出现的结构化重叠网络协议,是很多其后出现的重叠网络协议的基础.例如, BATON<sup>[10]</sup>和 VBI-tree<sup>[11]</sup>都是在 Chord 结构上建立的树型结构重叠网络;而 Mercury<sup>[13]</sup>则建立在多个 Chord 的基础上. Chord 协议简单、路由效率高、维护代价低,其路由代价为  $O(\log N)$  跳步(hop),数据更新所需的

fn(c)	price(n)	duration(n)	premiere(n)	cinema(c)
Garfield	60	75	2006	Guang
Lord of War	50	90	2005	Yongle
Silther	90	100	2006	Guang
The Break	90	110	2005	Yonghua
Thinking	100	125	2006	Yonghua
XMan	90	100	2005	Yonghua

数据库原组

Q1: fn="Lord of War"∧price>40∧price<80
Q3: premiere>2006∧premiere<2007∧cinema="Yongle"
Q2: duration>80∧price<90∧cinema="Guang"∧premiere>2006
Q4: fn="XMan"∧price<70∧premiere>06-03-12∧cinema="Guang"
Q5: fn="Thinking"∧price=50∧premiere>2006

查询

图 1 数据库与 MAQ 查询示例

消息数为  $O(\log N)$ . Chord 结构的主要特点为：

- (1) 每个节点以一个相同长度的二进制串作为节点标识(通常为 128 位)；
- (2) 每个节点维护一张路由表,其中每个表项指向一个与当前节点标识有一位差别的标识所对应的节点,如标识为 10101 节点的路由表指向 00101, 11100, 10001, 10111, 10100 5 个标识所对应的节点；
- (3) 所有的节点按照标识排序,序列中头尾相连,构成一个环,每个节点维护指向其前、后节点的指针.

我们称具有以上结构特点的重叠网络为类 Chord 重叠网络. 类 Chord 重叠网络上的 MAQ 查询处理主要的性能指标包括以下两项：

- (1) 查询处理延时. 即处理一个查询时,从发出查询开始到获取所需查询结果时的时间间隔. 由于分布式环境中,网络延时远大于本地处理所需时间,因此,我们以重叠网络中消息的跳步数作为衡量查询处理延时的指标；
- (2) 查询处理代价. 分布式系统中,网络的带宽资源是有限的. 查询处理代价即查询处理时对于带宽资源的消耗. 我们以查询处理过程中的重叠网络中消息总数和被访问的节点数量作为衡量查询处理代价的指标.

MAQ 查询中可能包含对于不同属性个数、不同属性、不同限制条件的查询,我们希望一个好的 MAQ 查询处理技术能够在各种查询条件下都具有较小的查询处理延时和查询处理代价. 此外,我们还希望相应的数据索引的维护代价较小,即当数据库

内容改变或者节点加入、退出时,重叠网络中的消息数、消息跳步数以及被访问的节点数均较小.

3 GChord 中的数据索引

GChord 数据索引的基本原理是将在某个属性上具有相近(或相同)值的元组存放 to 重叠网络上相近的节点上. 本节首先分别介绍数值型和种属型数据的编码,然后介绍多属性数据的索引.

3.1 数值型属性值和种属型属性值的编码

数值型属性的值域首先被均匀划分成若干段,划分后的段数为 2 的幂,其大小由系统决定(将在 3.2 节中介绍). 例如,对于值域为  $[1, 4] \cup [5, 17] \cup [31, 36]$  的属性,需要被划分为 4 段时,结果为  $\{[1, 4][5, 7]\}, \{(7, 12]\}, \{(12, 17]\}, \{[31, 36]\}$ . 被划分后的段按序编号后使用格雷码(Gray Code<sup>[16]</sup>)编码,如图 2 所示. 格雷码的特点是相邻数值的格雷码仅相差一位,因此,可以利用这一性质,将在某属性上具有相近值的元组映射到同一节点(当它们的属性值属于同一段)或相邻节点上(当它们的属性值属于相邻段).

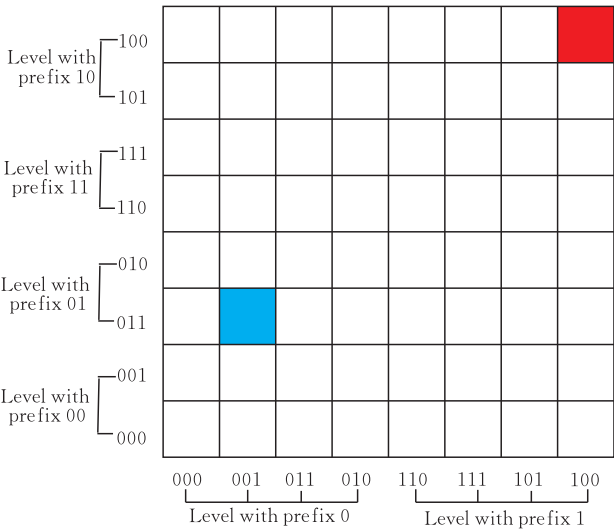


图 2 两个属性的数据空间划分及编码

在数据索引时,对于属性  $A$  的值  $v$ ,首先确定其所属的段的编号  $S_v$ . 然后可利用算法 1 得到其格雷码,其实现仅需要两条机器指令.

算法 1. 根据段编号获得格雷码.

S2GC:根据属性值的段号计算属性值的格雷码编码  
输入:  $S // v$  所在段的编号的二进制原码,其长度为  $m$   
输出:  $G // v$  所在段的格雷码  
1. BitString  $G[m]$ ;  
2.  $G \leftarrow S \oplus (S \gg 1)$ ;  $// \gg$  为移位操作,空余位补 0,  $\oplus$  为

按位异或操作

3. return G;

对于种属型属性,其编码为对值进行散列(hash)后取模的结果,即  $C_v = h(v) \bmod 2^m$ . 在实现时,GChord 使用 SHA-1 作为散列函数. 显然,相同属性值的编码相同,从而能够被映射到相同的重叠网节点上.

3.2 索引多属性数据

每个属性使用多少位编码在系统初始化时设定. 如果没有设定,则各属性均分 128 位的标识. 给定一个元组,按照上一小节所介绍的方法,分别对每个属性值计算得到其编码. 多属性数据索引主要解决两个问题:一是如何根据各个属性值的编码获得元组的编码,二是如何将元组根据其编码发布到各节点上.

属性值编码能够确保相近或者相同值的编码形式相近,但这并不表示两个在某属性上值相近的元组的编码整体上也相近. 这是因为这两个元组可能在其它属性上具有差别较大的值. 例如,图 1 中,第一和第二个元组虽然在 price 这一属性上值相近,但它们在所有其它属性上的值都差别较大. 由于类 Chord 重叠网络路由时总是将数据包先发送至左侧标识位与数据包目标地址相同的节点,因此如果只是简单地把所有属性值的编码顺序连接起来,那么排列在左侧的属性总是具有较高的选择率,而索引对排列在右侧的属性只具有很小的作用. 仍然以图 1 为例,所有元组通过编码后如图 3 所示,排列在 Chord 环中. 虽然第 2、4 和第 6 个元组在 price、duration、premiere、cinema 这 4 个属性上都相似,但是由于它们在属性 film name 上不同,因此它们的编码在类 Chord 网络上分散在各处. 当需要查询所有 cinema='Yonghua'的元组时,必须访问 4 个节点中的 3 个. 换句话说:在某属性上相似的元组被分配到的节点并不具有相关性.

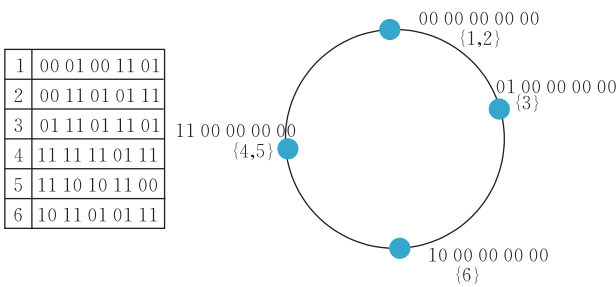


图 3 不适当的整体编码

为了避免出现以上情况,GChord 采用编码混洗(shuffle),即各个属性的编码被打散组成最终的

编码. 在混洗时,各属性的编码依次间隔排列. 图 1 中元组,如果仍然采用图 3 中的属性编码,但使用编码混洗,则可以得到图 4 所示的最终编码.

	混洗前	混洗后
1	00 01 00 11 01	00 01 00 10 11
2	00 11 01 01 11	01 00 10 11 11
3	01 11 01 11 01	01 01 01 11 11
4	11 11 11 01 11	11 10 11 11 11
5	11 10 10 11 00	11 11 01 00 10
6	10 11 01 01 11	11 00 10 11 11

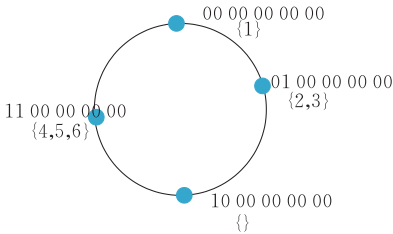


图 4 编码混洗

性质 1. 任意两个具有一位差别(one-bit difference)的元组编码所对应的元组的数值型属性值或者属于同一个段,或者属于相邻段.

性质 2. 具有相同前缀的元组编码,设它们不相同的后缀中对应于同一数值型属性的位数最多为  $k$ ,则它们所对应的元组的数值型属性值最多相差  $2^k$  个段.

性质 1 说明编码混洗保持了类 Chord 重叠网络上数值型属性的局部性;性质 2 说明编码混洗后,在种属型属性上相同,在各个数值型属性上都相似的元组被分配给同一个类 Chord 重叠网络节点的可能性较大.

4 基于 GChord 的查询处理

4.1 查询处理框架

虽然索引试图将种属型属性值相同、数值型属性值相近的元组存放在同一个节点上,但是,一个 MAQ 查询仍然可能需要访问多个节点. GChord 查询处理要解决两个问题:

- (1) 决定哪些节点可能存有 MAQ 查询结果;
- (2) 确定重叠网络上查询的分发路径.

性质 3. 存有 MAQ 查询  $q\{\langle attr_i, op_i, v_i \rangle\}$  的结果  $R(q)$  的节点的 ID 必定满足以下两个条件:

- (1) 对于种属型属性  $attr_i, attr_i = v_i$ ,则 ID 中所对应的编码  $code(attr_i) = code(v_i)$ ;
- (2) 对于数值型属性  $attr_i, attr_i op_i v_i$ ,则 ID 中所对应的编码  $code(attr_i) op_i' code(v_i)$ ,其中  $op_i'$  为数值型属性二进制比较符  $op_i$  在格雷码上的

表示。

性质 3 解决了查询处理要解决的第一个问题。对于所有可能有查询结果的节点,可以用一个带通配符的节点 ID 表示;那些受性质 3 限制的位为 0 或 1,而其余位为  $x$ 。根据该 ID,GChord 可构造一棵或多棵多播树(Multi-Cast Tree, MCT)。多播树的每个节点都是重叠网络上的节点,边为节点路由表指

针;所有可能有查询结果的节点都必须出现在 MCT 中。然后,GChord 将查询从 MCT 的根向所有叶子节点分发(多播)。每个收到查询的节点查询自己的本地索引,并将所有能够满足查询的元组发送给查询节点。

图 5 显示的是  $10xx1xx$  所对应的 MCT 及其在类 Chord 重叠网络上的位置。

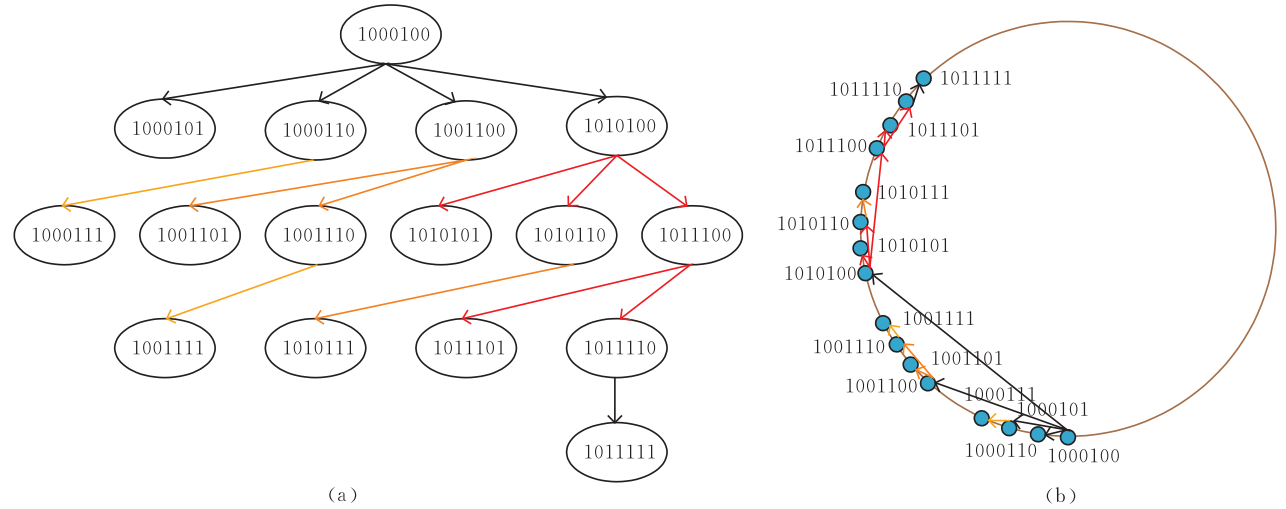


图 5  $10xx1xx$  所对应的多播树

于是,查询处理的核心问题就是多播树构造,或者说确定多播树所对应的 ID。

4.2 构造多播树

GChord 使用卡诺图(Karnaugh Map)<sup>[17]</sup>来构造多播树。首先,GChord 为每两个还未处理过的 MAQ 中出现的属性构造卡诺图。如果还剩一个属性未处理,则为此属性单独构造卡诺图。然后,将可能包含查询结果的节点在卡诺图中标注,并约简卡诺图,得到部分 MCT(Multicast Tree Proportion, MTP)。在得到所有 MTP 以后,GChord 按照编码混洗的方式得到 MCT 的 ID 表示。

构造和约简卡诺图的算法如算法 2 所示。

算法 2. 构造 MTP。

ATTR2MTP:根据两个属性及其上的谓词确定 MTP

输入: $\langle attr_1, op_1, v_1 \rangle, \langle attr_2, op_2, v_2 \rangle$

输出:MTP

1. 构造一个空的卡诺图,其长和宽分别是属性  $attr_1$  和  $attr_2$  的可能编码的个数;
2. 对卡诺图的每个方格,如果它所对应的编码满足  $\langle attr_1, op_1, v_1 \rangle, \langle attr_2, op_2, v_2 \rangle$  的编码表示,则标示 1,否则标示 0;
3. 对于两个相邻的相同大小的表示为 1 的格子,进行合并,并得到一个大小为  $2^i$ 、标示为 1 的格子;
4. 重复 3 直到没有其它可以合并的格子为止;

5. 为每个最终标示为 1 的格子产生 MTP 表示。

图 6 显示了一个约简后的卡诺图,其所对应的 MTP 表示为  $\langle x0x, x00 \rangle, \langle x1x, 11x \rangle$  以及  $\langle x1x, 101 \rangle$ 。

$\begin{matrix} a \\ b \end{matrix}$	000	001	011	010	110	111	101	100
000	1	1	0	0	0	0	1	1
001	0	0	0	0	0	0	0	0
011	0	0	0	0	0	0	0	0
010	0	0	0	0	0	0	0	0
110	0	0	1	1	1	1	0	0
111	0	0	1	1	1	1	0	0
101	0	0	1	1	1	1	0	0
100	1	1	0	0	0	0	1	1

图 6 两个属性上的卡诺图

不同属性对的 MTP 之间,进行两两组合,以得到最终 MCT 的属性表示。再通过混洗,就可以得到 MCT 所对应的 ID 表示。

MCT 构造不仅可以被用来处理单个查询,还可以对多个查询同时进行,从而进一步节省查询代价。

4.3 基于多播的查询处理

在获得了 MCT 的 ID 表示以后,就可以通过以



下 4 个步骤分发、处理查询：

1. 将 MCT 的 ID 表示中所有的  $x$  替换为 0, 将查询以及 MCT 的 ID(即  $\langle ID, q \rangle$ ) 发送给该节点；
2. 当一个节点  $ID'$  收到  $\langle ID, q \rangle$  以后, 如果该查询已经处理过, 则丢弃；否则在本地进行查询, 将结果返回给查询节点；
3. 对于未丢弃的查询, 对于每一个 ID 中为  $x$ ,  $ID'$  中为 0 的位, 构造  $ID''$ , 它的对应位为 1, 其它位与  $ID'$  相同, 将  $\langle ID, q \rangle$  发送给所有  $ID''$  对应的节点；
4. 步 2, 步 3 过程连续执行直到不存在符合条件的  $ID''$  为止。

图 5 展示了根据以上方法得到的一种多播树。

**性质 4.** 查询路由的跳步数(number of hops)的复杂度为  $O(\log_2 N + M)$ , 其中  $N$  是重叠网络中最多能够容纳的节点个数,  $M$  为 MCT 的 ID 表示中  $x$  的个数。

5 索引与查询优化

上一节介绍的 GChord 的基本索引和查询处理

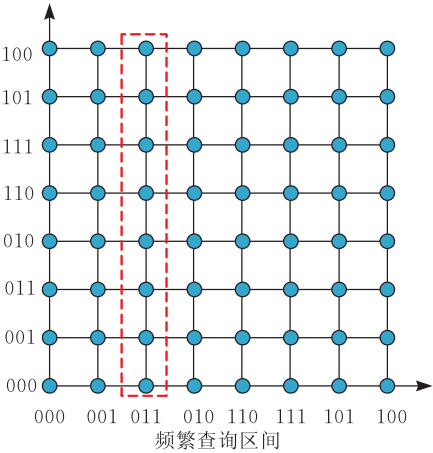


图 7 网络索引缓存

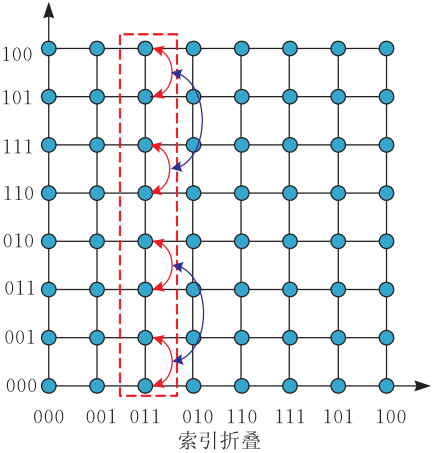
索引缓存要解决 3 个主要问题：“哪些数据需要缓存”、“在哪些相邻节点上缓存”以及“如何缓存”。

- (1) 索引信息交换. 每个节点定期将自己索引的数据的范围, 即数值型属性值所属于的段号, 发送给所有相邻节点. 这是因为类 Chord 重叠网络上的节点通常不是满的, 所以每个节点往往要负担多个逻辑节点的索引任务. 索引信息交换能让每个节点知道相邻节点的索引任务, 并由此以及路由经过的消息数估算相邻节点的负载。
- (2) 频繁查询区间检测. 对相邻节点的查询负载通过公式:  $F_{new}^i = F_{old}^i / 2 + N_{new}^i / T$  进行计算. 其中,  $T$  是预先设定的检测时间段,  $N_{new}^i$  为  $T$  时间段

- 方法. 在处理实际 MAQ 时, 可能存在两个问题：
- (1) 首先, 一些查询可能会查询非常大的范围, 同时很多查询的结果可能都包含某些元组, 于是, 频繁被查询到的节点可能过载(overload)；
  - (2) 另外, MCT 的个数是各属性对上 MTP 大小的乘积. MCT 可能非常多, 因此查询处理的代价很大, 即系统中处理查询所用的消息数可能非常多。
- 针对以上两个问题, 本节讨论索引缓存和多播树聚类技术, 以进一步优化索引与查询处理。

5.1 网络索引缓存

对于经常被访问的元组, GChord 不仅在它们对应的节点上进行存储, 还在那些节点的相邻节点上进行存储. 由于 GChord 的查询是通过相邻节点进行传送的, 因此, 这些被频繁查询的元组只需要较少的跳步就能被找到. 此外, 由于一个节点上的数据可能被缓存在相邻的多个节点上, 所以它也能缓解被频繁访问的元组的索引节点的负载. 图 7 是索引缓存的示意图. 其中, 虚线框表示被频繁访问的索引所在的节点, 缓存的数据被扩充缓存两次。



- 内, 通过本节点向相邻节点发送的对于属性  $attr_i$  的查询, 如果  $F_{new}^i$  大于预先设定的阈值  $F_{threshold}^i$ , 则表示相邻节点的负载较大, 需要在本地对相邻节点的数据进行缓存。
- (3) 索引缓存构造. 一旦确定了需要缓存的数据, 本地节点向相邻节点发出缓存请求, 指明并获得缓存数据. 当本地节点自身过载或缓存数据的访问频度回落, 本地节点通知缓存数据原索引节点撤销缓存。
  - (4) 索引缓存更新. 数据更新时, 索引节点与缓存节点同时更新数据。
- 5.2 多播树聚类**
- 由于类 Chord 重叠网络通常并不是满的, 因此

不同的节点 ID 可能最终对应于同一个物理节点. 而相近但不同的 MCT 事实上可能对应于相同的物理节点集合. 为了决定哪些 MCT 可以被聚成一类, GChord 中节点定期对整个重叠网络进行采样, 以确定整个网络的稀疏程度. 对于根节点距离小于重叠网络平均节点距离的 MCT, 它们将被打包一同发送给其中最小的那个根节点. 如果一个节点发现它收到了不属于自己的 MCT, 该 MCT 对应的查询将被进一步发送给正确的节点.

注意, 对于重叠网络采样的精确程度并不影响查询处理的正确度. 使用多播树聚类以后, 不同的 MCT 因为共享了消息, 因而可以极大地减少查询处理的开销.

6 实验结果

GChord 实验运行在 JDK 1.42 写的模拟器上. GChord 模拟实验总共使用 10000 个节点, 每个节点的标识符为 32 位二进制串.

模拟数据共有 5 个数值型属性和 1 个种属型属性, 各属性值均匀分布. 数据集共 100000 个元组. 由于网络传输代价远大于本地处理代价, 因此系统性能主要由查询需要访问的节点数决定, 平均每个节点 10 个元组能够使得同一个模拟器上运行尽量多的节点, 并且不影响对实验结果的分析. 查询中属性的个数、点查询 (point query) 和区域查询 (range query) 的中心、区域查询的范围 (不大于每个属性的 20%) 都按照随机分布随机生成. 索引缓存时, 每个节点可提供 20% 的存储容量用于缓存相邻节点的索引数据.

当 MAQ 查询中属性个数变化时, 实验测试了范围查询不超过 10% 时, GChord 查询处理消息的最大跳步数、系统的消息总数以及访问的节点数, 如图 8 所示. 查询中出现的属性越多, 系统的性能越好. 使用索引缓存时, 由于查询不需要访问最终索引节点, 因此可以明显减少查询所用的消息数和访问的节点数. 当使用多播树聚类时, 消息总数下降到基本 GChord 方法的 1/10. 多播树聚类技术对于查询中属性较少时性能的影响较明显.

实验进一步测试了查询属性个数固定 (4 个属性), 区域查询范围变化时的系统性能, 如图 9 所示. 当查询范围增大, MCT 的个数会增加, 因此性能会下降. 但多播树聚类能够极大地减轻 MCT 个数增多所带来的消息数增加问题. 索引缓存对于减少查询处理时访问的节点数具有很好的效果. 我们测试

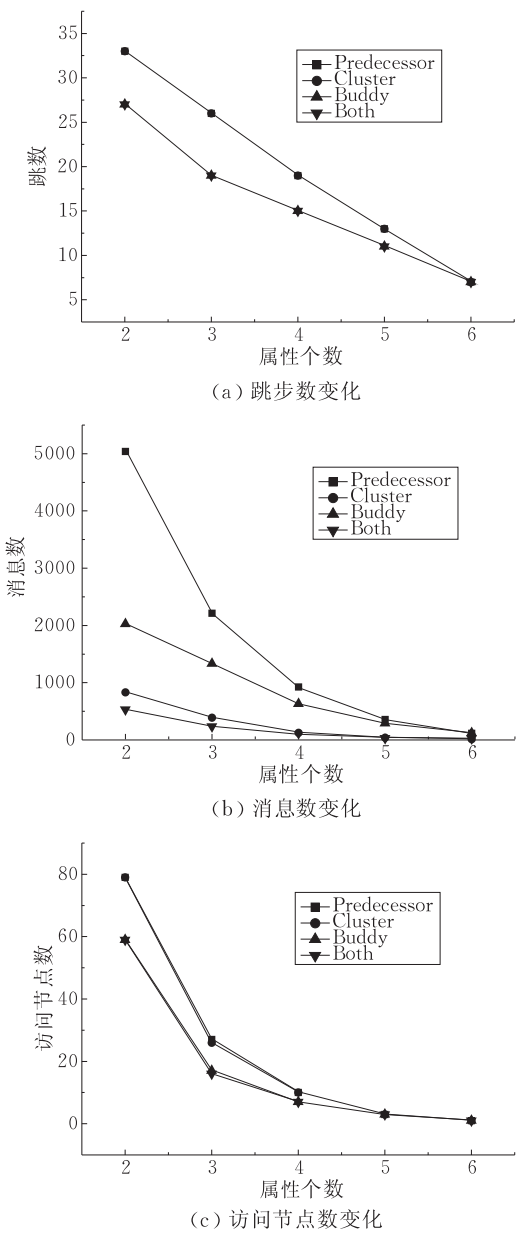


图 8 查询中属性个数变化时 GChord 的性能实验 (Predecessor 表示基本的 GChord 索引与查询处理, Cluster 表示采用了多播树聚类, Buddy 表示采用了索引缓存, Both 表示同时采用了两种优化技术)

了同时采用两种优化技术、热点查询的频繁程度变化时的系统性能, 如图 10 所示. 区域查询的最大范围设为 10%, 从实验结果可见, 热点越集中, 系统的性能越好. 具体而言, 当热点集中时, 索引缓存效果较明显, 因此大量查询能够在距离查询节点较近的缓存中得到结果, 因此跳步数下降明显. 查询越集中, 能够聚类的多播树越多, 因此, 在多播树聚类和索引缓存共同作用下, 系统消息数和访问节点数下降明显. 由于查询中属性个数多时, MCT 个数本身较少, 从而消息数和访问节点数本身较少, 因此系统

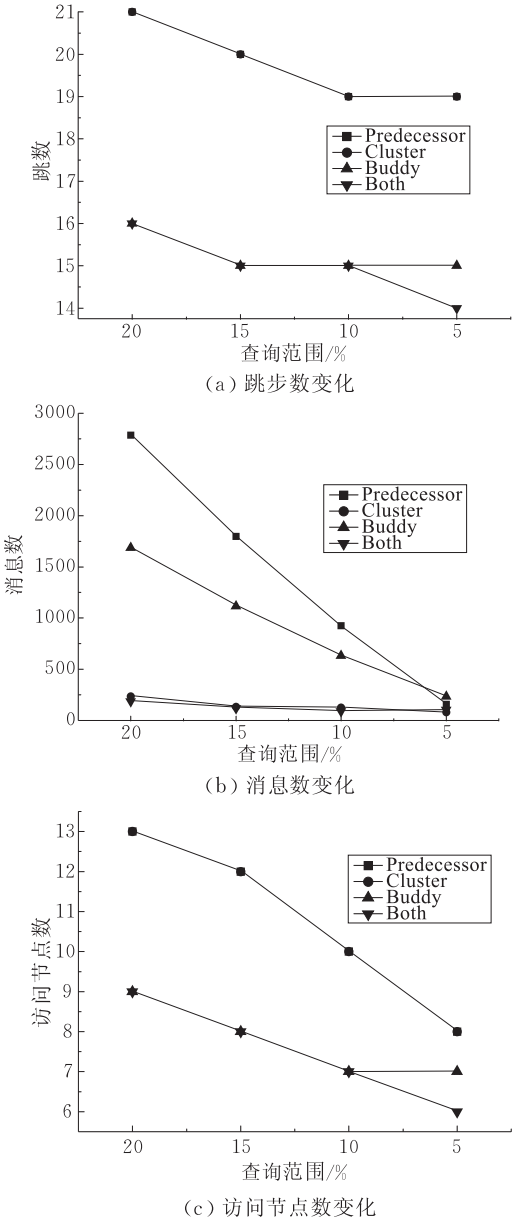


图 9 查询范围变化时 GChord 的性能实验 (Predecessor 表示基本的 GChord 索引与查询处理, Cluster 表示采用了多播树聚类, Buddy 表示采用了索引缓存, Both 表示同时采用了两种优化技术)

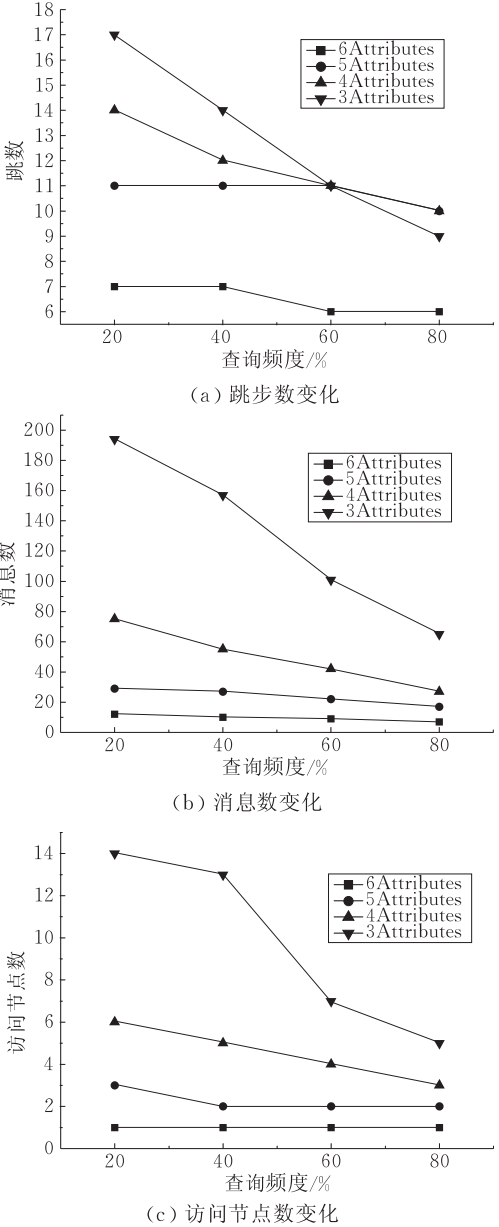


图 10 查询热点集中时 GChord 的性能实验

的“一位差异”编码与索引技术能够更有效地利用重叠网络链接, 具有较少的消息数.

7 相关工作

近年来, 大规模分布式系统由于其良好的可伸缩性和可靠性, 在 Web 搜索等海量数据处理应用中得到了广泛的使用<sup>[7-9]</sup>. 重叠网络 (overlay network) 是大规模分布式系统中的一项重要技术. Chord<sup>[14]</sup> 是一种重要的重叠网络协议. 它将所有节点组织成一个环, 并利用“一位差异”性质, 构建每个节点的路由表. Chord 能够利用  $O(\log N)$  的路由表达到  $O(\log N)$  代价的一维数据查找效率. 为了解决多维

性能受热点查询影响不大, 但是仍然能够看到最大跳步数的明显减少.

实验比较了 GChord 和 Mercury 的性能差异, 如图 11 所示. 由于 Mercury 需要在多个 (虚拟) Chord 中进行查询, 因而消息的跳步数和访问的节点数远高于 GChord. 同时, 由于 Mercury 中, 每个节点为每个属性都维护了一个 Chord 重叠网络, 因此能够快速定位维护了在某个属性上相邻值的节点. 所以 Mercury 在查询中属性较少时只需要使用较少的消息数. 但是, 当查询中属性较多时, GChord



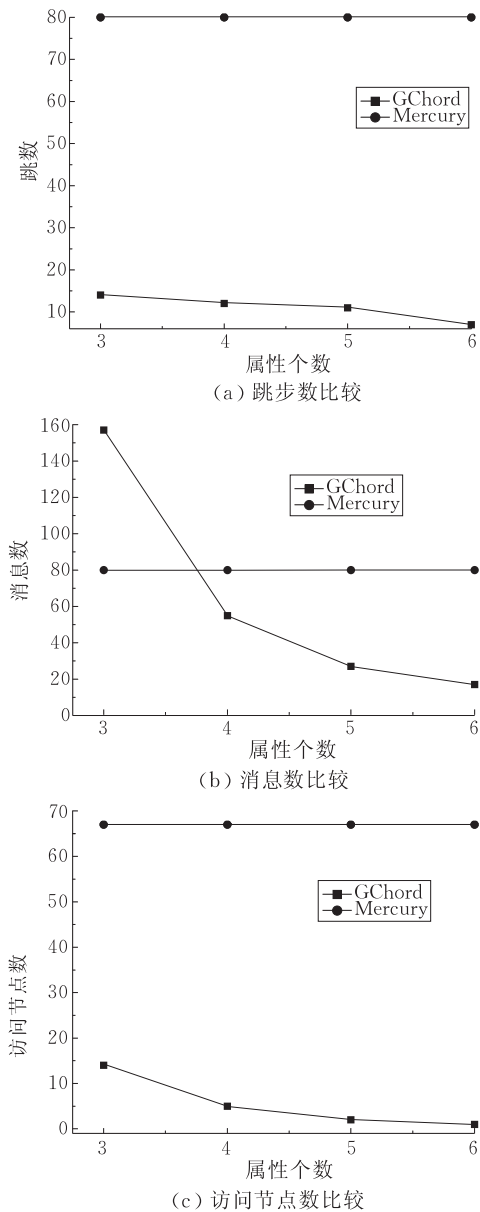


图 11 GChord 与 Mercury 的比较实验

数据在重叠网络上的索引和查询问题,现有方法主要可以分成 3 种:

(1) 利用空间填充曲线将多维数据投影到一维空间,再进行索引. BATON<sup>[10]</sup> 即属于这种技术. 它在 Chord 环的基础上,构造一棵平衡二叉树. 区域查询(range query)可以通过二叉树进行处理. 由于多维空间被映射到一维空间上,因此它对于 MAQ 查询无法进行有效的处理. 此外, BATON 对于包含种属型的数据无能为力.

(2) 直接构造分布式多维数据索引. VBI-Tree<sup>[11]</sup> 即属于这种技术. 它在传统的集中式多维数据索引技术<sup>[1]</sup> (如 R 树或 M 树)的基础上,解决了大规模分布式系统中的链接不可靠问题,其本质是类 Chord

环与多维数据索引技术的结合. VBI-Tree 对于多维近似搜索非常高效,但处理仅限制某些属性的 MAQ 查询时需要发送大量消息访问大量节点. 和 BATON 一样, VBI-Tree 对于包含种属型的数据无法处理.

(3) 为每一维数据构造一个重叠网络. MAAN<sup>[12]</sup> 和 Mercury<sup>[13]</sup> 都属于这类技术. 由于需要构造多个重叠网络,因此其维护代价较大. 此外,在处理 3 个以上属性时,由于单个重叠网络仅能过滤掉有限的无关数据,因此会使得这类方法浪费大量消息数去访问无关数据.

8 小 结

本文介绍了一种大规模分布式系统上的多属性查询处理技术 GChord. 它可在任意类 Chord 重叠网络上使用. GChord 能够处理对数值型和种属型属性的索引和查询. 它首先利用格雷码和散列技术,对两种类型的属性值进行编码,再利用编码混洗获得每个元组的最终编码. 这些编码可在重叠网络上利用“一位差异”性质进行索引. 在此基础上,多属性查询被转换为对于重叠网络上节点的访问. 查询通过多播树被发送到包含索引项的节点,以获得查询结果. 利用索引缓存和多播树聚类技术, GChord 可进一步减少查询消息的跳步数和网络中的消息总数. 实验结果表明,和现有利用多个重叠网络的多属性查询处理技术相比, GChord 占用更少的带宽、访问更少的节点,并能获得更少的查询延迟.

基于 GChord, 本文的后续研究工作包括对于 128 位节点标识符所能索引的最多属性的分析以及对于属性个数不定、数值型属性的值域未知的数据的索引和查询处理技术.

参 考 文 献

[1] Ramakrishnan R. Database Management Systems. New York: WCB/McGraw-Hill, 1998

[2] Özsu M T, Valduriez P. Principles of Distributed Database Systems. Second Edition. Upper Saddle River, NJ: Prentice-Hall, 1999

[3] Zhou A Y, Qian W N, Zhou S G, Ling B, Xu L H, Wee Siong Ng, Beng Chin Ooi, Tan Kian-Lee. Data management in peer-to-peer environment: A perspective of BestPeer. Journal of Computer Science and Technology, 2003, 18(4): 452-461

- [4] Shirky C. Listening to napster//Oram A ed. Proceedings of the Peer-to-Peer: Harnessing the Power of Disruptive Technologies. O'Reilly, 2002: 21-37
- [5] Aderson D. SETI@Home//Oram A ed. Proceedings of the Peer-to-Peer: Harnessing the Power of Disruptive Technologies. O'Reilly, 2002: 67-76
- [6] Miller J. Jabber: Conversational technologies//Oram A ed. Proceedings of the Peer-to-Peer: Harnessing the Power of Disruptive Technologies. O'Reilly, 2002: 67-76
- [7] Ghemawat S, Gobiolf H, Leung S-T. The Google file system//Proceedings of the SOSP 2003. New York, USA, 2003: 29-43
- [8] Dean J, Ghemawat S. MapReduce: Simplified data processing on large clusters//Proceedings of the OSDI 2004. San Francisco, CA, USA, 2004: 137-150
- [9] Chang F, Dean J, Ghemawat S, Hsieh W C, Wallach D A, Burrows M, Chandra T, Fikes A, Gruber R. Bigtable: A distributed storage system for structured data (awarded best paper!)//Proceedings of the OSDI 2006. Seattle, WA, USA, 2006: 205-218
- [10] Jagadish H V, Ooi B C, Vu Q H. BATON: A balanced tree structure for Peer-to-Peer networks//Proceedings of the VLDB 2005. Trondheim, Norway, 2005: 661-672
- [11] Jagadish H V, Ooi B C, Vu Q H, Zhang R, Zhou A Y. VBI-tree: A Peer-to-Peer framework for supporting multi-dimensional indexing schemes//Proceedings of the ICDE 2006. Atlanta, Georgia, USA, 2006: 34
- [12] Cai M, Frank M R, Chen J, Szekely P A. MAAN: A multi-attribute addressable network for grid information services//Proceedings of the GRID 2003. Phoenix, Arizona, USA, 2003: 184-191
- [13] Bharambe A R, Agrawal M, Seshan S. Mercury: Supporting scalable multi-attribute range queries//Proceedings of the SIGCOMM 2004. Portland, OR, USA, 2004: 353-366
- [14] Stoica I, Morris R, Liben-Nowell D, Karger D R, Kaashoek M F, Dabek F, Balakrishnan H. Chord: A scalable peer-to-peer lookup protocol for internet applications. IEEE/ACM Transactions on Networking, 2003, 11(1): 17-32
- [15] Crainiceanu A, Linga P, Machanavajjhala A, Gehrke J, Shanmugasundaram J. P-ring: An efficient and robust P2P range index structure//Proceedings of the SIGMOD Conference 2007. Beijing, China, 2007: 223-234
- [16] Gray F. Pulse code communications. In: U.S. Paten 2632058, 1953
- [17] Karnaugh M. The map method for synthesis of combination-logic circuits//AIEE Transactions, 1953, 72(1): 593-599



**ZHOU Ao-Ying**, born in 1965, Ph. D., professor, Ph. D. supervisor. His research interests include massive data management, Web data management and Web mining, Web services, data streams, and P2P computing systems.

**ZHOU Min-Qi**, born in 1980, Ph. D. candidate. His re-

search interest data management in Peer-to-Peer networks.

**QIAN Wei-Ning**, born in 1976, associate professor. His research interests include data stream query processing and mining, and large-scale distributed computing for database applications.

**ZHANG Rong**, born in 1978, Ph. D.. Her research interests are Peer-to-Peer computing, Web mining and knowledge grid resource management.

## Background

Peer-to-Peer (P2P) systems provide a new paradigm for information sharing in large-scale distributed environments. Though the success of file sharing applications has proved the potential of P2P-based systems, the limited query operators supported by existing systems prevent their usage in more advanced applications.

Much effort has been devoted to provide fully featured database query processing in P2P systems. There are several differences between query processing for file sharing and database queries. Firstly, the types of data are much more complex in databases than those in file names. Basically, numerical and categorical data types should be supported. Sec-

ondly, files are searched via keywords. Keyword search is often implemented by using exact match query. However, for numerical data types, both of the exact-match queries (or point queries) and the range queries should be supported. The last but not the least, user may issue queries with constraints on variant number of attributes for database applications. This last requirement poses additional challenges for database style query processing in P2P systems.

Therefore, to support complex query processing in a large-scale distributed environment is an important problem in massive data processing.