MCAN：一种可扩展改进的内容访问网络*)

彭小燕　杨寿保　陈东锋
（中国科学技术大学计算机科学系　合肥230026）

摘　要　特定数据项存储位置的定位效率是P2P应用面临的基本问题。本文提出的可扩展改进的内容访问网络(MCAN,Modified Content -Addressable Network)解决了数据项定位的效率问题。它将分布式哈希表和兴趣聚类紧密结合，根据查询数据的兴趣在类似于Internet规模的MCAN上高效地定位查询数据。理论分析结果表明，这种方法具有可扩展性、容错和完全自治的特点。本文通过仿真来阐述它的可扩展性和高效路由的特点。

关键词　MCAN，兴趣聚类，P2P

MCAN:A Scalable Modified Content -Addressable Network

PENG Xiao-Yan　YANG Shou-Bao　CHEN Dong-Feng
(Department of Computer Science,University of Science and Technology of China,Hefei 230026)

Abstract　A fundamental problem that confronts peer-to-peer application is to efficient locate the node that stored a particular data item. In this paper we introduce the concept of Scalable Modified Content -Addressable Network (MCAN) that combining Hash table and interesting cluster together and resolving the fundamental problem. We can locate requested data efficiently on MCAN system basing-on the interesting value of requested data. Results of theoretical analysis show that MCAN is scalable,fault-tolerant and completely self-organizing. We demonstrate its scalability and efficient routing properties through simulation.

Keywords　MCAN,Interesting-cluster,P2P

1　引言

P2P系统是每个节点都具有等同作用和交换信息的能力，并可直接进行通信的分布式系统。P2P文件共享系统被广泛地用于海量数据的共享和交换，如Gnutella[1]、Freenet[2]和Napster。除此之外，P2P将应用到更广泛的领域，比如语义Web服务的P2P网络提供服务的分布式访问。这些系统的利用都依赖于搜索机制的效率。而当前的P2P网络是在无组织的方式下组建的，具有可扩展性、网络负载和路由延迟问题，使得搜索时间超过了人们能够接受的范围。

目前在搜索机制效率和网络可扩展性上展开了许多工作。基于兴趣聚类的方法就是其中之一，它将系统中具有相同或者相似兴趣的节点聚成一类来改善内容定位效率。分布式哈希表(DHT)也用到许多P2P系统中——如Chord[3]，CAN[4]，这些系统利用DHT直接地定位内容使得搜索效率大大提高，同时也解决了系统的可扩展性问题。本文中提出的MCAN上的兴趣聚类将兴趣聚类和DHT相结合，在大规模的P2P系统上提供高效的内容搜索。

本文第2节描述相关工作，第3节介绍MCAN的基本设计，在接下来的第4节对MCAN基本结构做一些改善，第5节是仿真与分析，最后是总结。

2　相关工作

CAN[4]具有较好的可扩展性、容错性和完全自组等特点，它在d-虚拟坐标空间上利用DHT来实现内容定位。整个虚拟坐标空间在系统中的所有节点间动态地划分，以保证系统中的每个节点都拥有坐标空间的一个zone。系统中每个数据的*key*根据DHT映射到虚拟坐标空间的点P上，*key*所对应的(*key*,*value*)对就被存储到拥有P所在zone的节点上。当节点要发起数据查询时，它先利用DHT将查询数据的*key*映射到虚拟坐标空间的点P上，点P就是查询消息在坐标空间中的目的地址。节点根据自身的路由表将查询消息发送到离P最近的邻居上。对于一个具有N个节点的CAN系统，每个节点维护$O(d)$个状态，每次搜索的搜索代价是$O(dN^{1/d})$。

Chord[3]是一种基于DHT的分布式查询协议，系统中的每个节点维护其它$O(\log N)$个节点的路由信息，每次搜索只要付出$O(\log N)$的搜索代价就能准确地定位到查询数据。Chord仅提供一个简单的映射操作，它利用DHT将数据项的键值映射到节点n上，并将键值/数据项对存储到n上。这样通过将键值和数据项联合很容易在Chord上定位数据。

Gnutella上基于兴趣的定位[5]在具有相似兴趣的对等节点之间建立一条捷径(short-cut)。当进行查询时先用捷径定位内容，只有当捷径不能定位内容时才用Gnutella的洪泛法(flooding)定位。这种基于兴趣的定位在不改变Gnutella下层机制的条件下，提高了内容的定位效率。同时将基于兴趣定位作为一个单独层来实现使得这种基于兴趣定位的方法具有很强的可移植性，可以应用到不同的P2P系统中。

Hypercup[6]将系统中节点提供的n个基本服务类型进行归类，并将这些基本服务类型作为构造类型来构造d-外部超

*)本文得到国家自然科学基金项目(编号60273041)和国家863计划(编号2002AA104560)的资助。彭小燕　硕士研究生，主要研究方向为网络计算，P2P搜索机制；杨寿保　教授，博士生导师，主要研究方向为计算机网络及应用；陈东锋　硕士研究生，主要研究方向为网络计算。

立方体($d=\lceil \log n \rceil$)。外部超立方体中的每个顶点根据所在超立方体中的位置进行命名,表示提供相似服务的节点集合(概念聚类)。每个顶点表示的概念聚类又以超立方体的方式进行连接,构成一个内部超立方体。当一个节点发起查询时,查询消息先在外部超立方体上进行高效的广播,以保证每个顶点至多只能接收消息一次。当顶点是查询的目的概念聚类时,将查询消息发送到它的内部超立方体上确定查询内容所在位置,并将结果返回给发起者。这种方法和本文提出的MCAN有相似之处,内容定位都涉及两个不同的虚拟空间:聚类组成的虚拟空间和聚类内部节点组成的虚拟空间。

3 MCAN的基本设计

这里先对MCAN的基本形式进行描述,在下一节中将对MCAN做改进以提高其性能。

MCAN是从CAN[4](Content -Addressable Network)调整而来的,它假设每个节点都具有一个兴趣,每个兴趣都对应着一个唯一的兴趣值。分布式哈希表(DHT)将兴趣值哈希到d-圆环面的虚拟d-笛卡儿坐标空间的一点P上。具有相似兴趣的节点都被哈希到同一点上,形成一个兴趣聚类(interesting cluster)。兴趣聚类以Super-Peer[7]的方式组织,一个兴趣聚类可有多个*Super*节点,*Super*节点维护着一张路由表和它的对等节点的内容索引表。这样系统中就形成了两个不同的虚拟空间:兴趣聚类组成的d-坐标空间和节点组成的兴趣聚类空间。由于新兴趣的不断创建和兴趣的不断消亡,在不同时刻坐标空间中的点的数量是动态变化的。同时每个兴趣聚类中的节点数量(*Super*节点和对等节点)也是动态变化的。

在MCAN中数据存放在各个节点(*Super*节点和对等节点)上,要定位查询内容首先必须用DHT确定查询内容所在的兴趣聚类在d-坐标空间上的位置,在确定内容的兴趣聚类的位置后,再将内容的查询消息通过各个兴趣聚类中的*Super*节点路由到目标聚类上。最终的内容查找是在目标兴趣聚类上完成的。查询消息到达目标兴趣聚类的*Super*节点时,*Super*节点先查看查询数据是否在本机上存储,如果没有则查看它维护的内容索引确认它的对等节点上是否有查询的数据。

为了介绍MCAN的基本框架,下面将对MCAN中的三个最基本的部分进行描述:MCAN路由、MCAN结构的构造和MCAN结构的维护。

3.1 MCAN中的路由

从直观上来说,MCAN中的路由就是在d-虚拟坐标空间中的源兴趣聚类到目标兴趣聚类的直线路径和在目标聚类中内容定位到具体节点的路径之和。由于兴趣聚类采用Super-Peer的方式组织,目标聚类中的内容定位最多可在两跳内找到,因此MCAN的路由可以看成是d-坐标空间中从源兴趣聚类到目标兴趣聚类的路由。为了使d-虚拟坐标空间中的任意两个兴趣聚类之间都具有可达性,每个兴趣聚类的*Super*节点都维护着在d-坐标空间中和它相邻的$2d$个邻居的IP地址和邻居所在的兴趣聚类,在同一个兴趣中的*Super*节点具有相同的路由表。兴趣聚类C的第i个邻居是指将整个d-圆环面以C为中心展开的第i象限中离C距离最近的点(兴趣聚类),C中的*Super*节点的第i个邻居节点就是第i象限中离C最近的兴趣聚类的*Super*节点。

在了解路由表的构造之后,接下来对路由算法进行详细的介绍。由于MCAN有两个不同的虚拟空间,因此MCAN的路由也穿插于这两个不同虚拟空间内,因而路由被分成两个步骤:

步骤1 在d-虚拟坐标空间上的路由,确定查询内容所在的兴趣聚类。当一个节点发起一个查询时,如果该节点不是*Super*节点那么该节点将查询消息发送给它的*Super*节点S。当查询消息到达S后,S分析查询消息得出查询内容的兴趣值,然后根据这个兴趣值将这个兴趣值哈希到查询内容所在目标兴趣聚类上。在确定好目标聚类后,S节点根据自身维护的路由表将消息发送到离目标兴趣聚类距离最近的路由邻居节点上,这个过程不断重复直到查询消息到达目标兴趣聚类。

步骤2 当查询消息到达目标兴趣聚类后,目标兴趣聚类中的*Super*节点确定查询内容所在的具体位置。查询消息到达后,接收到查询消息的*Super*节点先查看查询内容是否在本机上,如果在本机上那么将内容返回给发起查询的节点,否则*Super*节点查看它维护的内容索引确定内容是否在它的对等节点上,如果不在它的对等节点上那么将查询消息发给聚类中其余所有的*Super*节点,其它*Super*节点确定内容的具体位置。图1是在二维坐标空间上的一个路由例子。

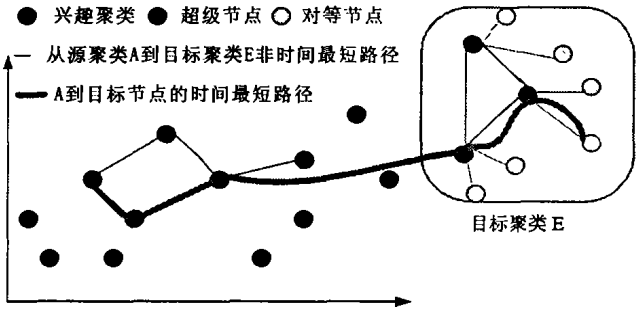

图1 从源聚类A到目标聚类E中的对等节点的路由路径

对于一个具有m个兴趣聚类d-坐标空间,如果这m个兴趣聚类所对应的坐标上的点集中在d-坐标空间上的一个集中的区域(该区域在各个维上都具有相同的长度)内,那么路由路径的长度是$O(\sqrt[d]{m})$跳,每个*Super*节点至多维护$2d$个邻居节点的路由信息。对路由的具体分析将在第5节的仿真中提到。

要提到的是在系统中任意两个兴趣聚类之间存在多条不同路径,因此即使其中的一个或者几个邻居节点发生失败也可以自动选择下一个最好的路径成功进行路由,除非一个兴趣聚类的所有邻居节点都失败,才可能出现路由失败的情况。因而对于系统中的任意两个节点,只要兴趣聚类的所有邻居节点以及节点所属的*Super*节点不失败时,就可以成功到达对方。

当*Super*节点的路由邻居失败时,*Super*节点将会重新选择一个邻居节点填补失败节点的空缺,这将会在3.4节中提到。

3.2 MCAN的构造

在构造MCAN当中,我们假定所有新节点加入时都知道自己的兴趣值。为了使MCAN具有可扩展性和低负载特性,新节点n加入系统不应该引起大量消息传递而加重系统负载,因此n加入只能对系统中已存在的少量节点产生影响。MCAN是通过下列方式处理新节点加入的,该过程只发送少量消息影响系统中的少量节点。

A 新节点加入 新节点n要加入系统必须先定位所属兴趣聚类的位置后才能加入到所属兴趣聚类中。节点n加入过

程是通过系统中已存在节点 n' 完成的。新节点 n 找到已存在于系统中的一个节点 n' 作为它加入系统的联系节点，并和 n' 进行联系，通过 n' 加入到 n 所属的兴趣聚类中。具体过程如下(考虑到 *Super* 节点的处理能力，我们假设每个 *Super* 节点至多拥有 k 个对等节点)：

步骤1：发送 JOIN 消息给 n'，如果 n' 不是 *Super* 节点则跳到第步骤5。

步骤2：节点分析 n 的 JOIN 消息，获得 n 的兴趣值，通过 DHT 将这个兴趣值哈希到 d-坐标空间的点 P 上。节点利用自身的路由表将 JOIN 消息路由到点 P，如果路由成功则跳到步骤3，否则跳到步骤4。

步骤3：点 P 所表示的兴趣聚类中的 *Super* 节点接收到 n 的 JOIN 消息，然后将 JOIN 消息发送给兴趣聚类中其它所有的节点，并从中选出对等节点数最少的 *Super* 节点 S 接收新节点 n。新节点 n 加入到 S 会出现两种情况：

· S 拥有的对等节点数小于 k，那么 n 可以直接加入到 S 的对等节点行列中。S 将对等节点数增加1，并增添 n 的内容索引，节点成功加入系统。

· S 拥有的对等节点数等于 k，n 加入到 S 对等节点行列中会引起 S 拥有的对等节点的分裂。在这种情况下 S 的具体操作是：S 将包括 n 和自己在内的它的所有节点划分成两个相等的节点集合，每个集合分别选出一个新的 *Super* 节点。S 将它的路由表信息传给这两个新的 *Super* 节点，到此节点加入完成。由于在 MCAN 中 *Super* 节点具有很重要的作用，所以希望选择的 *Super* 节点不是恶意节点，并且有比较好的处理能力和稳定性。因此我们将 EigenTrust[8] 应用到 MCAN 中来进行 *Super* 节点的选取(新 *Super* 节点的选取算法将在3.3节中提到)。

步骤4：节点发现点 P 在 d-坐标空间中的邻居兴趣聚类，并从各个邻居兴趣聚类中找出离 n 距离最近的 *Super* 节点作为它的路由邻居节点，建立起 n 的路由表。节点加入过程完成。

步骤5：n' 将 n 的 JOIN 消息发送给它的 *Super* 节点 S'，然后跳至步骤2。

B 系统中相关节点维护信息的更新　新节点 n 加入到系统后可能会引起某些节点信息状态的改变，为了保证这些节点状态的正确性，当新节点加入后要对这些节点的信息进行更新。节点加入后信息更新的情况如下：

1)新节点直接加入到一个已存在的兴趣聚类中并且没有引起任何的分裂操作，不引起节点信息更新。

2)节点加入到一个已存在兴趣聚类中，引起了节点集合的分裂，更新兴趣聚类和它的邻居的路由表。聚类的邻居重新从该聚类的所有 *Super* 节点中选择路由邻居更新路由表。同样聚类中的 *Super* 节点也从邻居聚类的所有 *Super* 节点中重新选择路由邻居节点。

3)节点加入引起新兴趣聚类的创建时，如果新兴趣聚类是它的路由邻居的第 i 个邻居，那么更新它的路由邻居的第 i 个路由表表项；如果新兴趣聚类在它的邻居聚类原来路由表的第 i 表项节点所在聚类 C 的第 j 象限内，那么查看 C 中 *Super* 节点的第 j 个路由表项，如果 n 所在聚类到达 C 的聚类要小于第 j 个表项所在聚类到达 C 的聚类，那么用 n 更新 C 中所有 *Super* 节点第 j 个路由表项。

从整个节点加入处理过程来看，新节点加入最多影响到 $4d$ 个节点状态的改变，所以新节点加入发送的消息数最多是 $m+2d$。

3.3 新 *Super* 节点选取算法

已知节点集合 $\{N_1, N_2, \cdots, N_i\}$(节点集合的 Session 时间集合为 $\{S_1, S_2, \cdots, S_i\}$)要从中选出一个超节点，我们先对这个集合中的所有节点用 EigenTrust[8] 算法进行评估得到它们信誉度的集合 $\{T_1, T_2, \cdots, T_i\}$。然后将各个节点的处理能力进行从小到大的排序，并为每个节点定义一个处理能力的秩 $\mathrm{Rank}(N_j)=N_j$ 在排序中的位置。根据式(1)为每个节点计算：

$$Q_j = p\times T_j + q\times \mathrm{Rank}(N_j) + (1-p-q)\times S_j (1\leqslant j\leqslant i) \tag{1}$$

其中 $0<p,q<1$。根据式(2)从节点集合中选出这个集合的超节点。

$$Q_j = \max_{k=1}^{i}\{Q_k\} \Rightarrow Super = N_j (1\leqslant j\leqslant i) \tag{2}$$

值得注意的是每次引起分裂都会选择出处理能力和稳定性较好的节点作为 *Super* 节点。当系统不断扩展时，*Super* 节点的处理能力、稳定性和信誉都会不断地提升，系统也会更趋于稳定。

3.4 节点的离开和失败

当节点离开时，如果离开节点是 *Super* 节点系统仍要保证该节点拥有的所有对等节点仍可以被正常访问。在这种情况下的一般做法是从离开节点的对等节点中选择一个新 *Super* 节点取代离开节点的位置。同时系统还要保证路由的正确性，所以也会更新路由表。具体情况如下：

1　离开节点只是一般的对等节点，只要从它的 *Super* 节点上删除它的内容索引即可。

2　离开节点是一个 *Super* 节点

2.1　离开节点有对等节点。离开节点先查看在同一个聚类中的其它 *Super* 节点中，是否有接管它的对等节点后拥有的对等节点数仍不超过 k 的 *Super* 节点。

·如果有这样的 *Super* 存在，那么将它的对等节点转交给此 *Super*。

·没有这样的 *Super* 存在，那么在该 *Super* 节点剩余的对等节点集合中选出一个 *Super* 节点。

2.2　*Super* 节点不拥有对等节点。将它的离开通知它的邻居和同一聚类中的其它 *Super* 节点。

Super 节点拥有它的对等节点的内容索引，为了保证索引的内容不是陈旧的，对等节点定期给它的 *Super* 节点发送刷新消息，*Super* 节点接收到刷新消息后返回一个应答。这种机制可以保证当一个节点发生失败时，可以检测到失败并采用相应的恢复措施。当一个 *Super* 节点在一个较长时间没有接收到对等节点 n 的刷新消息就可以判定 n 失败，那么直接删除 n 在它上面的索引。当对等节点 n 较长时间没有接收到它的 *Super* 节点 S 的刷新应答消息可以判断 S 失败，那么 n 发起选择 *Super* 节点的操作，从它们中选出一个 *Super* 节点代替失败的 *Super* 节点。替代的 *Super* 节点以失败 *Super* 节点的身份执行失败节点的离开过程。

4 MCAN 基本设计的改进

4.1 多路由表

在上面我们讲述了 MCAN 的基本设计，对于一个 d-坐标虚拟空间的 MCAN 来说，每个 *Super* 节点维护 $O(d)$ 个状态，当兴趣聚类集中在一个集中区域时它的路由跳数是 O

($\sqrt[d]{m}$)。这里的路由跳数指的是应用层的路由跳数，而不是IP层跳数。路由邻居虽然在应用层上只相差一跳，但是实际上可能是很多个IP跳，而在基本的MCAN设计中一个聚类中的所有 *Super* 节点都拥有相同的路由表，可能导致路由邻居节点并不是离它最近的邻居 *Super* 节点。如果同一兴趣聚类中 *Super* 节点可以具有不同的路由表，每个 *Super* 节点从邻居兴趣中选择到它距离最近的 *Super* 节点作为它的路由邻居，那么在路由时从所有 *Super* 节点的邻居节点中选择到达目的兴趣聚类距离最近的邻居节点转发路由可以减少每一跳的路由延迟，这也就减少了这个路由的路由延迟。

多个路由表可以增加系统的容错性，由于聚类中不同 *Super* 节点具有不同的路由表，这相当于一个聚类有多个路由表，只有当所有 *Super* 节点的路由邻居节点失败时才可能导致路由失败。当 *Super* 节点的一个或者几个邻居节点发生失败时，仍然可以选出下一个最好的路由。

4.2 多兴趣聚类

在MCAN的基本设计中每个节点只属于一个兴趣聚类，在节点加入到兴趣聚类后如果节点从别的兴趣聚类 C 中下载内容，这样会导致那些存储在节点中的下载内容不能被其它节点访问到。而且当节点存储的兴趣聚类 C 的内容比例增大时，会导致节点的兴趣的改变。当聚类 C 中具有这些下载内容的节点离开或失败时，使得实际上存在的内容(存在于 C 聚类外)变得不可用。为了增加数据的可用性并确保节点兴趣的正确性(即当节点的兴趣发生改变时，保证节点要加入到相应的兴趣聚类中)，我们对MCAN的基本设计做些改进——一个节点可以同时属于多个不同的兴趣聚类。

为了使节点 n 加入到多个兴趣聚类中，我们在节点 n 上设置一个计数器来计数 n 当前所属兴趣聚类的数目。我们假设一个节点可以同时加入到3个不同聚类中。而聚类的改变是根据节点 n 上存储内容的比例来确定的，也就是说当 n 中某个新兴趣聚类的内容上升到总内容的前3位时，那么就说 n 的兴趣发生了变化。当 n 的兴趣发生改变时，在 n 中所占比例最小的那个 *Super* 节点 S_1 执行如下操作：

1)如果 n 的计数器的计数小于3，那么 S_1 通知新兴趣聚类中具有最少对等节点的 *Super* 节点 S_2 接管节点 n，并将计数器的计数增加1；

2)n 的计数器的计数等于3，S_1 通知新兴趣聚类中具有最少对等节点的 *Super* 节点 S_2 接管节点 n，并将 n 在 S_1 中的相关信息删除，计数器的计数保持不变。

多兴趣的方法虽然增加了数据的可用性，但是同时也使得一个节点同时在多个 *Super* 节点上建立索引，这就增加了索引开销。

4.3 多兴趣查询

基本MCAN中的查询是单兴趣的，而实际上多兴趣查询是经常发生的。为了增强对实际应用的支持，在基本MCAN上扩充多兴趣查询，即查询内容可以涉及多个兴趣。多兴趣查询和单兴趣查询过程基本相似，只是增加了超级节点对查询内容的额外处理。例如节点 n 发起查询 L：兴趣 i V 兴趣 j V 兴趣 k，它的 *Super* 节点 S 对 L 进行最小项分解，即将 L 分解成单兴趣集合{兴趣 i，兴趣 j，兴趣 k}。把查询消息分别发送到兴趣 i、兴趣 j 和兴趣 k 所对应的目标聚类上。每个目标聚类都返回结果给 S，S 从多个返回结果中选择和 L 最匹配的结果返回给 n。

4.4 捷径

兴趣和兴趣之间并不是毫无关联的，它们之间可能存在紧密的关联。比如兴趣 i 中的节点经常访问兴趣 j 中的内容，如果兴趣 i 和 j 并不是邻居关系，那么每次兴趣 i 中的节点访问 j 中的内容时，都必须经过其它兴趣聚类的多次转发。如果在兴趣 i 和 j 之间建立一条捷径，每次 i 中节点访问 j 的内容时都通过这条捷径查询，不必通过多次路由转发，这样可以进一步提高路由效率和带宽利用率。这可以通过每个兴趣聚类维护一张大小为 r(常数)的捷径表来实现。为了反映兴趣之间的最新关联，兴趣聚类可以根据情况更新捷径表。

5 仿真与分析

本文的仿真是在2-坐标空间上针对路由路径长度(跳数)和带宽利用率(路由时所发送的消息数量)进行的。为了分析路由的路由长度以及带宽利用率，这里定义了几个参数：路由平均跳数avghop、路由最大跳数maxhop、平均消息数argmsg和最大消息数maxmsg，我们利用这几个参数对MCAN的性能进行分析。本文将五个1000次的路由作为一次仿真，取这五次仿真的平均值作为一个分析结果，最终通过三个这样的分析结果得出我们的结论。

5.1 兴趣分布和节点增长对路由的影响

假设坐标空间中具有 m 个兴趣聚类，在保持兴趣聚类 m 不变的情况下改变兴趣的分布(将相同数量的兴趣分布在坐标的不同区域内)，仿真结果如图2。在同一分布和兴趣数量情况下增加节点个数，仿真结果如图3所示。通过图2可以得出路由路径长度和兴趣在坐标中的分布有着很大的关系。若所有兴趣分布在一个能容纳它的最小的在各个维上都相等的区域内，那么路由具有最大的带宽利用率和最小的路由跳数 $O(\sqrt[d]{m})$——$\sqrt[d]{m}+3$跳。若兴趣随意分布在坐标中最大路由跳数可达到 $O(m)$。

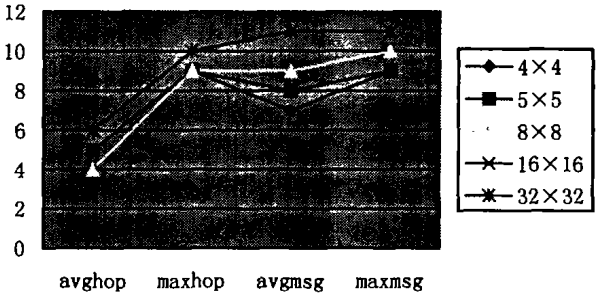

图2 12个兴趣分布在坐标空间的不同区域内

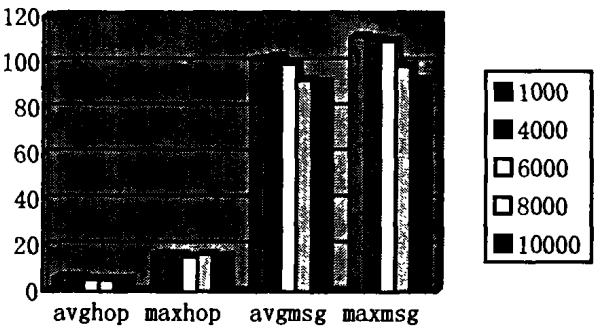

图3 172个兴趣分布在16×16区域内

从图3中可以看出，在兴趣分布和数量不变的情况下，节点的增加不会影响到路由和带宽的利用率。

5.2 同一分布条件下兴趣增长对路由的影响

为了分析兴趣增长对路由的影响，我们通过在坐标空间

上的16×16区域内不断增加兴趣数量来进行多次仿真(每次仿真的节点数目都是4000),仿真结果如图4所示。从图4可以看出,在同一分布和节点数的情况下,兴趣数量的增加会增加路由跳数和消息数。并且在最坏情况下路由跳数以 $O(m)$ 增长,而且消息数的增长会小于兴趣的增长。

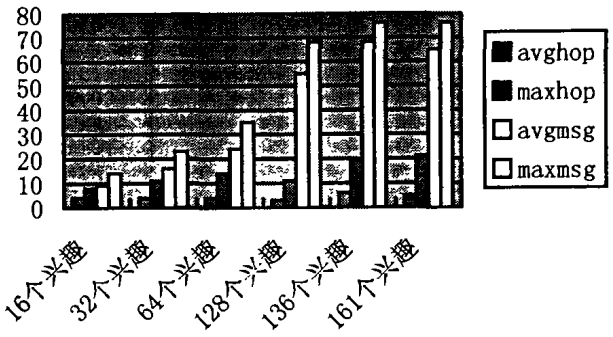

图4 同一兴趣分布下的兴趣增加

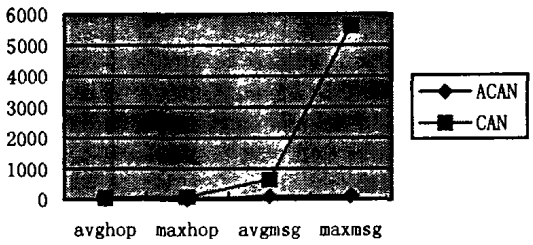

图5 MCAN 和 CAN 路由性能比较

5.3 MCAN 和 CAN 的比较

为了对 CAN 和 MCAN 两种情况下的路由进行比较,仿真将 MCAN 中10000个分布在16×16区域内的172个兴趣上的节点组织成一个二维 CAN 系统,对这10000个节点在这两种情况下的路由做分析(如图5)。从图5可以看出,CAN 中的路由路径长度是 $O(n^{1/d})$,消息数是 $O(n)$。由此可以得出结论,同样的节点在 MCAN 中可以获得比在 CAN 中更好的路由路径长度和路由消息数,即 MCAN 可以在保障系统可扩展性条件下改善路由性能。

总结 本文提出 MCAN 上的兴趣聚类方法,利用 DHT 定位目标兴趣聚类,各个节点可以完全分布地通过局部信息成功路由到目标节点上,使得 MCAN 具有很好的扩展性。新节点可以随时地加入和离开系统,并且当节点失败时 MCAN 具有容错性。MCAN 在每一跳上都选择最小的邻居节点来转发路由,因而具有较低的延迟。经仿真表明 MCAN 的路由路径与系统中的节点数量没有关系,只和兴趣的分布和数量有关——路由在最好情况下至多经过 $O(m^{1/d})$ 跳就能到达目标节点,最坏情况下经过 O(m) 跳能到达目标聚类。MCAN 仍存在很多不足,如节点的同时加入和离开的处理,以及如何应对拒绝服务攻击构建安全的 MCAN。在将来的工作中我们将对上面的两个问题进行研究,进一步完善 MCAN。

参 考 文 献

1 Gnutella website: www. gnutella. com

2 Clarke I, et al. Freenet: Adistributed anonymous information storage and retrieval system. Freenet projectat freenet. sourceforge. net.

3 Rowston A, Druschel P. Pastry: Scalable, decentralized object location and routing for large scale peer-to-peer systems. Microsoft Research Ltd. Cambridge

4 Stoica I, et al. Chord: A scalable peer-to-peer lookup service for internet applications. ACM Sigcomm, 2001

5 Xu Zhiyong, Hu Yiming. SBARC: A Supernode Based Peer-to-Peer File Sharing System. http://www. ececs. uc. edu/~ oscar/papers.

6 Sripanidkulchai K, Maggs B, Zhang Hui. Efficient Content Location Using Interest-Based Locality in Peer-to-Peer Systems. Carnegie Mellon University

7 Schlosser M, et al. HyperCup-Shaping Up Peer-to-Peer Nerworks. Stanfor University

8 Kamvar S D, Schlosser M T, Garcia-Molina H. The EigenTrust Algorithm for Reputartion Management in P2P Nerworks. http://www-db. stanford. edu/

(上接第125页)

值,从而提高用户生产率?怎样使 RTDBMS 通用化?这些都是须进一步研究的问题。

总结 为提高实时数据库系统事务处理的成功率和可靠性,考虑事务的执行需求和价值,降低资源竞争程度,提高利用率,使系统收益最大化,针对现实世界中存在的混合事务,提出一种混合实时事务的接纳控制策略。分析了不同混合实时事务的价值模型、各种接纳控制策略及 ACMHRTT 算法,与传统的接纳控制协议作了简单比较分析,提出了须进一步研究的问题。

参 考 文 献

1 Nagy S, Bestavros A. Admission control for soft-deadline transactions in ACCORD. In: Proc. of the 3rd IEEE Real-Time Technology and Applications Symposium, Montreal, Canada, 1997. 160~165

2 Nagy S, Bestavros A. Value-cognizant admission control for RTDB systems. In: Proc. of the 17th IEEE Real-time Systems Symposium, Washington D. C, USA, 1996. 230

3 Chakravarthy S, Hong D, Johnson T. Incorporating load factor into the scheduling of soft real-time transactions: [Technical Report TR94-024]. Department of Computer and Information Science, University of Florida, 1994

4 夏家莉. 适用于嵌入式实时数据库系统的接纳控制机制 IACM. 计算机学报, 2004, 27(3): 295~300

5 刘云生, 何冰, 冉龙波. 混合实时事务的延期单调速率调度算法及其可调度分析. 计算机学报, 2004, 27(3): 289~294

6 萨师煊, 王珊. 数据库系概论(第三版). 高等教育出版社, 2002. 2