第33卷 第19期 计 算 机 工 程 2007年10月
Vol.33 No.19 Computer Engineering October 2007
·软件技术与数据库· 文章编号：1000—3428(2007)19—0072—03 文献标识码：A 中图分类号：TP391

基于P2P主题索引网络的数据库搜索算法

马光志，杨 曦，廖家国，卢炎生
（华中科技大学计算机应用系，武汉 430074）

摘 要：传统的P2P单层网络难于兼顾搜索效率和高动态性，存在单点失效和负载不均等问题，该文利用“双层主题索引网络”构建系统，融合了无结构和有结构网络的优点，采用多哈希函数策略加入节点、发布资源。基于兴趣度cache缓存和相对距离，选取高优先级节点进行通信，使模型在搜索速度、查准程度、单点失效、负载均衡等方面有了很大的改进。

关键词：基于主题索引的P2P；多哈希；cache缓存；距离

Database Search Algorithm of Topic-based Structured P2P Network

MA Guang-zhi, YANG Xi, LIAO Jia-guo, LU Yan-sheng
(School of Computer Application, Huazhong University of Science and Technology, Wuhan 430074)

【Abstract】 Traditional P2P model can not give attention to both searching efficiency and dynamic nature, thus, problems, such as single point-invalidation and load-disproportion exist. This paper presents a new P2P model, topic-based structured P2P network (TS-P2P), which inosculates the merit of unstructured and structured P2P model, adopts multilateral hash to insert peers and publish database resources. Based on cache buffer and distance, it chooses best peer to communicate by comparing PRI. This model improves the capability of P2P networks on searching speed, veracity, single point-invalidation, and load-disproportion.

【Key words】 topic-based structured P2P(TS-P2P); multilateral Hash; cache buffer; distance

P2P网络的应用大多数是基于文件系统的，基于数据库的应用研究刚刚起步。基于泛洪搜索，无结构P2P网络由于消耗大量的带宽，因此扩展性较差。有结构P2P网络基于分布式哈希表，搜索速度快且具有较好的可扩展性，但在节点加入和离开时，需要额外的维护，不适合高度动态的网络环境[1]。

一些学者提出了新的P2P网络模型，先将网络中的对等节点按一定规则进行聚类，按类组织到一起，从而提高查询的搜索效率。基于主题的聚类将节点按主题聚簇，搜索资源时首先定位到相关主题的簇[2]；而基于位置的聚类按节点的距离聚簇，搜索资源时首先在邻近簇内查找[3~4]。这些方法仍然不能兼顾主题与距离，而且需要中心节点维护所管辖的节点信息，一旦中心节点失效，那么所管辖的所有节点也同时失效。笔者采用一种新的P2P网络模型[5]——基于主题索引的P2P (topic-based structured P2P, TS-P2P)模型。TS-P2P的上层基于主题的有结构网络，下层基于距离的无结构网络，保证了查询的搜索效率，兼顾了网络的高动态性；通过多哈希函数，改进资源并发布策略，将资源映射到多个上层节点，有效地解决了索引负载瓶颈和单点失效问题。

1 双层网络模型

1.1 双层网络模型的结构

模型的上层具有结构主题索引节点(topic peer, TP)，下层具有无结构的普通节点(common peer, CP)。上层主题索引节点TP构成有结构CAN网络，维护本地邻居表L_N和索引信息表L_T，为下层CP节点提供索引服务，其性能稳定，且在线时间长，离开系统时要执行相应的修复程序。主题索引网络用Key表示节点提供的资源，基于数据库组群的应用，Key值表示数据库的特征，即数据库名，Location(Key,CP)表示Key在CP上的位置信息。

普通“纯对等节点”CP，不负责节点的索引维护，可以自由加入和离开系统。CP提供本地资源Key，利用哈希函数将Key映射到TP上，并将Key和Location(Key,CP)存放到索引表中。通过多哈希函数索引，CP可与多个TP建立关系。

当CP搜索资源时，将搜索请求递交给与邻接的TP，TP遵循有结构的网络路由策略，将找到的目的节点反馈给发出请求的CP，此后，CP和目的节点直接通信并交换数据。

与传统的混合模型相比，双层模型的主题索引节点不直接索引普通节点，只索引通过哈希函数映射到其上的节点，混合模型的索引节点直接索引与之相邻的普通节点。在双层模型中，普通节点可以运用多哈希函数映射到的多个索引节点，而混合模型的普通节点只映射到一个索引节点。

1.2 基于多哈希策略的资源发布算法

当新节点CP_i加入系统，或CP_i变更数据库资源时，CP_i需向主题索引节点发布资源。CP_i联系相邻的索引节点TP_m，计算CP_i提供的资源Key的哈希值hash(Key)=ID，若网络上存在$node_{id}$等于ID的主题索引节点TP_n，则TP_m按上层CAN网络规则路由到TP_n，将Key及位置存放到TP_n的索引表中，即

$$\text{Topic-Index}(TP_n)=\text{Topic-Index}(TP_n)\cup\{(\text{Key},\text{Location}(\text{Key},CP_i))|\text{hash}(\text{Key})=\text{Nodeid}(TP_n)\}$$

若网络上没有$node_{id}$等于ID的主题索引节点，则按照CAN规则将普通节点CP_i加入到上层索引网络中，并将其作为一个主题索引节点，建立自己的索引表，即

基金项目：湖北省自然科学基金资助项目(2006ABA082)
作者简介：马光志(1964－)，男，副教授，主研方向：数据库与数据挖掘；杨 曦、廖家国，硕士；卢炎生，博士生导师
收稿日期：2006-11-28 **E-mail：**maguangzhi@hust.edu.cn

Topic-Index(CP_i)={(Key,Location(Key,CP_i))|hash(Key)=$Node_{id}$(CP_i)}

双层模型通过采用多哈希策略，将数据库资源发布到多个主题索引节点上，如图 1 所示。

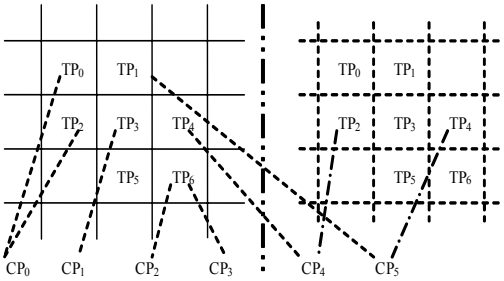

图 1　基于主题索引的双层 p2p 网络模型

以 2 个哈希函数为例，对CP_4待发布的数据库资源Key：$hash_1$(Key)=ID_1,$hash_2$(Key)=ID_2。ID_1为主题索引节点TP_4的ID号，ID_2为TP_2的ID号，则TP_m分别路由到TP_4和TP_2，这样Key就在上层索引网络上保存了 2 个副本，相当于建立 2 个虚拟的CAN覆盖网络。索引表中的索引分区存放，按哈希函数的不同标记为$hash_1$和$hash_2$索引区，它们在逻辑上是相互独立的。若某个主题索引节点失效，则启动另一个哈希函数，解决了单点失效问题，同时，当某一主题索引节点过于繁忙时，也可以启动第 2 条路径查找，缓解了单点的负载压力。笔者以 2 个哈希函数为例，给出资源发布算法。

输入 CP_i为发布资源节点，TP_m为CP_i的邻居索引节点，ID_1=$hash_1$(Key),ID_2=$hash_2$(Key)。

输出 Topic-Index(TP)索引表

Search(TP_i), TP_i∈TP; //搜索主题索引节点

Publish(ID_1)；Publish(ID_2);

其中，Publish(ID)为资源发布函数。

```
Publish(ID){
  if(Node_id(TP_n)=ID)//网络中存在目标索引节点
  {  Route(Key, TP_m,TP_n); //按CAN方式路由到TP_n
     Topic-Index(TP_n)=
     Topic-Index(TP_n)∪{(Key, Location(Key, CP_i)} }
  else
  {  Let CP_i be a TP, join into the CAN;
     //CP_i作为主题索引节点
     Topic-Index(CP_i)={(Key,Location(Key,CP_i))} }
}
```

采用多哈希函数发布资源，主题索引节点维护的索引信息增多了，需要为每个哈希函数分配一个专用的索引区。虽然网络在资源发布的初期增加了工作量，但是索引区分片策略使“多哈希”逻辑独立，不会降低资源的搜索效率。

2　资源搜索策略

2.1　上层网络资源搜索路由算法

如图 1 所示，当有查询请求时，类似于资源发布过程，CP_i联系与之相邻的主题索引节点TP_m，TP_m通过查询请求的Key得到ID=hash(Key)，按CAN规则路由到节点号等于ID的节点TP_n，如果节点TP_n不在线或者忙，则启动第二哈希函数，路由到TP_k上。找到主题索引节点TP_n或TP_k后，搜索本地索引表，找到符合查询条件的节点集合，即

A={CP_j| CP_j∈Topic-Index(TP)}

基于双哈希函数的搜索算法描述如下。

输入 CP_i发起查询的节点，TP_m为CP_i的邻居节点，TP_n为$hash_1$得到的索引节点，TP_k为$hash_2$得到的索引节点。$state_1$、$state_2$分别为两哈希函数搜索索引节点返回状态。

输出 集合 A。

```
if (state1=free)    Choose(TPn); // TP_n为目标节点
else if(state1=not-exist)
{ if((state2=not-online or not-exist)) Return failed;
   if((state2=free or bussing)) Choose(TPk);
 }
else If (state1=not-online or bussing)
{ if(state1=not-online)&(state2=not-online or not-exist)
      Return failed;
   if(state1=not-online)&(state2=free or bussing)
      Choose(TPk);
   if(state1=bussing)&(state2=not-online or bussing or not-exist)
      Choose(TPn);
   if(state1=bussing)&(state2=free)
      Choose(TPk);
}
Send(A, Location(Key, CP_j), CP_i);
//搜索本地索引表，将符合条件的节点集合 A 返回给查询节点
```

在主题索引节点得到的集合A中，CP_j可能不存在，或有一个或多个。当A为空时，返回无结果信息给源节点；当A只有一个CP_j时，源节点和目的节点通过底层协议直接通信；当A存在多个CP_j时，需执行下面的搜索后处理算法。

2.2　下层网络资源搜索后处理算法

搜索后的处理算法，考虑到A中包含多个CP_j的情况，通过计算节点距离和cache缓存技术，将多个节点按优先级排序并找出最佳目的节点。假定Internet为v维物理拓扑参照系，每一维设立一个坐标原点，P2P节点通过Ping这些坐标原点，得到该节点在每一维的物理坐标值。参照系的维数v越大，坐标原点分布越均匀，则坐标系描述的节点的物理位置越精确[3]。

定义 1 在坐标系各维特征变量量纲一致的情况下，即量纲为节点与坐标原点的网络传输延时时间，节点 x、节点 y 之间的物理距离为欧氏距离 $D_v(node_x, node_y)$。即

$$D_V(node_X, node_Y)=\left(\sum_{i=1}^{v}\left|X_i-Y_i\right|^2\right)^{1/2}$$

其中，$node_x$ 坐标为 $\langle X_i\rangle$；$node_y$ 坐标为 $\langle Y_i\rangle$。

每个节点设置一个cache缓存区，用于存储最近访问的数据库信息。如果CP_i本次访问CP_j，那么下次再访问CP_j的几率较大。cache缓存表按表 1 构造。

表 1　cache 缓存

节点权值	8	7	6	5	4	3	2	1
历史节点	P_0	P_1	P_2	P_3	P_4	P_5	P_6	P_7

其中，P_i表示最近访问节点的ID号，按先进先出原则写入cache表；数字代表权值q，最近访问的节点拥有最高的权值，其余节点的权值依次降低。

定义 2 节点的权值Q是其在cache表中出现的位置的权值[MGZ1]之和，即Q(P)=∑q_i，q_i为P所在位置的权值。

假如CP_0的ID号出现在P_0, P_2位置，CP_1出现在P_1, P_3, P_7，则

$Q(CP_0)$=8+6=14,$Q(CP_1)$=7+5+1=13,$Q(CP_0)>Q(CP_1)$

定义 3 cache表节点集合B={CP_i| CP_i为cache表中不同ID号的节点}，CP_i可以是普通节点，也可以是包含资源的索引节点。

当A有多个CP_j时，计算A∩B中节点优先级，并按照排序

结果选取目的节点发出查询请求，如果：

(1)A∩B= Φ ,则随机选取 A 中 *K* 个节点(*K* 为节点选取阈值)送到发起查询的源节点，计算源节点与目的节点的物理距离 *D*，以物理距离最小的目的节点的优先级最高，依次取优先级较高的目的节点发出查询请求。

(2)A∩B=C,C≠ Φ ，若 C 中节点个数大于 *K*，则取 C 中 *K* 个节点按权值优先级排序；若小于 *K*，则取 C 中所有节点并随机另取 A 中 D=*K*-|C|个节点，*K* 个节点按物理距离优先级排序，由于 C 中的节点是最近访问节点，因此在排序时可强行将其优先级提前。然后，按优先级的高低向目的节点发出查询请求。

最优节点选取、搜索后处理算法如下：

输入 CP_i发起查询的节点，TP_n为目标索引节点，节点集合A和B，阈值*K*。

输出 包含 Key 的节点的优先级排序队列 E。

```
if |A|>1//有多个符合条件节点，计算优先级
{if(A∩B= Φ )
  {if(|A|<K)   E=A;
   else Choose K' peers from A constitute E;
   //取 A 中 K 个节点组成 E
   Calculate Dv ( node_cpi , node_cpj );//计算距离优先级
   Sort these peers by Dv DEC; }
 if(A∩B=C，C≠ Φ )
  {if(|C|>K)
    {Choose K' peers from C constitute E;
     Calculate the Q(CPj)//计算权值优先级
     Sort these peers by Q(CPj)DEC; }
   if (|C|<K)
     {Choose (K-|C|)' peers from C constitute D, E=D∪C
      Calculate the Q(CPj), CPj∈C;
      Sort these peers by Q(CPj)DEC;
      Calculate Dv ( node_cpi , node_cpj );
      Sort these peers by Dv DEC;
      Insert D queue behind C queue;   //两队列合并为 E } }
}
```

3 仿真实验及结果分析

实验目的是测试双层网络模型的路由算法效率，与传统的 CAN 路由算法对比，评估其性能。网络拓扑产生模块采用了美国乔治大学的 GT-ITM 软件包，GT-ITM 是著名的开源代码的网络拓扑产生系统，可产生多种模型的网络拓扑结构，能够较好地模拟 Internet 网络拓扑结构。

3.1 平均路由长度对比

不同网络模型的平均路由长度对比如图 2 所示，CAN 为 2 维模型，每个主题索引节点平均维护 10 个普通节点。

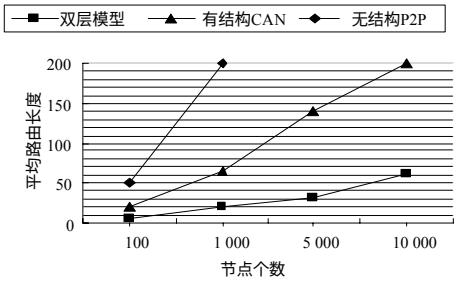

图 2 平均路由长度对比

实验结果表明，双层网络模型在路由跳数上优势显著，即使每个主题索引节点只维护 10 个普通节点，其查询效率也比单层 CAN 高很多。以 1000 个节点为例，CAN 模型的平均路由长度为 65，而双层模型的平均路由长度仅为 20，路由效率提高了 69.2%。

3.2 查准率比较

传统的 CAN 找到符合条件的多个节点后，随机地选取一个或几个节点通信。可以通过 cache 缓存技术和距离计算策略，对多个节点按优先级排序，得到最佳目的节点，让查询者和成功几率高的节点[MGZ2]进行通信，提高了查准率。2 种方式查准率的比较见图 3。

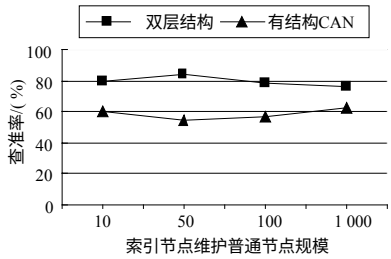

图 3 不同索引规模查准率比较

实验结果表明，双层模型的平均查找成功率为 78.25%，高于 CAN 的 60.5%约 18 个百分点。可以看出，双层模型搜索的成功率是很高的。

3.3 单点失效率比较

双层模型利用多哈希函数，将关键字映射到不同的上层索引网络上，既可避免单点失效，又能控制负载平衡。采用的哈希函数个数越多，单点失效率就越低，而负载也就更均衡。同时采用的哈希函数越多，主题索引节点上的索引表就越多，因为不同的哈希映射索引按哈希函数分区标记，逻辑上相互独立，所以并不会影响资源的搜索速度。

多哈希策略可以减少节点单点失效对查找的影响，哈希函数越多单点失效的影响越小。图 4 给出了节点失效率与查找成功率的关系。

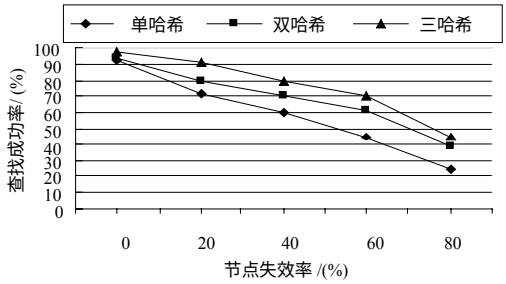

图 4 查找成功率与节点失效率的关系

4 结束语

在该双层主题索引网络模型中，上层主题索引网络基于 CAN 结构，具有较高的搜索速度和查询效率，下层为无结构网络，具有高动态性。资源发布和资源搜索都采用多哈希策略，解决了节点负载不均和单点失效问题；引入基于兴趣度的 cache 缓存和相对距离，对符合查询条件的节点进行排序，提高了查询的准确性，改善了网络性能。

参考文献

1 Hales D, Arteconi S. SLACER: A Self-organizing Protocol for Coordination in Peer-to-Peer Networks[J]. IEEE Computer Society, 2006, 6(3): 29-35.

（下转第 84 页）