第33卷 第21期 计 算 机 工 程 2007年11月
Vol.33 No.21 Computer Engineering November 2007
·网络与通信· 文章编号：1000—3428(2007)21—0102—03 文献标识码：A 中图分类号：TP393

Chord路由表结构的改进与优化

刘晓锋，吴亚娟，钟乐海
(西华师范大学计算机学院，南充 637002)

摘 要：如何高效定位所需资源是P2P网络和网格计算中的一个核心问题。P2P资源定位协议Chord的路由表结构含有一定的冗余信息，导致查询效率不高。针对该缺陷，文章对其进行改进与优化，使平均查询路径长度由$\frac{1}{2}\log N$缩短到$\frac{1}{2}\log N-\frac{1}{4}\log R(N)$，查询效率明显提高。

关键词：P2P；路由表；Chord；分布式哈希表

Optimization and Improvement of Finger Table in Chord Protocol

LIU Xiao-feng, WU Ya-juan, ZHONG Le-hai
(College of Computer, China West Normal University, Nanchong 637002)

【Abstract】 How to locate efficiently the node that stores desired resource is a core problem in P2P networks and grid computing. Chord is a successful resource location protocol in P2P systems, but its lookup efficiency is lower because finger table has redundant information. An improved and optimal finger structure is presented for removing the shortages of the original chord finger and getting higher location efficiency. The average query path length can be shortened from $\frac{1}{2}\log N$ to $\frac{1}{2}\log N-\frac{1}{4}\log R(N)$, and lookup efficiency can be heightened obviously.

【Key words】 peer-to-peer(P2P); finger table; Chord; DHT

高效定位所需资源是P2P系统和网格计算中的一个研究热点，目前资源定位算法主要有集中式和分布式2大类。在分布式资源定位系统中，Chord以其路由表具有良好的结构性、查找具有确定性、有较好的可扩展性等优点得到了广泛研究，但其结构具有一定的冗余信息，定位效率不高。本文针对Chord路由表结构的冗余信息，对其进行优化与改进，设计了一种不含冗余信息的Chord路由表结构。

1 Chord简介

Chord[1]协议算法是由MIT研发的一种分布式查询协议。基本思想是将代表资源信息的关键字和服务器节点统一映射到一个虚拟位置空间(virtual location space, VLS)中，见图1。

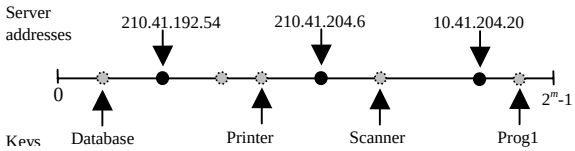


图1 虚拟位置空间

Chord根据节点的IP地址和关键字进行哈希运算得到全局唯一的标识(identifier)。为了适应分布式系统的动态性特点、负载均衡和避免哈希冲突，它采用一致哈希[2](consistent hashing)函数来对关键字和节点进行哈希运算，如SHA-1[3]。SHA-1能将任意字符串映射到2^{160}个不同的位置，这就保证了几乎不可能将任意2个数据对象或节点映射到VLS中的同一个位置。

在Chord中，每个节点只须知道其他少量节点的路由信息。如在由N个节点组成的网络中，每个节点只须维护其他$O(\log N)$个节点的信息，每次查找也只需$O(\log N)$条消息。

在VLS中，节点标识对应一个实际的网络节点，而关键字标识并不对应实际网络节点，是用户查询资源和确定对应资源在网络中的存放节点的依据。资源在网络中的存放规则是：关键字标识为K的资源信息存放在节点标识为*successor*(K)的节点上。*successor*(K)是节点标识大于或等于K的第1个节点，称为K的后继节点。

Chord资源的发现克服了查找的盲目性。资源消费者发出资源请求，然后传递该请求给它的后继节点，若该后继节点拥有相应资源，则响应消费者，否则继续向前传递该请求，直到到达被查询的关键字标识的后继节点为止。

在一个大型的P2P系统中，尽管可以采取一些措施(如双向搜索)来提高搜索效率，但以上述线性方式搜索所有节点是很低效的，因为每次搜索所涉及到的节点平均数为$N/2$。为了加快搜索速度，Chord要求网络中每个节点维护一张称为finger的路由表。在finger中有m(每个标识的位数)个表项，其第i个表项为$(Id+2^{i-1}) \bmod 2^m (1\leq i\leq m)$。图2(a)表示$m$=6时的Chord结构及节点$N_8$的finger表，finger表的第1列为$(N_8+2^{i-1}) \bmod 2^m$，第2列是存放对应该标识的资源信息的节点标识，即*successor*$((N_8+2^{i-1}) \bmod 2^m)$。

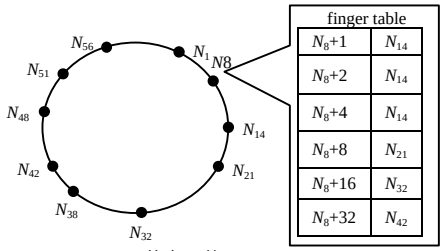
Chord执行查询操作，其目标是查找所需关键字标识的后继节点。如图2(b)，节点N_8需要查询K_{54}，而K_{54}存放在节点N_{56}上，因此，N_{56}就是N_8要查询的目标，即节点N_{56}的IP地址。其相应的查询算法[2]描述如下：

```
n.find_successor(id)
if(id∈(n, successor))
   return successor ;
```

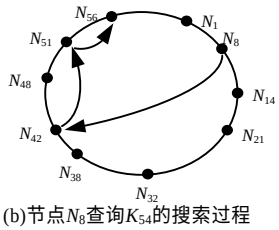
作者简介：刘晓锋(1972－)，男，讲师、硕士研究生，主研方向：基于网络的计算机应用，网格计算，并行算法理论；吴亚娟，硕士；钟乐海，教授、博士

收稿日期：2006-12-11 **E-mail：**xhxfliu@163.com

```
else n'=closest_preceding_node(id) ;
return n'.find_successor(id) ;
n.closest_preceding_node(id)
   for i=m downto 1
      if(finger[i]Î(n,id)
         return finger[i] ;
 return n;
```



(a)节点N_8的finger table



(b)节点N_8查询K_{54}的搜索过程

图 2　Chord 的 finger 表及搜索过程(m=6)

2　Chord 路由表结构分析

由图 2(a)可见，从节点N_8开始，以 2^i为跨度来寻找路由表中的后继节点，这会在finger中产生一定的冗余量，导致Chord的查询效率不高。根据finger表的构造思想，产生不同程度的冗余信息是必然的，如果能去掉这些冗余，添加相应数量的其他有效信息，会大大提高Chord的查询效率。

假设Chord环的大小为 2^m，网络节点数为$N=2^n(m>>n)$，且所有节点均匀分布在Chord环中，则任意 2 个相邻节点之间的间隔 $\Delta=\frac{2^m}{2^n}$，而路由表的大小为m。如果第 1 个节点是N_0，则第 2 个节点，第 3 个节点，……，第 2^n个节点分别是$N_0+\Delta$, $N_0+2\Delta$,…， $N_0+(2^n-1)\Delta$，其中，节点Id的路由表可表示为$(Id+2^{i-1})\bmod 2^m$, $(1\le i\le m)$。经研究发现，只有当 $2^{i-1}\le\Delta(i\le m-n+1)$时，在路由表中才有冗余，当$i>m-n+1$ 时，路由表中存在冗余的概率小到可以忽略不计。因此，有$successor((Id+2^{1-1})\bmod 2^m)=successor((Id+2^{2-1})\bmod 2^m)=\cdots=successor((Id+2^{(m-n+1)-1})\bmod 2^m)=Id+\Delta$，很显然，这些都属于节点$Id$的路由表上的冗余信息，由此可得冗余度$red=\frac{m-\lceil n\rceil}{m}\times 100\%$，相应的冗余量$R(N)=m\times red$。如在大小为 2^m (m=20)的Chord环中，共有 10 000 个网络节点，约 $2^{13.29}$，即n=14，则$red=\frac{20-14}{20}\times 100\%=30\%$。在图 2 中，$m$=6，节点数为 10，约 $2^{3.32}$，即n=4，则$red=\frac{6-4}{6}\times 100\%=33.3\%$，其冗余量$R(N)$=2。冗余度$red$与$m$和$n$的关系如图 3 所示。

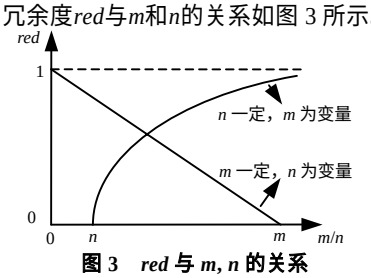
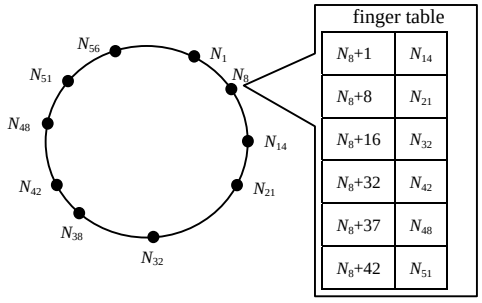


图 3　red 与 m, n 的关系

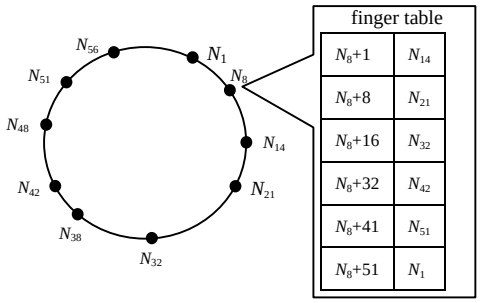
3　Chord 路由表的改进优化及性能分析

3.1　Chord 路由表的改进与优化

根据原finger表的构造方法，节点Id的finger表的最后一项入口是$(Id+2^{m-1})\bmod 2^m$，这隐含了节点Id的finger表只能覆盖整个Chord环的一半，只要目的节点落入这一半之外，节点Id就必须通过其他至少一个中间节点才能找到。另外，finger表的冗余信息几乎都出现在finger表前面，因此，除去finger表前面的冗余信息，添加其他节点的路由信息到finger表的尾部来消去finger的冗余。添加方法可以有以下 2 种方案：(1)在finger表未能覆盖到的区域$(Id+2^{m-1}, Id+2^m-1)$中顺序选择$R(N)$多个节点加入到finger表的尾部(见图 4(a))；(2)在区域$(Id+2^{m-1}, Id+2^m-1)$中均匀选取$R(N)$多节点添加入finger表的尾部，使finger表覆盖整个Chord环(见图 4(b))。本文针对第 2 种方案进行改进。



(a)节点N_8的finger表改进方案 1



(b)节点N_8的finger表改进方案 2

图 4　finger 表的改进示意图

对finger改进后，原finger表能覆盖的这一半区域$[Id+2^0, Id+2^{m-1}]$，finger表不变，但在finger末尾添加的$R(N)$个节点信息的入口表达式不再是 $(?+2^{i-1})\bmod 2^m$(？表任意节点标识)。可按如下方法来确定这样一个入口表达式：

根据假设，N个节点均匀地分布在Chord环上，即有$N/2$个节点分布在区域$(Id+2^{m-1}, Id+2^m-1)$内，现从该区域内均匀(等距离)地选取$R(N)$个节点添加到finger表的尾部，使finger能覆盖整个Chord环。假设通过这种策略选定了某个节点id，即节点id要成为某些资源信息的后继节点，则在finger表中相应入口为$(Id+d)\bmod 2^m$，其中，$d\in(predecessor(id)-Id+k.2^m, id-Id+k.2^m)$，$k$=0 或 1。这就保证$successor((Id+d)\bmod 2^m)=id$成立，$predecessor(id)$表示节点$id$的前驱节点。根据所选节点标识$id$与$Id$的大小关系来定$k$的取值，即 $\begin{cases} id>Id, & k=0 \\ id<Id, & k=1 \end{cases}$。如图 5(b)，选择节点$id=N_{51}$和$N_1$添加在节点$Id=N_8$的finger表尾部。$predecessor(N_{51})=N_{48}$，$k$=0，有$d\in$(48-8, 51-8)，即$d\in$(40, 43)，因此，取$d$=41。

3.2 性能分析

路径长度是衡量路由协议性能的一个重要指标。在Chord中，路径长度是指在执行查询操作中所访问的节点数(或跳数，hops)，而且已知原Chord的平均查询路径长度为$O(\log N)$ [1, 4]。为了确定改进后的平均查询路径长度，先讨论一个关于平均路径长度的性质。

性质 在有$N(=2^n)$个节点的Chord网络中，假设其平均查询路径长度为$L_{avg}(N)$，则对该Chord网络的任意$\alpha(0<\alpha\leq 1)$部分进行查找的平均查询路径长度为$L_{avg}(\alpha N)$。

该性质的正确性是明显的。一个有N个节点的Chord网络的α部分，就相当于一个由$M=\alpha N$个节点组成的一个新的Chord网络，其平均查询路径长度为$L_{avg}(M)= L_{avg}(\alpha N)$。

根据前面的讨论，每个节点的finger表中有$R(N)=m\times red$的冗余量，改进的finger表中是去掉了这些冗余量，从$(2^{m-1}, 2^m-1)$中均匀选择$R(N)$项添进finger表，则只要目的节点落入区域$(2^{m-1}, 2^m-1)$内的平均查询路径长度就不能按$\frac{1}{2}\log N$ [1, 4]计算，但改进后的Chord的平均查询路径长度可分成2种情况：(1)目的节点落入区域$[0, 2^{m-1}]$，按$\frac{1}{2}\log N$计算；(2)目的节点落入区域$(2^{m-1}, 2^m-1)$，不能按$\frac{1}{2}\log N$计算，但节点落入区域$[0, 2^{m-1}]$和$(2^{m-1}, 2^m-1)$是等概事件，其概率均为1/2，因此，改进后的Chord的平均查询路径长度为$L'_{avg}(N)=\frac{1}{2}(L_{avg1}+L_{avg2})$，其中$L_{avg1}=\frac{1}{2}\log\frac{N}{2}$(由性质1)，$L_{avg2}$对应第2种情况。

为了计算L_{avg2}，把改进finger表剖析成如图5所示。在区域$[0, 2^{m-1}]$内按原Chord的finger表结构，每个表项按2的方幂增加，在区域$(2^{m-1}, 2^m-1)$内按改进Chord的finger表结构，每个表项是等距的。

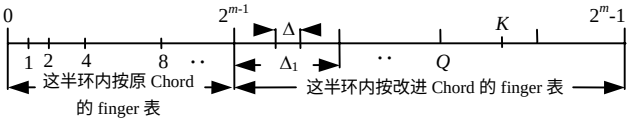


图5 改进Chord的finger表结构

其中，$\Delta=\frac{2^m}{2^n}$，表示$N=2^n$个节点均匀分布在位置空间内的间距；$\Delta_1=\frac{2^{m-1}}{R(N)}$，表示在$(2^{m-1}, 2^m-1)$内均匀选取$R(N)$个节点的间距，$R(N)$表示冗余量。这二者之间存在如下关系：$\Delta_1=\frac{2^{n-1}}{R(N)}\Delta$，这关系体现了在$\Delta_1$内有$\frac{2^{n-1}}{R(N)}$个节点。

现假设待查关键字$key=K$，产生查询请求的是节点n。按图3的查询算法，只要$K\notin(n, successor(n))$，就需要将该查询请求路由到离K最近的前驱节点Q上(如图6所示)，由Q继续执行该查询任务。显然，K一定会落入由Q开始的Δ_1内，由于$\Delta_1<<2^{m-1}$，因此在Q节点的finger表中是按原Chord的finger结构来继续查询该关键字。根据性质有，由Q执行该查询的效率为$O(\frac{1}{2}\log\frac{2^{n-1}}{R(N)})$。另外，由节点$n$到$Q$，在改进Chord中只需一步，而在原Chord中至少需要一步，因此有$L_{avg2}=1+\frac{1}{2}\log\frac{2^{n-1}}{R(N)}$，改进Chord的平均查询长度为

$$L'_{avg}(N)=\frac{1}{2}(\frac{1}{2}\log\frac{N}{2}+1+\frac{1}{2}\log\frac{2^{n-1}}{R(N)})=\frac{1}{2}\log N-\frac{1}{4}\log R(N)$$

其中，N表示网络中节点总数；$R(N)$表示冗余量。如$m=20$, $N=2^n$, $n=12$，则改进的平均查询路径长度$L'_{avg}(N)=5.25$，而原平均查询路径长度$L_{avg}(N)=6$。查询效率明显改善，提高了12.5%。

4 仿真实验及结果分析

Chord协议可采用迭代和递归[1,4]2种方法来实现。迭代方式，就是查询节点为一个查询初始化所有通信，即查询节点从自己的路由表中访问一系列节点的信息，使得在Chord环中每移动一次就更接近目标节点；递归方式，每一个中间节点推进查询请求到下一个节点，直到到达目标节点。本仿真实验采用迭代方式来实现。

本仿真实验主要验证原Chord和改进Chord在查询路径长度上的差异，没有考虑每一跳的访问延迟。所采用的试验环境为，每个标识用6位二进制数表示，节点数分别为8, 9, 10, 11, 12的小型网络，每个节点上存放10个文件左右，对每一组值实验10次。其结果如图6所示，可以看出，平均查询路径长度与节点数有近似的对数关系，改进Chord较原Chord在查询效率上有明显改善，而且随着节点数的增加，这2条曲线越接近，这与理论分析的结果较吻合。

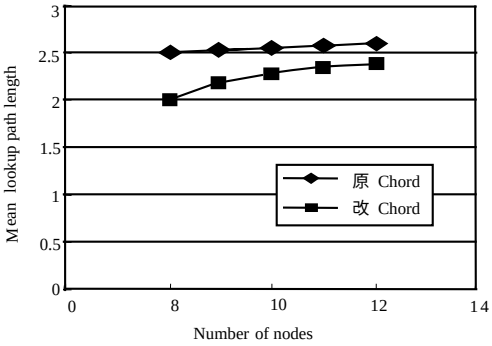


图6 仿真实验结果对比图(m=6)

在试验中还发现，当m与n的差值越大，改进Chord表现越好，当m与n的差值越小，改进Chord越接近原Chord，这与理论结果也是一致的。因为m越大，节点数越小，导致$R(N)$增大，$L'_{avg}(N)$就越小，查询效率越高；相反$L'_{avg}(N)$越大，查询效率越低，最后趋近于$L_{avg}(N)$。

参考文献

1 Stoica I, Morris R, Karger D, et al. Chord: A Scalable Peer-to-peer Lookup Service for Internet Applications[C]//Proceedings of the ACM SIGCOMM'01, San Diego, CA. 2001-08.

2 Karger D, Lehman E, Leighton T, et al. Consistent Hashing and Random Trees: Distributed Caching Protocols for Relieving Hot Spots on the World Wide Web[C]//Proceedings of the 29th Annual ACM Symposium on Theory of Computing. 1997-05.

3 FIPS 180-2-02 Secure Hash Standard, Federal Information Processing Standards Publication[S]. Department of Commerce, NIST, 2002-08.

4 Stoica I, Morris R, Liben-nowell D, et al. Chord: A Scalable Peer-to-peer Lookup Protocol for Internet Applications[J]. IEEE/ACM Transactions on Networking, 2003, 11(1).