兴趣局部性在DHT资源定位技术中的应用

王 芳[1],李 梁[2],郑明春[1]
(1.山东师范大学 计算机系,山东 济南 250014;
2.山东省聊城市财政局信息中心,山东 聊城 252000)

摘 要:P2P系统的一个核心问题是如何高效定位所需资源。文中提出了一种基于兴趣局部性(locality)的分布式哈希表(DHT)资源定位技术,将非结构化对等网络引入到Chord中,充分利用节点在物理网络上的邻近性和节点间兴趣的局部性。模拟测试表明,该技术在路径长度和访问延迟方面的性能优于原Chord。

关键词:Chord;资源定位;分布式哈希表;兴趣局部性
中图分类号:TP393 **文献标识码**:A **文章编号**:1005-3751(2006)04-0038-03

A DHT Resource Location Technique Using Interest-Based Locality

WANG Fang[1], LI Liang[2], ZHENG Ming-chun[1]
(1.Dept. of Computer, Shandong Normal University, Jinan 250014, China;
2. Info. Center, Liaocheng Financial Administration, Shandong Province, Liaocheng 252000, China)

Abstract: A core problem of peer-to-peer system is the efficient location of the node that stores desired resource. This paper presents a DHT resource location technique using interest-based locality. This method introduces unstructured P2P to Chord, making the best of the locality both in the underlying physical network and the interest among peers. Simulation tests show that this technique is superior to original Chord at path length and access latency.

Key words: Chord; resource location; distributed Hash table; interest-based locality

1 问题的提出

目前P2P系统的资源定位技术基本上可以分为三类。第一类基于中央服务器(或服务器群),如Napster系统、中央服务器维护系统中所有共享资源的位置索引。该技术的优点是高效、易于实现,但存在单一故障点、可扩展性差等问题。另外两种技术将共享资源的索引分散在系统中的结点上,根据资源定位机制是否依赖特定的系统拓扑结构,分为非结构化和结构化定位技术。非结构化定位技术,如Gnutella,以洪泛(flooding)方式发送定位请求,其优点是没有单一故障点,但存在带宽消耗大、可扩展性差等问题。结构化定位技术基于分布式哈希表(DHT),每个节点都维护一个路由表,定位资源时通过路由表进行选择性转发,在一定的跳数内定位到资源。近年来,各国研究人员提出了各种结构化分布式查找系统,如Chord[1]。这类技术解决了非结构化定位机制的可扩展性问题,但由于没有考虑节点物理位置上的关系,只是优化节点间在逻辑网络上的跳数,使查找跳数较大,访问延迟较长[2]。研究表明[3,4],当前的P2P系统都具有地域上的邻近性和兴趣上的局部性。对于节点A,如果它在地理位置上靠近节点B且以后可能会为A提供服务,那么A和B存在地域上的邻近性,即地域上的局部性。如果节点C在过去一段时间内响应了节点A的请求,节点C可能以后还会满足A的其它请求,那么A和C具有相似的兴趣,即兴趣上的局部性。由于P2P系统中的各点是平等的,节点可能会忽视一些"临近"的、通信效率较高的点,而与那些"遥远"的、相互之间通信效率不高的点进行数据交换。当前的资源定位机制由于没有全面利用P2P系统中内在的两种局部性,无法同时实现搜索高效和可扩展性。

文中面向Internet规模的P2P应用,对Chord进行改进,提出一种基于Chord的查找服务系统,在结构化P2P的基础上建立一个基于兴趣的非结构化覆盖网络,充分利用底层物理网络节点的邻近特性和节点间兴趣的局部特性。

2 技术介绍

2.1 Chord简介

Chord使用相容哈希为系统中的每个节点和关键字产生惟一的标识符Id,如IP地址为120.10.11.1的节点的Id为51,关键字"Letitbe"的Id为54。关键字保存在它

收稿日期:2005-10-18
基金项目:山东省优秀中青年科学家科研奖励基金(304068)
作者简介:王 芳(1975-),女,山东枣庄人,硕士研究生,主要研究方向为对等网络;郑明春,教授,研究生导师,主要研究领域为对等网络和无线网络。

的后继节点(successor)中,后继节点是标识符大于或等于关键字 k 的第一个节点,记为 successor(k)。如图 1 中关键字"Letitbe"存放在 successor(54),即节点 N56。如果 m 是关键字和结点标识符的位数(采用二进制表示),每个节点只需维护一张最多由 m 个表项组成的路由表,称之为指针表(finger table)。节点 n 的指针表中第 i 个表项包括的是 $s = \text{success}(n + 2^{i-1}) \bmod 2^m (1 \leqslant i \leqslant m)$,$s$ 称为节点 n 的第 i 个指针。图 1 中节点 N8 的指针表内容如表 1 (a)。进行资源定位时,节点根据请求的关键字 Id 和指针表,将查询请求转发到节点距离请求关键字最近的节点上,实现高效查找。

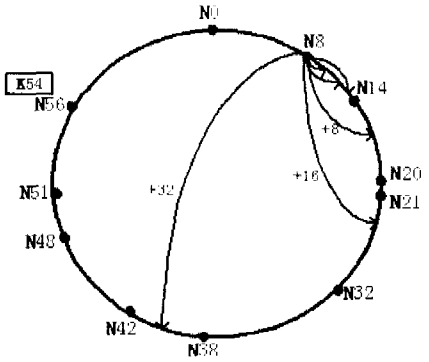

图 1　Chord 示意图

2.2　基于兴趣的资源定位技术

为了利用地域上的局部性,反映节点之间在物理网络上邻近特性,采用邻近邻居选取(proximity neighbor selection,PNS)的方法改进 Chord 的指针表:节点 n 的第 i 个指针是从标识符介于 $n + 2^{i-1}$ 和 $n + 2^i$ 的节点中选择前 x 个节点(可能之间的节点数少于 x)中,选择距离 n 最近的节点作为节点 n 的第 i 个指针。例如图 1 中,节点 N32 和 N38 同处于节点 N8 的指针表第 5 个指针的标识符跨度,如果节点 N38 比节点 N32 距离节点 N8 更近,节点 N8 的第 5 个指针为节点 N38。使用 PNS 方法改进后,节点 N8 的指针表如表 1(b)所示。

表 1　节点 8 的指针表

start	interval	node
9	[9,10)	14
10	[10,12)	14
12	[12,16)	14
16	[16,24)	20
24	[24,40)	32
40	[40,8)	42
successor	14	
predecessor	0	

(a)

start	interval	node
9	[9,10)	14
10	[10,12)	14
12	[12,16)	14
16	[16,24)	20
24	[24,40)	38
40	[40,8)	51
successor	14	
predecessor	0	

(b)

文中提出的资源定位技术依赖的原则是基于兴趣的局部性[5],即从与自己具有共同兴趣的节点处更有可能找到所需资源。这里称具有共同兴趣的节点为朋友节点,节点 n 的所有朋友节点构成了节点 n 的兴趣社区。为了构建兴趣社区,节点首先要确定自己的朋友节点。文中基于节点 X 和节点 Y 当前共享内容的相似程度,判断两个节点是否是朋友。

节点的兴趣社区可以看作是一张图,顶点是节点,当两个节点间有共同兴趣即共享了相同的文档时,在图中相应的两个顶点间建立一条弧。每条弧可以根据不同的测度设置权重,文中使用两个节点共享相同文件的个数作为弧的权重。创建社区的节点称种子节点,所有节点的兴趣社区构成了一个非结构化的对等网络,创建算法描述如下。图 2 是节点 N8 和节点 N42 的兴趣社区,虚线表示深度为 2 的朋友节点。

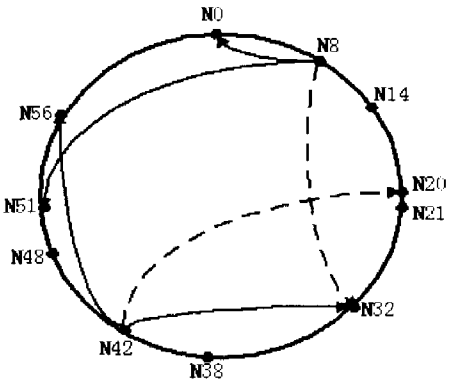

图 2　节点 8 和节点 42 的朋友节点示意图

Step1　种子节点开始创建兴趣社区,种子节点的深度为 0。

Step2　对于深度 $i = \{0,1\}$ 的每一个节点 p:

从 p 的本地文档中随机选取 k 个文档,这些文档的关键字构成集合 r;从 p 所知道的节点中随机选取 n 个节点,向这些节点请求 r。

对于至少满足一个文档的每个节点 q:

将 q 加入到图中,其深度为 $i + 1$;在 p 和 q 间添加一条弧,其权为 p 和 q 共享相同文件的个数。

Step3　在图中加入一个收点,并在该点和图中深度为 2 的节点间添加弧,每条弧的权都为 1。

Step4　计算图中种子节点到收点的最大流。

Step5　在种子节点的朋友列表中加入由 Step4 得到的节点。

在算法第二步的迭代中,加入到图中的节点可能与种子节点没有任何相同的文档,但与深度为 1 的节点有相同的文档。尽管种子节点和深度为 2 的节点当前没有共同的文档,但由于它们都与深度为 1 的节点有联系,所以很可能不久后它们会至少有一个相同文档。因此,加入这些节点有助于改善社区的效率。每个兴趣社区中的节点数都设置上限,种子节点对社区进行更新时,若社区中的节点数已到达上限,社区中共享文档数最少的节点被取代。种子节点根据兴趣社区图建立并维护一个朋友节点列表,保存朋友节点的标识、共享相同文档的个数及深度。当节点的共享内容变化时,根据朋友节点列表通知朋友,并重

新衡量与朋友节点的兴趣相似程度。

确定朋友节点后，种子节点保存朋友节点上共享文档的关键字，构造自己的兴趣列表，其主要内容是关键字和相应文档的存储位置。由于结构化 P2P 系统中每个节点为了共同利益存储其他节点共享文档的关键字，因此节点共享文档的关键字和节点存储的关键字在很大概率下是不同的。种子节点保存的是朋友节点上共享文档的关键字，而不是朋友节点的 Id 所负责的关键字，增加了种子节点能够定位的资源数量。由于对网络中的每对节点进行兴趣比较代价太大且不现实，故限制了每个节点比较的文档数是节点共享资源更新数目的函数。也就是说，只有当本地共享资源变化了一定比例时才更新朋友列表。

进行资源定位时，节点首先从兴趣列表中查找关键字，如果兴趣列表的关键字可以匹配请求，只需再多一跳就可以到达目的节点；如果兴趣列表无法满足搜索请求，节点使用底层 Chord 的定位算法继续执行搜索定位。

3 仿真与性能分析

基于 MIT 的 Chord 模拟器实现了基于兴趣的资源定位技术的性能模拟，并与原有 Chord 进行了比较，test1 对应于原 Chord 算法，test2 对应于基于兴趣局部性的 Chord 算法。实验参数设置：节点个数、文档个数、请求个数的比例为 1∶1∶10，节点个数分别为 100，500，1000，2000，4000 和 5000。每个节点的兴趣社区中的节点个数为 $1/2\log N$，即指针个数的一半。由文献[6] 的实验结果可知，节点 n 的第 i 个指针是标识符介于 $n+2^{i-1}$ 和 $n+2^i$ 的前 16 个节点中距离 n 最近的节点，可使 Chord 达到近似最佳的延迟。

首先，测试了兴趣列表的命中率。命中率是指通过由兴趣列表直接满足的请求与总请求的比值。如果命中率高，则兴趣列表有助于资源定位。图 3 给出了不同网络规模下兴趣列表的平均命中率为 36.6%～56.3%。

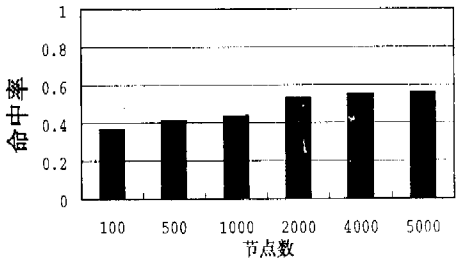

图 3 兴趣表的命中率

实验分别从下面两个测度，对文中提出的技术与原 Chord 进行比较。

* 路径长度：是指请求在到达目标节点时经历的覆盖网络跳数。节点 p 发出请求，如果请求不能在本地得到匹配，根据 Chord 进行查找，其复杂度为 $O(\log N)$。如果请求目标可以在本地的兴趣列表中得到匹配，则只需再多一跳到达目的节点就可以满足请求，这种情况下满足请求的复杂度是 $O(1)$，极大地缩短了查找路径长度。测试结果如图 4 所示，基于兴趣的 Chord 的平均路径长度比原 Chord 缩短了 25%～43%。

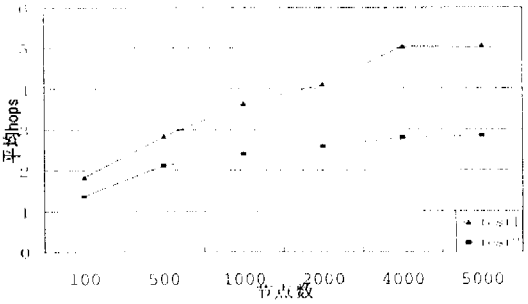

图 4 查找路径长度比较

* 访问延迟：用户感觉到的延迟减小，主要是由于两个原因：尽可能选择物理网络上最近的节点作为邻居；查找路径长度的缩短也使延迟有所减小。两种测试访问延迟的结果如图 5 所示，基于兴趣的 Chord 在访问延迟方面是原 Chord 的 38%～50.6%。

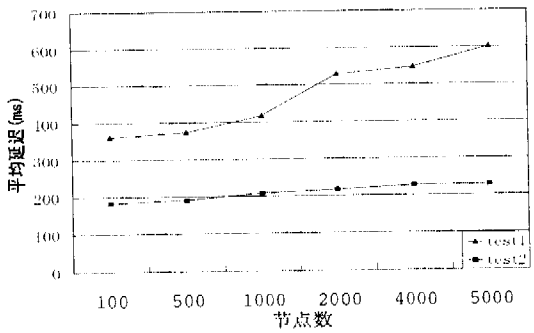

图 5 访问延迟比较

当前客户机的存储能力日益增强，存储朋友节点列表、兴趣列表不会对性能产生影响。每个节点的朋友节点数随网络规模按指数级增长，因此这种定位技术具有可扩展性，适用于 Internet 规模的 P2P 应用。同时，这种技术没有改变 Chord 的关键字分布方法，因此继承了 Chord 在负载平衡方面的优点。

4 结束语

提出一种基于兴趣局部性的资源定位技术，将非结构化对等网络引入到 Chord 中，继承了 Chord 原有的优点，同时又缩小了查找路径长度和访问延迟，改善了服务质量。下一步将研究 P2P 系统中其他因素，如负载平衡对性能的影响，以及开发适合基于兴趣局部性的特定 P2P 系统应用。

参考文献：

[1] Stoica I, Morris R, Karger D, et al. Chord: A Scalable Peer - to - Peer Lookup Service for Internet Applications[A]. Proceedings ACM Sigcomm 2001[C]. San Diego, CA, USA: [s.n.], 2001.149 - 160.

[2] Yang B, Garcia - nolina H. Efficient Search in Peer - to - Peer Networks[R]. CA, USA: Stanford University, 2001.

(下转第 43 页)

不同的是每次发送和接收数据的数量。

4 实验结果

4.1 实验条件和任务概述

根据第 3 节描述的程序,在建立的 MPI 并行环境上分别运行不同规模的按行划分以及按列划分的矩阵向量乘法。由于并行计算适合于数据量较大的计算任务,所以矩阵规模分别采取 50 × 50,100 × 100,200 × 200,300 × 300。实验分别使用 1 个结点,2 个结点,3 个结点,4 个结点(每个结点都是单 CPU,型号都为 PIV 1.5GHz,512kL2,1ML3,操作系统都为 Windows XP)来完成计算。

在实验结果分析时需要注意影响并行计算性能的因素:

1)网络性能。在测试过程中,并行计算的时间是不稳定的,有时时间会变得很大。原因是本次实验的网络环境是利用实验室的局域网。这种抢占式的带宽分配造成并行计算网络环境的不稳定性,同时也影响了网络并行计算的性能;

2)矩阵和向量采用随机值初始化,不同特性的矩阵和向量相乘对运行时间有一定影响;

3)MPI 的任务调度。类似于把 8 个进程分配给 2 个结点,不等于每个结点有 4 个进程,因为在 MPI 的程序中只指定了进程的数目,而把不同进程进行组合和映射的工作是在 MPI 的底层部分实现,对用户透明[5]。

基于以上 3 点因素,对每一种情况下并行计算时间的确定,采用多次运行取平均值的方法。

4.2 实验结果分析

按行划分的并行计算时间(以秒为单位)如表 1 所示(括号内的她字表示进程数)。

表 1　按行划分的并行计算时间

矩阵规模	1 个结点(4)	2 个结点(8)	3 个结点(12)	4 个结点(16)
50×50	0.0269687	0.0448418	0.179386	0.1001190
100×100	0.0304387	0.0564664	0.300662	0.0870834
200×200	0.0461752	0.0672561	0.205023	0.0716447
300×300	0.0748596	0.0900141	0.258670	0.0985663

按列划分的并行计算时间(以秒为单位)如表 2 所示(括号内的数字表示进程数)。

表 2　按列划分的并行计算时间

矩阵规模	1 个结点(4)	2 个结点(8)	3 个结点(12)	4 个结点(16)
50×50	0.0233023	0.0262456	0.0319666	0.0560001
100×100	0.0287801	0.0386619	0.0459031	0.0636264
200×200	0.0332701	0.0517920	0.0577157	0.0631534
300×300	0.0476856	0.0738858	0.0656160	0.0744067

对比按行划分和按列划分的计算结果,可以得出以下几点结论:

1)在同等条件下,按行划分的时间比按列划分的时间大。由于测试案例的行和列都相等,导致在按行划分和按列划分情况下进程之间的通信时间相差很少以及所有结点的总计算量是一样的,所以运行时间的主要差别是按行划分需要花费广播向量至其他的进程和其他的结点的时间,而按列划分不需要。

2)3 个结点下,按行划分的时间比按列划分的时间大一个数量级。因为在这种情况下,向量广播是按行划分计算的瓶颈,比多个任务并行计算的代价大的多。在这种条件下,最佳方案是按列划分。

5 结　论

文中笔者对按行划分以及按列划分的矩阵向量乘法在原理以及程序的执行时间方面的异同进行了分析。由于按列划分的算法在文献上涉及甚少,并且按行划分和按列划分在原理和执行时间上存在差异,所以在算法原理和程序上比较它们的异同,以便在不同条件下采用最佳的算法,具有一定的意义。

参考文献:

[1] 陈国良.并行计算 - 结构、算法、编程(修订版)[M].北京:高等教育出版社,2003.

[2] 陈国良.并行算法的设计与分析(修订版)[M].北京:高等教育出版社,2002.

[3] 张丽君,金绥更.向量算法与并行算法[M].北京:国防工业出版社,1993.

[4] 李晓梅,莫则尧,文尚猛.面向结构的并行算法 - 设计与分析[M].北京:国防科技大学出版社,1996.

[5] 都志辉.高性能计算之并行编程计算 - MPI 并行程序设计[M].北京:清华大学出版社,1999.

(上接第 40 页)

[3] Gummadi K P, Dunn R J, Saroiu S, et al. Measurement, Modeling, and Analysis of a Peer - to - Peer File - sharing Workload[A]. Proceedings of the 19th ACM Symposium on Operating Systems Principles (SOSP - 19)[C]. Bolton Landing, NY, USA:[s.n.],2003. 314 - 329.

[4] Fessant F L, Handurukande S, Kermarrec A M, et al. Clustering in Peer - to - Peer File Sharing Workloads[A]. Proceedings of the 3rd International Workshop on Peer - to - Peer Systems (IPTPS)[C]. San Diego, USA:[s.n.], 2004.

[5] Sripanidkulchai K, Maggs B, Zhang H. Efficient Content Location using Interest - based Locality in Peer - to - Peer systems[A]. Proceedings of 22nd Annual Joint Conference of the IEEE Computer and Communications Societies[C]. San Francisco, CA, USA:[s.n.],2003.

[6] Dabek F, Li J Y, Sit E J, et al. Designing a DHT for Low Latency and High Throughput[A]. Proceedings of the 1st Symposium on Networked Systems Design and Implementation (NSDI '04)[C]. Berkeley, CA, USA:[s.n.],2003. 85 - 98.