文章编号:1001 - 9081(2006)12 - 2803 - 03

Chord 改进算法在文件共享系统的实现

高集荣,苏振林
(中山大学 计算机科学系,广东 广州 510275)
(gaojr@mail.sysu.edu.cn)

摘 要:Chord 是一种比较有效的 P2P 路由算法,它能够快速地查找到该资源的位置。但 Chord 算法对网络中传送的消息利用率过低,路由表存在严重的信息冗余,网络维护量大。为此,提出了一种 Chord 的改进算法,解决了 Chord 算法中存在的一些问题,提高了网络查询效率,增强了网络的容错能力。将改进的 Chord 算法用于一个文件共享系统的设计,实践表明该 Chord 改进算法是有效的。

关键词:Chord;文件共享;路由表;DHT;P2P
中图分类号: TP393.07 **文献标识码**:A

Implementation of an improved Chord algorithm in the file sharing system

GAO Ji-rong, SU Zheng-lin
(*Department of Computer Science, SUN Yat-Sen University, Guangzhou Guangdong* 510275, *China*)

Abstract: Chord is a kind of comparatively effective P2P route algorithm, and it can locate the target resources quickly. But because of the over low utilization of the message during transmission, the routing table suffers serious information redundancy and great network maintenance is needed. Therefore, this text put forward an improved Chord arithmetic, and resolved some existing problems. As a result, the network search efficiency was raised and fault-tolerance ability of the network was strengthened. The improved one was implemented in a design of the file sharing system and proved to be effective.

Key words: Chord; file sharing; router table; Distributed Hash Table (DHT); P2P

0 引言

在庞大的 P2P 网络中快速找到用户所需资源的位置是 P2P 网络要解决的技术难题。DHT(Distributed Hash Table)算法能够很好地解决 P2P 网络资源定位的问题。该算法把网络中的路由信息分布到网络的节点中。

Chord 算法[1,2]是一种 DHT 网络算法。该算法能够在 N 个节点的网络中,在 $O(\log N)$ 的时间复杂度内找到网络上的资源。当网络的节点加入或者退出网络的时候,该算法能够保证在一定的时间内通过节点的运算,重新稳定网络。

在 Chord 网络中,只要给出一个关键字,就能够很快地找到保存该关键字的资源的索引。但是传统的 Chord 算法存在以下不足:

1) Chord 链的维护的困难,维护 Chord 所花的信息过多,导致在带宽较小的环境下表现不好。

2) Chord 必须要构成一个大的环状结构,原来的算法没有考虑该环在大的网络中可能的构成困难。

3) 容错性不够好。当 Chord 环出现错误的时候,不能保持查找的正确性。

4) 维护路由表要耗费大量的工作,没有利用网络的有效的信息对路由表进行处理。

5) 单一的查找过程,无法进行并行查找,查找的速度较慢。

6) Chord 网络的安全问题。

1 Chord 算法改进的主要策略

1) 充分利用网络传输的信息

主要的目的就是要充分利用网络中所传送的信息,使网络中的节点能够从这些信息中提取重要的内容来更新节点保存的路由表。

在 Chord 网络中的消息常常要通过转发的方式传送到下一个节点中去。Chord 算法中没有利用这些信息作为节点的路由更新信息。

改进的算法中会把发送信息的节点的本节点、前驱节点和后继节点的信息放到消息的头部。当有节点收到这个消息后会更新路由表。

2) 改变路由表的结构,使之适应新的消息机制

原算法每次对路由表进行更新时,都必须要对这些确定的 id 在网络上调用 find_successor 的操作。这样的操作无疑对网络的带宽带来影响,而且要维护这个路由表的代价也是很大的。

改进的算法中采用新的路由表结构,使得可以从收到的消息去更新路由表的信息,节省了维护路由表的耗费。改进算法的路由表结构如表 1 所示。

在改进的算法中,不单独保留 successor 和 predecessor,而是在路由表中保存的节点中查找 successor 节点和 predecessor 节点。在改进的算法中也不存在 successor list 列表。如果一个 successor 节点在网络中丢失,那么节点就会在 finger table

收稿日期:2006 - 06 - 26;**修订日期**:2006 - 08 - 07

作者简介:高集荣(1960 -),男,陕西西安人,副教授,硕士,主要研究方向:网络技术及其应用、数据库技术; 苏振林(1984 -),男,广东湛江人,硕士,主要研究方向:网络及网络软件.

中寻找一个节点来作为下一个 successor 节点。这样,在改进的算法中就减少了原来算法所保留的很多冗余数据,使每个节点所保留的数据量减少。

表 1 改进算法的路由表结构

标记	定义
finger[k]. start	$(n+2k-1) \bmod 2m, 1 \leqslant k \leqslant m$
finger[k]. interval	[finger[k]. start, finger[k+1]. start)
finger[k]. nodeList	在 finger[k]. interval 中的节点的列表,该列表的最大长度由长度常数 Bucket 决定。在每个 finger 项中,可能存在一些项的节点列表是不满的,有一些的 finger 项是一个节点也没有的。

3) 对网络中破坏节点的检测,并将这些节点排除到这个网络之外

网络中会存在一些节点,由于它们频繁加入和退出网络,导致整个网络的不稳定,这样的节点也会影响路由表的正确性.

在网络中存在比较久的节点在下一个时间段内仍然存在的可能性比较大,这是在真实环境的网络下的一个重要的特点.根据这个特点,在改进的算法中,系统会对每个点进行计数.每当有来自某个节点的信息时,如果该节点在 finger table 中,那么就会将该节点的计数加一,然后每个 finger table 项中的 nodeList 就会根据这个计数排序.有新节点的信息加入时,先检测该链中最小的节点是否还存在网络中,如果不存在,那么就丢弃原来的节点,并加入这个新的节点;否则的话,就丢弃新的节点的信息.这样路由表中就会保留网络中的稳定的节点,而那些不稳定的节点就无法加入到网络中.

2 优化后的算法与原算法的效率比较

1) 路由表更新效率的比较

原算法更新网络中的路由表时,更新一项需要转发的消息数为 $\log(N)$,每条消息更新的路由表项的数量为 $\mu/\log(N)$(其中 $0 \leqslant \mu < 1$).

为了在网络中尽可能地传输路由信息,可以在消息的头部加入若干个最近更新的节点的信息,这时每条消息包含的节点信息大于 3.因此,在路由表更新效率的比较上,改进算法的更新效率与原算法更新效率的比大于等于 $3/[\mu/\log(N)] = 3\log(N)/\mu$.

2) 网络查找效率的比较

改进算法在查找的改进是采用了并行查找的方法,同时向资源 id 的 successor 节点所在的区间中的若干的节点同时发出请求.而原算法是串行查找的方法.

3) 节点加入网络的效率比较

原算法加入的过程比较简单,但是加入网络以后不能马上利用路由表进行查询,必须等到一定的时间(160 个周期运行 fix_finger 函数,才能保证路由表的正确性).在路由表还没有正确的时候,查询不能达到 $O(\log(N))$ 的复杂度.在改进的算法中,当节点加入网络以后,节点上的路由表就能够保证基本正确,查询的性能在节点加入网络后就能够保证.

4) 网络传输数据量的比较

在一个周期内,一个节点要产生的消息为 $6 + 2\log(N)$ 个,网络中要传送的消息总数为 $6N + 2N\log(N)$ 个.

改进算法的网络传输数据量为:周期性运行 stabilize 函数,该函数发送 NOTIFY_MESSAGE 消息,等待 ACK 消息,网络传输为 2 个消息,这样网络中传送的消息总共为 $2N$.

3 改进算法在文件共享系统的实现

3.1 文件共享系统功能

文件共享系统是一种主机之间相互提供文件访问的网络系统,其中的节点可以相互提供文件的下载,浏览等功能.文件共享系统要解决的是文件的定位问题,即如何在网络中正确地找到某个文件的位置.传统的解决方法是在网络中加入一些服务器,当文件共享系统中的用户加入到网络的时候,该用户把共享的文件的信息放到服务器的数据库中.其他的用户要查找文件的时候,只要向服务器发出查询的请求就可以解决问题.这种实现的方法比较简单,但是对网络中的服务器有很大的依赖.当用户的数量增大的时候,服务器的负担很大,会出现系统的瓶颈.

使用改进 Chord 算法的 DHT 系统可以很好地解决这一问题,加入网络的用户不需要网络服务器的帮助就可以很快找到网络上共享的文件.

3.2 系统的结构

系统分为四层,分别封装了系统的功能模块,如图 1 所示.系统的最底层是网络层,负责监听网络的消息和发送消息.网络层是一个异步的 UDP socket,当收到消息时,把消息传到上一层去处理;当系统要发送消息的时候,把消息封装到一个 UDP 包中再传出去.第二层负责对收到的 UDP 包解包,并转换成系统可以处理的消息,然后通知上层的系统来处理.当上层的系统要传送消息,这一层把这些消息打成 UDP 包,然后传到下一层去把消息发送出去.第三层是系统的核心模块,负责算法的实现.这一层处理节点的加入、退出,周期性地维护网络系统和资源的查找任务.第四层负责共享本机的资源,在网络中注册共享文件,查找共享文件并提供文件共享信息的访问的功能.

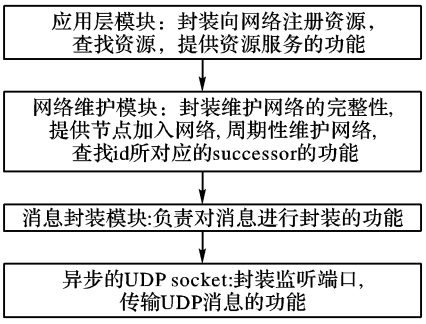

图 1 使用改进算法的系统结构

3.2.1 系统通讯层的设计

系统最底层开一个 UDP 端口监听网络的消息.当收到一个 UDP 包消息的时候,系统向上层发送收到的消息.当系统要传输一个消息的时候,也是通过该端口发出 UDP 包的.

系统底层负责数据传输的类是 AsyncSocket.该类中的 Observers 变量保存注册到该类中的所有 Observer 对象.类在主机上开一个 UDP socket 的端口负责网络数据传输.该类继承 Runnable 和 Observed 接口,能够作为线程运行和通知 Observer 发生事件.所运行的线程用来监听系统的一个 UDP 端口,当收到消息的时候,调用 notifyObserver 函数,通知注册

在该接口上的所有 Observer 收到的消息. 当要传送消息的时候, 调用该类的 send 函数, 从该 UDP 端口中发送一个消息. 系统通过这个类与网络中其他的主机进行通信.

3.2.2　系统消息处理层的设计

当 AsyncSocket 类收到消息以后, 通知上一层的类 Node 来处理这个消息. Node 实现 Observer 接口, 把 Node 对象注册到 AsyncSocket 中, 当系统收到 UDP 消息的时候, 就会调用 Node 对象的 onNotify 函数执行.

当系统收到一个 UDP 包的时候, 调用 onNotify 函数. 函数首先把收到的包转换成为一个 Message 消息, 然后根据这个消息和发送这个消息的主机地址生成一个 event 对象. 最后获得一个 transaction 处理这个消息.

该类提供 getServerTransaction 和 getClientTransaction 两个函数, 其作用分别是取得一个处理服务事务和处理作为客户事务.

服务事务的工作是处理由于收到请求而产生的事件. 在系统收到一个消息的以后, 系统会产生一个事件通知 Listener 类. 由于系统可能同时收到多个请求, 所以为了保证并发性, 由 Node 类产生一个服务的事务来处理服务的事件. 事务在执行的时候会产生一个线程来执行操作.

当上层的类要向网络发送消息时, 上层的类会通过 Node 类产生一个 ClientTransaction 来处理消息的传输的任务. ClientTransaction 产生一个 CallID 保留在消息列表中, 如果在一定时间内没有收到应答, 则系统会产生一个超时事件, 通知系统执行 timeout 事件. 在保留已发送但没有收到应答的消息列表以后, ClientTransaction 调用下层 AsyncSocket 的 send 函数把消息传送出去.

类 Node 的作用是把底层收到的字符串消息转化成系统的消息对象, 再把该对象通知更上层的类. 在改进的算法中, 为了标志每个不同的消息, 每个消息都有一个 CALL_ID 的标志, 该标志记录每个请求. 当请求在限定的时间内没有回复时, 系统就调用上层的类 Listener 的 processTimeout 函数来处理请求不能接收到请求节点回复的事件.

程序会周期性地运行 checkTimeout 函数, 该函数检查每个请求的计数有没有超过 timeout 值, 如果超过的话, 就发生 TIME_OUT 事件. 如果在 TIME OUT 时间内收到消息, 那么节点就取消 waitingMessage 内部的消息.

3.2.3　网络维护层的设计

网络维护层负责实现 DHT 算法. 当网络中收到消息时, 这一层负责对接收到的消息进行处理, 从而实现 P2P 网络.

第二层的 Node 类接收到底层的消息以后, 把消息封装到事件中, 然后通知 Listener 对象所发生的消息. Node 类产生相应的事件, 然后生成 ServerTransaction 处理发生的事件. ServerTransaction 根据事件的不同种类, 调用 Listener 中不同的处理函数. Listener 中有三种函数来处理不同的事件: 当网络上的其他节点向该节点发送请求的时候, Listener 调用 processRequest 函数来处理; 当收到的是对本节点的请求的应答时, Listener 调用 processResponse 函数来处理; 当本节点发出请求, 而不能够在限定时间内得到应答的时候, Listener 就会调用 processTimeout 来进行相应的处理.

类 ChordMend 实现 Listener 接口. 把 ChordMend 对象注册到 Node 上以后, 当 Node 接收到消息的时候, 就会向 ChordMend 节点发送事件, 让 ChordMend 节点对该事件进行处理.

ChordMend 类是实现整个 DHT 网络的关键核心. 在该类中, 节点加入网络、查找资源以及维护网络的稳定都是靠这一层的类来工作的.

当节点加入网络的时候, 系统调用 join 函数来寻找网络的启动点. join 函数首先向网络发出广播, 当网络中有其他节点存在时, 这些节点接收到启动信息后就会向发出该信息的节点发出应答. 节点收到应答后, 选择其中几个节点来进行加入网络的工作.

节点找到启动点的以后, 向启动点发出加入网络的请求. 启动点查找该节点 id 所对应的 successor 节点, 然后把查询结果返回到该节点, 节点把查询到的结果放到路由表中, 这样就完成了加入的过程.

刚加入网络的节点的路由表所含的网络节点的信息比较少. 当网络上其他节点给该节点发出请求以后, 该节点记录下消息中的节点信息. 经过一段时间的运行, 节点的路由表就能够逐渐丰富.

每个 FingerTable 有 160 个 NodeList 对象, 每个 NodeList 对象含有零到多个的 PeerNode 对象. PeerNode 类保存网络上节点的信息. 调用 FingerTable 的 getSuccessor 的方法就会返回 FingerTable 中与本节点距离最近的节点的 PeerNode 的消息. 因为 PeerNode 中包含了该节点的 id 号和网络地址, 所以可以根据 PeerNode 对象的属性访问这些节点.

3.2.4　应用层的系统设计

应用层的工作是提供文件注册的功能. 改进算法和 Chord 一样, 每个节点负责维护本节点 id 与 successor 节点 id 之间的名字空间的资源. 在应用层中要提供的操作是查找一个资源, 在网络中注册一个资源和取消注册的资源的任务. 为了提供对其他节点的应用服务. 类 AppService 实现 Listener 接口, 然后注册到 Node 中. 当收到消息时, AppService 负责处理与文件相关的服务.

FileShareNode 类继承 ChordMend 类, 提供对文件服务的访问操作. 该类中有一个 fileService 的属性, 该属性实现的是对文件服务的功能.

4　结语

对 Chord 算法进行了改进, 解决了它没有对网络消息进行有效利用的缺点, 改变了 Chord 中路由表的结构, 使路由表更加灵活, 能够利用网络的消息, 改进了原 Chord 算法不具备的并行查找的能力. 根据改进的算法, 成功地设计了基于该算法的一个文件共享软件, 实践表明该 Chord 改进算法是有效的.

参考文献:

[1] CLARKE I, HONG T, MILLER S, *et al*. Protecting free expression online with Freenet[J]. IEEE Internet Computing, 2002, 6(1): 40-49.

[2] ROWSTRON A, DRUSCHEL P. Pastry: Scalable, distributed object location and routing for large-scale peer-to-peer systems[A]. Proceedings of the 18th IFIP/ACM International Conference on Distributed Systems Platforms(Middleware 2001)[C]. 2001.