针对访问成功率的 P2P 动态网络对象定位模型*

綦宏伟+, 代亚非, 李晓明

(北京大学 计算机科学技术系,北京 100871)

A P2P Objects Location Model for Higher Access Success Rate over Dynamic Networks

QI Hong-Wei+, DAI Ya-Fei, LI Xiao-Ming

(Department of Computer Science and Technology, Peking University, Beijing 100871, China)

+ Corresponding author: Phn: +86-013691368487, E-mail: qi_hongwei@pku.org.cn

Received 2004-03-06; Accepted 2004-11-03

Qi HW, Dai YF, Li XM. A P2P objects location model for higher access success rate over dynamic networks. *Journal of Software*, 2005,16(5):894−902. DOI: 10.1360/jos160894

Abstract: According to the requirement of network massive storage applications, this paper puts forward a P2P based objects distribution and location model, supporting the logic network dynamically composed by a large number of voluntary nodes. The model is discussed in detail as follows: global mapping relation, routing table, object locating and routing algorithm, object indices distribution scheme, and maintenance algorithm when nodes join and leave the network. In particular, a novel scheme for distributing objects indices is provided to improve the average success rate of objects access, and each part of the model is improved. Through analysis, the model fulfills the five objectives given in the introduction. Finally, a simulation program built on this model verifies the expected abilities for balancing the load distribution and improving the objects access efficiency.

Key words: peer-to-peer; objects location; object indices distribution; global mapping relation; routing table

摘 要: 针对网络海量存储系统的应用需求,提出了一个基于 Peer-to-Peer 思想的对象分布和定位模型,能够支持众多节点自发组成的动态网络结构.对该模型进行了比较完整的论述,依次建立了全局映射关系、路由表、对象定位和路由算法、对象索引分布方案和节点加入、退出时的维护算法,特别是提出了新的对象索引分布方案,提高了对象的平均访问成功率,围绕此方案,对模型的各组成部分进行了改进,实现了提出的 5 个性质.最后,通过建立模拟程序,验证了模型的分析预测结果,能够提供均衡的负载分布和较好的对象访问效率.

关键词: peer-to-peer;对象定位;对象索引分布;全局映射关系;路由表

中图法分类号: TP393 **文献标识码**: A

* Supported by the National Natural Science Foundation of China under Grant No.90412008 (国家自然科学基金); the National Grand Fundamental Research 973 Program of China under Grant No.2004CB8318204 (国家重点基础研究发展规划(973))

作者简介: **綦宏伟**(1978−),男,浙江龙游人,硕士,主要研究领域为分布式系统,网络存储,P2P 计算;**代亚非**(1958−),女,博士,教授,博士生导师,主要研究领域为分布式系统,网络存储,P2P 计算;**李晓明**(1957−),男,博士,教授,博士生导师,主要研究领域为计算机系统,Web 信息处理,网络计算.

在当今开放、互联的网络中,数据、计算和存储等各种共享资源的数量急剧增长,推动了海量级网络应用系统的发展,一个具体例子就是网络文件共享系统.

网络海量应用需要解决的基本问题是数量巨大的节点和对象之间的关系.加入网络的计算机节点是计算资源和存储资源的统一体.而完整的数据信息可用虚拟对象来表示,该对象被赋予给定的标识,记录了数据的属性,封装了数据的内容.

基于 Peer-to-Peer 思想的对象分布和定位模型使得任一节点可访问其他节点上的任一对象.节点之间组成动态逻辑结构,节点可自主地进入或离开这个结构,能自主地承担一定的计算负载和存储负载,对象在节点上动态分布,节点能够通过结构定位到存放在其他节点上的对象.问题的核心是节点的平均负载和对象的平均访问效率之间的权衡.

本文主要讨论的就是一个基于 Peer-to-Peer 思想的改进的对象分布和定位模型.该模型主要参考了 Plaxton 等人提出的 PRR 模型[1].PRR 模型构造了一个静态节点结构,将对象和对象位置信息分离,通过先找到对象位置信息来定位对象,该模型使每个节点上存放的对象位置信息数量在理论上是均等的,而对象平均访问效率较高,假设有 N 个节点,则总的位置信息存储开销为 $O(N\log N)$,对象访问的平均节点跳数为 $O(\log N)$.其后,Tapestry[2] 和 Pastry[3]等人将该模型扩展为动态结构.

本文将 PRR 模型扩展为动态结构,保持了平均负载均衡且低的优点,并主要针对对象平均访问效率中的平均访问成功率进行了讨论.访问效率是以下指标的综合衡量:平均节点跳数、平均相对延迟和平均访问成功率.节点跳数是指一次访问过程中从发起节点路由到目标节点所经过的中间节点的个数.相对延迟是指两个节点之间通过逻辑结构建立的路由过程延迟与直接通过底层物理网络进行路由的最短延迟之间的比值.访问成功率是指通过路由过程定位到的节点上存在所需对象位置信息的概率.在已有的参考模型中,相对前两个指标,访问成功率需要进一步考虑.

本文讨论的对象定位模型其基本结构为:假设存在这样的全局映射关系,对于每个节点和对象都具有全局唯一的 *id*,对象 *oid* 和节点 *nid* 之间是多对一的,对于每一个对象 *oid* 必定映射到唯一的节点 *nid*.对于每一个对象,在某节点存放该对象时,建立对象索引,包括对象 *oid*、存储对象的节点位置等信息,并将该对象索引存放到映射后的唯一节点 *nid* 上.当其他节点需要定位该对象时,通过路由算法到达存放有对象索引的节点 *nid*,然后由对象索引得到存储对象的节点位置,从而能够访问到对象.由于逻辑结构的动态性和单个节点所知信息的局部性,全局映射关系需要节点间相互协作,在路由的过程动态体现.

本文提出的模型能够实现以下 5 个性质:

性质 1. 全局映射关系使得对象索引随机、均匀地在节点上分布.

性质 2. 全局映射关系能够动态维护.

性质 3. 确定性路由.给定对象 *oid*,从任一节点出发,路由算法总能确定地到达通过全局映射关系决定的唯一节点.

性质 4. 路由算法的平均节点跳数和平均相对延迟低.

性质 5. 对象定位的成功率高.

本文的主要贡献在于,提出一个对象索引分布方案,围绕该方案,在保持原 PRR、Tapestry 等模型的优点的基础上,改进了对象分布和定位模型,实现了以上 5 个性质,对对象定位的成功率提供了一种量化控制机制.

本文第 1 节依次讨论模型中的全局映射关系、路由表、对象定位和路由算法、对象索引分布方案和节点加入、退出时的维护算法.第 2 节通过模拟程序验证模型结果.第 3 节讨论相关的研究工作.第 4 节是结论.

1 对象分布和定位模型

1.1 全局映射关系

全局映射关系的第 1 步是生成对象 *oid* 和节点 *nid*,可以由安全 Hash 函数很好地得以实现.

安全 Hash 函数,比如 SHA-1(160 位),具有如下性质:可作用于任意长度的数据块,产生定长输出数字串,并且

满足随机均匀分布,使得能够很容易地构造出符合条件的节点 *nid* 和对象 *oid*.

将节点的物理地址(如 IP 和端口号)经过安全 Hash 函数转换得到数字串,作为节点 *nid*;将对象的名称或内容经过类似转换,得到对象 *oid*;节点 *nid* 和对象 *oid* 都足够长,并且前者的长度要小于后者,使得每个节点都能对应较多的对象索引.比如节点 *nid* 的基数 *B* 为 16,长度 *L* 为 16,对象 *oid* 的基数 *B* 为 16,长度 *L* 为 32.

根据安全 Hash 函数的性质,转换得到的 *nid* 和 *oid* 能够满足随机均匀分布.

设 *nid* 的长度为 *x*,*oid* 的长度为 *y*,oid_x 为 *oid* 的前 *x* 位数字串,定义映射关系及根节点如下:

定义 1(映射关系 *I*). 定义域为所有 *oid*,值域为所有 *nid*,多对一映射.在所有节点中,存在数值上与 oid_x 最接近的节点,且个数只有 1 个或 2 个两种情况:如果为 1 个,则令映射值 *I*(*oid*)等于此 *nid*;如果为两个,则这两个节点必定相对 oid_x 左右对称,则令映射值 *I*(*oid*)等于数值大的 *nid*.

定义 2(根节点 *Root*(*obj*)). 对于任一对象 *oid*,通过映射关系 *I* 得到唯一的节点 *nid*=*I*(*oid*),令此节点为对象的根节点 *Root*(*obj*).

每一对象都有对应的根节点,每一根节点可能对应 0 个或多个对象,且不同的根节点其对应的对象集互不重叠.对象索引存放在对应的根节点上.

性质 1 的证明:对象 *oid* 和节点 *nid* 都是随机均匀分布的,映射关系 *I* 是根节点 *nid* 空间对对象 *oid* 空间的一次划分,划分后各对象子空间内 *oid* 的个数依旧均匀分布,因此根节点承担的对象索引存储开销是均衡的. □

性质 2 的证明:当对象加入或退出逻辑结构时,根据后面的维护算法在邻居集中重新分布对象索引,使之满足新的全局映射关系. □

1.2 路由表

路由表是对象定位和路由算法思想的体现,本模型和 PRR 模型[1]一样,其路由表和路由算法的建立受立方体模型的启发.

假设立方体模型共有 8 个节点,每个节点的 *nid* 用 3 位二进制串表示,如 110,每个节点的路由表由 3 个相邻节点构成,这 3 个节点分别与它只有一位数字的区别,节点 110 的路由表为{010,100,111}.路由算法按前缀匹配方式进行,当节点 110 想要访问节点 001 时,它从路由表中选择第 1 位为 0 的节点(即节点 010)进行路由请求转发,节点 010 从自己的路由表中选择前两位为 0 的节点(即节点 000)进行路由请求转发,最后由节点 000 路由到节点 001.

立方体模型使得每个节点具有同样开销的路由表,每次路由过程能够确定性地到达目标节点.但它的结构是静态的,没有考虑节点加入与退出,没有考虑路由表项为空的情况.

为了有利于度量访问效率和规划合理的节点空间,参考 PRR 模型,采用如下通信成本,该通信成本已证明可适用于多数物理网络.

定义 3(节点间通信成本 *c*). 给定节点集 *V*,设 *c* 为任意两个节点间传递固定长度消息经过的物理网络通信延迟,*c* 为实数值.对于 *V* 中的任一节点 *u* 和任一实数值 *r*,令 *N*(*u*,*r*)代表节点集{$v \in V: c(u,v) \Leftarrow r$},可以把 *N* 假想为以 *u* 为中心,*r* 为半径的球集,则存在实常数值 $\delta > 2^3$ 和 Δ,使得

$$\min\{\delta|N(u,r)|,|V|\} \Leftarrow |N(u,2r)| \Leftarrow \Delta|N(u,r)|,$$

其中|*N*|和|*V*|分别表示集合 *N* 和 *V* 中的节点个数,min{}表示取最小值.

上述左右不等式限定了节点空间密度的变化,保证了如图 1 所示路由表的构造.

0---	**1---**	2---	3---	4---	∅	6---	∅
∅	11--	∅	**13--**	14--	∅	16--	17--
∅	131-	∅	∅	134-	**135-**	∅	137-
1350	∅	∅	∅	∅	∅	∅	**1357**

Fig.1 Routing table (nid_{self}=1357)

图 1 路由表(nid_{self}=1357)

如图 1 所示,路由表 RT 由行和列构成.行数对应节点 *nid* 的长度 *L*,列数对应节点 *nid* 的基数 *B*.假设 *L*=4,*B*=8.对于第 *I* 行和第 *J* 列(0⇐*I*<*L*,0⇐*J*<*B*)的路由表项用 *RT*[*I*,*J*]表示,每个路由表项中存放节点队列,队列长度最小为 0(称为空表项,用∅表示),最大为 *K*(*K* 取 *O*(log*N*),*N* 为预估节点总数).对于队列中的任一节点 *nid*,具有相同的性

质:$PREFIX(nid_{self},I)=PREFIX(nid,I)$并且 $nid[I+1]=J$,其中 $PREFIX(nid,I)$表示节点 nid 的前 I 位子串,当 $I=0$ 时,$PREFIX$ 为空串,$nid[I]$表示节点 nid 的从高到低的第 I 位值.节点队列中的节点按照通信成本从小到大排列,首节点称为主要节点,其余节点称为次要节点.路由表的每一行都存在一个路由表项,其主要节点为自身.路由表的存储开销小于 LBK.

为了建立正确的路由算法,路由表必须尽可能地满足以下两条性质,在后面节点动态维护部分我们再给出证明.

路由表性质 1(同空性). 如果任一节点 nid 的路由表项 $RT[I,J]$为空,则结构中不存在对应前缀的节点.

路由表性质 2(最近优先). 如果任一节点 nid 的任一非空路由表项 $RT[I,J]$的节点队列长度为 S,则它们就是在对应前缀的所有节点中通信成本最小的前 S 个节点.

1.3 对象定位和路由算法

对象定位分为两步,首先根据对象 oid 路由到存放对象索引的根节点,然后直接从其对象索引中的位置信息定位对象所在的节点.

开始时,发起节点向自己发送对象定位消息,消息参数包括发起节点 nid,目标对象 oid,起始查找行号=0,查找标志为正常查找(另两种为最小上界查找和最大下界查找).

消息处理过程首先判断自己是否就是候选根节点.如果自己的 nid 就是目标对象 oid 的前缀或者起始查找行号已经到达了路由表的最大行数,那么自己就是候选根节点,检查自己是否存储目标对象索引,如果有,就向发起节点发送回对象索引,否则告知没有对象索引.

如果自己不是候选根节点,就根据查找标志分别处理.

(1) 如果查找标志是正常查找,就从起始查找行开始根据目标对象 oid 进行最大前缀匹配,如果当前行对应路由表项的主要节点等于自身,则行号增加,继续查找,直到找到一个非空路由表项,其主要节点不等于自身,或者找到一个空表项.

(1.1) 如果找到非空节点,就向主要节点发送对象定位消息,消息参数中发起节点和目标对象不变,起始行号等于当前查找行号+1,查找标志为正常查找.

(1.2) 如果找到空表项,则同时查找路由表中的最小上界和最大下界节点.

(1.2.1) 在查找最小上界节点时,从当前行的当前空表项向右查找,如果能找到非空表项,则取其节点队列中的 nid 数值最小节点为下一步的消息转发节点,如果没有非空表项,则下降一行,从自身节点所在的路由表项开始向右继续查找,采取同样的处理方式.转发消息参数中发起节点和目标对象不变,起始行号等于当前查找行号+1,查找标志改为最小上界查找.如果直到第 0 行仍没有找到可转发的节点,则认为自身就是最小上界节点,按候选根节点处理,检查自己是否存储目标对象索引,如果有,就向发起节点发送回对象索引,否则告知没有对象索引.

(1.2.2) 查找最大下界节点类似于查找最小上界节点,区别是它是向左查找,寻找数值最大的节点,转发消息时查找标志改为最大下界查找.如果直到第 0 行仍没有找到可转发的节点,则认为自身就是最大下界节点,按候选根节点处理,检查自己是否存储目标对象索引,如果有,就向发起节点发送回对象索引,否则告知没有对象索引.

(2) 如果查找标志是最小上界查找,则在给定的起始查找行中从第 0 列向右开始,查找数值最小的节点,由于每一行肯定至少存在自身节点,则必定能找到一个节点,向该节点转发对象定位消息,消息参数中发起节点和目标对象不变,起始行号等于当前查找行号+1,查找标志仍为最小上界查找.

(3) 如果查找标志是最大上界查找,则在给定的起始查找行中从最大列向左开始,查找数值最大的节点.同样,由于每一行肯定至少存在自身节点,则必定能找到一个节点,向该节点转发对象定位消息,消息参数中发起节点和目标对象不变,起始行号等于当前查找行号+1,查找标志仍为最小上界查找.

算法最终肯定能够终止.因为如果正常查找中找不到候选根节点,肯定会同时进行最小上界和最大下界查找,而在它们的查找过程中行数是不断递增的,因此最终会达到最大行数,从而最终找到候选根节点.

发起节点最终会得到一个或两个结果消息.通过比较消息返回节点的 *nid*,就能判定最终根节点.如果返回的对象索引非空,就能从中取出对象的位置信息,从而定位到存储对象的节点.如果对象索引为空,则认为结构中不存在该对象.

性质 3 的证明:(反证法)假设现在由两个不同的节点发起相同的对象定位请求,得到了两个不同的根节点 *A* 和 *B*,设这两个根节点的最长共同前缀为 *s*,*s* 的长度为 *L*,且从节点 *A′*和 *B′*开始分别进行前缀长度为 *L*+1 的路由.根据路由算法,这两个过程都发生在空表项处理之后.根据路由表性质 1,这两个过程选择的列号都是一样的,从而两个路由过程到达同样的最终节点,与假设发生矛盾.因此给定对象 *oid*,从任一节点出发,路由算法总能确定地到达通过全局映射关系决定的唯一节点. □

性质 4 的证明:参考 PRR 模型,并根据本模型中有限的空表项处理,可以得到类似的结果,其平均节点跳数为 $O(\log N)$,平均相对延迟为 $O(1)$. □

1.4 对象索引分布方案

每个对象 *oid* 映射到单个根节点,但是对象索引的实际存放位置是由分布方案决定的,对象索引分布方案影响动态结构中对象定位的成功率.

对于静态结构的 PRR 模型[1],对象索引分布可以预先设定.对于动态结构,存储对象的节点根据路由算法找到根节点,在其上建立对象索引.根节点可以通过软状态方式维护对象索引,每隔 T_{obj} 时间,存储对象的节点就向根节点重新发布索引信息.如果经过多个 T_{obj} 时间,根节点未收到发布信息,就认为对象已不可访问,因此删除对象索引.

但是一旦根节点突然退出结构,则映射关系调整后的新根节点可能还没有该对象索引,在存放对象的节点重新发布索引信息之前,对该对象的访问总是失败的.为了保证很高的对象定位成功率,T_{obj} 时间必须尽可能地小,这样就会耗费过多的网络资源.

针对这个问题,Tapestry[2]采用在路由中间节点增加对象索引备份和引入多套映射关系相结合的方法,每套映射关系得到一个根节点,对象索引同时存在于多个根节点上,前者增加了中间节点的存储开销,并且索引信息可能不能及时更新,后者的多根节点访问会增加时延和占用更多带宽.Pastry[3]采用每个节点建立邻居节点集的方法,但其将对象直接存放在邻居节点中,使得传输对象本身会占用很多资源.Bamboo[4]也支持邻居节点集的方法,证明它能更好地保持 *nid* 空间的一致性,具有更好的可靠性和效率.

本模型保持单套映射关系和单个根节点,增加邻居节点集.定义根节点主管的对象索引前缀范围为[*nid*_l, *nid_r*],其中 *nid*_l 和 *nid_r* 分别对应数值上最接近的左右邻居节点,定义根节点的 *M* 邻居集为 $M(M>=1)$个在数值上最接近的节点,每个根节点把自己主管的对象索引都请求 *M* 邻居集节点存放.经过 T_{nb} 时间,根节点把自己主管的对象索引的改变情况通知 *M* 邻居集进行更新.当根节点退出时,路由过程会自动定位到新根节点,只要新根节点属于原根节点的 *M* 邻居集,则对象定位会无缝地完成.当下次新根节点维护自己的 *M* 邻居集时,会将对象索引信息扩散出去.

本方案不改变原有的对象定位算法,可以允许较大的 T_{obj} 时间,每个根节点可以灵活地设定自己的 *M* 邻居集大小,调整对象索引的存储开销.

如图 2 和图 3 所示,分别给出了当 *M*=1 和 *M*=2 时分布方案的示意图.设节点 *nid* 的长度 *L*=4,对象 *oid* 的长度为 6,基数均为 8.两图都有上下数轴,上数轴表示节点 *nid* 的分布,用黑圈代表,其范围为[0000,7777],下数轴表示对象 *oid* 的分布,用白圈代表,其范围为[000000,777777].给定一个节点,其下的左右虚线限定了其存放的对象索引范围.从图中可以看出,当 *M*=1 时,平均每个对象索引存放在 2.4 个节点上,当 *M*=2 时,平均每个对象索引存放在 3.6 个节点上,因此,在同样的动态变化情况下,后者的对象定位成功率将明显高于前者.

性质 5 的证明:根据以上分析,给定一个合适的 *M* 和邻居维护周期 T_{nb} 时间,较大的对象重新发布周期 T_{obj} 时间,在网络结构的变化较快时,可以提高对象定位成功率. □

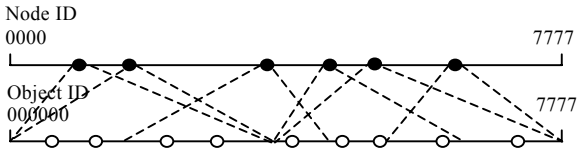

Fig.2 Object indices distribution example (M=1)

图 2 M=1 时的对象索引分布示意图

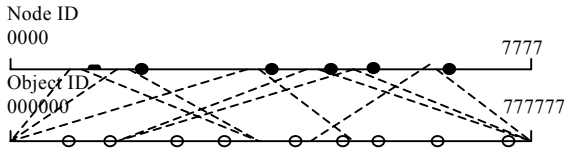

Fig.3 Object indices distribution example (M=2)

图 3 M=2 时的对象索引分布示意图

1.5 节点动态加入、退出维护算法

动态维护算法针对 3 种情况:节点加入、节点事先通知的正常退出、节点未通知的异常退出.

在算法中涉及到反向路由表和代理节点.反向路由表的作用是当节点退出时,能很方便地通知其他节点从路由表中删除自己,当且仅当其他节点的路由表中存在自己,才把它放入反向路由表中.某节点的代理节点类似于对象的根节点,它们的 *nid* 在数值上最接近,通过路由算法可动态获得.

节点加入算法.

(1) 节点路由表的生成;

(1.1) 首先通过预先已知的节点路由到代理节点,以它的路由表作为自己的初始路由表.

(1.2) 如果自身和代理节点的最大匹配前缀长度为 S,则对路由表中第 S 行及其以上节点进行组播.对于组播到的每个节点,都返回新节点确认消息,加入临时节点列表.

(1.3) 从第 S–1 行开始递减填充路由表.新节点要求临时节点列表中的每一个节点返回其第 S–1 行的所有节点,填入路由表中的第 S–1 行,测量与这些节点的通信成本,取前 K 个节点.

(1.4) 以第 S–1 行节点为新的起点,向它们请求得到第 S–2 行节点,进行同样的填入、测量和筛选,重复进行直到完成第 0 行.

(2) 邻居关系结构和对象索引分布的改变;

(2.1) 参考代理节点,生成自己的左右邻居和 M 邻居集,获得自己主管的对象索引集.

(2.2) 通知其他节点改变左右邻居、M 邻居集.

(2.3) 把自己主管的对象索引集存放到 M 邻居集节点上.

(3) 对路由表中的每个节点,通知它们将自己加入反向路由表中.代理节点通知反向路由表中的节点修改路由表.

节点的正常退出算法.

(1) 通知其他节点修改路由表;

通知反向路由表中的每个节点,如果与该节点的最大匹配前缀长度为 S,则附加路由表中的第 S–1 行节点.得到消息的节点删除退出节点,并选择附加节点填入路由表.

(2) 通知邻居节点修改左右邻居、M 邻居集,并转移对象索引信息;

(3) 通知路由表中的每个节点将自己删除出它们的反向路由表.

针对其他节点异常退出的维护算法.

当节点异常退出时,其他节点需要通过一定的时间延迟才能发现(如节点软状态的维护,消息通信的不可达).

(1) 需要修改路由表时

根据退出节点的 *nid* 路由到其代理节点,通知节点异常退出.

如果先前没有其他节点通知过,则代理节点对路由表中第 S–1 行节点进行组播,其中 S 为代理节点与退出节点的最大匹配前缀长度.组播到的节点将第 S–1 行节点传回代理节点,代理节点将节点列表发给通知节点和所有组播到的节点,修改路由表.

如果先前已有其他节点通知过,则直接返回以前的节点列表.

(2) 需要修改邻居关系时

计算新的左右邻居和 M 邻居集,重新划分存放的对象索引集.

路由表性质 1 的证明:节点退出不影响该性质,考虑节点加入情况.假定在某新节点加入前该性质满足,如果新节点存在空表项,则该空表项或者对应新节点和其代理节点的匹配前缀,或者相对代理节点更靠近新节点,对于前者,由加入算法在代理节点必定存在对应空表项;对于后者,根据代理节点的定义,则在结构中必不存在对应的节点. □

路由表性质 2 的证明:对于节点正常退出容易满足,节点异常退出类似于节点加入.假定节点加入前满足该性质,则根据队列长度 $K=O(\log N)$,通过代理节点能够尽可能地得到所有最大匹配前缀的节点,并且定义 3 中的通信成本使得点集空间的扩张受限,如果已经获得的某行节点能够尽可能地满足通信成本最优,那么能够尽可能地获得满足性质的低一行节点. □

2 模拟程序和测试结果

2.1 模拟程序说明

模拟程序通过 GT-ITM 网络拓扑产生器建立具有 5 000 个物理节点的 Transit-Stub 模拟广域网络.逻辑结构中的节点 *nid* 和对象 *oid* 都是通过 SHA-1 安全 Hash 函数得到的,路由表的行数为 8,列数为 8.节点受控于一定的生命周期以模拟加入和退出网络,会周期性发布对象索引信息、维护路由表和邻居信息(设这 3 个周期时间分别为 P,R 和 N),会随机地发起对象定位请求.

2.2 测试结果

图 4 给出了随节点数变化的两种不同的空间开销.前者是路由表项队列平均长度,后者是 M=2 时存放对象索引的平均节点个数.从图中可以看出,队列平均长度随节点个数的增加而不断增加,但是控制在 $O(\log N)$范围内,存放对象索引的平均节点个数都是 2,验证了空间开销的均衡.

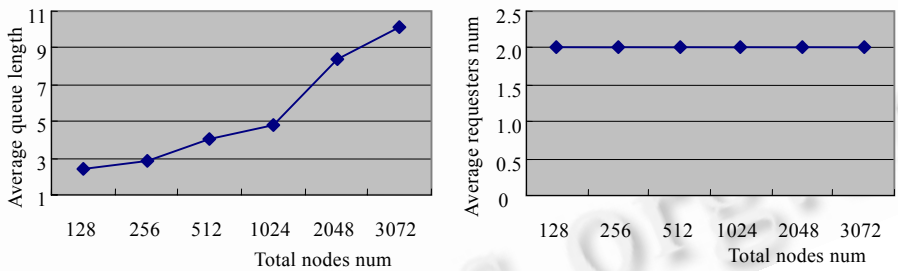

Fig.4 Average queue length, average requesters num vs. total nodes num

图 4 路由表项队列平均长度、存放对象索引节点的平均个数随节点总数变化的关系

图 5 和图 6 分别给出了平均相对跳数和平均相对延迟.总节点数为 512.从两图中可以看出,两者的结果是比较相似的.当路由表维护周期时间较长(2000s 以内)时,相对跳数和相对延迟总能控制在 2.5 以下.

图 7 给出了路由表瞬时正确率的变化情况.总节点数为 1024,路由表维护周期时间为 100s,每隔 500s 测量一次.从图中可以看到,虽然路由表正确率有一定的变化,但是都在 99.5%以上,说明了节点维护算法的有效性.

图 8~图 12 是给定 5 个不同的对象索引发布周期时间 P 时,对于不同的 M,平均对象定位成功率 S 随邻居结构维护周期时间 N 的变化关系.对于其中的每个子图,总节点数都是 512.如果认为 S>=95%比较理想,则从这 5 个图中可以观察到:无论 P 和 N 取何值,M=2 和 M=4 下的 S 都较接近,而 M=0 下的 S 相对较差,几乎总在理想线以下;当 P<=1000,N<=1000 时,M=2 和 M=4 时的 S 比较理想;当 P=2000,N<=100 时,M=2 和 M=4 时的 S 比较理想;当 P=4000,N<=10 时,M=4 下的 S 比较理想.

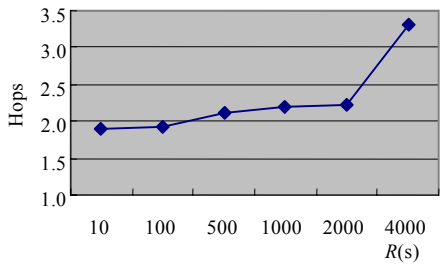

Fig.5 Average relative hops

图 5 平均相对跳数

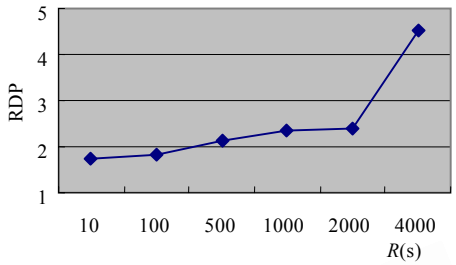

Fig.6 Average relative delay penalty

图 6 平均相对延迟

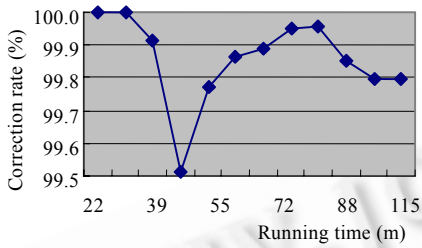

Fig.7 Routing table items correction rate variation

图 7 路由表正确率随时间变化

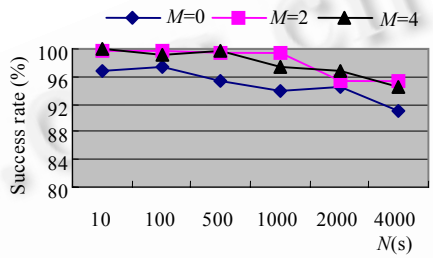

Fig.8 Object access success rate (P=100s)

图 8 对象访问成功率(P=100s)

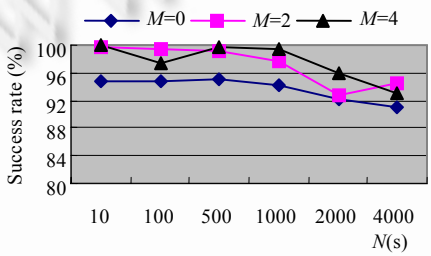

Fig.9 Object access success rate (P=500s)

图 9 对象访问成功率(P=500s)

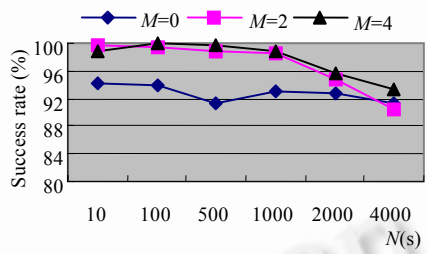

Fig.10 Object access success rate (P=1000s)

图 10 对象访问成功率(P=1000s)

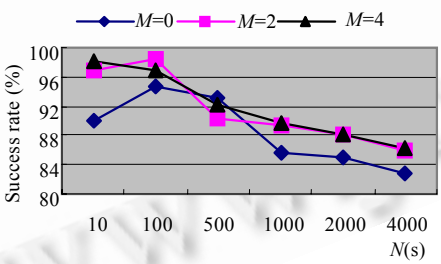

Fig.11 Object access success rate (P=2000s)

图 11 对象访问成功率(P=2000s)

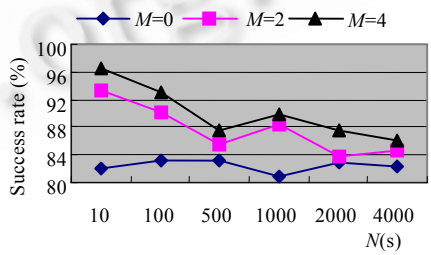

Fig.12 Object access success rate (P=4000s)

图 12 对象访问成功率(P=4000s)

3 相关工作

目前,基于 Peer-to-Peer 思想的对象定位方案可以分为两类:一类是非结构化的,侧重于按照某统计特征维护网络拓扑[5,6],当节点数量很大时,定位效率会降低很多,这类的实例如 Gnutella 等.另一类是结构化的,根据分布式 Hash 表建立全局映射关系,维护节点逻辑结构,把对象的定位过程与节点之间的路由过程绑定,PRR[1],Tapestry[2],Pastry[3],Emergint[5],Chord[7],CAN[8]和本模型都属于此类,其中 Chord,CAN 采用各自不同的路由算法,建立路由表时不考虑物理网络中的通信开销,其他方案都参考 Plaxton 等人提出的 PRR 模型[1],采用相似的路由表结构和对象定位算法.针对这类模型建立的分布式文件系统有 Oceanstore[9],Past[10],CFS[11],燕星[12]等.

本模型的路由和对象定位算法参考了 Tapestry[2]和 Emergint[5],但是由于针对提高对象访问成功率,对于根节点的定义和空表项处理都有不同,把它们与对象索引分布进行了统一.本模型的对象索引分布方案参考了 Pastry[3]和 Bamboo[4],在根节点维护邻居节点集,而且在节点维护算法中考虑了对象索引的动态维护,保证了路由和定位算法的正确性.

4 结 论

本文提出、分析并模拟验证了一个基于 Peer-to-Peer 思想的对象分布和定位模型,并主要针对提高访问对象的平均成功率提出了新的对象索引分布方案.

本模型的各组成部分紧密联系并统一完整,依次建立了全局映射关系、路由表、对象定位和路由算法、对象索引分布方案和节点加入、退出时的维护算法,实现了引言中提出的 5 个性质.

本文最后建立模拟程序,验证了模型的分析预测结果:具有均衡的节点存储负载,路由表中表项队列平均长度控制在 $O(\log N)$范围内,邻居集大小控制在 $O(M)$范围;具有较高的路由表正确率(99.5%以上);具有高效的路由算法,路由过程平均相对跳数和相对延迟控制在 2.5 以下;具有较高的对象定位成功率,且可进行量化控制,当邻居集 $M>=2$ 时,如果对象索引发布周期时间为 1000s,邻居结构维护周期时间为 1000s,则平均对象定位成功率在 95%以上.

因此,本模型适合于建立网络海量应用,特别是文件共享系统,能够适应众多节点自发组成的动态网络结构,提供均衡的负载分布和较好的对象访问效率.

References:

[1] Plaxton CG, Rajaraman R, Richa AW. Accessing nearby copies of replicated objects in a distributed environment. Theory of Computing Systems, 1999,32(3):241−280.

[2] Hildrum K, Kubiatowicz JD, Rao S, Zhao BY. Distributed object locationin in a dynamic network. Theory of Computing Systems, 2004,37:405−440.

[3] Rowstron A, Druschel P. Pastry: Scalable, decentralized object location and routing for large-scale peer-to-peer systems. In: Proc. of the 18th IFIP/ACM Int'l Conf. on Distributed Systems Platforms (Middleware). 2001. 329−350.

[4] Rhea S, Geels D, Roscoe T, Kubiatowicz J. Handling churn in a DHT. In: Proc. of the USENIX Annual Technical Conf. 2004. 127−140.

[5] Albert R, Barabasi AL. Statistical mechanics of complex networks. Review of Modern Physics, 2002,74(1):47−97.

[6] Iamnitchi A, Ripeanu M, Foster I. Locating data in (Small-World?) peer-to-peer scientific collaborations. In: Proc. of the 1st Int'l Workshop on Peer-to-Peer Systems. Cambridge: Springer-Verlag, 2002. 232−241.

[7] Stoica I, Morris R, Karger D, Kaashoek MF, Balakrishnan H. Chord: A scalable peer-to-peer lookup service for Internet applications. In: Proc. of the ACM SIGCOMM 2001. San Diego, 2001. 149−160.

[8] Ratnasamy S, Francis P, Handley M, Karp R, Shenker S. A scalable content-addressable network. In: Proc. of the ACM SIGCOMM 2001. San Diego, 2001. 161−172.

[9] Rhea S, Eaton P, Geels D, Weatherspoon H, Zhao B, Kubiatowicz J. Pond: the OceanStore prototype. In: Proc. of the 2nd USENIX Conf. on File and Storage Technologies (FAST 2003). San Francisco, 2003.

[10] Rowstron A, Druschel P. Storage management and caching in PAST, a large-scale, persistent peer-to-peer storage utility. In: Proc. of the 18th ACM SOSP. Chateau Lake Louise: ACM, 2001. 188−201.

[11] Dabek F, Kaashoek MF, Karger D, Morris R, Stoica I. Wide-Area cooperative storage with CFS. In: Proc. of the 18th ACM SOSP 2001. Chateau Lake Louise: ACM, 2001. 202−215.

[12] Han H. Studies on Internet oriented distributed massive file storage system [Ph.D. Thesis]. Beijing: Department of Computer Science and Technology, Peking University, 2002 (in Chinese with English abstract).

附中文参考文献:

[12] 韩华.面向 Internet 的分布式海量文件存储系统研究[博士学位论文].北京:北京大学计算机科学技术系,2002.