文章编号: 1000-5862(2012)03-0292-05

重叠型 P2P 网络中的查询负载均衡策略研究

王　珏

(华东交通大学软件学院, 江西 南昌 33013)

摘要: 提出了一种资源发布和查询过程中的负载均衡策略. 通过将超级结点的资源信息组织为 B^+树, 利用 B^+树中叶子结点的均衡性来实现超级结点的负载均衡. 实验结果表明: 在网络中资源发布数和查询数较多的情况下, 该方法能够在相对不降低搜索性能的前提下, 有效地解决重叠型 P2P 网络中超级结点负载不均衡的问题.

关键词: 重叠型 P2P 网络; 超级结点; 负载均衡; B^+树
中图分类号: TP 393.09　　文献标志码: A

0　引言

P2P 网络可以分为结构化和非结构化网络. 结构化 P2P 网络因为其可控性被广泛应用于各种环境中, 其中一种主流技术是基于分布式哈希表(distributed hash table, DHT)的结构化 P2P 网络. 在结构化 P2P 网络中, 所有的结点不论其能力如何, 在资源查询和访问中都承担着相同的角色: 各结点通过承载一定的键值空间来分担网络负载. 在理想情况下, 各结点负载均匀分布, 系统具有良好的负载均衡并能发挥最优性能. 从已有的研究成果来看, 结构化 P2P 网络的性能受到多种因素制约: (1)在 P2P 网络中, 角色相同的结点, 其各自的能力值(包括 CPU 能力、内外存空间和网络带宽等)有很大的差别[1]; (2)结点在网络中的稳定性也各不相同, 网络中多数结点可任意加入或离开网络, 造成网络结构的频繁变动[2]; (3)P2P 网络中能力值低、或稳定性差的结点被称为弱结点, 大量弱结点势必会使得 P2P 网络变得不稳定且性能更差[3].

为了解决结构化 P2P 网络的上述问题, 多种不同的 P2P 网络结构被提出, 其中最为广泛应用的是重叠型 P2P 网络. 本文研究基于一种基于超级结点(super-peer, SP)的重叠型 P2P 网络, 在网络中能力值高、稳定性强的结点称为超级结点, 根据重叠型网络的异构性, 超级结点在查询中承担较多的负载. 与其他结构化 P2P 网络相比, 重叠型 P2P 网络包含多个以超级结点为中心的局部网络, 使集中式查询更有效率, 同时超级结点和各结点互联, 具备自治性和动态适应性, 提高了 P2P 系统的查询和传输性能[4].

然而, 在重叠型 P2P 网络中进行查询也存在着问题. 由于超级结点 DHT 使用的哈希函数或者查询关键字请求量(热度)不相同等原因, 导致超级结点的负载分布不均衡. 本文通过分析重叠型 P2P 网络中超级结点查询索引的分布情况, 参考 B^+树中结点键值分布策略和相关算法, 提出了一种超级结点负载均衡的策略, 使得负载在查询过程中根据查询索引请求量动态平衡, 并设计相关实验过程模拟了重叠型 P2P 网络的查询过程, 观察查询过程中超级结点的负载变化情况, 以及查询算法的效率变化情况.

1　重叠型 P2P 网络介绍

1.1　组织结构

重叠型 P2P 网络是由超级结点 Chord 网络和普通结点网络叠加而成的混合型网络, 由上下两层网络重叠而成: 上层仅包含超级结点, 按 DHT 协议组成 P2P 网络, 一般情况下, 超级结点遵照 Chord 标准组织[5]; 在某些重叠型 P2P 网络中, 超级结点按照完全图的结构进行组织.

收稿日期: 2011-10-23
基金项目: 江西省教育厅科技计划(GJJ12305), 江西省科技厅青年基金(20114BAB2011018)和华东交通大学校立科研基金(09RJ06)资助项目.
作者简介: 王　珏(1978-), 男, 江西南昌人, 讲师, 硕士, 主要从事计算机网络与信息安全的研究.

下层网络由普通结点组成. 下层网络中的多个普通结点与上层网络中的 1 个超级结点连接, 构成 1 个自治域. 普通结点与超级结点的连接方式多以星型为主, 相互之间不相交; 或者与同一个超级结点相连的普通结点相互连接, 但与其他普通结点保持独立; 本文重叠型 P2P 网络结构如图 1 所示, 普通结点不仅与超级结点相连, 自身按照结构化 P2P 网络(图中为 Chord 标准)结构组织在一起, 能够在上层网络失效的情况下保证普通结点继续工作, 也有益于结点资源的动态调整.

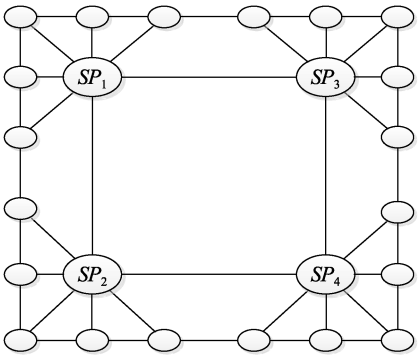

图 1　重叠型 **P2P** 网络组织结构示意图

1.2　资源查询策略

每个普通结点 *CP* 保存其拥有的资源索引 *ResIndex*, 资源索引由结点的 IP、资源 ID、文件名等信息组成. 而连接多个普通结点的超级结点上的拥有一张资源索引列表 $ResIndexList = \{ ResIndex_1, ResIndex_2, \cdots, ResIndex_n \}$, 该列表记录了与其连接的普通结点的资源索引. 普通结点发布和查询资源须通过超级结点.

重叠型 P2P 网络中资源查询的过程如下: (1)普通结点 CP_1 向其连接的超级结点 SP_1 发送查询消息 Q, 包括自身的结点 IP、资源 ID 等内容; (2)若 SP_1 失效, 则 CP_1 将查询转发给自己的直接后继结点 CP_m, CP_m 将 Q 提交给自己连接的超级结点; (3)超级结点 SP_1 接收到普通结点 CP_1 发的查询消息 Q 后, 依据资源 ID 等信息在列表 *ResIndexList* 中查找, 若发现匹配资源, 则将包含该资源的普通结点 CP_n 的资源索引 $ResIndex_n$ 的回复消息发送给 CP_1; (4)若没有匹配资源, SP_1 根据 Chord 协议的查询算法, 在上层网络中找到超级结点 SP_x, 转发 Q 到超级结点 SP_x; (5)SP_x 处理请求 Q, 若匹配资源成功, 则向 CP_1 发送回复消息; (6)若 SP_x 匹配失败, 则将 Q 转发给自己的直接后继超级结点继续处理.

上述查询过程的优点是: (1)将能力值高、稳定性好以及网络带宽高的结点作为超级结点, 负责执行查询, 能够高效处理任务; (2)查询请求从普通结点到超级结点, 并在超级结点之间传递, 经过的结点数目较少, 大大减少了请求数据包的数量[6]; (3)双层网络逻辑上保持独立性, 当某个超级结点失效时, 下层网络依然可以转发查询请求.

同时, 在重叠型 P2P 网络中的查询也存在着缺点: (1)超级结点的负载能力有限, 如果与其相连接的普通结点的数量太多, 则超级结点必须存放所有这些普通结点所拥有的资源信息, 查询时需要处理大量的请求, 从而造成结点负担过重; (2)如果某个资源在查询中被频繁地请求, 则这些请求最终都由与拥有该资源的普通结点相连接的超级结点处理, 此时该超级结点的负载明显高于其他超级结点.

2　重叠型 P2P 网络中负载均衡的查询策略

2.1　超级结点的组织结构

在本文提出负载均衡查询策略中, 超级结点以资源索引为关键字构成一个 m 阶 B^+树, 该 B^+树中, 叶子结点为超级结点, 其存储的关键字为资源索引列表 *ResIndexList* 中元素关键字的集合; 上层结点为虚拟结点, 记录下层结点的最大关键字. 有关 B^+树数据结构的详细说明, 请参见文献[7].

因此, 超级结点的信息结构需进行以下改造: (1)超级结点的资源索引列表 *ResIndexList* 的记录需按照关键字由小至大进行排序; (2)每个超级结点需存储并维护一个 B^+树, 该 B^+树用“上层结点列表” *BTreeNodeList* 表示, 该列表存储了在 B^+树中所有上层结点; (3)*BTreeNodeList* 中每个结点还包含 2 个列表: 关键字有序列表 $KeysList = \{ Key_1, Key_2, \cdots, Key_m \}$ 和孩子结点列表 $ChildsList = \{ ChildID_1, ChildID_2, \cdots, ChildID_n \}$(若孩子结点是超级结点, 则 *ChildID* 为 Chord 网络中的结点关键字, 否则为该虚拟结点在 *BTreeNodeList* 中的 ID). 2 个列表记录的关系是: 元素 Key_i 为元素 $ChildID_i$ 结点所包含的关键字集合中的最大值.

2.2　资源信息的发布

2.2.1　*资源信息的发布过程*　资源信息的发布过程除了将资源信息提交到超级结点之外, 另外也是在超级结点上构建 B^+树的过程. 资源信息发布步骤为: (1)如果普通结点 CP_1 要申请发布资源, 先将资源的

名称、类别、关键字等关键信息与该结点自身的信息, 包括 ID、IP 地址和能力值等封装资源信息 R, 提交给与之相连的超级结点 SP_1; (2)SP_1 在自己的 $ResIndexList_{SP_1}$ 中查找资源标识 $ResIndex_R$, 若存在, 则停止发布 R; (3)若 $ResIndex_R$ 不在 $ResIndexList_{SP_1}$ 中, 则分以下 3 种情况进行处理: ① $ResIndex_R.Key < \min\{ ResIndex_i.Key, ResIndex_i \in ResIndexList_{SP_1} \}$, 即 $ResIndex_R$ 的关键字小于 $ResIndexList_{SP_1}$ 中所有元素的关键字或 $ResIndexList_{SP_1}$ 为空, 则根据 Chord 查询算法[8]在上层网络中查找结点 SP_p, 满足以下条件: $ResIndexList_{SP_p} = \min\{ ResIndexList_j, ResIndex_R.Key < \min\{ ResIndex_i.Key, ResIndex_i \in ResIndexList_j\} \}$, 即是在所有 $ResIndexList$ 元素关键字大于 $ResIndex_R$ 的超级结点中, 包含元素关键字最小的. 然后将 $ResIndex_R$ 插入 $ResIndexList_{SP_p}$ 的首位置; ② $ResIndex_R.Key > \max\{ ResIndex_i.Key, ResIndex_i \in ResIndexList \}$, 即 $ResIndex_R$ 的关键字大于 $ResIndexList$ 中所有元素的关键字, 则根据 Chord 查询算法在上层网络中查找结点 SP_n, 满足以下条件: $ResIndexList_{SP_n} = \max\{ ResIndexList_j, ResIndex_R.Key > \max\{ ResIndex_i.Key, ResIndex_i \in ResIndexList_j\} \}$, 即在所有 $ResIndexList$ 元素关键字小于 $ResIndex_R$ 的超级结点中, 包含元素关键字最大的. 然后将 $ResIndex_R$ 插入 $ResIndexList_{SP_n}$ 的末位置, 并更新 $BTreeNodeList$; ③ 当 $ResIndex_R.Key$ 介于 $ResIndexList$ 元素关键字最大值和最小值之间, 或者在上述情况中未找到指定结点, 则将 $ResIndex_R$ 插入 $ResIndexList_{SP_1}$ 相应位置.

在以上插入操作中, 插入后如果 $ResIndexList$ 中的元素个数 > m(m 为 B^+树的阶数), 则需要进行资源均衡处理.

2.2.2 *资源信息发布中的负载均衡* 当超级结点 SP 的 $ResIndexList$ 元素个数超过 B^+树的阶数 m 时, 需要对其包括的资源及包含资源的普通结点进行负载转移的操作.

负载转移过程如下: (1)将超级结点 SP_i 的 $ResIndexList$ 中第 $m/2$ 个元素之前(含第 $m/2$ 个元素)的元素保留, 对第 $m/2$ 个之后的元素(不含第 $m/2$ 个元素)进行转移; (2)从 SP_i 负责的关键字空间[9]中选取负载为 0 的超级结点 SP_0, 将第 $m/2$ 个之后的元素存放在 SP_0 的 $ResIndexList$ 中; (3)将第 $m/2$ 个元素加入 $BTreeNodeList$ 中 SP_i 的双亲结点的有序列表 $KeysList$ 中相应位置 j, 并将 SP_0 的 ID 插入列表 $ChildsList$ 中的 j+1 位置; (4)若双亲结点的 $KeyList$ 和 $ChildList$ 的元素个数大于 m, 则按照步骤(2)~(3)相同的方法分裂元素, 将需转移的元素存放在新的虚拟结点中, 并继续向上更新祖先结点, 具体过程可参见 B^+树的插入算法[10]; (5)更新 SP_i 的 $BTreeNodeList$, 并将 $BTreeNodeList$ 复制到所有超级结点中; (5)若没有超级结点的负载为 0, 则采用 S.Zoels 等[11]提出的针对层次 DHT 网络的负载均衡方法, 实时观察超级结点的实时负载, 此时如果有普通结点申请加入网络, 则将该普通结点与负载轻的超级结点连接.

在该方法中, 负载均衡取决于 B^+树的阶数 m 以及负载为0的超级结点的个数, 当发布的资源较多或者 B^+的阶数 m 选择不合理时, 该方法就变为文献[11-13]提出的算法, 而该算法会引起上层网络中查询消息的洪泛, 其效果并不是很好.

解决该问题的方法是在查询的过程中统计资源的查询"热度", 将"热度"最低的资源从超级结点中删除, 在删除资源信息的过程中空闲出超级结点. 该方法将在信息查询的负载均衡中再加以叙述.

2.3 资源信息的查询

2.3.1 *资源信息的查询过程* 资源信息的查询实际上是 B^+树的查询, 其过程如下: (1)普通结点 CP_1 首先提交一个查询资源 R 的请求 Q 到与之相连的超级结点 SP_1 上; (2)SP_1 从 Q 中解析出资源关键字 key, 并在 SP_1 的 $ResIndexList_{SP_1}$ 中进行匹配, 若匹配成功, 则返回资源信息 R, CP_1 根据 R 中的结点信息进行资源获取; (3)若 SP_1 中匹配失败, 则从 $BTreeNodeList_{SP_1}$ 中找到根结点, 从根结点开始在 B^+树中每个结点的 $KeysList$ 中进行查找, B^+树的查找算法见文献[10], 若查找成功, 则转到 $ChildsListkey$ 中对应 $ChildID$, 转向 $ChildID$ 代表的超级结点 SP_t, 并从 SP_t 的 $ResIndexList_{SP_t}$ 中找到并返回资源信息 R, CP_1 根据 R 中的结点信息进行资源获取; 若查询失败, 则返回错误消息.

2.3.2 *资源信息查询中的负载均衡* 由于每个超级结点都存储相同的 B^+树, 所以资源信息的查询可在任意一个超级结点上进行, 所以当某个超级结点的查询负载超过指定阈值时, 可将查询请求转发给前驱或后继结点进行.

在资源发布时, 负载均衡策略取决于是否存在空闲超级结点, 如果不存在空闲超级结点, 则负载均衡策略的效率会大大降低. 为了使超级结点能够在查询过程中不断被回收利用, 在资源信息的查询中, 为每个资源设立查询"热度", 以此控制资源的删除和超级结点的回收. 具体过程如下: (1)超级结点的 $ResIndexList$ 中, 设立查询热度值 n, 初始值为

0; (2)查询过程中, 某资源被查询成功 1 次, 则在超级结点的 *ResIndexList* 中, 其对应的 *ResIndex* 的查询热度加 1; (3)每个超级结点设立定时器, 每隔时间 $T(T=f[m], m$ 为 B^+树的阶数)检查 *ResIndexList* 中所有资源的查询热度, 删除查询热度为 0 的 1 个元素; (4)若删除后超级结点 SP_i 的 *ResIndexList* 中元素个数小于 $m/2$, 则从其前驱(后继)结点中分担 1 个资源, 更新 *BTreeNodeList* 中 *KeysList* 对应结点的关键字, 并更改该资源对应普通结点与超级结点的连接; (5)若前驱(后继)结点无法向 SP_i 提供资源(因为自身的 *ResIndexList* 元素个数减 1 后也小于 $m/2$), 则将 SP_i 的 *ResIndexList* 合并至前驱(后继)结点, 将超级结点 SP_i 的所有普通结点的连接转移到前驱(后继)结点, 并更新前驱(后继)结点的 *BTreeNodeList*(更新方法见文献[10]). 从而使 SP_i 成为空闲超级结点.

该方法的优点有: (1)定期释放超级结点, 保证了资源发布时的负载平衡; (2)超级结点删除"冷门"资源, 提高了超级结点的负载能力; (3)以查询热度为依据删除资源, 使得超级结点的负载能力能够集中在频繁查询的资源上, 避免了能力的闲置和浪费.

2.4　策略分析

上述重叠型 P2P 网络中查询负载均衡策略具有以下的特点: (1)集中性: 当 P2P 网络中发布的资源数较少的时候, 资源负载主要集中在少数的超级结点上, 此时的负载均衡是在这些有负载的超级结点之间进行的, 其余超级结点空闲(或备用). 此举能够充分利用超级结点的负载能力, 在负载未超出超级结点的能力范围之时, 不采取负载转移; (2)均衡性: 利用 B^+树的特性进行负载转移, 能够将超级结点的负载都控制在 $m/2$~m 之间; (3)多样性: 除了负载转移的方法进行均衡之外, 资源负载量小于 $m/2$ 的超级结点采用负载分担的方式均衡相邻结点的资源负担; (4)动态性: 根据资源的查询热度动态调整超级节点的负载量, 并以此为基础, 通过对超级结点的利用和释放调整上层 Chord 网络的规模.

3　实验过程和结果

3.1　实验过程设计

实验首先使用 GT2ITM 拓扑生成器[13], 创建一个基于 Transit2stub 模式的 P2P 网络的拓扑图, 图中包含 100 个结点, 其中包括 10 个超级结点和 90 个普通结点, B^+树的阶数 m 设为 10, 超级结点更新资源的时间间隔为 100 s. 在该 P2P 网络中进行 10 次查询实验, 在第 1 次实验中, 由普通结点提出 11 次资源发布的请求, 并提出 11 次资源查询的请求, 以后每次实验增加 10 次资源发布请求和 10 次查询请求, 最后取实验结果的平均值.

在实验中设立参数负载均衡因子 r, 其值为 $r=R_{\min}/R_{\max}$. 其中 $R_{\min}$ 表示所有超级结点中, 负载最少的结点的负载, $R_{\max}$ 表示负载最多的结点的负载数. $r\in[0, 1]$. 在此范围内, r 的取值值越大, 表示超级结点的查询负载越趋于均衡. 实验比较在重叠型 P2P 网络中, 采用传统查询方法和负载均衡的查询方法时, 比较各个超级结点的负载均衡因子的变化.

3.2　实验结果分析

本文主要分析系统的查询延时和不同的查询方法超级结点的负载情况.

图 2 比较了 2 种搜索方法的资源发布时的延时. 可以看出, 由于采用了负载均衡的策略, 当资源数较少时, 在资源发布时的延时大于传统的方法; 而当资源数较多, 接近和超过超级结点负载阈值时, 负载均衡方法的延时开始小于传统的方法.

图3 比较了 2 种搜索算法的资源查询的延时. 采取了负载均衡策略的查询可在任意一个超级结点上进行, 其平均延时几乎呈线性增长; 而传统方法的查询延时由于基于 Chord 网络进行, 其延时要大于采

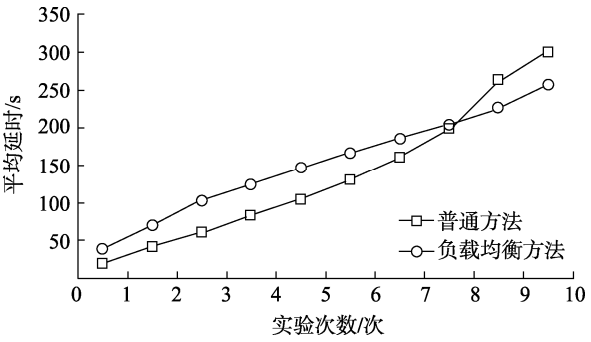

图 2　发布资源时的延时

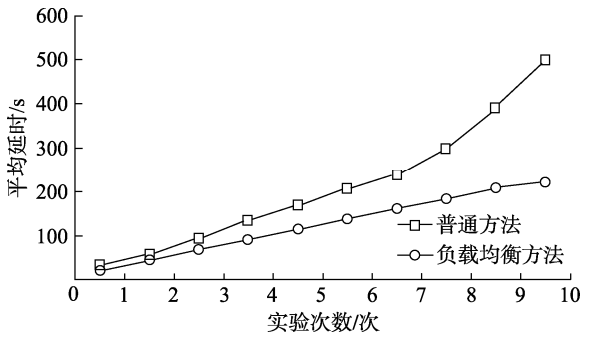

图 3　查询资源时的平均延时

取负载均衡的查询，且在超过超级结点负载阈值后，延时有明显的增加.

图 4 给出了 10 次实验中，每次网络中超级结点负载因子的变化情况(不包括负载为 0 的超级结点). 很明显，采用了负载均衡的查询策略中，各个超级结点的负载更为均衡.

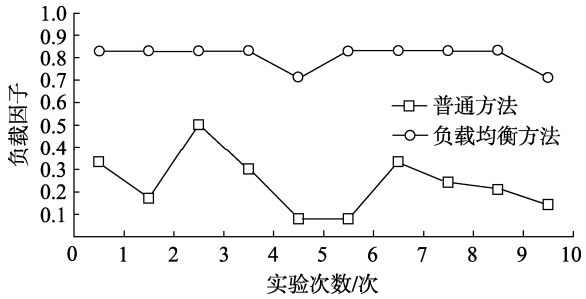

图 4　发布和查询资源时的负载因子

4　结束语

本文首先介绍了在重叠型 P2P 网络中查询资源的基本原理，并指出了该方法使得超级结点存在负载均衡的问题. 文中针对超级结点负载均衡的问题，将各个超级结点所包含的资源信息组织为 B^+树，使得资源信息分布到多个超级结点上. 通过仿真实验可知，在网络中资源发布数和查询数较多的情况下，该方法能够在相对不降低搜索性能的前提下，有效地解决重叠型P2P网络中查询时，超级结点负载不均衡的问题，有益于重叠型 P2P 网络的进一步应用.

鉴于本文对重叠型 P2P 网络的查询技术的研究还处于初级阶段，尚有许多问题需要解决，例如提高同步 *BTreeNodeList* 数据同步的效率，设定和变更 B^+树的阶数 m 以及计算超级结点定时器的间隔时间 T 等，有待于在后续的研究工作中继续完善.

5　参考文献

[1] Saroiu S, Gummadi P K, Gribble S D. A measurement study of peer-to-peer file sharing systems [C]// Proceedings of the Multimedia Computing and Networking. San Jose, USA: IEEE Computer Society Press, 2002: 156-170.

[2] Krishnamurthy S, El-Ansary S, Aurell E, et al. An analytical study of a structured overlay in the presence of dynamic membership [J]. IEEE Transactions on Networking, 2008, 16(4): 814-825.

[3] 张宇翔，张宏科. 一种层次结构化 P2P 网络中的负载均衡方法 [J]. 计算机学报, 2010, 33(9): 1580-1589.

[4] 汪永琳. 超节点结构 P2P 中负载均衡的信息索引机制 [J]. 计算机工程与科学, 2009, 31(8): 107-109.

[5] Joung Y J, Wang J C. Chord2: A two-layer chord for reducing maintenance overhead via heterogeneity [J]. Computer Networks, 2007, 51(3): 712-731.

[6] 余敏，李战怀，张龙波. 基于 super-peer 的连续查询策略 [J]. 计算机工程与应用, 2006(1): 9-12.

[7] 严蔚敏，吴伟明. 数据结构 [M]. 北京：清华大学出版社, 1997.

[8] 周伟平，刘卫国. 基于节点异构的双向查询 Chord 系统 [J]. 计算机工程, 2009, 35(2): 95-97.

[9] 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 . San Diego, USA: IEEE Computer Society Press, 2001: 149-160.

[10] Thomas H C, Charles E L, Ronald L R, 等. 算法导论 [M]. 北京：机械工业出版社, 2006.

[11] Zoels S, Despotovic Z, Kellerer W. Load balancing in a hierarchical DHT-based P2P system [C] // Proceedings of the International Conference on Collaborative Computing: Networking, Applications and Worksharing. New York, USA: IEEE Computer Society Press, 2007: 353-361.

[12] 李明，叶继华，李建莲，等. 基于 Hash 聚合的网络流量管理研究 [J]. 江西师范大学学报：自然科学版, 2011, 35(3): 174-177.

[13] 邱彤庆，陈贵海. 一种令 P2P 覆盖网络拓扑相关的通用方法 [J]. 软件学报, 2007, 18(2): 381-390.

The Research of Load Balancing Strategy for Query in Overlapped P2P Network

WANG Jue

(School of Software, East China Jiaotong University, Nanchang Jiangxi 330013, China)

Abstract: A load balancing strategy in the process of resource publishing and query, in which, all super-peers have been organized a B^+ tree , based on the property of B^+ tree′s leaf nodes, load on super-peers can be in state of balance using this algorithm. The simulations indicate that the strategy effectively solves the load unbalance problem in the overlapped P2P network.

Key words: overlapped P2P network; super-peer; load balancing; B^+ tree

(责任编辑：冉小晓)