负载敏感的 P2P 覆盖网①

王 雷，董彬如

(中国科学技术大学 自动化系，合肥 230027)

摘 要：P2P 网络较好地实现了大范围分布式环境下的节点自组织，但面向实际应用时，由于节点能力的差异带来了负载均衡问题。按照混合层次网络架构，基于 Treap 树设计了一种 P2P 覆盖网，根据负载率的优先级构造最小堆，并动态维护，实现稳定化操作。节点通过 Treap 树的信息汇聚机制获取后代节点的负载率，以此为基础实现负载均衡策略。仿真结果表明，这种覆盖网协议对于解决负载均衡问题是切实有效的。

关键词：结构化对等网络，负载均衡，二叉堆

Load-Aware Overlay Network Based on Structured P2P

WANG Lei, DONG Bin-Ru

(Department of Automation, University of Science and Technology of China, Hefei 230027, China)

Abstract: Load balancing problem is an important issue in nowadays structured P2P networks due to the heterogeneous capacity of each node. This paper presents a new overlay network based on Treap which takes the load of each node as the priority of Treap node to stabilize network topology. This overlay network provides the interface for load balancing by implementing the load information gathering procedure. It is proved to be effective according to the simulation result.

Key words: structured P2P network; load balancing; binary heap

在实际的网络应用中，由于存在着节点异构性以及用户和业务运营商对服务质量的需求，无论采用何种体系的网络架构，都需要考虑负载均衡的问题。就传统的 C/S 结构服务器集群而言，多数采取的方法是在请求到达实际服务器之前会经过一个负载均衡模块，对一些热点内容进行缓存，复制以及在某个服务器负载超重时进行重定向和重分配是基本的策略和思路。在 BT 等混合分布式网络中，由于超节点的存在，可以在更高层次的网络拓扑中实现负载均衡，此外，超节点可以获取部分的全局信息，可以在自己所知的网络拓扑中进行一些局部的优化配置并进行内容的分发，其基本思路与传统的 C/S 结构类似。而洪泛查询的 P2P 网络并无 QoS 要求，尚不能保证检索成功率，也无从谈起节点的内容负载均衡。

在第三代的结构化 P2P 网络中，由于单个节点无法获取全局信息，通常来说内容分布的不均衡性和查询请求的不均衡性相当的明显，如何平衡各节点负载显得格外重要。

针对内容负载，目前的解决思路包括了 VS (Virtual Server)策略和多 ID 策略[1,2]。而针对查询请求及副本转移带来的负载，目前的主要思路是通过修改路由表项以尽快的完成内容复制并将部分的负载转移到轻载节点上。这些算法通常会使用树形混合网络的结构，在原有的结构化 P2P 网络中建立另一层覆盖网以达到负载消息的收集分发以及负载的转移[3,4]，带来了一些消息传递的开销和计算开销[5]。同时，这些树形结构的覆盖网依赖于超级节点的存在，且有着单点失效的问题。

本文提出一个基于 Treap 树结构[6]的负载敏感的 P2P 覆盖网，并在此基础上对负载均衡策略进行测试。

① 基金项目:国家高技术研究发展计划(863)重大项目(2008AA01A317)
收稿时间:2011-04-14;收到修改稿时间:2011-05-29

1 负载敏感的Treap协议

为了测试这种为解决负载均衡问题而设计的树形结构覆盖网的有效性，首先需要对覆盖网节点的负载进行定义，其次，网络拓扑的设计也是影响覆盖网性能的重要方面。

1.1 负载率

由于加入拓扑网的节点能力大小不一，通常采用负载率作为衡量节点性能的统一指标。影响负载率的因素包括如下几项：存储空间、存储文件大小、CPU占用率、连接数、带宽占用率和上线时间。为了更好地体现不同类型的影响因素对负载率的影响程度，本文采用静态负载和动态负载分类方式，其中，存储率（存储文件大小与存储空间之比）和运算速度属于静态负载，而带宽占用（包括查询带来的连接、带宽占用以及 P2P 网络中文件转移带来的连接和带宽占用）和上线时间是动态负载。指标定义如下：

$L_{Node}=\alpha\beta^{T}+\varepsilon\eta^{T}$，其中

L_{Node} 是负载率；

$\alpha=[W_{cpu},W_{cap}]$ 中，W_{cpu} 表示 CPU 占用率，W_{cap} 表示存储空间占用率；

$\beta=[P_{cpu},P_{cap}]$ 中，P_{cpu},P_{cap} 分别为 CPU 占用率和存储空间占用率的权重系数；

$\varepsilon=[W_{link},W_{time}]$ 中，W_{link},W_{time} 分别为连接数与最大连接数之比和上线时间与节点平均在线时间比例；

$\eta=[P_{link},P_{time}]$ 为以上两项的权重系数。

1.2 Treap 树

Treap 本身是一棵二叉搜索树，它的左子树和右子树也分别是一个 Treap。和一般的二叉搜索树不同的是，Treap 为每个节点随机分配了一个优先级。Treap 在以关键码构成二叉搜索树的同时，还按优先级来满足堆的性质。但是这里要注意的是 Treap 和二叉堆有一点不同，就是二叉堆必须是完全二叉树，而 Treap 可以不是。

Treap 的节点插入和删除操作均用到了类似 AVL 和 RB 树的旋转操作，如图 1 所示。

Treap 树的主要操作包括：插入、删除、分离、合并。

（1）插入操作

给节点随机分配一个优先级，先和二叉搜索树的插入一样，先把要插入的点插入到一个叶子上，然后跟维护堆一样，如果当前节点的优先级比根大则需要考虑旋转，如果当前节点是根的左儿子则右旋，如果当前节点是根的右儿子则左旋。

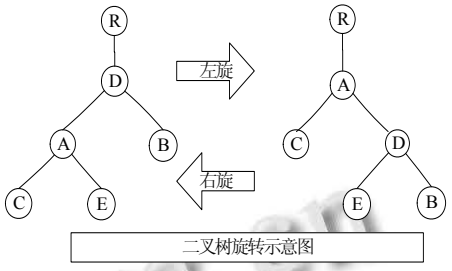

图 1 二叉树旋转示意图

插入操作采用递归形式，在递归调用完成后判断是否满足堆性质，如果不满足就旋转。由于旋转是 O(1)的，最多进行 h 次(h 是树的高度),插入的复杂度是 O(h)的，在期望情况下 h=$O(\log N)$，所以它的期望复杂度是$O(\log N)$。

（2）删除操作

因为 Treap 满足堆性质，当需要删除节点时，类似堆的操作，需要把要删除的节点旋转到叶节点上，然后直接删除。具体的方法就是每次找到优先级最大的儿子，向与其相反的方向旋转，直至叶节点，然后删除。删除最多进行 O(h)次旋转，期望复杂度是$O(\log N)$。

（3）分离操作

要把一个 Treap 按大小分成两个 Treap，在需要分开的位置加一个虚拟节点，然后旋至根节点删除，左右两个子树即为所需的两个 Treap 了。根据二叉搜索树的性质，这时左子树的所有节点都小于右子树的节点。时间相当于一次插入操作的复杂度，亦即$O(\log N)$。

（4）合并操作

合并是指把两个平衡树（或者其他的有序表）合并成一个平衡树（有序表），其中第一个树（表）的所有节点都必须小于或等于第二个树（表）中的所有节点，这也是上面的分离操作的结果所满足的条件。Treap 的合并操作的过程和分离相反，只要加一个虚拟的根，把两棵树分别作为左右子树，然后把根删除。时间复杂度和删除一样，也是期望$O(\log N)$。

1.3 Treap 协议

基于如上的负载率定义和 Treap 树的结构，本文

提出一种 P2P 覆盖网的自组织协议，以节点 ID 作为 Treap 树节点的关键码，以节点负载率为 Treap 树节点的旋转优先级，从而达到负载敏感的目的。以下为 Treap 协议的基本操作。

（1）节点的初始化

在节点加入到 Treap 网络中时，需要对节点自身的状态做评估和初始化，包括节点的索引，负载率，而置父节点，左右子节点为空。

节点本身的数据结构定义如下:

```
struct TreapNode{
    Treap_key key;
    Treap_priority load;
    TreapNode left, right, parent;
    };
```

（2）节点的加入

当第一个节点加入时，由于 Treap 网络为空，因此，它作为 Treap 树的根节点，其负载即为整个网络的负载。其后的节点加入时，随机选取 Treap 网络中的节点加入，上溯至根节点为其分配索引及所负责键值空间(注意，由于 Treap 树所具备的最小堆的性质，根节点总是有能力承担类似的任务)，由于尚无带宽占用和上线时间，因此在加入时的节点负载率被定义为从根节点所获知的 Treap 网络平均负载与自身静态负载的平均数。按照 Treap 树的加入操作，节点被插入到叶节点的位置后，以负载率作为优先级，进行堆的维护操作，将整个 Treap 网络维护为一个动态的最小堆。此外，为了方便查询操作，每个节点需维护左子树和右子树的最小和最大索引值。

（3）节点的退出

注意一点，在给定优先级的情况下，只要是用符合要求的操作，通过任何方式得出的 Treap 都是一样的。因此，在负载信息收集的周期内，整个 Treap 树是一颗确定优先级的树，其删除和加入的操作并不会导致 Treap 树的退化而影响 Treap 树操作的期望值。因此，类似 Treap 树的删除操作，只需要找到将要离开的节点较大优先级的儿子，按相反方向进行旋转直至到达叶节点为止即可。

在节点出现意外退出的情况下，由于其左子树和右子树仍保持着一个 Treap 的性质，因此，仅需要将其左子节点或右子节点重新加入到 Treap 网络中，在下一个负载均衡周期时进行动态调整的操作即可。

（4）节点的动态调整

本文在负载率的计算时考虑到了动态负载的问题，因此，可以预见在一个有着频繁文件交换操作和节点加入退出的动态网络中，其网络拓扑的恢复能力至关重要，此外，在调整周期内需要保证 Treap 树的拓扑正确性和不退化性，按照负载均衡的周期来进行调整是一个较好的选择。动态调整与普通二叉堆维护的操作类似，但与普通二叉堆不同的是，Treap 并不保证平衡，亦即，同一节点的左子树和右子树高度相差可能大于 1，因此需要从叶节点开始操作，逐步保证最小堆的性质，使得负载率较低的节点始终处于 Treap 网络中的较高层位置。注意，无论是在节点的加入过程或是动态调整过程中，Treap 仍保有二叉查找树的性质，亦即，同层中节点是按索引升序排列的。

```
stabilize(T)
    r←Root(T)
    do cur←cur.right
        while cur.right!=null
    do
        stabilize(cur)
        if cur.load<cur.parent.load
        &&cur.parent.right=cur
            Rotate_Left(cur.parent)
        if cur..load<cur.parent.load
            &&cur.parent.left=cur
            Rotate_Right(cur.parent)
    cur=cur.parent
    while cur!=r
```

（5）节点的查找操作

如前所述，在节点的加入操作中，节点从根节点获得索引值，根节点将此索引值与当前记录的最小索引与最大索引进行比较以进行更新后，根据二叉查找树的性质将其分配给左子节点或右子节点递归执行此操作。因此，在 Treap 网络中每个非叶节点均保有其左子树的最小索引和右子树的最大索引。在进行索引查找时，随机选取 Treap 网络中的非叶节点进行查询，若不在其记录的最小索引和最大索引之间，则递归向父节点查询，直至根节点；否则根据查找树的查找规则，向子树查找。

```
find_nearest(cur, x)
    cur←Random
```

```
r←Root(Random)
if cur.key=x
      return cur;
if cur.key!=r
      if cur.min>x|| cur.max<x
            do find_nearest(cur.parent, x)
                  while cur!=r
      else if cur.min<x<cur.max
                  if cur.key<x
                     do find_nearest(cur.right, x)
                        while cur.right!=null
                         &&cur.left!=null
                  else do find_nearest(cur.left, x)
                        while cur.right!=null
                         &&cur.left!=null
return false;
```

1.4 Treap 网络的负载均衡策略

在 Treap 网络中，由于以负载率为优先级创建了一个 Treap 树的拓扑，使得负载均衡策略变得格外简单，描述如下：

（1）节点统计自己所维护的后代节点数，用算术平均方法计算包括自身与后代节点的平均负载率，周期性的向父节点发送节点数和平均负载率，以及自身的存储空间、连接数和带宽占用率等局部网络的信息；

（2）父节点收到子节点信息后，同样统计自己所维护的 Treap 节点数和平均负载率，向自己的父节点发送维护节点数、平均负载率、自身存储空间、连接数、带宽占用率等信息，递归调用，直至根节点。

（3）根节点在每个周期完成负载信息收集之后进行计算，并将平均负载率配额沿树下溯到叶节点，各节点根据自身的负载状况，向父节点和子节点发送相关的过载或轻载的消息，再根据回馈决定采取推或者拉的措施，实现负载均衡。

其它树型结构覆盖网只能在固定的超级节点上完成上述工作，存在单点失效问题，从而导致负载均衡失败。本文提出的架构，可以根据节点负载率动态产生超级节点。当超级节点失效时，还可以通过主动进行分离和合并操作，产生新的超级节点，从而保证了负载均衡策略的执行。由于动态调整所需的负载率信息是在负载消息传递过程进行捎带更新的，也可以有效地节约网络的维护代价。

2 仿真实现与分析

2.1 仿真环境

仿真实验的硬件平台 CPU 主频 2.60GHz，内存 2G，软件方面包括 Ubuntu10.04，jdk-1.6.0，Eclipse, Peersim-1.4.0。

在 Peersim 的环境下实现了该协议并测试了如上所描述负载均衡策略。Peersim 是一个开源的 Java 实现的 P2P 网络仿真平台，支持周期驱动模型和事件驱动模型两种不同的方式分别来模拟静态和动态网络的模型。

2.2 仿真过程与结果分析

为了数据处理和曲线拟合的便利，假定节点的负载能力是一个呈现 Gnutella 式分布的均为十的幂次方，同时在负载均衡时对节点初始负载采取整数初始化的方式。在测试负载均衡策略的有效性时，对节点的负载率初始化采取了一个较极端的方式，亦即，节点负载率是一个在 0-100(%)之间的均匀分布的情形。

图 2 是根据网络规模不同而仿真得出的 Treap 查找跳数与经典的结构化 P2P 网络 Chord 查找跳数的比较。在网络规模较小时，Treap 网络与 Chord 的查找平均跳数非常的接近，而在网络规模逐渐变大时，Treap 网络略优于 Chord 网络。

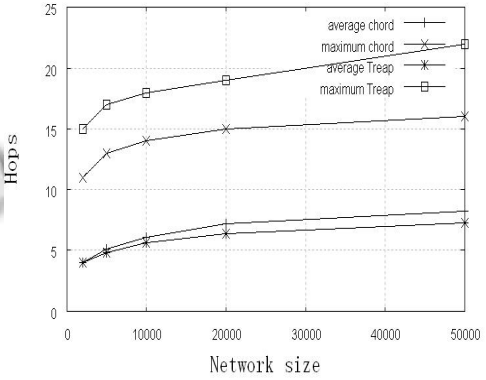

图 2 路由跳数比较

同时，在图中可以观察到，Treap 网络的路由最大跳数要高过 Chord，但是对于系统性能无本质影响。

节点的初始负载率分布呈现一个从 0-100(%)均匀分布的态势，经过一个负载均衡周期后，轻载节点和过载节点均大幅度减少，总的分布呈现一个类正态分布的态势。对比表明，该覆盖网协议及负载均衡策略效果良好。

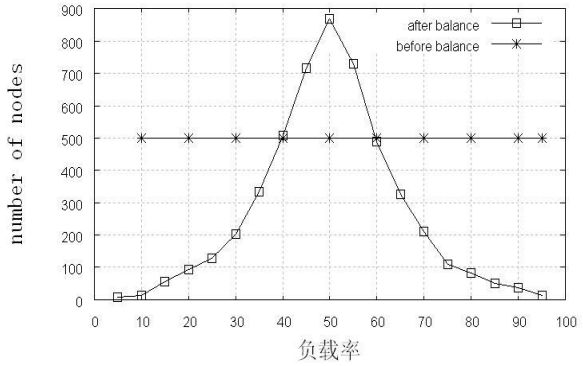

图 3　负载均衡策略前后

3　结语

本文提出了一种负载敏感的树形混合层次覆盖网，为解决传统的结构化 P2P 网络的负载均衡问题提出了一种新的思路。与一些普通的多叉树的混合层次网络相比，Treap 具备一个二叉堆的可合并性与可分离性，从而解决了单点失效的问题。同时，该网络也可以独立运作，提供一个动态 P2P 网络的基本功能。经过仿真模拟，该方案切实可行且运行良好。在此覆盖网协议的基础上，可以考虑将其与传统的结构化 P2P 网络相结合，深入的探讨不同负载率分布的情况下，不同负载均衡策略的有效性。

参考文献

1 A-Theotokis S, Spinellis D. A Survey of P2P Content Distribution Technologies. ACM Computing Surveys, December,2004,36(4):335–371.

2 Xia Y, Chen SG, Korgaonkar V. Load Balancing with Multiple Hash Functions in P2P Networks. Proc. of the 12th International Parallel and Distributed Processing Symposium, 2006.

3 Godfrey B, lakshiminarayanan K, et al. Load Balancing in Dynamic Structured P2PSystems. Proc. of 2nd International Workshop on P2P Systems.2003.

4 Zhu YW, Hu YM. Towards Efficient Load Balancing in Structured P2P Systems. Proc. of the 18th International Parallel and Distributed Processing Symposium, 2004.

5 熊伟,谢冬青.一种结构化 P2P 协议中的自适应负载均衡方法.软件学报,2009,20(3):660–670.

6 Martinez C, Rouva S. Randomized Binary Search Trees. Journal of ACM, 1998, 45(3).

（上接第 45 页）

7 Perona P, Malik J. Scale space and edge detection using anisotropic diffusion. IEEE Transactions on Pattern Analysis and Machine intelligence PAMI,1990,12:629–639.

8 M Kass, A Witkin D. Terzopoulos. Snakes: Active contour models. International Journal of Coumputer Vision, 1988,1(4):11.

9 王大凯,侯榆青,彭进业.图像处理中的偏微分方程方法.北京:科学出版社,2008.

10 祝轩,周明全,朱春香,等.曲率驱动与边缘停止相结合的非线性扩散及其在图像去噪中的应用.光子学报,2008, 37(3):609–612.

11 Osher S, Rudin LI. Feature oriented image enhancement using shock filters. SlAM.J. Num. AnaL,1990,(27)919–940.

12 Ghung DH, Sapiro G. On the level lines and geometry of vector-valued images. IEEE, IP,2000,9(7):241–243.

13 陆金甫,关治.偏微分方程数值解法.第 2 版.北京:清华大学出版社,2004.254–256.