

结构化对等网络中基于物理邻近性的负载均衡

李 英, 刘广亮

(南阳师范学院 软件学院, 河南 南阳 473061)

摘 要: 结构化 P2P 网络中的负载均衡是当前 P2P 研究领域中的一个热点, DHT 的使用以及节点性能的不同导致系统中节点负载不均衡而影响整个系统的效率. 文中提出了负载均衡算法. 每个节点周期性的收集其邻居节点负载信息, 然后选择链路延迟较小的节点进行负载转移, 并且对不同的瓶颈资源提出不同的负载衡量标准. 仿真实验结果表明, 算法能有效的进行负载均衡.

关键词: 对等网络; 物理邻近性; 负载均衡; 虚拟服务器; 分布式哈希表

中图分类号: TP393

文献标识码: A

文章编号: 1000-7180(2013)03-0071-04

Physical Proximity Based Load Balancing in Structured P2P Network

LI Ying, LIU Guang-liang

(School of Software, Nanyang Normal University, Nanyang 473061, China)

Abstract: Load balancing of structured P2P network is now a hotspot in the research of P2P area. The use of DHT and the heterogeneity of node capacities led to load imbalance which affected the efficiency of the system. A load balancing algorithm is presented in this paper. Every node gathers load information of its neighbor nodes periodically, chooses a physically close node to transfer the load and adopts different criteria for different bottleneck resources. Experimental results prove that the algorithm can balance the load effectively.

Key words: P2P; physical proximity; load balancing; virtual servers; DHT

1 引言

DHT 方法是结构化 P2P 系统的关键技术. 为了确定节点中 DHT 网络中的位置, 以及它在逻辑网络中的路由表, 在节点加入 DHT 网络时, 我们需要为每个节点生成一个能够唯一标识该节点的标识节点 NodeId. 并且每个节点上都存储着 (key, value) 对, 其中 key 表示目标节点, value 表示目标节点上的一些信息. 这些信息有可能是代表内容的索引, 也可以是内容本身. 节点通过存储 (key, value) 对的方式来维护资源的信息^[1].

DHT 的使用使得系统中节点的分布是随机的, 从而每个节点负责的标识符空间的大小也是不同

的, 某些节点可能拥有大段的标识符空间而导致存储大量的数据. 因此, 这种资源存储方式必然会导致负载的不均衡.

现有的负载均衡算法^[2-5]存在以下不足:

(1) 负载的转移并没有考虑节点之间的物理邻近性, 当两个节点之间的链路延迟较大时, 就会增加负载转移的开销.

(2) 只考虑了一种瓶颈资源, 比如存储空间. 而对于存储节点上的不同的文件, 其下载请求的数量也是不同的, 并且各个节点的性能也有着差异, 比如说带宽不同, 如果节点上存储的不是文件的索引信息, 而是文件的具体内容时, 负载不均衡会造成网络流量分布不均衡, 导致瓶颈产生.

收稿日期: 2012-06-20; **修回日期:** 2012-07-22

基金项目: 河南省基础与前沿技术研究计划项目(122300410302); 河南省科技攻关项目(102102210388); 南阳师范学院科研基金(QN2012048)

本文提出了负载均衡算法,在物理邻近的节点之间进行负载转移,减少了负载转移开销,并且对存储空间、接入带宽这两种常见的瓶颈资源选择不同的负载衡量标准.

2 负载均衡算法

为了解决底层物理网络和叠加网络拓扑的构建不匹配的问题,研究人员进行了大量的研究^[6-8].文献[6]介绍了使用 IP 地址来判断 P2P 节点的物理拓扑;文献[7]提出了网络相关簇技术;文献[8]提出了将节点分组的方法.本文提出的负载均衡算法前提是利用网络相关簇技术,将节点组织成簇,从而得到节点的邻居信息.

节点的路由表由三个部分构成:第一部分是 Successor()表示后继活动节点的地址,用 Predecessor()表示前驱活动节点的地址.第二部分由 Finger 表组成.第三部分是 Neighbor 表,用来记录节点的邻居信息,包括该节点的负载、节点存储能力、带宽以及与本节点之间的链路延迟等信息.

2.1 虚拟服务器(virtual servers)

负载均衡算法使用了虚拟服务器^[9].可以将虚拟服务器看成是 DHT 中的节点.每个物理节点可以对应着多个虚拟服务器.如果物理节点超载,就需要用 DHT 中的删除和插入操作来将一些虚拟服务器从超载节点转移到轻载节点.

转移的虚拟服务器大小和参与本次转移的节点之间链路延迟的乘积和表示为负载转移开销.在均衡节点负载时应使负载转移开销较小,以占用较小的系统带宽资源.

2.2 存储负载衡量

节点 A 的存储负载 $LS_A = \sum_{i=1}^m Fsize_i$, 其中: m 为该节点上存储的文件数目, $Fsize_i$ 为文件 i 的大小.

节点 A 的存储利用率表示为 $utlS_A = LS_A / CS_A$, 其中: LS_A 为 A 的存储负载, CS_A 为 A 的存储能力.

节点 A 的邻近节点(包括 A)的存储负载和与存储能力和之比,表示为节点 A 系统局部存储利用

率,即 $utlS_A = \frac{\sum_{i=1}^p LS_i}{\sum_{i=1}^p CS_i}$, 其中 p 为 A 的邻居节点个数.

当 LS_A 小于或等于 $(\frac{\sum_{i=1}^p LS_i}{\sum_{i=1}^p CS_i} + \epsilon) CS_A$ 时认为节点

为存储轻载 LNoS(Light Node of Storage), 反之则为超载节点 HNoS(Heavy Node of Storage). 其中 ϵ 为可调参数,在实际应用中 ϵ 的值可以根据具体情况而定,当 ϵ 为 0 时,负载均衡质量达到最好,但此时负载转移开销也达到最大.

2.3 带宽负载衡量

节点 A 的带宽负载 $LB_A = \frac{\sum_{i=1}^m n_i \times Fsize_i}{B_A}$

其中:

m 为该节点上存储的文件数目;

$Fsize_i$ 为文件 i 的大小;

n_i : 该文件 i 在特定时间内被请求下载的次数,即频率;

B_A : 用来表示节点 A 的带宽.

LB 反映节点的负载受到文件的影响.如果节点上文件数目越多,文件越大,文件的请求下载次数越多,则 LB 也就越大,则该节点的带宽负载也就越大.如果 LB 过大可能会导致该节点成为瓶颈.这里需要考虑两种情况,第一种情况,考虑不同节点的性能差异,例如节点的带宽,有些节点虽然接受的下载请求不多,但是由于该节点的带宽较小,所有分配给每个请求的带宽就很小,这样可能会成为瓶颈.第二种情况,需要考虑节点上的文件对带宽的影响,例如文件在某一时刻的下载量过大时,也可能导致瓶颈的产生.

LB 的值不是固定不变的,会随着时间的变化而变化.因此需要每隔一段时间第 LB 的值进行更新.而 LB 的值变化主要体现为文件的下载频率 n 的变化,所以就需要对 n 进行更新, n 的值可根据下式进行更新^[10]:

$$n_{i+1} = (1 - \alpha)n_{i-1} + \alpha n_i$$

其中:

n_{i-1} : 第 $i-1$ 次的下载频率值;

n_i : 第 i 次的下载频率;

n_{i+1} : 第 $i+1$ 次的下载频率;

α : 表示更新的频率值和原值之间的关联性的权重,权重可根据实际情况选取, α 值越小,则关联性越大.

当节点 A 的下载量 $DL_A = \sum_{i=1}^m n_i \times Fsize_i$ 小于

或等于 $\frac{\sum_{i=1}^p DL_i}{\sum_{i=1}^p B_i} \times B_A \times (1+\delta)$ 时认为节点为带宽轻

载 LNoB(Light Node of Bandwidth), 反之则为超载节点 HNoB(Heavy Node of Bandwidth). 其中 p 为 A 的邻居节点个数, δ 为可调参数, 可以根据实际情况, 用来调解负载均衡质量和负载转移开销的关系.

2.4 算法描述

对于超载节点, 选择将要删除的虚拟服务器, 将其转移给其邻居表(Neighbor 表)中相应的轻载节点. 为了解决热点文件问题, 可以将热点文件进行复制从而解决负载均衡问题, 但是为了维护数据的一致性, 与将其转移到有高接入带宽的节点上相比产生额外开销较大. 当然, 将复制或缓存融入负载均衡算法中能有效提高系统性能, 在以后的工作中将对此进行研究.

函数 Choose-VS-To-Remove 选择需要删除的虚拟服务器, 函数 Choose-Node-To-Accept 选择接收虚拟服务器的物理节点. 负载均衡算法通过把负载较重节点上的虚拟服务器转移给负载较轻的节点, 使得各个节点负载达到均衡. 文中使用启发式算法首先选择超载节点上需要删除的虚拟服务器, 再查找能够接收该虚拟服务器的物理节点, 最后将虚拟服务器插入到该节点. 分布式负载均衡算法如下:

Load-Balancing-Algorithm(A , Neighbor-Table)

 Compute utl_A ;

 if($utl_A > 1$) then

$utl_A = 1$;

 end if

 Target Load of Storage

$$TLS_A = \left(\frac{\sum_{i=1}^p LS_i}{\sum_{i=1}^p CS_i} + \epsilon \right) CS_A;$$

 Target Load of Bandwidth

$$TLB_A = \frac{\sum_{i=1}^p n \times Fsize}{\sum_{i=1}^p B_i} \times B_A \times (1+\delta);$$

 if($LS_A \leq TLS_A$ and $DL_A \leq TLB_A$) then

 return; /* A is LNoS and LNoB */

 end if

 Removed-VS = Choose-VS-To-Remove(A);

 Accept-Node = Choose-Node-To-Accept(Removed-VS, Neighbor-Table);

 if(Accept-Node != nil) then

 Transfer Removed-VS from node A to Accept-Node;

 end if

在函数 Choose-VS-To-Remove 中, 当节点 A 为 HNoS 时, 选择能使节点从超载节点变为轻载节点的存储量最小的虚拟服务器, 若不存在, 则选择存储量最大的虚拟服务器转移. 当节点 A 为 HNoB 时, 选择能使节点从超载节点变为轻载节点的存储量最小的虚拟服务器(减少负载转移开销), 若不存在, 则选择下载量最大的虚拟服务器转移. 当节点 A 为 HNoS 且 HNoB 时, 选择能使节点同时从存储超载和带宽超载变为轻载节点的存储量最小的虚拟服务器, 若不存在, 则选择能使节点从 HNoS 变为 LNoS 的虚拟服务器中下载量最大的, 若不存在使节点从 HNoS 变为 LNoS 的虚拟服务器, 则选择存储量最大的虚拟服务器转移. 由于负载均衡算法是周期性的执行, 因此可以认为经过多次负载均衡后节点 A 将会变为非超载节点. 所以算法不考虑一次转移多个虚拟服务器.

节点 I 在接收虚拟服务器后, 需要计算对虚拟服务器的利用率, 可以用 utl_I 来表示利用率. 两个节点之间存在着链路延迟, 用 lat_{A-I} 表示节点 A 和节点 I 之间的链路延迟. 当 $p < 0$ 时, 说明找到了更合适的节点来接受虚拟服务器. 对负载均衡效果和负载转移速度的重视程度用可调参数 a, b, c 来反映. 如果没有合适的节点可以接收虚拟服务器, 函数将返回 nil. 此时节点 A 需要等到下一个周期进行负载均衡. 选择接收虚拟服务器的物理节点如下:

Choose-Node-To-Accept(Removed-VS, Neighbor-Table)

 Accept-Node = nil;

 for each node I in Neighbor-Table do

 if($LS_I + LS_{Removed-VS} \leq TLS_I$ and $DL_I + DL_{Removed-VS} \leq TLB_I$) then

 if(Accept-Node == nil) then

 Accept-Node = I ;

 else

$$utl_imp = \frac{utl_I - utl_{Accept_Node}}{utl_{Accept_Node}};$$

$$lat_imp = \frac{lat_{A-I} - lat_{A-Accept_Node}}{lat_{A-Accept_Node}};$$

$$p = a \times utl_imp + b \times LB_imp + c \times lat_imp$$

 if($p < 0$) then

 Accept-Node = I ;

 end if

 end if

```

end if
return Accept—Node

```

3 仿真实验

3.1 实验设置

仿真实验是在结构化 P2P 系统 Chord 协议上完成的. 首先将随机产生 1024 个节点, 节点的带宽在 $[500\text{KB/s}, 1\text{MB/s}]$ 之间, 存储空间在 $[500\text{M}, 1\text{G}]$ 之间, 将这些节点分成 16 个簇, 簇内节点之间的链路延迟在 $[50\text{ms}, 200\text{ms}]$ 之间. 然后保持网络处于静态状态, 插入 100000 个文件 (包括 50 个热门文件), 文件大小在 $[1\text{MB}, 10\text{MB}]$ 之间. 令 $a=b=c$, $\epsilon=(1-\text{utilS}_A)/2$, $\delta=0.5$.

3.2 实验结果及分析

节点存储利用率如图 1 所示. 可以看出, 执行一次负载均衡算法后, 节点存储利用率基本分布在 $20\% \sim 80\%$ 之间, 与均衡前相比有了很大的改进.

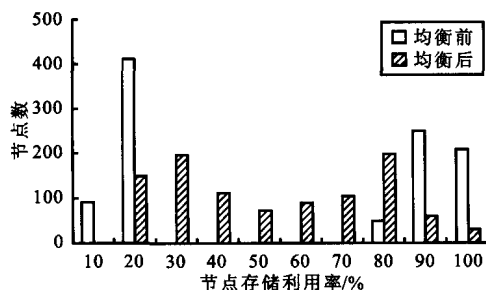


图 1 节点存储利用率对比

实验通过对已插入的 100000 个文件中随机选择 10000 个进行下载请求, 这里的 10000 个文件不包括 50 个热门文件, 以及对 50 个热门文件分别进行 10 次到 100 次的下载请求. 假设节点的带宽会均匀分配给每个请求, 因此节点接受到新的请求时, 就需要重新计算每个请求的下载时间, 并且更新对各个文件的连接数. 图 2 反映了随热门文件下载次数增加平均下载时间变化的情况.

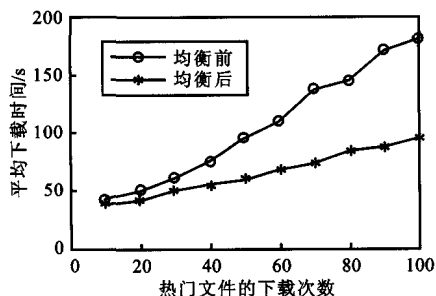


图 2 平均下载时间随热门文件下载次数对比

当同一时刻请求热门文件下载的数量增加时, 会增加对同一个热门文件的请求次数, 从而导致该

文件所在节点的负载增加, 那么节点上的每个下载请求所被分配的带宽也就相应的减小, 这样使得在这个节点上下载文件需要花费更长的时间. 因此无负载均衡的 Chord 算法的文件平均下载时间随热门文件下载次数迅速增长. 而负载均衡算法发现节点超负载时, 会将下载请求转移到负载较轻的节点上, 使文件的平均下载时间的增长速度在一定程度上得到了减缓.

4 结束语

本文针对结构化对等网络提出了一种负载均衡算法. 每个节点周期性的收集其邻居节点负载信息, 然后选择链路延迟较小的节点进行负载转移, 并同时考虑存在存储空间和带宽两种瓶颈资源时的负载均衡问题, 适合于例如共享文件系统等的 P2P 应用. 仿真实验结果表明, 算法能有效的均衡负载和减少负载转移开销.

参考文献:

- [1] Ion Stoica, Robert Morris, David Liben—Nowell, et al. Chord: a scalable Peer—to—Peer lookup protocol for internet applications [J]. IEEE/ACM Transactions on Networking, 2003, 11(1): 17—32.
- [2] Rao A, Lakshminarayanan K, Surana S, et al. Load balancing in structured P2P systems [C]// IPTPS. USA, Berkeleg, CA, 2003.
- [3] Godfrey B, Lakshminarayanan K, Surana S, et al. Load balancing in dynamic structured P2P systems [C]// IEEE INFOCOM. Hong Kong, 2004.
- [4] Zhu Y, Hu Y. Efficient, proximity—aware load balancing for DHT—based P2P systems [J]. IEEE Trans on Parallel and Distributed Systems, 2005, 16(4): 349—361.
- [5] Byers J, Considine J, Mitzenmacher M. Simple load balancing for distributed hash tables [C]// IPTPS. Berkeleg, CA, USA, 2003.
- [6] Padmanabhan V N, Subramanian L. An investigation of geographic mapping techniques for internet hosts [C]// Proceeding of the ACM SIGCOMM 2001 Conference on Application, Technologies, Architectures, and Protocols for Computer Communication. San Diego, CA, USA, 2001: 173—185.
- [7] Krishnamurthy B, Wang J. On network—aware clustering of web clients [C]// Proceeding of the ACM SIGCOMM 2001 Conference on Application, Tech—

(下转第 79 页)

[5] 唐笑秋. 基于负载均衡的无线 Mesh 网络路由算法研究[D]. 北京: 北京邮电大学, 2008.

[6] 赵蕴龙, 马延龙, 李香. 无线 Mesh 网链路不相交多路径路由策略[J]. 哈尔滨工程大学学报, 2008, 29(9): 968—972.

[7] 杨孟珂, 杨亚涛, 白中英. 基于 HWMP 协议的无线 Mesh 网络多网关路由协议研究[J]. 微电子学与计算机, 2009, 26(12): 4—8.

[8] 丘小兰, 李陶深, 葛志辉. 无线 Mesh 网络多网关选播 QoS 路由模型研究[J]. 微电子学与计算机, 2010, 27(8): 116—118.

作者简介:



李陶深 男, (1957—), 博士, 教授. 研究方向为网络计算与信息安全、无线 Mesh 网络、分布式数据库.



王春霞 女, (1985—), 硕士, 助教. 研究方向为无线 Mesh 网络.



葛志辉 男, (1976—), 博士, 副教授. 研究方向为无线网络、网络协议性能优化.

(上接第 74 页)

nologies, Architectures, and Protocols for Computer Communication. Stockholm , Sweden, 2000: 97 — 110.

[8] Sylvia Ratnasamy, Mark Handley, Richard Karp, et al. Topologically — aware overlay construction and server selection [C] // Proceedings of INFOCOM. USA; New York, 2002: 1190—1199.

[9] Dabek F, Kaashoek M F, Karger D, et al. Wide—area cooperative storage with CFS [C] // The 18th ACM Symp Operating Systems Principles (SOSP), Banff,

Canada, 2001.

[10] 周健, 洪佩琳, 李津生. DHT 网络中一种基于树型结构的负载均衡方案[J]. 小型微型计算机系统, 2006, 27(11): 2042—2046

作者简介:

李 英 女, (1982—), 助教, 硕士. 研究方向为计算机网络、自然语言处理.

刘广亮 (1978—), 讲师, 硕士. 研究方向为计算机网络.